

DEVELOPPEMENT MOBILE

DR. KABORE RAOGO

PLAN DU COURS

► INTRODUCTION

► GENERALITES

► ARCHITECTURE ANDROID

► DEVELOPPEMENT D'APPLICATION AVEC ANDROID

☐ CM : 15 H
☐ TP : 20 H
☐ Projet

BIBLIOGRAPHIE

- ▶ <http://developer.android.com/>
- ▶ Support de cours Jean François Lalande – Développement sous Android
- ▶ Développer des applications mobiles pour les Google phones :
<http://kslimi.files.wordpress.com/2011/01/dunod-android.pdf>
- ▶ Développement d'applications Android de M. Dalmau, IUT de Bayonne

OBJECTIFS

- ▶ être capable de développer une application fonctionnant sur la plateforme Android en utilisant le SDK fournit par Google.
- ▶ Connaître les spécificités du développement mobile et en particulier d'Android
- ▶ savoir utiliser les fonctionnalités spécifiques aux Smartphones et Tablettes Android

QUI SUIS-JE?

Dr. KABORÉ Raogo

[rkabore@yahoo.fr]

- Bac E, Lycée Technique de Bouaké
- Ingénieur Informaticien, Institut Africain d'Informatique (I.A.I), Libreville, Gabon
- Mastère Spécialisé en Cyber sécurité, Institut Mines-Télécom Atlantique, Brest, France
- Doctorat Cyber sécurité des Infrastructures Critiques, Institut Mines-Télécom Atlantique, Brest, France

INTRODUCTION



INTRODUCTION

Smartphones & Tablettes



INTRODUCTION

- ▶ ANDROID QU'Est-ce QUE C'EST?
 - ▶ Système d'exploitation pour terminaux mobiles
 - ▶ Basé sur Linux
 - ▶ Open Source (licence Apache)

INTRODUCTION

► FONCTIONNALITES

- Framework applicatif avec réutilisation et remplacement possible des composants
- ART : Android RunTime (machine virtuelle optimisée pour les périphériques mobiles)
- Navigateur intégré basé sur le moteur WebKit (Open Source)
- Librairie 2D dédiée
- Gestion de la 3D basée sur une implémentation d'OpenGL ES 1.0 (avec support de l'accélération matérielle)
- Base de données SQLite
- Gestion des écrans tactiles et du Multitouch

INTRODUCTION

► FONCTIONNALITES

- MultiMedia : support de la plupart des formats classiques d'images, de vidéos et audios (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- Téléphonie GSM (selon hardware)
- Bluetooth, EDGE, 3G, 4G, 5G et Wi-Fi
- Camera, GPS, compas et accéléromètre
- Environnement de développement riche incluant :
 - Un émulateur (avec une interface de contrôle)
 - Des outils de débogage
 - Outils de profiling mémoire et performance
 - Un plugin pour l'IDE Eclipse

INTRODUCTION

▶ HISTORIQUE

- ▶ Développé par la startup Android Inc.
- ▶ Juillet 2005 : Rachat par Google
- ▶ Novembre 2007 : Open Handset Alliance (OHA)
 - Texas Instruments, Broadcom Corporation, Google, HTC, Intel, LG, Marvell Technology Group, Motorola, NVidia, Qualcomm, Samsung Electronics, Sprint Nextel, T-Mobile
 - Décembre 2008 : ARM Holdings, Atheros Communications, Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp, Vodafone

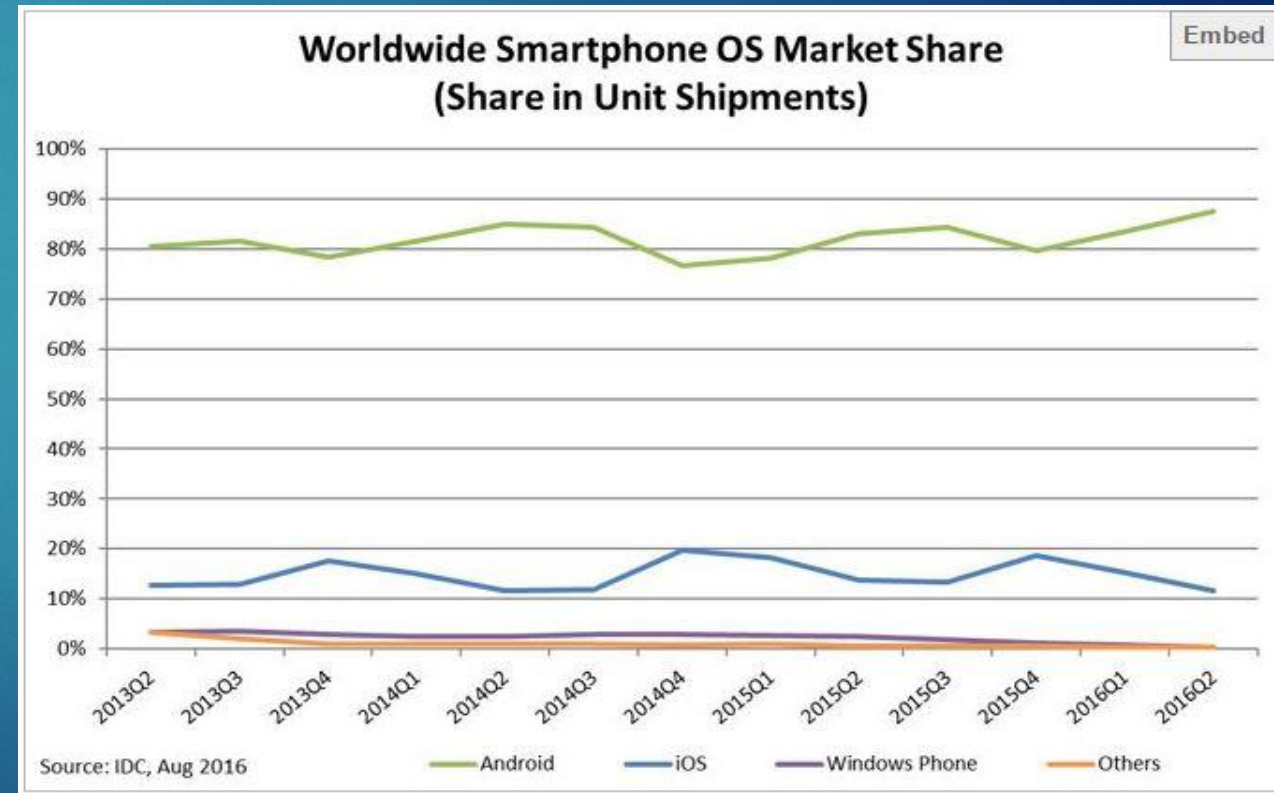
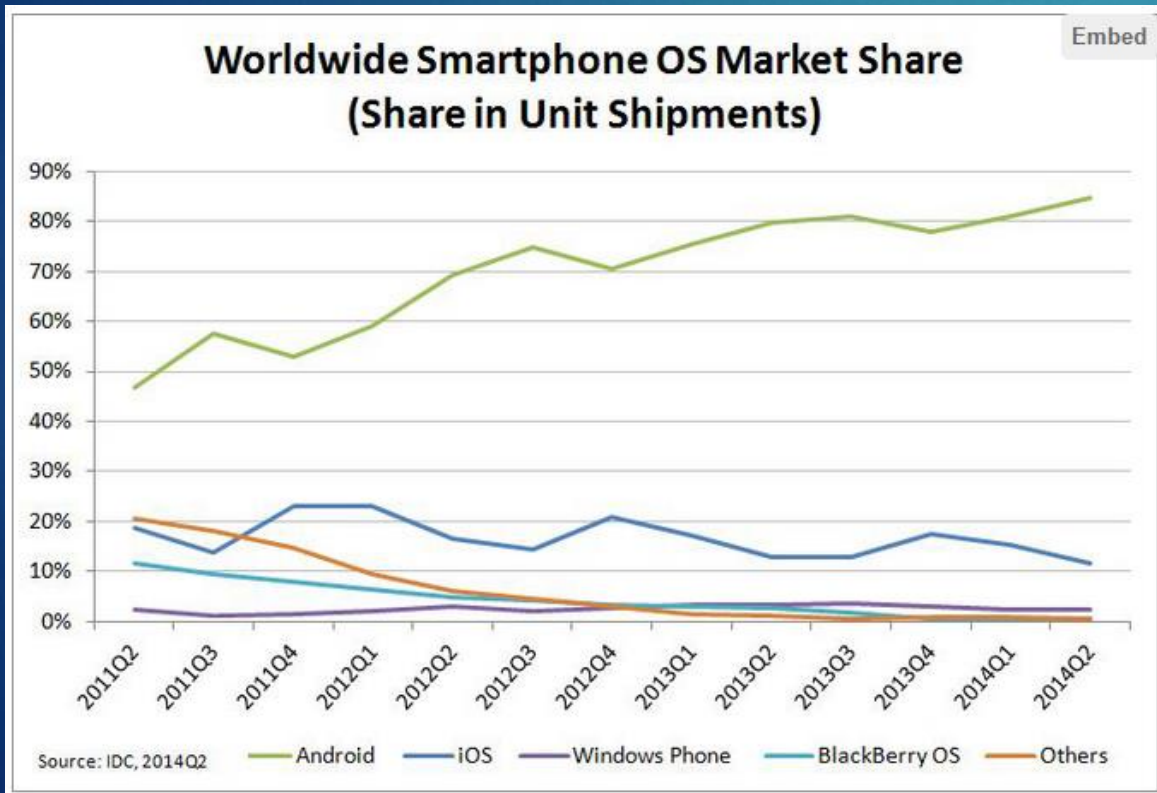
INTRODUCTION

