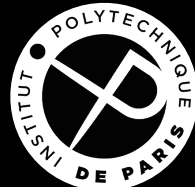


# Développement d'applications mobiles avec Android : Introduction

James EAGAN

james.eagan@telecom-paris.fr



Mise à jour : septembre 2019

1

## Who am I?

## James EAGAN

MAÎTRE DE CONFÉRENCES EN INTERACTION HOMME-MACHINE



Associate Prof. at Télécom Paris  
Adjunct Researcher at LTCI



2008 — Georgia Tech  
M.S., Ph.D. Computer Science



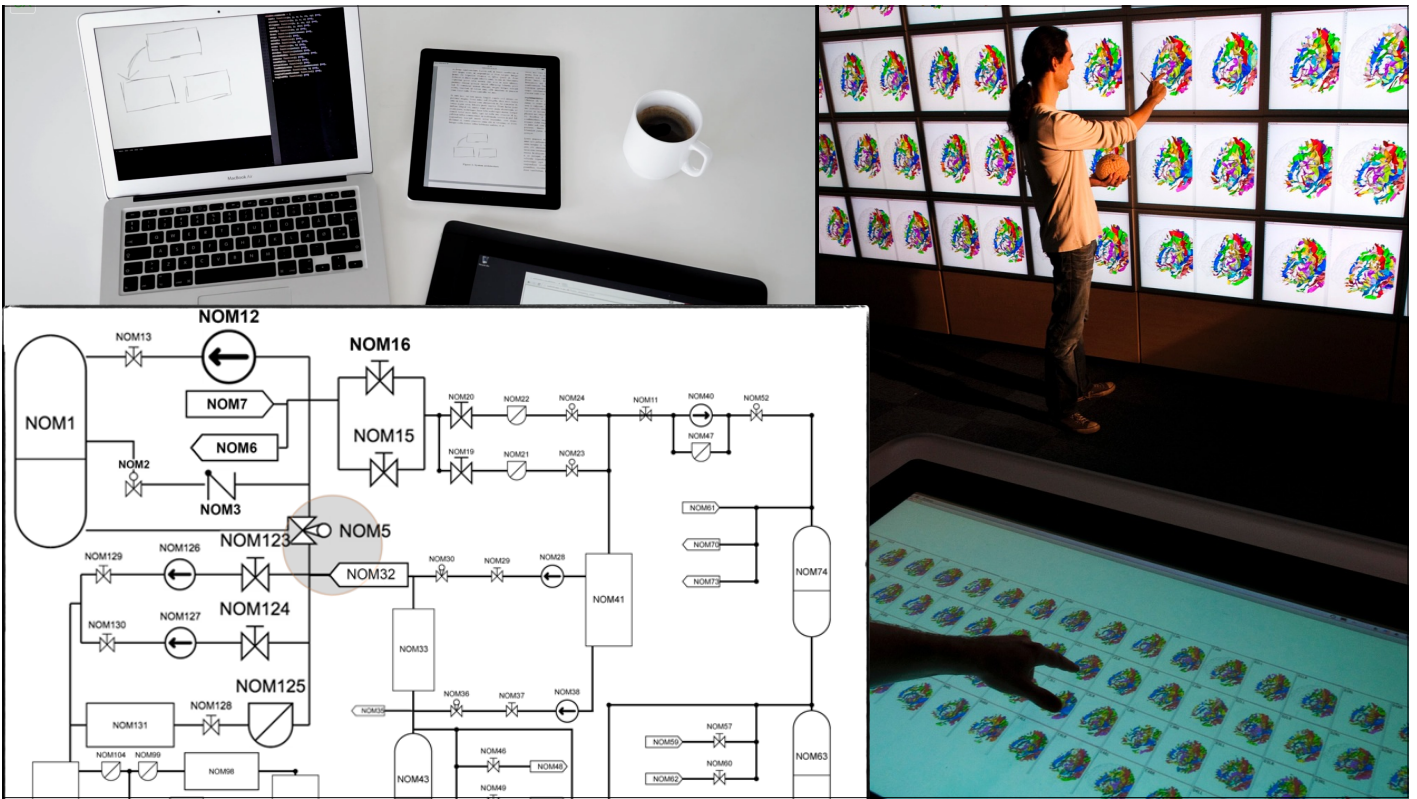
2000 — Lawrence University  
B.A. Mathematics/Computer Science

james.eagan@telecom-paris.fr



2

2



3

# Agenda

- Introduction
- Applications mobiles
- Anatomie d'une application Android
- Composants d'interface (widgets)
- Outils de développement

