



# [Initiation à Android]

Télécom Bretagne  
Décembre 2017



Horacio Gonzalez

# Initiation à Android



## [Qui sommes-nous ?]



Horacio Gonzalez

# Initiation à Android



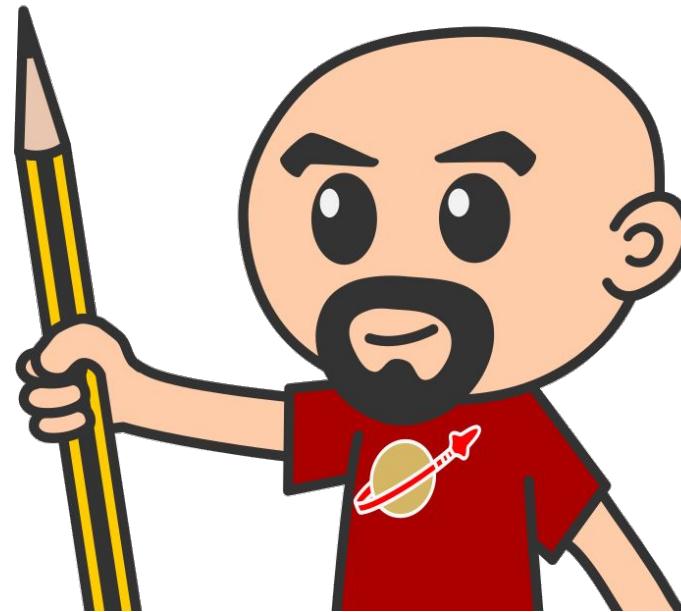
## Horacio Gonzalez

@eceLostInBrittany

Spaniard lost in Brittany,  
developer, dreamer and  
all-around geek



<http://cityzendata.com>



Finist  
Devs



# Initiation à Android



[ Objectifs ]





## Objectifs de la formation

Maîtriser et être autonome sur les éléments suivants:

- Activity
- Fragment
- AsyncTask
- View & Layout

Les bonus:

- Material Design
- Stockage de données
- Gestion des différents écrans



# Initiation à Android



[ Android ]



Horacio Gonzalez

# Initiation à Android



## C'est quoi Android ?

- Système d'exploitation sur noyau Linux

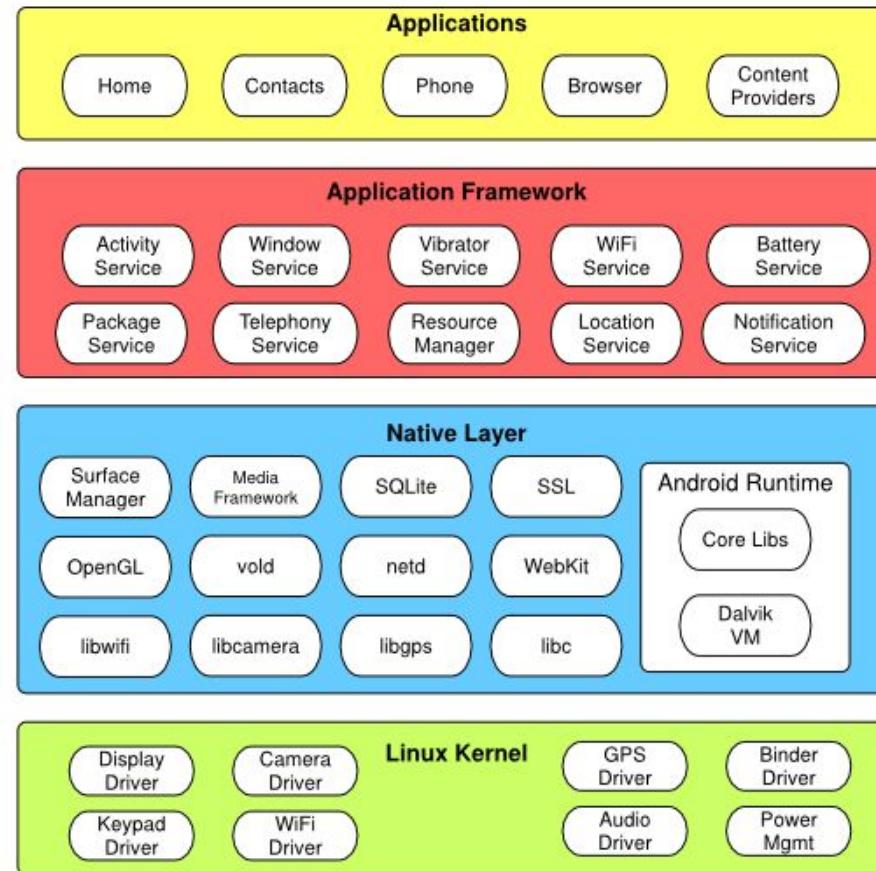


Image : [newcircle](#)



Horacio Gonzalez

# Initiation à Android



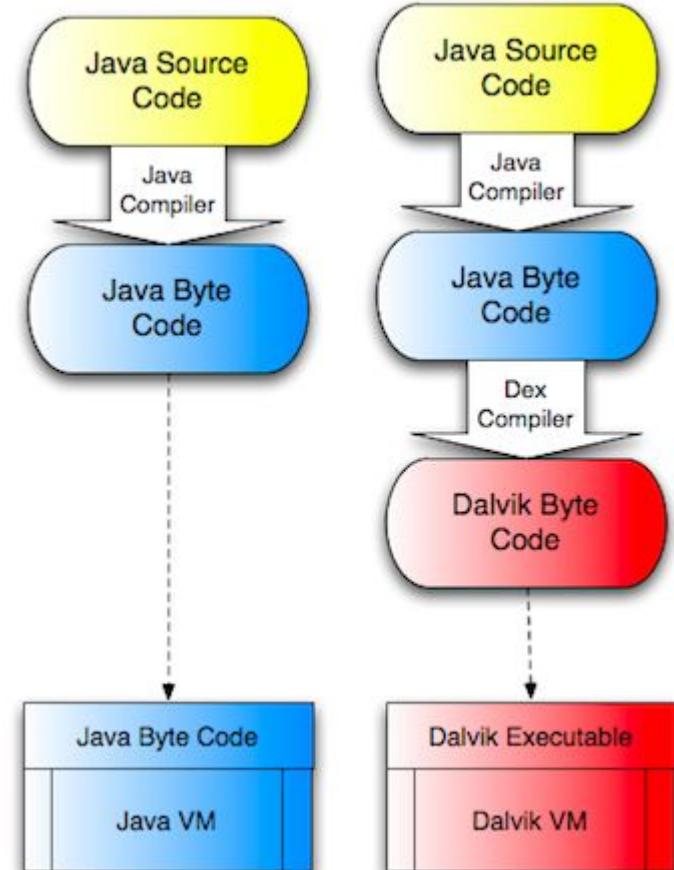
## Dalvik ?