Concurrents

- ▶ Apple iPhone OS : un des leaders en téléphonie, fermé...
- ▶ Windows Phone : stagne, fermé...
- ▶ Blackberry : en chute
- ▶ Symbian : passage en open source octobre 2009

INTRODUCTION

► PART DE MARCHE DES OS MOBILES



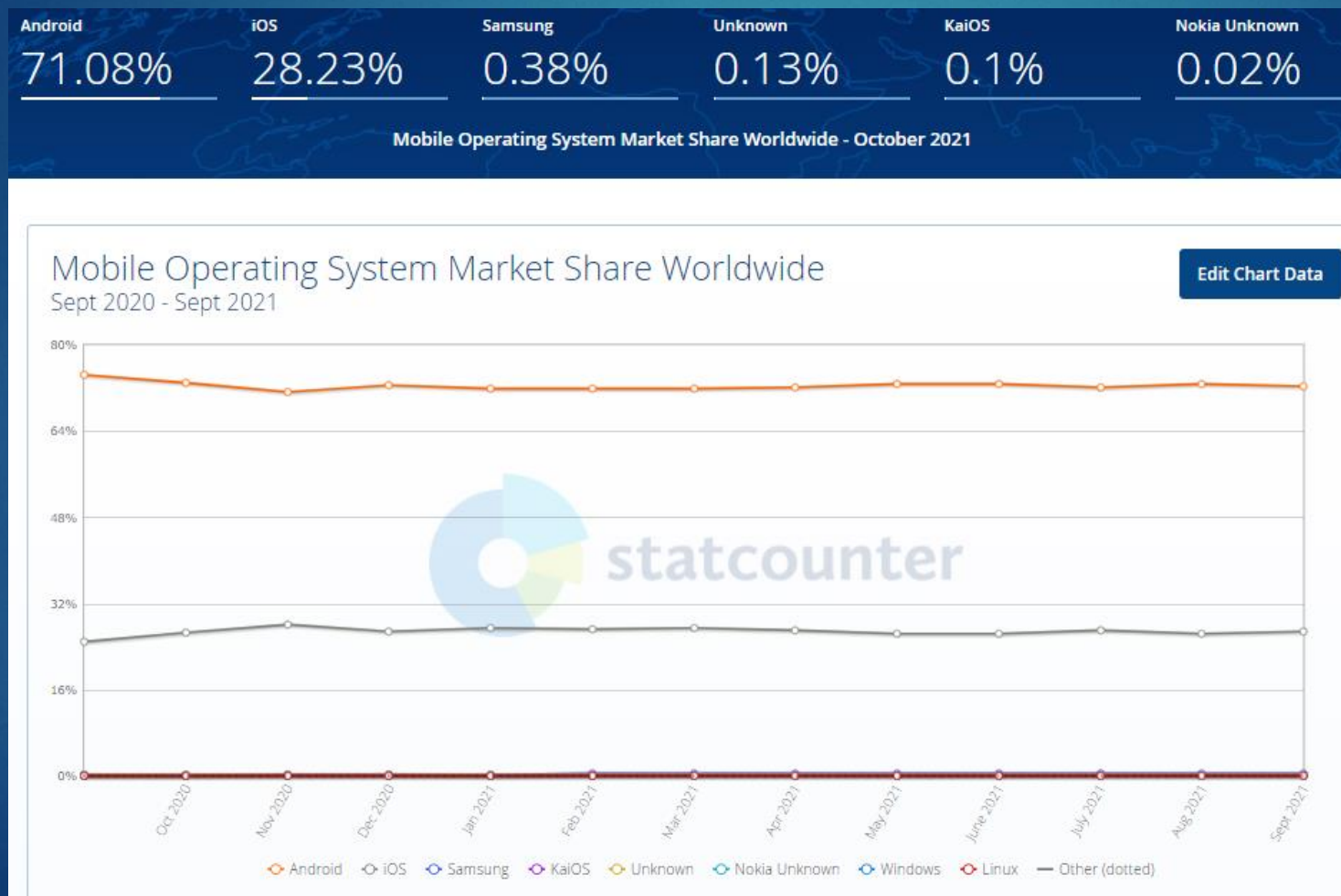
INTRODUCTION

► PART DE MARCHE DES OS MOBILES

Period	Android	iOS	Windows Phone	Others
2015Q3	84.3%	13.4%	1.8%	0.5%
2015Q4	79.6%	18.6%	1.2%	0.5%
2016Q1	83.4%	15.4%	0.8%	0.4%
2016Q2	87.6%	11.7%	0.4%	0.3%

INTRODUCTION

► PART DE MARCHE DES OS MOBILES



GENERALITES

► VERSIONS D'ANDROID

ANDROID VERSIONS LIST: A COMPLETE HISTORY & FEATURES



Cupcake
1.5



Donut
1.6



Eclair
2.0/2.1



Froyo
2.2



Gingerbread
2.3



Honeycomb
3.0/3.1



Ice Cream Sandwich
4.0



Jelly Bean
4.1/4.2/4.3



KitKat
4.4



Lollipop
5.0



Marshmallow
6.0



Nougat
7.0



Oreo
8.0



Pie
9.0



android

GENERALITES

► VERSIONS D'ANDROID



GENERALITES

- ▶ Dernière version d'Android : Android 12 (API 31)
- ▶ Date sortie : 4 octobre 2021



GENERALITES

▶ PUBLICATION DES APPLICATIONS

- ▶ Google Play
- ▶ Marchés alternatifs
 - ▶ GetJar
 - ▶ SlideMe
 - ▶ Amazon App store for Android
 - ▶ F-Droid
 - ▶ AppsLib
 - ▶ AndroidPIT App Center