# Objectifs

- À la fin de ce module, vous :
  - Comprendrez des contraintes particulières aux applis mobiles
  - Pourrez créer une appli Android
  - Saurez comment intégrer une appli avec d'autres

5

5

## Qu'est-ce qui est spécial d'applis mobiles ?

- Elles sont limitées en ressources
- Elles “connaissent” leur contexte
- Elles sont omniprésentes
- Elles sont intrusives

6

6

# Attention aux ressources

- Éviter le gaspillage de ressources
- Personne ne veut :
  - Recharger son mobile plus d'une fois par jour
  - Faire banqueroute à cause de l'utilisation de données
  - Un téléphone 🐢 car une appli utilise tout le CPU
  - Une appli qui plante à cause d'une appli qui bouffe toute la mémoire
  - Remplir tout le stockage avec les données d'une appli
  - **Les ressources limitées encouragent la créativité**

7

7

# Considérations IU de base

- Une appli tourne en plein écran
- Elles doivent être réactives
- Elles doivent prendre en compte l'orientation de l'écran
- Elles tournent sur des matériels différents
- Elles n'ont pas forcément un clavier, boutons, *etc.*
- Elles peuvent être arrêtées brutalement

8

8

# Applis peuvent être omniprésentes

- Une appli mobile peut souvent :
    - Capturer des données de l'environnement (lumière, accélération, direction, GPS, ...)
    - Communiquer sur Internet
      - Pour échanger avec des services distants
      - Pour charger des ressources publiques
      - Pour faire un calcul à distance
      - Pour recevoir des données poussées vers le dispositif mobile
    - Communiquer avec d'autres dispositifs (BlueTooth, NFC, ...)
- Mais elle doit être encore utilisable face à une connexion faible ou absente**

9

9

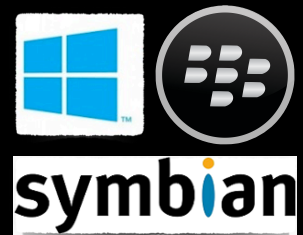
# Applis peuvent être intrusives




- Une appli peut avoir un accès :
  - aux contacts de l'utilisateur
  - aux agendas de l'utilisateur
  - aux photos & vidéos de l'utilisateur
  - à la géolocalisation du dispositif
  - au micro du dispositif
  - aux caméras du dispositif
- ... Et si tout ça était envoyé à un tiers méchant ?

10

10

# Part du marché



 74 %  
 49 %  
 76 %

25 %  
 50 %  
 22 %

1 %  
 0,3 %  
 1,6 %

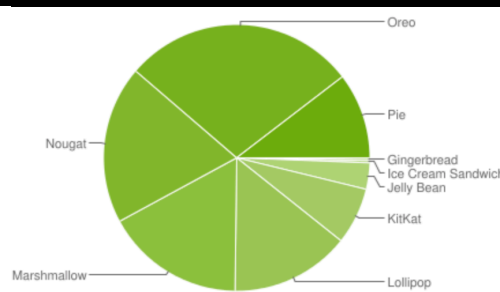
[ GlobalStats statcounter, août 2019 ]

11

11

# Quelle version Android ?

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%



Version	Name	%	Sortie
9.0	Pie	10,4	08/2018
8.0	Oreo	28,3	08/2017
7.0	Nougat	19,2	08/2016
6.0	Marshmallow	16,9	10/2015
5.0	Lollipop	14,5	11/2014
4.4	KitKat	6,9	10/2013