Machine virtuelle *open-source* utilisée sur Android avant Android 5.0 Lollipop



- Exécutant des fichiers **.dex**
  - Format différent du bytecode Java
  - Instructions basées sur des registres
    - pas sur une pile comme en Java
  - Avec des optimisations
    - pas de duplication de Strings constantes...
- Pas compatible avec une JVM

Le compilateur Dex compile le bytecode Java en bytecode Dex



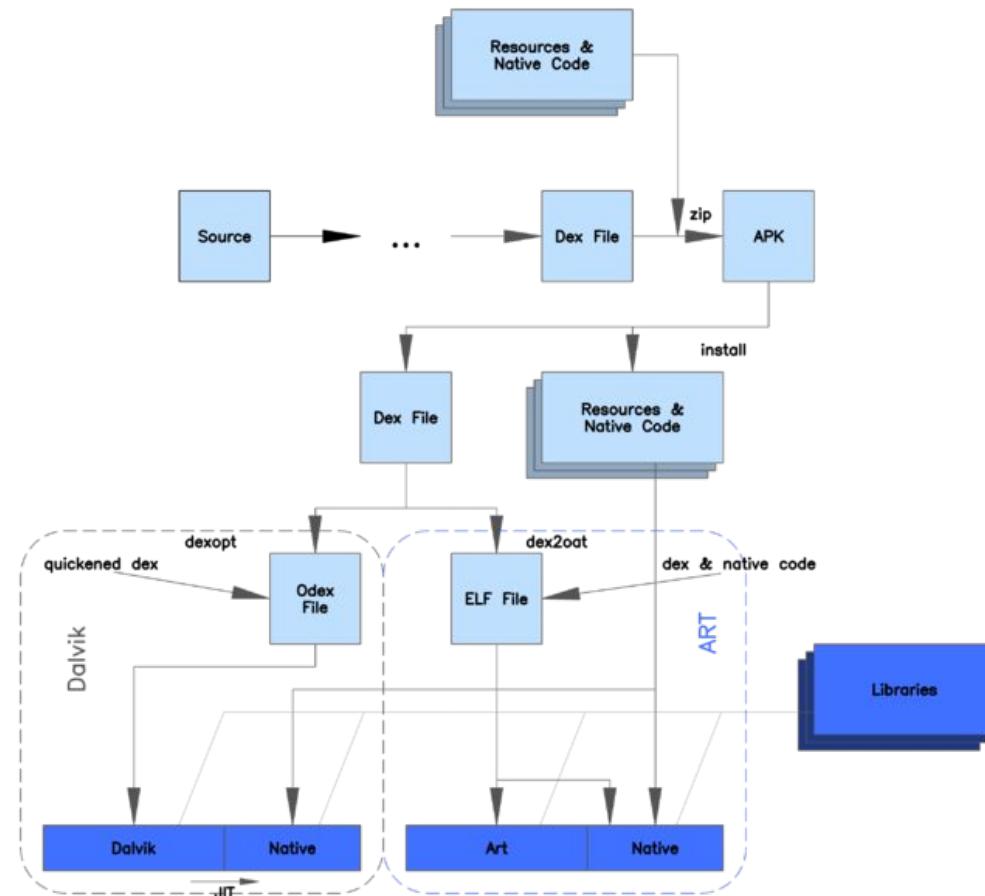
# Initiation à Android



## ART (Android Runtime) ?

Machine virtuelle *open-source*  
utilisée sur Android depuis  
Android 5.0 Lollipop

- Compilation des fichiers **.dex** à l'installation (AoT)
- Depuis Android 7.0 Nougat une compilation JIT complémentaire et améliore les performances de l'AoT



# Initiation à Android



## Terminaux Android

Partout !

- Smartphones
- Tablettes
- Netbooks
- Téléviseurs
- Autoradios
- Domotique
- Open-hardware
- ...



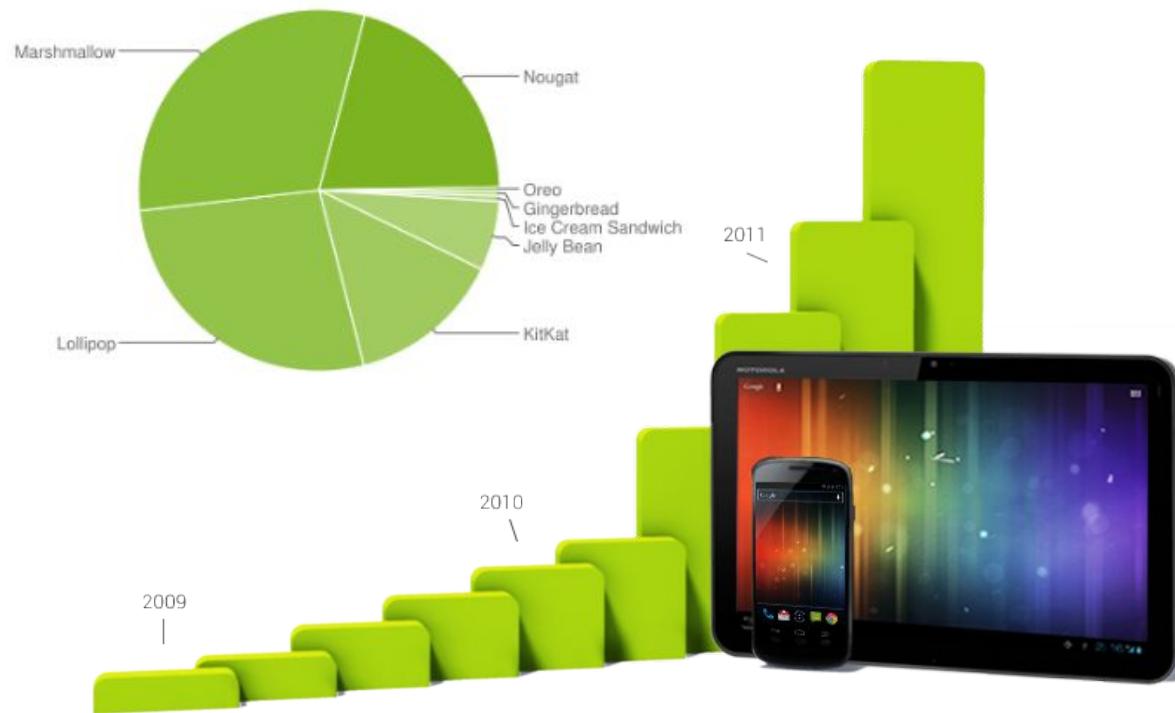
# Initiation à Android



## Quelques chiffres

- 1 500 millions de terminaux activés
- 1,5 millions d'activations par jour
- 50 milliards d'applications installés

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.5%
4.1.x	Jelly Bean	16	2.2%
4.2.x		17	3.1%
4.3		18	0.9%
4.4	KitKat	19	13.8%
5.0	Lollipop	21	6.4%
5.1		22	20.8%
6.0	Marshmallow	23	30.9%
7.0	Nougat	24	17.6%
7.1		25	3.0%
8.0	Oreo	26	0.3%