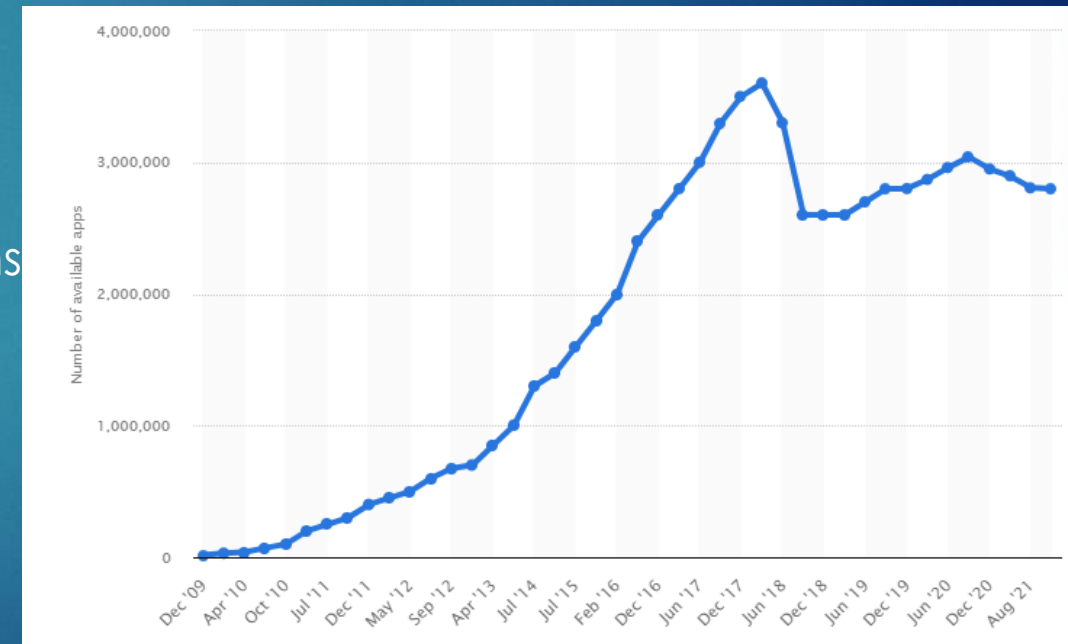
GENERALITES

- ▶ GOOGLE PLAY [PLAY STORE]
 - ▶ Plateforme pour distribuer, vendre et faire la publicité de son application
 - ▶ Il existe plusieurs Android Markets
 - ▶ Google Play est le plus grand et le plus visité

GENERALITES

► GOOGLE PLAY [PLAY STORE]

- Créé le 22 octobre 2008 sous le nom Android Market
- Rebaptisé Play Store le 6 mars 2012
- Mars 2009, 2 300 applications
- Août 2013, plus de 1 300 000 d'applications
- Janvier 2020, plus de 3 000 000 d'applications



GENERALITES

- ▶ PUBLIER DES APPLICATIONS SUR PLAY STORE
 - ▶ Créer un compte développeur : <https://play.google.com/apps/publish> .
 - ▶ 25 \$ USD de frais à payer une seule fois
 - ▶ 70% au développeur 30% à Google
 - ▶ Toutes les applications doivent disposer d'une clé cryptographique
 - ▶ Taille maximale du fichier APK : 50 MB
 - ▶ Barre des 50 milliards de téléchargement franchie

GENERALITES

► GOOGLE PLAY

Pays ^	Inscription des développeurs possible	Inscription des marchands possible	Devise par défaut des développeurs
Afrique du Sud	✓	✗	☆
Albanie	✓	✗	☆
Algérie	✓	✗	☆
Allemagne	✓	✓	EUR
Angola	✓	✗	☆
Antigua-et-Barbuda	✓	✗	☆
Antilles néerlandaises	✓	✗	☆
Arabie saoudite	✓	✓	SAR
Argentine	✓	✓	USD
Arménie	✓	✗	☆
Aruba	✓	✗	☆
Australie	✓	✓	AUD
Autriche	✓	✓	EUR

GENERALITES

► GOOGLE PLAY

Cité du Vatican	✓	✗	☆
Colombie	✓	✓	COP
Corée du Sud	✓	✓	KRW
Costa Rica	✓	✓	CRC
Croatie	✓	✗	☆
Côte d'Ivoire	✓	✗	☆
Danemark	✓	✓	DKK
Espagne	✓	✓	EUR
Estonie	✓	✗	☆
Fidji	✓	✗	☆

GENERALITES

► GOOGLE PLAY – ACHAT D'APPLICATIONS

Pays ^	Télécharger des applications gratuites	Télécharger des applications payantes	Gamme de prix et devise des acheteurs
Afrique du Sud	✓	✓	10 – 2 000 ZAR
Albanie	✓	✓	★
Algérie	✓	✓	★
Allemagne	✓	✓	0,50 – 199 EUR
Angola	✓	✓	★
Antigua-et-Barbuda	✓	✓	★
Antilles néerlandaises	✓	✓	★
Arabie saoudite	✓	✓	4 – 750 SAR
Argentine	✓	✓	★
Arménie	✓	✓	★
Aruba	✓	✓	★
Australie	✓	✓	0,99 – 200 AUD

GENERALITES

► GOOGLE PLAY – ACHAT D'APPLICATIONS

Chypre	✓	✓	0,50 – 199 EUR
Colombie	✓	✓	2 000 – 390 000 COP
Corée du Sud	✓	✓	999 – 220 000 KRW
Costa Rica	✓	✓	500 – 100 000 CRC
Croatie	✓	✓	★
Côte d'Ivoire	✓	✓	★
Danemark	✓	✓	6 – 1 200 DKK
Espagne	✓	✓	0,50 – 199 EUR
Estonie	✓	✓	0,50 – 199 EUR
Fidji	✓	✓	★
Finlande	✓	✓	0,50 – 199 EUR