[ [developer.android.com](http://developer.android.com), 05/2019 ] 12

12

# Développement en Android vs. iOS

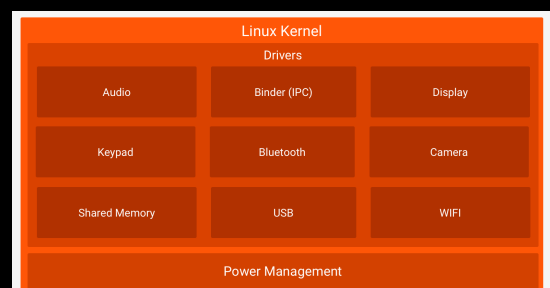
	Android	iOS
Execution	ART/Dalvik	native
Langage	Java / Kotlin / C++	Objective-C / Swift
Famille d'OS	Linux	Darwin (BSD)
App Store	\$25	\$99/an
Test	Emulation* / Simulation	Simulation

13

13

## Architecture Android

- Noyau Linux
  - Fils, gestion de mémoire, processus, etc.
- HAL — Hardware Abstraction Layer
- Environnement d'exécution Android (ART)
- Java APIs
- Applications Système



[ [developer.android.com](http://developer.android.com) ]

14

14

# Machine Virtuelle Dalvik

- Exécute du bytecode Dalvik, traduit du bytecode Java
- Possibilité d'utiliser n'importe quel langage qui compile en bytecode Java (*e.g.*, Kotlin, Scala, Jython, JRuby, ...)
- Pas possible si génération de bytecode est dynamique (*e.g.*, Clojure)
- Exécutables, ressources mis ensemble dans un .apk

15

15

# Applis Android vs. bureautiques

- Comparée à une application bureautique, une appli Android :
  - peut avoir  $\geq 1$  point d'entrée
  - peut implicitement s'intégrer avec des services d'autres applis
  - peut proposer des services aux autres applis
  - peut interagir avec d'autres applis inconnues
- Une appli est organisée en activités :
  - Quand l'écran change, c'est souvent une nouvelle activité
  - Une activité peut en lancer une autre et peut lui fournir des données
  - Une activité peut recevoir un résultat d'une autre activité
- Par exemple : scanner un flashcode

16

16



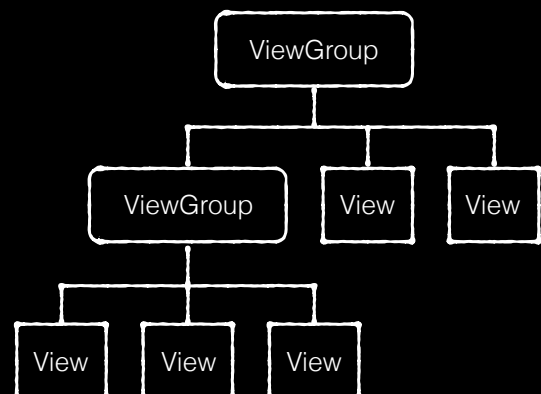
# Interface Utilisateur

17

17

## View, ViewGroup

- L'IU affichée est une hiérarchie de composants
- Pour commencer une activité, appelez `setContentView()` en donnant une référence vers la racine de la vue
- La plupart du temps, on définit cet arbre dans un XML layout
- On peut aussi le manipuler en Java



18

18

# Layout en XML

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity" >

    <EditText android:id="@+id/edit_euros"
        android:inputType="numberDecimal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/euros_label" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_convert"
        android:onClick="convert" />

</LinearLayout>
```

19

19

# Widgets

- Android définit des widgets pour construire l'interface :
  - Widgets de base : boutons, cases à cocher, champs de texte, ...
  - Widgets un peu plus complexes : horloge, zoom, date picker
- Tous les widgets sont définis dans le paquetage `android.widget`

20

20

# Interaction

- Deux méthodes existent pour gérer une interaction :
  - Abonner des listeners à la view.
    - Cette méthode est conseillée.
  - Créer une sous-classe, puis surcharger les méthodes de rappel des événement de saisie.

21

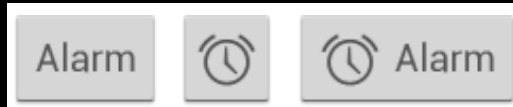
21

Widget	Classe
Bouton	Button
Champs de texte	EditText, AutoCompleteTextView
Case à cocher	CheckBox
Bouton radio	RadioGroup, RadioButton
Interrupteur à bascule	ToggleButton
Liste déroulante	Spinner
Sélecteur (de date, horaire, ...)	DatePicker, TimePicker

22

22

# Bouton



- Peut avoir un label, icône, ou bien les deux
- XML : android:onclick
- Code :  
setOnClickListener,  
View.OnClickListener

Dans le layout XML :