## [Outils de Développement]



Horacio Gonzalez



# Du point de vue d'un développeur

Des points forts :

- Développement en Java ou en Kotlin
- Un SDK complète, *open-source* et multi-plateforme
  - Des IDEs basés sur des outils confirmés
    - IntelliJ IDEA
  - Des exemples, de la documentation
  - Des APIs riches
  - Un émulateur



# Android Studio

- IDE pour Android

- Basé sur IntelliJ IDEA
- Actuellement en v.3.0.x
- IDE Android complet
  - Syntaxe, refactoring, corrections, templates
  - Editeur graphique d'écrans (layouts)
  - Emulateur des différents terminaux
- Outilage additionnel :
  - Construction : Gradle
  - Performance : Lint
  - Signature d'applications : ProGuard





## Fondements d'une application

- Applications écrites en Java ou Kotlin
- Compilation avec SDK Android
  - Génération d'un fichier .apk
- Les applications tournent dans un sandbox
  - User, VM et processus Linux isolés par application
  - Principe du moindre de privilèges
- Communication entre applications



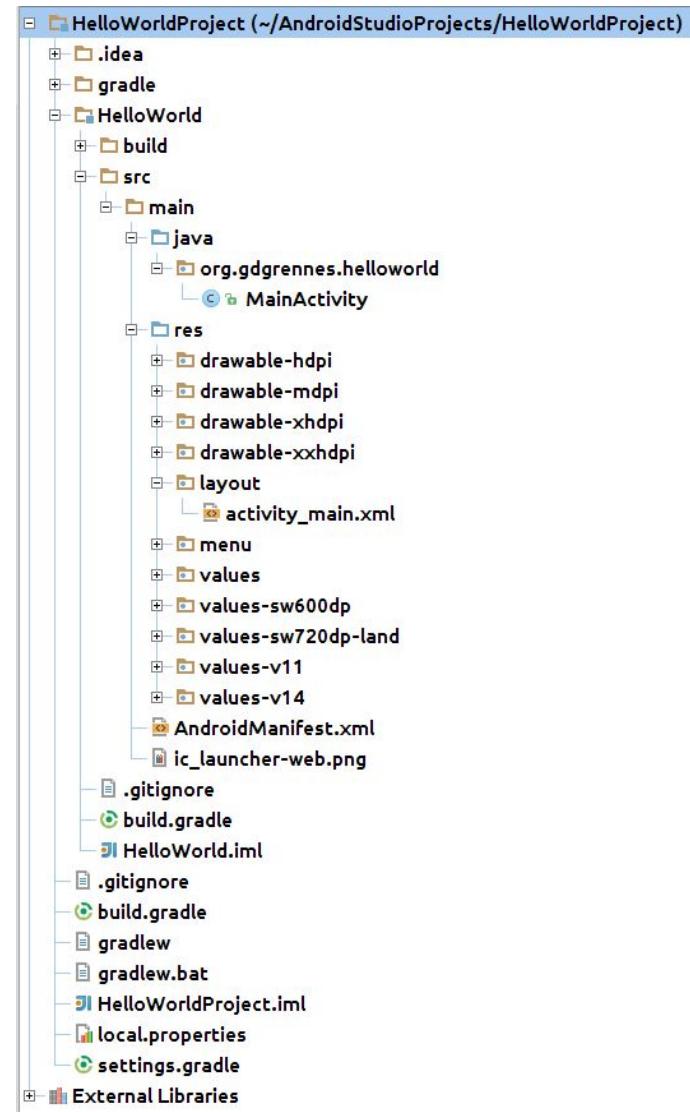
## [Anatomie d'une application Android]





# Arborescence

- /src
  - /src/java
  - /src/res
    - /src/res/layout
    - /src/res/menu
    - /src/res/values-xxx
    - /src/res/drawable-xxx
  - /src/AndroidManifest.xml
- /build.gradle



# Initiation à Android



## Anatomie d'une application Android

- *Activities*
- *Services*
- *Intents & Intent Filters*
- *Processes & Threads*
- *Permissions*
- *Manifest*

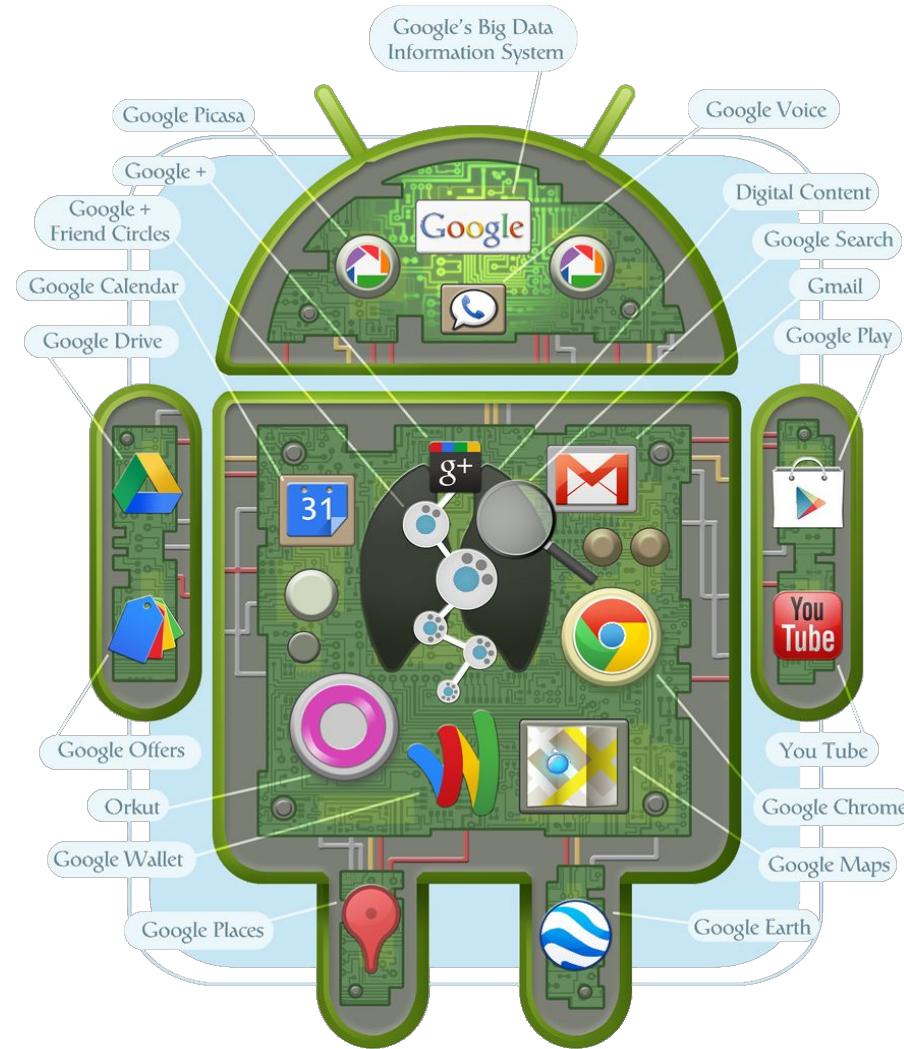


Image : [Android4Fans](#)