ENVIRONNEMENT DE DEVELOPPEMENT

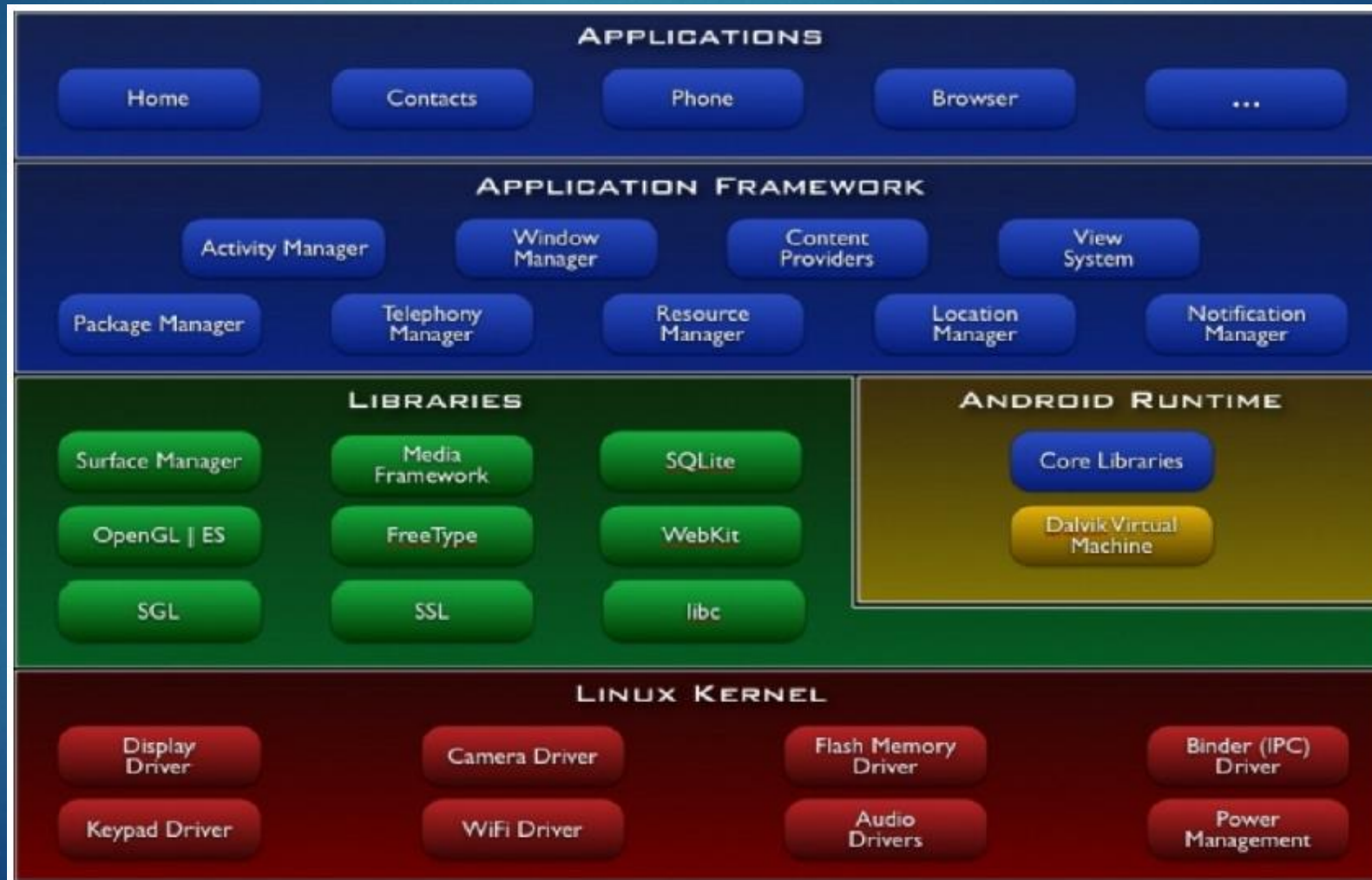
▶ OUTILS

- ▶ Eclipse
- ▶ SDK Android
- ▶ ADT : Android Development Tools (plugin eclipse)
- ▶ AVD : Android Virtual Device
- ▶ ADB : Android Debug Bridge (ligne de commande)
- ▶ Android Studio



ARCHITECTURE ANDROID

► ARCHITECTURE GENERALE



ARCHITECTURE ANDROID

- ▶ 5 couches
 - ▶ Noyau
 - ▶ Bibliothèques natives (Librairies)
 - ▶ Runtime
 - ▶ Framework
 - ▶ Application

ARCHITECTURE ANDROID

LE NOYAU



ARCHITECTURE ANDROID

▶ LE NOYAU

- ▶ Android repose sur un noyau Linux version 2.6
- ▶ Gestion de la sécurité
- ▶ Gestion de la mémoire
- ▶ Gestion des processus
- ▶ Gestion réseau
- ▶ Drivers
- ▶ ...
- ▶ Ce noyau agit comme une couche d'abstraction entre le matériel et le restes des couches applicatives.

ARCHITECTURE ANDROID

▶ LE NOYAU

- ▶ Noyau Linux 2.6 (mais modifié)
- ▶ Choisi pour sa stabilité, sa maturité et l'ouverture du code
- ▶ Principal Changement : Suppression des IPC SysV remplacé par Binder
 - ▶ Binder proche de CORBA.
 - ▶ Economique en ressource dédié aux architectures qui ne reposent pas activement sur la gestion de processus.
- ▶ Gestion de la mémoire différente. SHM POSIX mais simplifié.
- ▶ Partage de mémoire entre processus via Binder
- ▶ Système embarqué oblige l'accès aux journaux ne peut pas se faire via `/var/log/*`
- ▶ Intégration d'un logger
- ▶
- ▶ En standard pas de fonction pour terminer l'application
- ▶ Viking Killer (Out Of Memory Management)

ARCHITECTURE ANDROID

► LIBRARIES

- Elles fournissent un accès direct aux ressources du système
- Une couche d'abstraction au Framework Java Android



ARCHITECTURE ANDROID

► LIBRARIES

- Android inclus un ensemble de librairies C/C++
- Utilisées par les applications Android
- Accessibles au développeur via le SDK
- Quelques-unes de ces librairies
 - Librairie Système C : une implémentation dérivée de l'implémentation BSC des librairies standard C (libc)
 - LibWebCore : Un moteur de navigateur internet moderne utilisé autant pour navigateur Android que pour les vues web intégrables
 - SQLite : un système de gestion de base de données relationnel léger et puissant disponible pour toutes les applications.



ARCHITECTURE ANDROID

► ANDROID RUNTIME



ARCHITECTURE ANDROID

- ▶ ANDROID RUNTIME

- ▶ ART : Android Runtime Depuis Android 5,0 Lollipop
- ▶ Compilation AOT (Ahead-Of-Time)
- ▶ Compile une SEULE FOIS, AOT
- ▶ Compilation native
- ▶ Utilise moins de Threads
- ▶ Garbage Collector non-bloquant
- ▶ Adopte l'architecture 64 bits

ARCHITECTURE ANDROID

- ▶ ANDROID RUNTIME : COMPILATION
 - ▶ Deux passages :
 - ▶ .JAVA vers .CLASS
 - ▶ Concaténation des .CLASS en .DEX
 - ▶ Une application c'est :
 - ▶ Le bytecode DEX
 - ▶ des ressources (images, sons...)
 - ▶ Le tout regroupe dans un package .APK

ARCHITECTURE ANDROID

- ▶ ANDROID RUNTIME
- ▶ Android inclus un ensemble de librairies de base proposant ainsi la quasi-totalité des fonctionnalités disponibles dans le langage de programmation Java.
- ▶ Chaque application sous Android utilise sa propre instance d'une ART.
 - ▶ - Pas de problème d'interaction entre les applications
 - ▶ - Espace protégé
 - ▶ - Pas de risque de plantage général
 - ▶ - D'où la nécessité d'une VM optimisée !

ARCHITECTURE ANDROID

► FRAMEWORK APPLICATIF



ARCHITECTURE ANDROID

- ▶ FRAMEWORK APPLICATIF

- ▶ Framework écrit en Java.
- ▶ Fournit tout ce dont les applications ont besoin.
- ▶ API du Framework décrite dans la documentation du SDK



ARCHITECTURE ANDROID

▶ FRAMEWORK APPLICATIF

▶ Éléments du Framework :

- ▶ Activity Manager : cycle de vie des applications (backstack).
- ▶ Assure le multi tâche
- ▶ Package Manager : Manipulation du format .apk
- ▶ Window Manager : utilise Surface Manager.
- ▶ Resource Manager : Tout ce qui n'est pas du code.



ARCHITECTURE ANDROID

▶ FRAMEWORK APPLICATIF

- ▶ Content Manager : Partage des données entre processus
- ▶ View System : equivalent d'un toolkit GTK+. Gère le rendu HTML
- ▶ Telephony Service : fournit l'accès aux services GSM, 3G,GPRS
- ▶ Location Service : fournit l'accès à la gestion du GPS.
- ▶ Bluetooth Service
- ▶ Wi-Fi Service
- ▶ Sensor Service



ARCHITECTURE ANDROID

▶ FRAMEWORK APPLICATIF

- ▶ Plateforme de développement Ouverte
 - ▶ - Permet des applications riches et variées
 - ▶ - Accès au matériel
 - ▶ - Accès aux informations de localisation
 - ▶ - Lancement de services de fond
 - ▶ - Mise en place d'alarmes, de notifications
 - ▶ - .
- ▶ Architecture conçue pour simplifier la réutilisation des composants
- ▶ Publication des capacités des applications
- ▶ Les autres applications peuvent utiliser ces capacités
- ▶ Permet de charger facilement les apps.

ARCHITECTURE ANDROID

► FRAMEWORK APPLICATIF

- Une application est composée d'un ensemble de services et de systèmes incluant :
- Un ensemble de vues "Views" utilisées pour construire l'application (listes, grilles, zone de saisies, boutons ou encore navigateur web intégrable)
- "Content Provider" permettant aux applications d'accéder aux données d'autres applications (Contacts...) ou de partager leurs propres données.