```
<Button ...  
    android:id="@+id/button_send"  
    android:text="@string/button_send"  
    android:onClick="sendMessage" />
```

Dans le code de l'Activity :

```
/* Appelée lorsque l'utilisateur tape le bouton */  
public void sendMessage(View view) {  
    // Fais qqc ici  
}
```

[ <https://developer.android.com/guide/topics/ui/controls/button.html> ]

23

23

# Rajouter un listener en code

Alternatif, dans l'Activity, sans utiliser le XML :

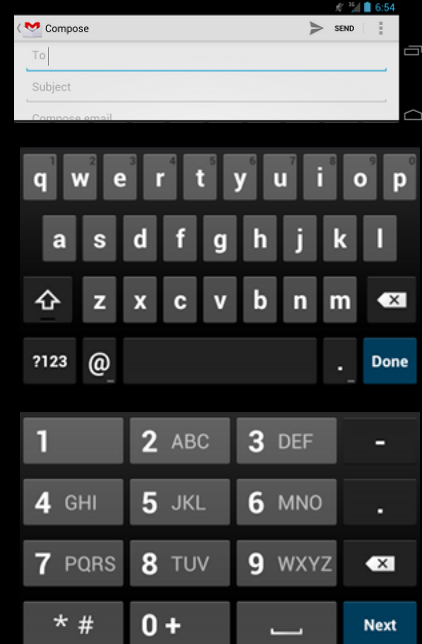
```
Button button = (Button) findViewById(R.id.button_send);  
button.setOnClickListener(new View.OnClickListener() {  
    public void onClick(View v) {  
        // Fais qqc ici  
    }  
});
```

24

24

# Champs de texte

- Permet de saisir du texte
- Peut utiliser un clavier adapté avec `android:inputType` : `text` (clavier normal), `textUri`, `textEmailAddress`, `phone`, ...
- Options : `textCapSentences`, `textCapWords`, `textAutoCorrect`, `textPassword`, `textMultiLine`, ...
- Peut combiner les options avec | : `android:inputType="text|textCapWords"`



[ <https://developer.android.com/guide/topics/ui/controls/text.html> ]

25

25

# Rajouter un EditText

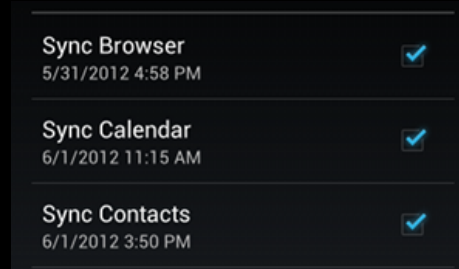
```
<EditText
    android:id="@+id/email_address"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/email_hint"
    android:inputType="textEmailAddress" />
```

26

26

# Cases à cocher

- Permet de choisir zéro, un, ou plusieurs options
- Gère les tapes comme un bouton :  
`android:onClick`,  
`View.OnClickListener`
- État peut être changé en code avec  
`setChecked(boolean)` ou  
`toggle()`



[ <https://developer.android.com/guide/topics/ui/controls/checkbox.html> ]

27

27

## Cases à cocher (2)

Dans l'Activity :

Dans le layout XML :

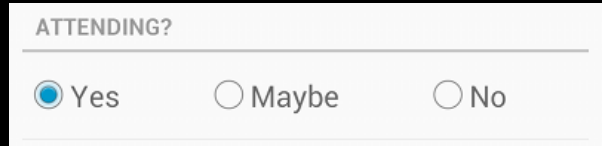
```
<LinearLayout
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <CheckBox android:id="@+id/checkbox_meat"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/meat"
    android:onClick="onCheckboxClicked"/>
  <CheckBox android:id="@+id/checkbox_cheese"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/cheese"
    android:onClick="onCheckboxClicked"/>
</LinearLayout>
```