# Initiation à Android



{ Activity }



Horacio Gonzalez





# Activités (Activities)

- Une Activité peut être assimilée à un écran qu'une application propose à son utilisateur
  - La transition entre deux écrans correspond au lancement d'une Activité ou au retour sur une Activité placée en arrière-plan
- Une Activité est composée de deux volets :
  - La logique et la gestion du cycle de vie
    - Implémentées en Java dans une classe héritant de **Activity**
  - L'interface utilisateur, qui peut être définie
    - Soit dans le code de l'Activité
    - Soit dans un gabarit (*layout*)
      - Fichier XML placé dans les ressources de l'application
      - Approche plus générale à favoriser

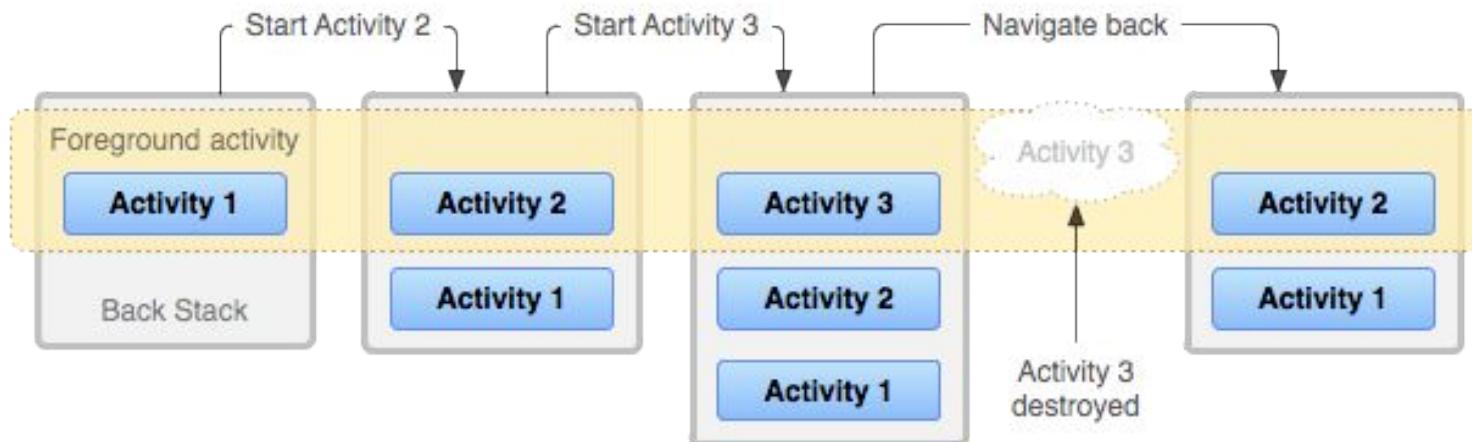


# Initiation à Android



## Plusieurs Activités par application

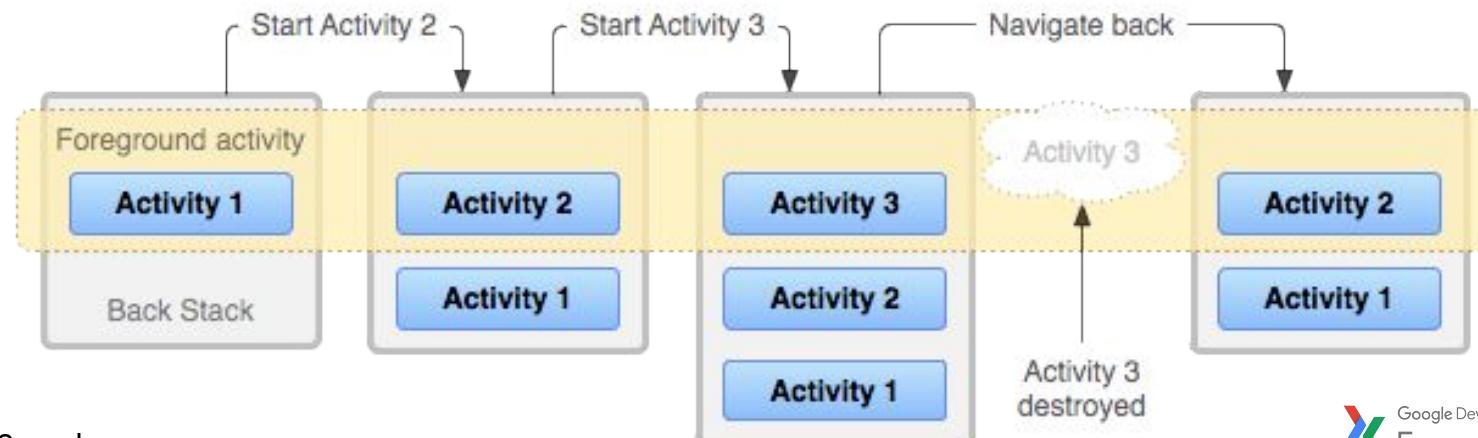
- Une application peut avoir plusieurs Activités
  - Une Activité par écran de l'application
  - Une seule Activité est active à un instant donné
    - L'Activité précédente est mis en pause
  - Les Activités d'une application sont gérées dans le *Back Stack* (une pile)





# Activités et Tâches (Tasks)

- Tâche : série d'activités avec lesquelles l'utilisateur interagit pour faire quelque chose
  - Unité d'exécution des Activités
    - Une pile d'Activités par Tâche
  - Les Activités d'une application s'exécutent dans la même Tâche
  - Des fois des Activités de plusieurs applications s'exécutent aussi dans une même Tâche
    - Si une application a besoin de lancer des Activités d'une autre
    - Exemple : navigateur qui ouvre un lien `mailto` dans gmail