ARCHITECTURE ANDROID

► FRAMEWORK APPLICATIF

- "Resource Manager" permettant d'accéder à des ressources tel que des chaînes de caractères, des images ou des "layout" (le tout paramétrable selon de multiples critères : taille de l'écran, internationalisation...)
-
- Mais aussi :
- "Notification Manager" permettant à chaque application d'utiliser la barre de statut générale pour y intégrer ses propres informations.
- "Activity Manager" : composant qui gère le cycle de vie d'une application et fournit les outils de navigation applicative.

ARCHITECTURE ANDROID

► LES APPLICATIONS



ARCHITECTURE ANDROID

▶ LES APPLICATIONS

▶ 2 parties :

- ▶ Les activités : des fenêtres interactives
- ▶ Les services : tâches de fond.

▶ Les applications tournent dans leurs SandBoxes

▶ Communications entre applications : Les "intent"

▶ Intent = intention : formule une demande

▶ Plusieurs composants peuvent répondre à un "intent" .

ARCHITECTURE ANDROID

▶ LES APPLICATIONS

- ▶ Dernière couche sur Android
- ▶ Plusieurs sont intégrées dans le système :
- ▶ Ecran "Home"
- ▶ Gestion des Emails
- ▶ Gestion des SMS/MMS
- ▶ Gestion de la téléphonie
- ▶ Google Maps...
- ▶ Application supplémentaires installables
- ▶ Toutes les applications sont écrites via le même SDK !

DEVELOPPEMENT D'APPLICATIONS ANDROID

► GENERALITES

- Les applications sont écrites en Java
- Le code compilé "dex" ainsi que les ressources (images, layout...) sont regroupés dans une archive au format "apk" par les outils du SDK
- Cette archive "apk" est un tout permettant la distribution et l'installation de l'application sur n'importe quelle plateforme Android.

DEVELOPPEMENT D'APPLICATIONS ANDROID

► GENERALITES

- Les applications sont écrites en Java
- Le code compilé "dex" ainsi que les ressources (images, layout...) sont regroupés dans une archive au format "apk" par les outils du SDK
- Cette archive "apk" est un tout permettant la distribution et l'installation de l'application sur n'importe quelle plateforme Android.

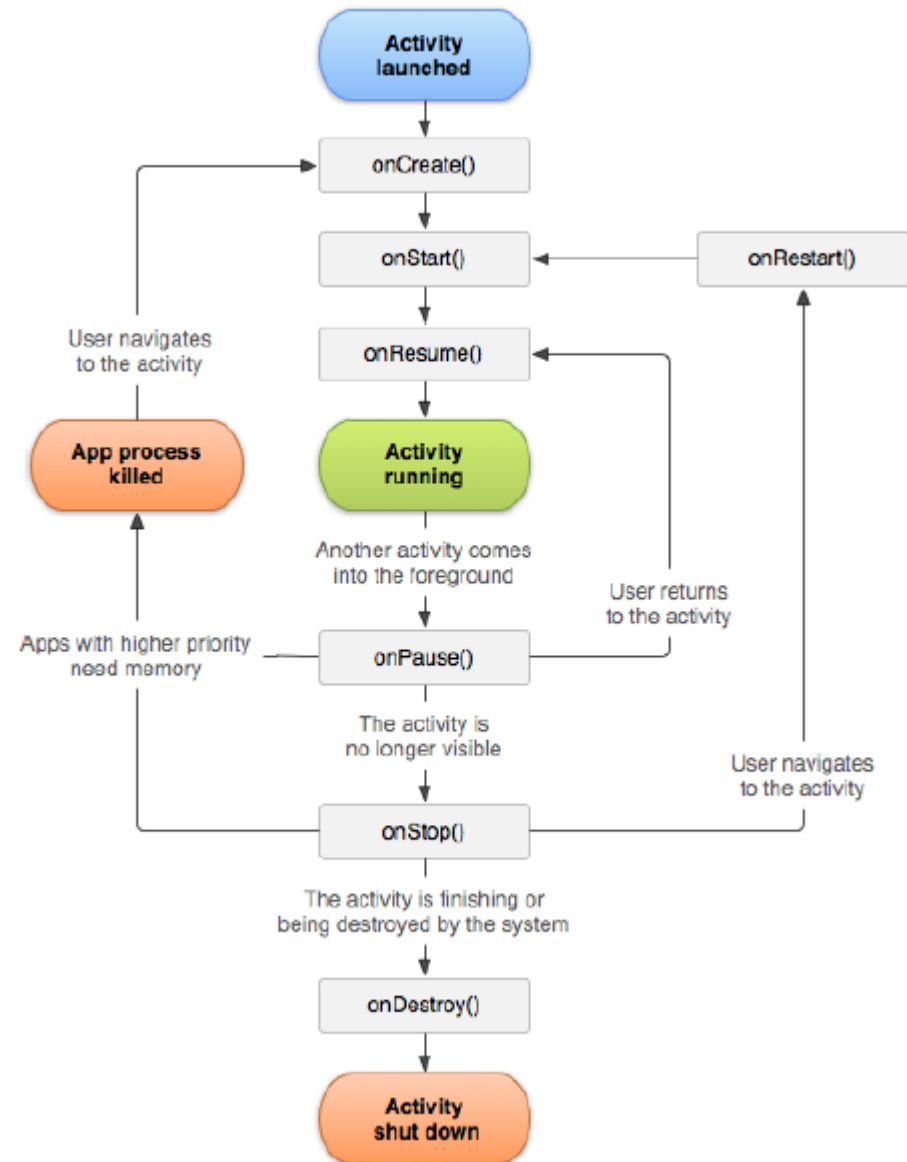
DEVELOPPEMENT D'APPLICATIONS ANDROID

▶ ELEMENTS FONDAMENTAUX

- ▶ Activity
- ▶ Service
- ▶ BroadcastReceiver
- ▶ ContentProvider
- ▶ Intent

DEVELOPPEMENT D'APPLICATIONS ANDROID

► CYCLE DE VIE

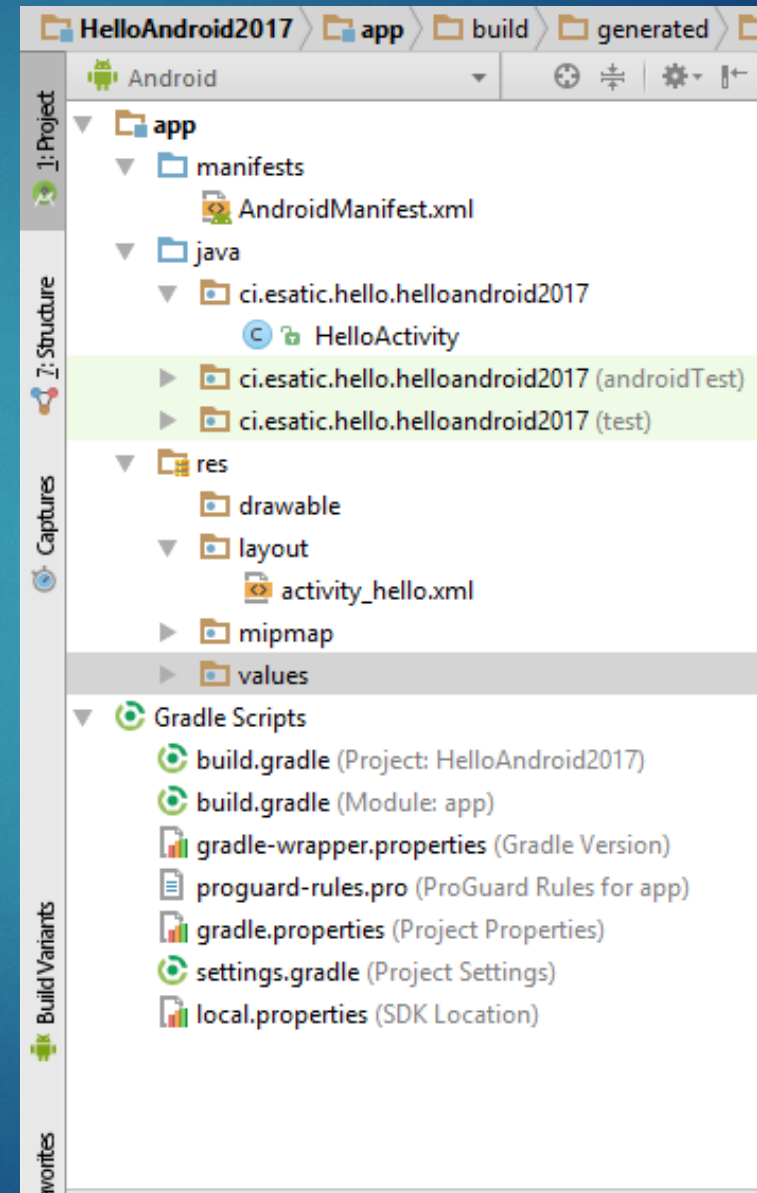


DEVELOPPEMENT D'APPLICATIONS ANDROID

- ▶ VOTRE PREMIERE APPLICATION : HELLO ANDROID !!!!