```
public void onCheckboxClicked(View view) {
  // La case est-elle cochée ?
  boolean checked = ((CheckBox) view).isChecked();

  // Quelle case a été cochée ?
  switch(view.getId()) {
    case R.id.checkbox_meat:
      if (checked) {
        // Rajouter de la viande
      } else {
        // Enlever la viande
      }
      break;
    case R.id.checkbox_cheese:
      if (checked) {
        // Rajouter du fromage
      } else {
        // Pas de fromage
      }
      break;
    // TODO: Végétarien ...
  }
}
```

28

# Boutons radio



- Pour sélectionner une option parmi plusieurs, quand on veut voir toutes les options à la fois
- Gérés comme des `CheckBox`, mais avec `RadioButton` : `android:onClick`, `setChecked(boolean)`, `toggle()`
- Appartiennent à un `RadioGroup`

```
<RadioGroup
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <RadioButton android:id="@+id/radio_pirates"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/pirates"
        android:onClick="onRadioButtonClicked"/>
    <RadioButton android:id="@+id/radio_ninjas"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/ninjas"
        android:onClick="onRadioButtonClicked"/>
</RadioGroup>
```

[ <https://developer.android.com/guide/topics/ui/controls/radiobutton.html> ]

29

29

# Adapters

- On a souvent besoin de s'intégrer avec une source de données, c.f. pour afficher des résultats dans une liste.
- Un adaptateur permet de relier une source de données statiques ou dynamiques à un tel widget.
- Relier un adaptateur à un curseur d'une BD peut aider à efficacement gérer la mémoire et de construire les vues à la demande.

30

30

# Opérations lentes

- Android utilise un fil d'exécution pour l'UI
  - Si une opération (potentiellement) de longue durée s'exécute sur le fil principal (e.g., sleep, accès au réseau, BD, ...), l'interface ne répond pas.
  - Aucun autre fil ne puisse modifier l'UI.
- Pour exécuter une longue opération :
  - Un nouveau *worker thread* est créé pour la faire.
  - Le *worker thread* envoie des messages au *UI thread* via un `Handler`.
  - À la réception du message, le `Handler` tourne sur l'UI thread.

31

31

# Et si on n'utilisait pas de thread ?

- Lorsqu'un événement arrive (comme un tap), il est passé à l'UI thread.
- Android surveille le temps de gestion d'un événement
- Si l'événement n'est pas géré assez vite, Android traite l'appli comme non-réactive et propose à l'utilisateur de la fermer brutalement.
- Le *strict mode* peut être activé pour signaler ce genre d'erreur
- Utiliser les bons outils dès le début est fortement conseillé : `Handler`, `IntentService`, `AsyncTask`, `runOnUiThread`.

32

32




# AsyncTask & runOnUiThread

- La classe `AsyncTask` permet de faire une tâche lente sur un autre fil d'exécution tout en affichant une barre de progrès. Elle utilise des types génériques et peut être spécialisée aux besoins.
- La méthode `runOnUiThread` de la classe `Activity` permet d'exécuter du code sur le fil principal à partir d'un fil secondaire.
- Les deux solutions utilisent les mêmes mécanismes que les `Handlers`.

33

33

# Sages conseils

- Éviter les méthodes longues et compliquées.
  - Suivez le principe du  : Keep It Simple, Stupid !
- Si votre solution est difficile à comprendre ou à implémenter, changez de cap, elle n'est probablement pas la bonne solution.

34

34

# UI Guidelines

- Google a publié de très bonnes guidelines
  - Pour aider aux applis tierces de ressembler à celles du système
  - Pour aider les utilisateurs à reconnaître des symboles et idiomes communs
  - Pour faciliter l'adaptation aux écrans, densités, et formats différents
- Suivez-les autant que possible.

35

35

# Cartes


- Il est possible de placer des items sur une carte :
  - Des widgets Google Maps sont disponible avec un clef API (gratuit), lié à l'appli
  - Mapsforge propose une API similaire basée sur OpenStreetMap

36

36

[developer.android.com](http://developer.android.com)



 [ Certains transparents adaptés de ceux de Samuel Tardieu ]