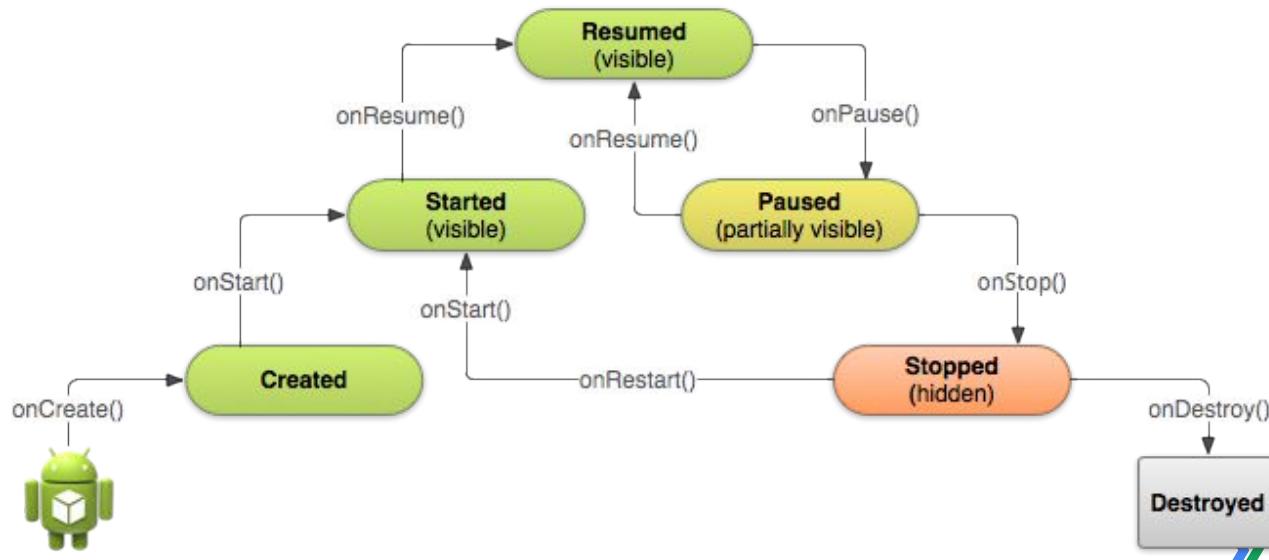




# Cycle de vie d'une Activité

## États principaux d'une Activité :

- Active (*active* ou *resumed*)
  - Activité visible qui détient le focus et attend les entrées utilisateur.
- Suspendue (*paused*)
  - Activité au moins en partie visible à l'écran mais sans le focus
- Arrêtée (*stopped*)
  - Activité non visible, mais encore en vie





# Cycle de vie d'une Activité

- Démarrage → Active
  - Méthode `onStart()`
- Active → Suspendue
  - Méthode `onPause()`
- Suspendue → Arrêtée
  - Méthode `onStop()`
- Suspendue → Active
  - Méthode `onResume()`
- Arrêtée → Détruite
  - Méthode `onDestroy()`

```
public class Main extends Activity {  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.acceuil);  
    }  
    protected void onDestroy() {  
        super.onDestroy();  
    }  
    protected void onPause() {  
        super.onPause();  
    }  
    protected void onResume() {  
        super.onResume();  
    }  
    protected void onStart() {  
        super.onStart();  
    }  
    protected void onStop() {  
        super.onStop();  
    }  
}
```



# Sauvegarde du contexte : *Bundle*

- La gestion de mémoire est faite par le système
  - Une application qui n'a plus le focus, peut être déchargée de la mémoire
- Lorsqu'une Activité est suspendue, on appelle à `onSaveInstanceState(Bundle outState)`
  - Par défaut il sauvegarde tous les champs avec identifiant
    - Formulaires de saisie...
- Si l'application est déchargée, le système doit la redémarrer lorsqu'elle récupère le focus
  - Pour récupérer l'état, il lui donnera le bundle dans `onCreate(Bundle savedInstanceState)`



# Initiation à Android



## {Interfaces graphiques}



Horacio Gonzalez





# Vues et Gabarits

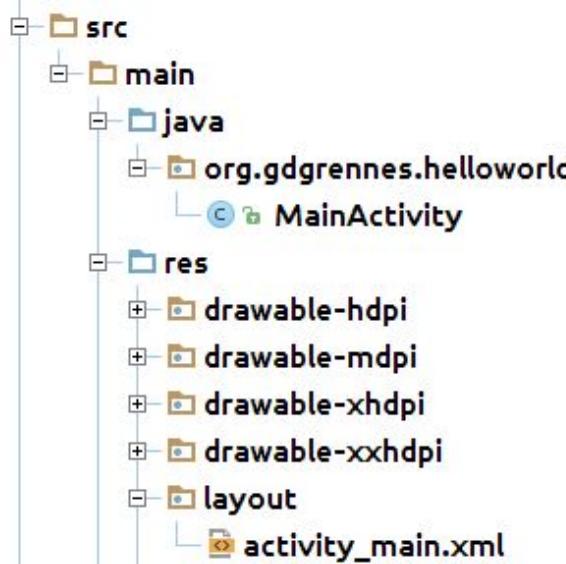
- Vues (Views) : composants graphiques de l'interface
  - Héritant de la classe `View`
- Les Vues sont groupées dans des Gabarits (Layouts)
  - Le Gabarit décrit l'interface associée à chaque Activité
- Le type de Gabarit définit la disposition des Vues dans l'écran
  - `LinearLayout` : éléments de gauche à droite et du haut vers le bas
  - `RelativeLayout` : élément placé par rapport au précédent
  - `TableLayout` : éléments placés selon une disposition matricielle
  - `FrameLayout` : éléments qui s'empilent les uns sur les autres



# Gabarits

## Descriptions en XML de l'interface de l'Activité

- Placé dans les ressources de l'application
  - Dans /src/res/layout/



```
<?xml version="1.0" encoding="utf-16"?>
<LinearLayout
  xmlns:android=
    "http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >

  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />

  <Button
    android:id="@+id/bouton1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />

  <EditText
    android:id="@+id/edit1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
</LinearLayout>
```