STRUCTURE DUN PROGRAMME ANDROID

- ▶ ACTIVITE
- ▶ CLASSE R
- ▶ LES RESSOURCES
- ▶ LAYOUT
- ▶ LE MANIFEST



LES ACTIVITES(HelloActivity.java)

```
package ci.esatic.hello.helloandroid2017;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class HelloActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hello);
    }
}
```

INTERFACE UTILISATEUR (activity_hello.xml)

```
<?xml version="1.0" encoding="utf-8" ?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_hello"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="ci.esatic.hello.helloandroid2017.HelloActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
</RelativeLayout>
```

LE MANIFEST (AndroidManifest.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ci.esatic.hello.helloandroid2017">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".HelloActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

LE MANIFEST (Permissions)

<manifest ... >

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.SEND_SMS" />

<uses-permission android:name="android.permission.RECEIVE_SMS" />

<uses-permission android:name="android.permission.CALL_PHONE" />

<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

</manifest>

GESTION DES EVENEMENTS

- ▶ Interface **OnClickListener**
- ▶ L'activité doit implémenter la méthode `onClick` de l'interface `OnClickListener` afin de gérer les évènements
 - ▶ `Import android.view.View.OnClickListener`
 - ▶ `public class MainActivity implements OnClickListener{`
 - ▶ `b1 = (button)findViewById(R.id.btn1);`
 - ▶ `b1.setOnClickListener(this)`
 - ▶ `public void onClick(view v){`
 - ▶ `}`
 - ▶ `}`

SPLASH SCREEN

```
import android.os.Bundle;
import android.os.Handler;
import android.view.Menu;
import android.view.MenuItem;

public class SplashActivity extends ActionBarActivity {
    private static int SPLASH_TIME_OUT = 3000;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

SPLASH SCREEN (suite)

```
new Handler().postDelayed(new Runnable() {  
  
    @Override  
    public void run() {  
        Intent intent = new Intent(getApplicationContext(),  
MenuActivity.class);  
  
        startActivity(intent);  
  
        finish();  
  
    }  
}, SPLASH_TIME_OUT);  
}
```

LES INTENTS

- ▶ NAVIGATION ENTRE PLUSIEURS ACTIVITES
- ▶ APPEL D'ACTIONS SYSTEMES
 - ▶ LANCEMENT PAGE WEB (ACTION_VIEW)
 - ▶ APPeler UN NUMERO (ACTION_CALL)
 - ▶ OUVRIR LISTE CONTACT
 - ▶ LOCALISER UN POINT A L'AIDE DE GOOGLE MAP

MULTIMEDIA

- ▶ JOUER DE L'AUDIO
- ▶ JOUER DE LA VIDEO

MULTIMEDIA - AUDIO

- ▶ Créer un dossier `raw` dans le répertoire `res` qui contiendra les fichiers audios, vidéos, ...
- ▶ Classe `MediaPlayer` (pour jouer l'audio)
 - ▶ `import android.media.MediaPlayer;`
- ▶ Classe `AudioManager` (pour le contrôle de volume)
 - ▶ `import android.media.AudioManager`

MULTIMEDIA - AUDIO

- ▶ ...
- ▶ `Private MediaPlayer mp;`
- ▶ ...
- ▶ `setContentView(R.layout.main);`
- ▶ `setVolumeControlStream(AudioManager.STREAM_MUSIC);`

MULTIMEDIA - AUDIO

```
// Libérer les ressources d'un MediaPlayer antérieur
if (mp != null) {
    mp.release();
}
// Créer un nouveau MediaPlayer pour jouer le son
mp = MediaPlayer.create(this, resId);
mp.start();
```

MULTIMEDIA - VIDEO

En plus de la classe MediaPlayer :
VideoView : Pour afficher la video

Dans le Layout :

- ▶ <VideoView
- ▶ android:id="@+id/video"
- ▶ android:layout_width="wrap_content"
- ▶ android:layout_height="wrap_content"
- ▶ android:gravity="center" />

MULTIMEDIA - VIDEO

Dans l'Activité :

```
Import android.widget.videoView;  
Import android.net.Uri;  
...  
setContentView(R.layout.main);  
VideoView video = (VideoView)findViewById(R.id.video);  
String uriPath = "android.resource://com.android.Video/" + R.raw.small;  
    Uri uri = Uri.parse(uriPath);  
    video.setVideoURI(uri);  
    video.requestFocus();  
video.start();
```


MULTIMEDIA - VIDEO

Dans l'Activité :

```
Import android.widget.videoView;  
Import android.net.Uri;  
...  
setContentView(R.layout.main);  
VideoView video = (VideoView)findViewById(R.id.video);  
String uriPath = "android.resource://com.android.Video/" + R.raw.small;  
    Uri uri = Uri.parse(uriPath);  
    video.setVideoURI(uri);  
    video.requestFocus();  
video.start();
```

BASE DE DONNEES

- ▶ SQLite : Minuscule, mais puissante base de données
- ▶ Créée en 2000 par Dr. Richard Hipp.
- ▶ Peut être la base de données SQL la plus déployée dans le monde.
- ▶ Se retrouve dans :
 - ▶ Android, Apple iPhone, Symbian phones, Mozilla Firefox, Skype, PHP, Adobe AIR, MacOS X, Solaris ...
- ▶ Gratuit
- ▶ Très petit (environ 150 KB)
- ▶ Pas d'administration, pas de configuration, pas de serveur

BASE DE DONNEES

- ▶ La Base de données SQLite est juste un fichier
- ▶ Android stocke ce fichier dans le dossier `/data/data/nompackage/databases`
- ▶ Utiliser le File Explorer de Eclipse pour le voir le fichier
- ▶ Android utilise des commandes SQL pour accéder aux fichiers
- ▶ Un Helper Class est utilisé pour masquer la complexité

BASE DE DONNEES

- ▶ Interface **BaseColumns**

- ▶ Contient la constante **_id** qui est la clé primaire d'une table

- ▶ Classe **SQLiteOpenHelper**

- ▶ Cette classe contient les routines de création d'une base de données et la gestion des versions de la BD

BASE DE DONNEES

► APPLICATION -Events

- Application de gestion d'évènements
- Stocke les différents évènements dans une DB pour les restituer plus tard
- Project name: Events
- Build Target: Android 2.2
- Application name: Events
- Package name: org.example.events
- Create Activity: Events
- Min SDK Version: 8

BASE DE DONNEES

- Pour stocker les constantes décrivant la base de données nous allons d'abord créer une **interface Constants**
- **Constants.java**