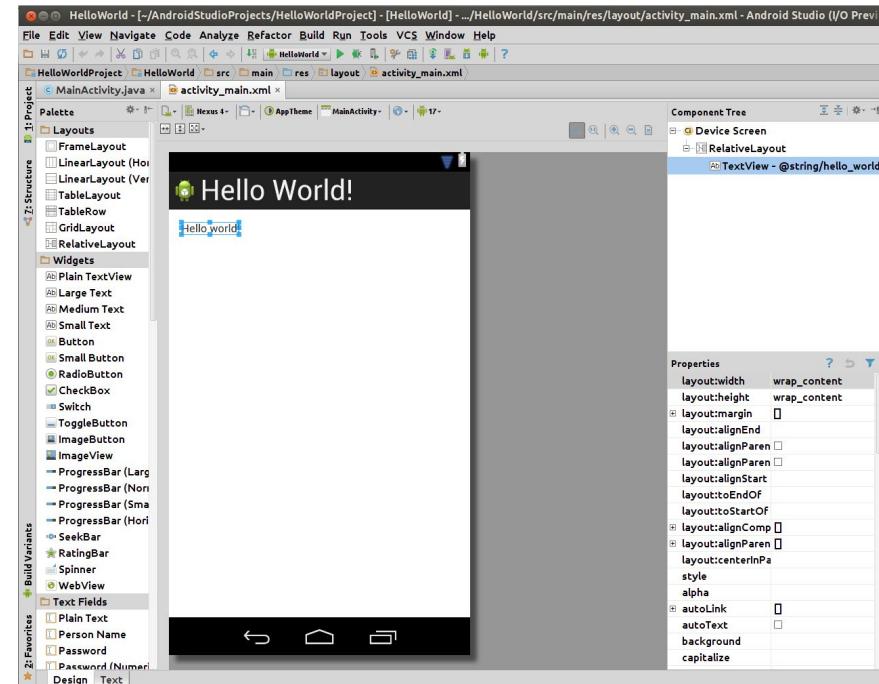
# Composants graphiques

- On ajoute des composants
  - Soit directement en XML
  - Soit en utilisant l'éditeur graphique

```
<?xml version="1.0" encoding="utf-16"?>
<RelativeLayout
    xmlns:android=
        "http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```





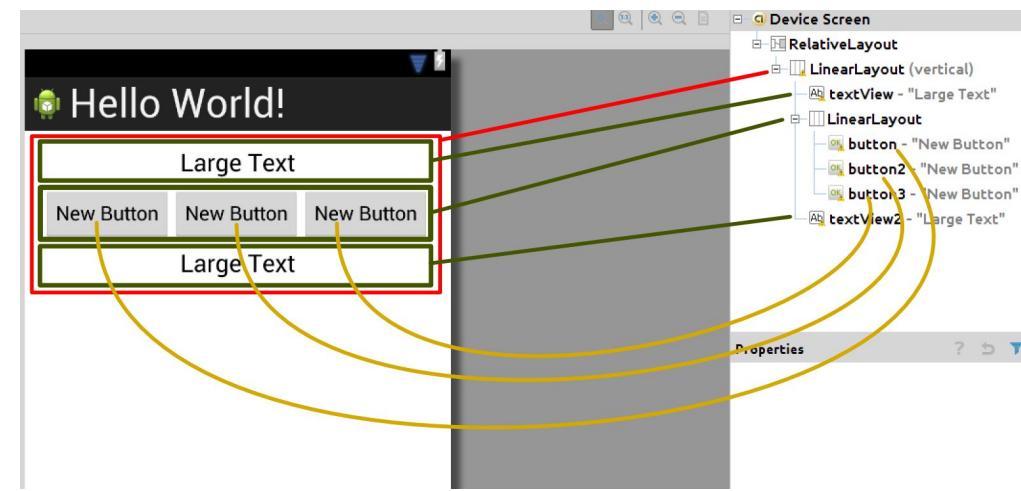
# Imbrication de gabarits

- Les Gabarits peuvent inclure d'autres Gabarits
  - Pour faire du positionnement avancé
    - Un layout horizontal à l'intérieur d'un vertical, par exemple
  - Pour permettre la réutilisation du code
    - En utilisant le mot clé `include`

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

    <include android:id="@+id/included_accueil"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        layout="@layout/acceuil" />

</LinearLayout>
```





# Attributes du Gabarit

Permettent de fournir des propriétés globales du Gabarit

- Les plus importants sont :

- android:layout\_width et android:layout\_height
  - définit la place occupé par le gabarit par rapport au père
  - valeur `fill_parent` - l'élément remplit l'élément parent
  - valeur `wrap_content` - prend la place nécessaire à l'affichage
- android:orientation
  - définit l'orientation d'empilement
  - valeurs `vertical` et `horizontal`
- android:gravity
  - définit l'alignement des éléments
  - valeurs `top, bottom, left, right...`

```
<LinearLayout
    xmlns:android=
        "http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:id="@+id/accueilid" >
    [...]
</LinearLayout>
```





# Accès depuis le code au gabarit

Et si dans mon code je veux modifier un composant ?

- On utilise la classe **R**
  - Le nom du fichier XML permet de retrouver le Gabarit dans le code
    - pour `accueil.xml` on passe par `R.layout.accueil`
- Associer un Gabarit à une Activité : `setContentView()`

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

- On peut accéder à tout élément par son **id**

Dans le code :

```
LinearLayout l =  
    (LinearLayout)findViewById(R.id.mongabarit);  
l.setBackgroundColor(Color.BLACK);
```

Dans le gabarit :

```
<LinearLayout  
    [...]  
    android:id="@+id/mongabarit" >  
    [...]  
</LinearLayout>
```

# Initiation à Android



{manifest}



Horacio Gonzalez



## Manifeste (*Manifest*)

- Toute application doit en avoir → `AndroidManifest.xml`
  - Dans le répertoire racine de l'application
- Montre l'information essentielle sur l'application
  - Nom du paquet Java de l'application
    - Identifiant unique pour l'application
  - Description des composants de l'application
    - Activités, services, récepteurs et fournisseurs de contenus
  - Nom des classes implémentant chaque composant
  - Capacités de chaque composant, *Intent Filters* auxquels il répond
- Déclare les permissions que l'application doit avoir
  - Pour accéder aux parties protégée de l'API
  - Pour interagir avec d'autres applications
  - Les permissions que les autres applications nécessitent pour appeler les composants de l'appli
- Déclare le niveau minimal requis de l'API Android

# Initiation à Android



## Fichier AndroidManifest.xml

Déclare l'ensemble des éléments de l'application

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.gdgrennes.android.bootcamp"
    android:versionCode="1" android:versionName="1.0">

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Main" android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service>...</service>
        <receiver>...</receiver>
        <provider>...</provider>
    </application>
</manifest>
```



## {Resources}





# Les ressources

Ressources : composants graphiques, chaînes, menus, images, tailles, couleurs...

- Utilisées au travers de la classe statique R
  - Régénérée à chaque changement du projet
  - Incluant les ressources
    - déclarées dans le Manifeste
    - pour lesquelles les fichiers associés sont dans le bon répertoire
- Utilisation : `android.R.type_ressource.nom_ressource`  
En récupérant une instance de la classe Resources :

```
Resources res = getResources();
String hw = res.getString(R.string.hello);
XXX o = res.getXXX(id);
```



Image : [Constant Contact Labs](#)



# Ressources : chaînes de caractères

- Déclarées dans `/src/res/values/strings.xml`
  - Cela permet l'internationalisation

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World!</string>
    <string name="app_name">My Android Hello World</string>
</resources>
```



Image : [Constant Contact Labs](#)

- Utilisation dans le code → `R.string.name`

```
Resources res = getResources();
String hw = res.getString(R.string.hello);
```

# Initiation à Android



## Ressources : chaînes et i18n

- Pour chaque langage un répertoire  
`/src/res/values-XX/`
  - `xx` : code de la langue (`en`, `fr`, `es`...)
- Avec un fichier `strings.xml` à l'intérieur
  - Le `name` de chaque chaîne doit être le même
- Example : `/src/res/values-fr/`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Bonjour le monde!</string>
    <string name="app_name">Mon Hello World Android</string>
</resources>
```

- Android chargera bon fichier de ressources
  - En fonction de la langue du système



Image : [Constant Contact Labs](#)



# Ressources : composants graphiques

- D'autres ressources sont spécifiables dans res:

- les menus (`R.menu`)
- les images (`R.drawable`)
- des dimensions (`R.dimen`)
- des couleurs (`R.color`)



Image : [Constant Contact Labs](#)



Horacio Gonzalez

# Initiation à Android



[LiveCoding]

[Hello World!]





# LiveCoding : Hello world!

- Objectif : Une application minimalist
  - Elle affiche un Hello World sur l'écran
- Concepts à voir
  - Création d'une application
  - Création d'une Activité
  - Ressources
  - Gabarit
  - Chaîne de caractères
  - Déploiement et test



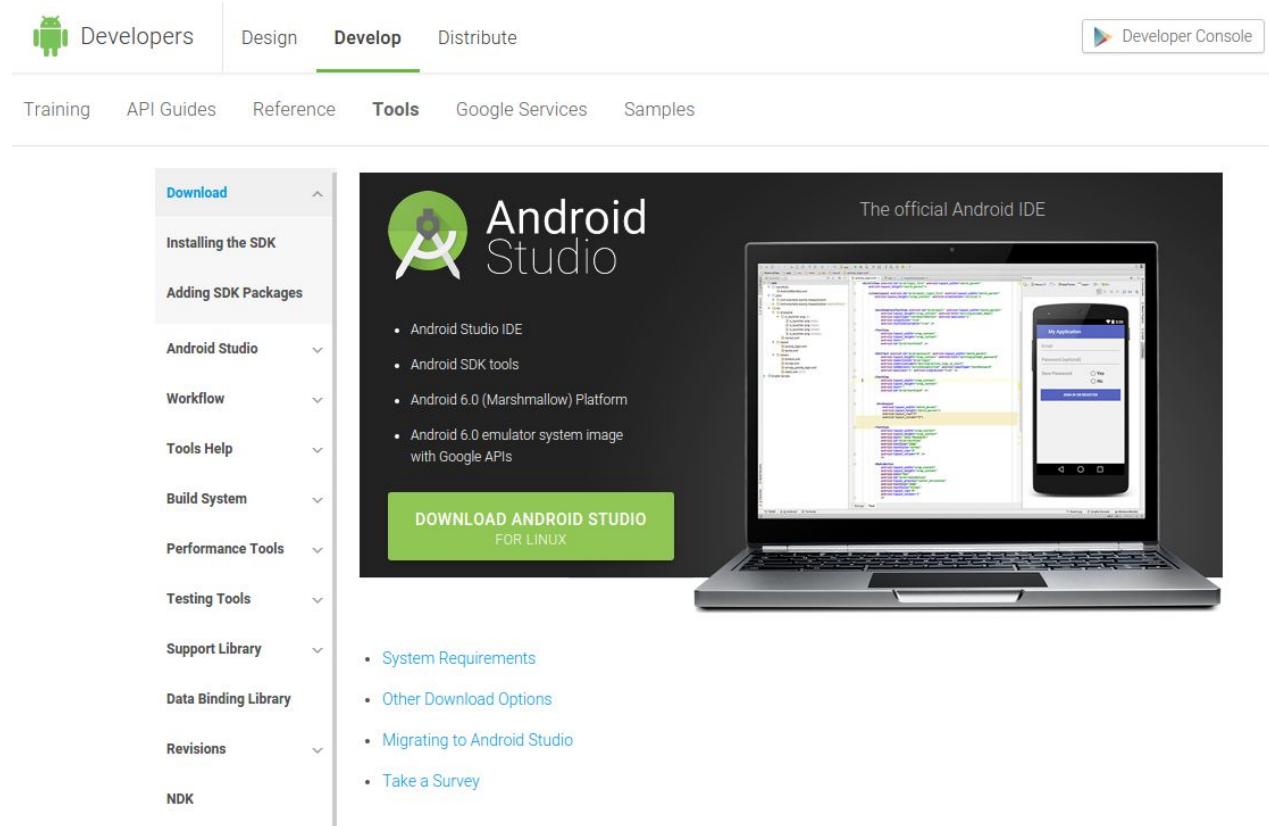
# Initiation à Android



## LiveCoding : Hello world!

### 0. Installer Android Studio :

<http://developer.android.com/sdk/index.html>



The screenshot shows the official Android Developers website. The top navigation bar includes links for Developers, Design, Develop (which is underlined), Distribute, Developer Console, Training, API Guides, Reference, Tools (which is also underlined), Google Services, and Samples. On the left, a sidebar titled 'Download' lists various sections: Installing the SDK, Adding SDK Packages, Android Studio (with a dropdown menu), Workflow, Tools Help, Build System, Performance Tools, Testing Tools, Support Library, Data Binding Library, Revisions, and NDK. The main content area features a large image of a laptop displaying the Android Studio interface, which includes a code editor with Java code and a preview window showing a mobile application. Above the laptop, the text 'The official Android IDE' is displayed. Below the laptop, a green button says 'DOWNLOAD ANDROID STUDIO FOR LINUX'. To the right of the laptop, there is a list of links: System Requirements, Other Download Options, Migrating to Android Studio, and Take a Survey.