```
package org.example.events;  
import android.provider.BaseColumns;  
public interface Constants extends BaseColumns {  
    public static final String TABLE_NAME = "events" ;  
    // Colonnes dans la BD Events  
    public static final String TIME = "time" ;  
    public static final String TITLE = "title" ;  
}
```

BASE DE DONNEES

- ▶ Chaque évènement sera stocké comme une ligne dans la table `events`
- ▶ Chaque ligne aura une colonne `_id`, `time` et `title`.
- ▶ `_id` représente la clé primaire, déclarée dans `BaseColumns` dont nous nous avons héritée.

BASE DE DONNEES

- ▶ CREATION DE LA BASE DE DONNEES
- ▶ Nous allons créer une helper class appelée `EventsData` qui représente la base de donnée elle même.
- ▶ `EventsData` hérite de `SQLiteOpenHelper`
- ▶ Ecrire juste le constructeur et implémenter les deux méthodes

BASE DE DONNEES

► EventsData.Java

```
package org.example.events;

import static android.provider.BaseColumns._ID;
import static org.example.events.Constants.TABLE_NAME;
import static org.example.events.Constants.TIME;
import static org.example.events.Constants.TITLE;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
```

BASE DE DONNEES

► EventsData.Java (suite)

;

```
public class EventsData extends SQLiteOpenHelper {  
    private static final String DATABASE_NAME = "events.db" ;  
    private static final int DATABASE_VERSION = 1;
```

```
    /** Create a helper object for the Events database */  
    public EventsData(Context ctx) {  
        super(ctx, DATABASE_NAME, null, DATABASE_VERSION);  
    }
```


BASE DE DONNEES

► EventsData.Java (suite)

@Override

```
public void onCreate(SQLiteDatabase db) {  
    db.execSQL("CREATE TABLE " + TABLE_NAME + " (" + _ID  
    + " INTEGER PRIMARY KEY AUTOINCREMENT, " + TIME  
    + " INTEGER," + TITLE + " TEXT NOT NULL);" );  
}
```

BASE DE DONNEES

► EventsData.Java (suite)

@Override

```
public void onUpgrade(SQLiteDatabase db, int oldVersion,  
int newVersion) {  
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);  
    onCreate(db);  
}  
}
```

BASE DE DONNEES

► Programme principal

Nous allons stocker les événements dans une base de données locale et il les rendra sous forme de chaîne de caractères dans un TextView

BASE DE DONNEES

► Main.xml

Nous allons stocker les événements dans une base de données locale et il les rendra sous forme de chaîne de caractères dans un TextView

BASE DE DONNEES

► Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent" >
<TextView
android:id="@+id/text"
android:layout_width="fill_parent"
android:layout_height="wrap_content" />
</ScrollView>
```


BASE DE DONNEES

► Events.java

```
package org.example.events;

import static android.provider.BaseColumns._ID;
import static org.example.events.Constants.TABLE_NAME;
import static org.example.events.Constants.TIME;
import static org.example.events.Constants.TITLE;
import android.app.Activity;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.widget.TextView;
```

BASE DE DONNEES

► Events.java (suite)

```
public class Events extends Activity {  
    private EventsData events;
```

```
@Override
```

```
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        events = new EventsData(this);
```

BASE DE DONNEES

► Events.java (suite)

```
try {  
    addEvent("Hello, Android!" );  
    Cursor cursor = getEvents();  
    showEvents(cursor);  
}  
finally {  
    events.close();  
}  
}
```

BASE DE DONNEES

- **Events.java (suite)** [Ajout d'une ligne – méthode addEvent()]

```
private void addEvent(String string) {  
    // Insère un nouvel enregistrement dans la source de données Events  
  
    SQLiteDatabase db = events.getWritableDatabase();  
    ContentValues values = new ContentValues();  
    values.put(TIME, System.currentTimeMillis());  
    values.put(TITLE, string);  
    db.insertOrThrow(TABLE_NAME, null, values);  
}
```

BASE DE DONNEES

- **Events.java (suite)** [Exécuter une requête– méthode getEvents()]

```
private static String[] FROM = { _ID, TIME, TITLE, };  
private static String ORDER_BY = TIME + " DESC" ;  
private Cursor getEvents() {  
  
    SQLiteDatabase db = events.getReadableDatabase();  
    Cursor cursor = db.query(TABLE_NAME, FROM, null, null, null,  
        null, ORDER_BY);  
    startManagingCursor(cursor);  
    return cursor;  
}
```


BASE DE DONNEES

- **Events.java (suite)** [Afficher les résultats– méthode showEvents()]

```
private void showEvents(Cursor cursor) {  
    // On les met tous dans une grosse chaine de caractères  
    StringBuilder builder = new StringBuilder(  
        "Evènements enregistrés:\n" );  
    while (cursor.moveToNext()) {  
        //On peut utiliser getColumnIndexOrThrow() pour récupérer les indexes  
        long id = cursor.getLong(0);  
        long time = cursor.getLong(1);  
        String title = cursor.getString(2);
```

BASE DE DONNEES

- `Events.java (suite)` [Afficher les résultats– méthode `showEvents()`]

```
builder.append(id).append(": " );  
builder.append(time).append(": " );  
builder.append(title).append("\n" );  
}  
//Afficher à l'écran  
TextView text = (TextView) findViewById(R.id.text);  
text.setText(builder);  
}
```

GEOLOCALISATION



- ▶ Le système Global Positioning System (GPS) a été développé pour un usage militaire par les américains et converti à un usage civil.
- ▶ Le GPS envoie un faisceau de signaux très précis à des récepteurs sur la terre tel que les téléphones Android.
- ▶ La puce GPS peut déterminer votre position à 15 mètres près !
- ▶ À part le GPS, Android utilise aussi les informations des pylônes pour calculer votre position ainsi que les Hotspots Wifi.

GEOLOCALISATION

LES PERMISSIONS

- Pour avoir accès aux informations de localisation, il faut indiquer les permissions dans AndroidManifest.xml

- Avant le tag <Application> mettre :

```
<uses-permission
```

```
android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

```
<uses-permission
```

```
android:name="android.permission.ACCESS_FINE_LOCATION" />
```

- ACCESS_FINE_LOCATION  GPS
- ACCESS_COARSE_LOCATION  Pylônes

GEOLOCALISATION



LES CLASSES UTILISEES

- ▶ `import android.location.Location;`
- ▶ `import android.location.LocationListener;`
- ▶ `import android.location.LocationManager;`
- ▶ `import android.location.LocationProvider;`

GEOLOCALISATION



Méthode getSystemService()

- ▶ Le point de départ des services de géolocalisation est la méthode getSystemService()
- ▶ getSystemService() est appelé avec l'argument **LOCATION_SERVICE**
- ▶ Il retourne une classe **LocationManager** qui permet de gérer la géolocalisation

GEOLOCALISATION

Mise à jour localisation

- ▶ Pour mettre à jour les information de localisation, on appelle la méthode `requestLocationUpdates()` de l'objet instance de `LocationManager`
- ▶ `requestLocationUpdates()` utilise 4 paramètres :
 - Nom du provider
 - le délai pour limiter les MAJ trop fréquentes
 - une distance minimale (changement inférieures ignorés)
 - Objet `LocationListener`

GEOLOCALISATION



Interface `LocationListener` (4 méthodes)

- ▶ `public void onLocationChanged(Location location)`
- ▶ `public void onProviderDisabled(String provider)`
- ▶ `public void onProviderEnabled(String provider)`
- ▶ `public void onStatusChanged(String provider, int status, Bundle extras)`

GEOLOCALISATION



Interface `LocationListener` (4 méthodes)

- ▶ Méthode `onLocationChanged()`
- ▶ C'est la méthode la plus importante
- ▶ Appelée chaque fois que le provider remarque un changement de position de l'appareil

GEOLOCALISATION




Interface `LocationListener` (4 méthodes)

- ▶ Méthodes `onProviderDisabled()`, `onProviderEnabled()`, et `onStatusChanged()`
- ▶ Peuvent être utilisées pour changer de provider au cas où le premier choix devient indisponible.

GEOLOCALISATION

EXAMPLE : Application LocationTest



The screenshot shows an Android application interface with a title bar 'LocationTest'. The status bar at the top displays icons for 3G, signal strength, battery, and the time 6:24 PM. The main content area displays the following text:

```
Location providers:  
LocationProvider[name=gps,enabled=true,  
getAccuracy=fine,getPowerRequirement=high,  
hasMonetaryCost=false,requiresCell=false,  
requiresNetwork=false,requiresSatellite=true,  
supportsAltitude=true,supportsBearing=true,  
supportsSpeed=true]  
  
Best provider is: gps  
  
Locations (starting with last known):  
  
Location[unknown]  
  
Provider status changed: gps, status=temporarily  
unavailable, extras=Bundle[mParcelledData.  
dataSize=52]  
  
Location[mProvider=gps,  
mTime=1247973839000,mLatitude=37.422006,m  
Longitude=-122.084095,mHasAltitude=true,  
mAltitude=0.0,mHasSpeed=false,  
mSpeed=0.0,mHasBearing=false,  
mBearing=0.0,mHasAccuracy=false,  
mAccuracy=0.0,mExtras=Bundle[mParcelledData.  
dataSize=52]]
```

GEOLOCALISATION



EXAMPLE : Application LocationTest

Project name: LocationTest

Build Target: Android 2.2

Application name: LocationTest

Package name: org.example.locationtest

Create Activity: LocationTest

Min SDK Version: 8

Access to location information

GEOLOCALISATION



EXEMPLE : Application LocationTest

Fichier **AndroidManifest.xml**

```
<uses-permission  
    android:name="android.permission.ACCESS_COARSE_LOCATION" />  
<uses-permission  
    android:name="android.permission.ACCESS_FINE_LOCATION" />
```

GEOLOCALISATION



EXEMPLE : Application LocationTest

Fichier `mainLayout.xml`

Nous allons imprimer toutes les information de géolocalisation dans un gros scrolling
TextView

GEOLOCALISATION

EXEMPLE : Application LocationTest

Fichier `mainLayout.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent" >
<TextView
android:id="@+id/output"
android:layout_width="match_parent"
android:layout_height="wrap_content" />
</ScrollView>
```


GEOLOCALISATION



EXEMPLE : Application LocationTest

Fichier LocationTest.java

```
package org.example.locationtest;
```

```
import java.util.List;
```

```
import android.app.Activity;
```

```
import android.location.Criteria;
```

```
import android.location.Location;
```

```
import android.location.LocationListener;
```

```
import android.location.LocationManager;
```

```
import android.location.LocationProvider;
```

```
import android.os.Bundle;
```

```
import android.widget.TextView;
```

GEOLOCALISATION



EXAMPLE : Application LocationTest

Fichier `LocationTest.java`

```
public class LocationTest extends Activity implements  
    LocationListener {  
    private LocationManager mgr;  
    private TextView output;  
    private String best;
```

GEOLOCALISATION

EXEMPLE : Application LocationTest

Fichier LocationTest.java

```
// définition de constantes
private static final String[] A = {"invalid","n/a","fine","coarse"};
private static final String[] P = {"invalid","n/a","low","medium","high"};
private static final String[] S = {"out of service","temporarily unavailable","available"};

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

GEOLOCALISATION



EXEMPLE : Application LocationTest

Fichier LocationTest.java

```
mgr = (LocationManager) getSystemService(LOCATION_SERVICE);
```

```
output = (TextView) findViewById(R.id.output);
```

```
log("Location providers:" );
```

```
dumpProviders();
```

GEOLOCALISATION



EXEMPLE : Application LocationTest

Fichier LocationTest.java

```
Criteria criteria = new Criteria();  
best = mgr.getBestProvider(criteria, true);  
log("\nBest provider is: " + best);  
  
log("\nLocations (starting with last known):" );  
Location location = mgr.getLastKnownLocation(best);  
dumpLocation(location);  
}
```


GEOLOCALISATION



EXEMPLE : Application LocationTest

Fichier `LocationTest.java`

```
private void log(String string){  
    output.append(string + "\n");  
}  
  
private void dumpProviders(){  
    List<String> providers = mgr.getAllProviders();  
for(String provider:providers)  
    {  
        dumpProvider(provider);  
    }  
}
```

GEOLOCALISATION



EXEMPLE : Application LocationTest

Fichier LocationTest.java

```
private void log(String string){  
    output.append(string + "\n");  
}  
  
private void dumpProviders(){  
    List<String> providers = mgr.getAllProviders();  
for(String provider:providers)  
    {  
        dumpProvider(provider);  
    }  
}
```

GEOLOCALISATION



EXAMPLE : Application LocationTest

Fichier `LocationTest.java`

```
private void dumpProvider(String provider){  
    LocationProvider info = mgr.getProvider(provider);  
    StringBuilder builder = new StringBuilder();  
    builder.append("\nLocationProvider [")  
        .append("name = ")  
        .append(info.getName())  
        .append(", enabled = ")
```

GEOLOCALISATION

EXAMPLE : Application LocationTest

Fichier `LocationTest.java` (suite dumpProviders)

```
.append(mgr.isProviderEnabled(provider))  
.append(", getAccuracy = ")  
.append(A[info.getAccuracy()+1])  
.append(", getPowerRequirement ")  
.append(P[info.getPowerRequirement()+1])  
.append("name = ")  
.append(",hasMonetaryCost=" )
```

GEOLOCALISATION



EXEMPLE : Application LocationTest

Fichier `LocationTest.java` (suite dumpProviders)

```
.append(info.hasMonetaryCost())  
.append(",requiresCell=" )  
.append(info.requiresCell())  
.append(",requiresNetwork=" )  
.append(info.requiresNetwork())  
.append(",requiresSatellite=" )  
.append(info.requiresSatellite())  
.append(",supportsAltitude=" )  
.append(info.supportsAltitude())  
.append(",supportsBearing=" )
```


GEOLOCALISATION



EXEMPLE : Application LocationTest

Fichier `LocationTest.java` (suite et fin dumpProviders)

```
append(info.supportsBearing())  
.append(",supportsSpeed=" )  
.append(info.supportsSpeed())  
.append("]");  
log(builder.toString());  
  
}
```

GEOLOCALISATION



EXEMPLE : Application LocationTest

Fichier `LocationTest.java`

```
private void dumpLocation(Location location){  
  
    if (location == null)  
        log("\nLocation[unknown]");  
    else{  
        log("\n" + location.toString());  
        log("\nLatitude = " + location.getLatitude());  
        log("\nLongitude = " + location.getLongitude());  
    }  
}
```

GEOLOCALISATION



EXEMPLE : Application LocationTest

Fichier `LocationTest.java`

`@Override`

```
protected void onResume(){
```

```
super.onResume();
```

```
// Début les mises à jour (la documentation recommande un délai >=60 000 ms)
```

```
mgr.requestLocationUpdates(best, 15000, 1, this);
```

```
}
```

`@Override`

```
protected void onPause(){
```

```
super.onPause();
```

```
// Arrête les mises à jour pour économiser la batterie quand l'appli est en pause
```

```
mgr.removeUpdates(this);
```

```
}
```

GEOLOCALISATION

EXEMPLE : Application LocationTest

Fichier `LocationTest.java`

```
public void onLocationChanged(Location location) {  
    dumpLocation(location);  
  
}  
  
@Override  
public void onProviderDisabled(String provider) {  
    log("\nProvider disabled :"+ provider);  
  
}
```

GEOLOCALISATION



EXEMPLE : Application LocationTest

Fichier LocationTest.java

@Override

public void onProviderEnabled(String provider) {

log("\nProvider enabled :"+ provider);

}

@Override

public void onStatusChanged(String provider, int status, Bundle extras) {

log("\nProvider status changed :"+ provider + ", status ="

+S[status] + ", extras =" + extras);

}