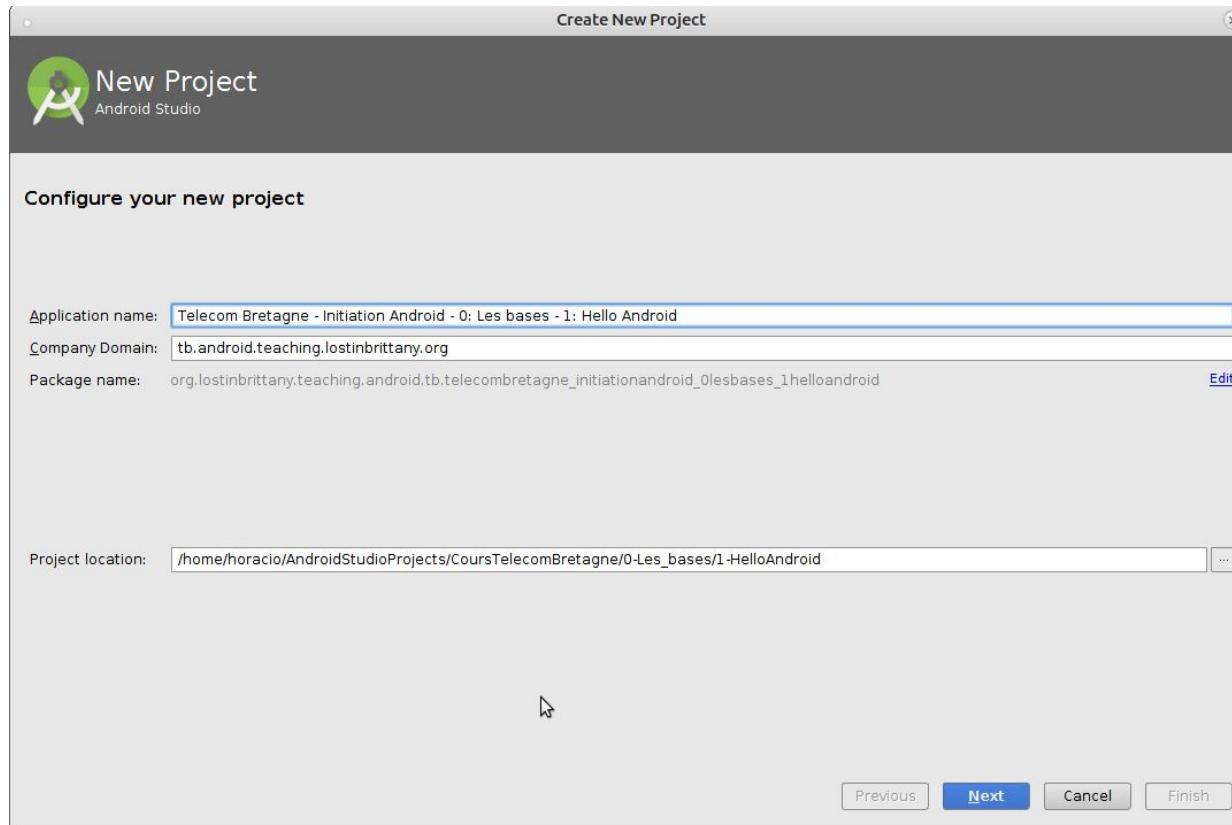


# Initiation à Android



## LiveCoding : Hello world!

1. Démarrer Android Studio
2. Créer une nouvelle application **Hello World!**

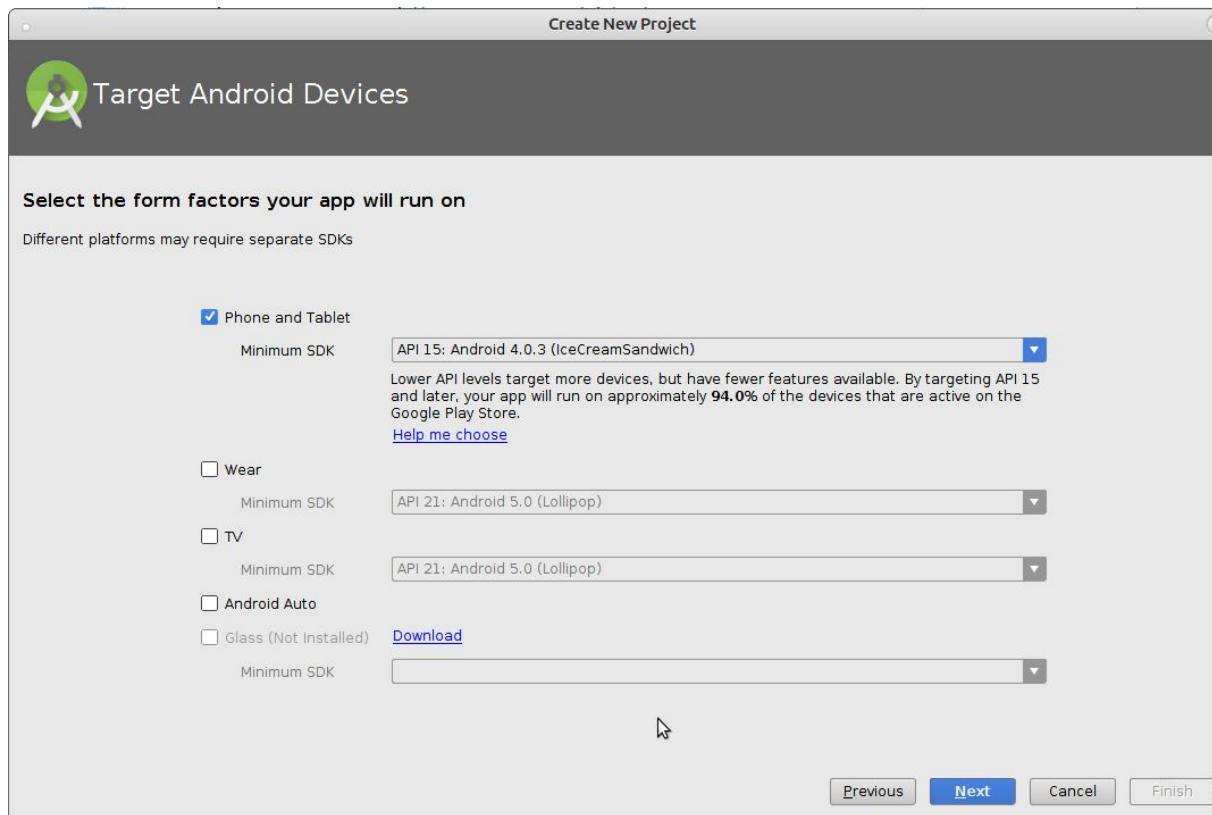


# Initiation à Android



## LiveCoding : Hello world!

3. Accepter le choix de niveau d'API par défaut
4. Créer une application smartphone/tablette





# LiveCoding : Hello world!

## 5. Ajouter une Activity vide MainActivity

The screenshot shows the App Inventor 'Create New Project' interface. On the left, there's a sidebar with icons for 'Add an activity to Mobile' (Blank Activity, Empty Activity), 'Google Maps Activity', 'Login Activity', and 'Master/Detail Flow'. The 'Empty Activity' template is selected. On the right, the main window is titled 'Customize the Activity' and displays the configuration for creating a new empty activity. The 'Activity Name:' field contains 'MainActivity', with 'Generate Layout File' checked and 'Layout Name:' set to 'activity\_main'. At the bottom, there are buttons for 'Previous', 'Next', 'Cancel', and 'Finish'.

Create New Project

Add an activity to Mobile

Add No Activity

Blank Activity

Empty Activity

Creates a new empty activity

Activity Name:

Generate Layout File

Layout Name:

Empty Activity

The name of the activity class to create

Previous Next Cancel Finish





# Initiation à Android



## LiveCoding : Hello world!

The screenshot shows the Android Studio interface with the project '1-HelloAndroid' open. The main window displays a virtual smartphone screen showing the text 'Hello World!' in a blue header bar and a white body. The left side of the interface features a Project structure tree, a Palette with various UI components like Layouts and Widgets, and a Component Tree panel. The right side includes a Properties panel and a Device Screen preview. At the bottom, there are tabs for Design and Text, along with various toolbars and status indicators.



Horacio Gonzalez

# Initiation à Android



## LiveCoding : Hello world!

### 6. Créer un terminal virtuel Android

The screenshot shows the Android Studio interface with the 'AVD Manager' tab selected in the top navigation bar. A modal dialog titled 'Create new Android Virtual Device (AVD)' is open on the left, prompting the user to define the virtual device's configuration. The configuration includes:

- AVD Name: Virtual\_Nexus\_4
- Device: Nexus 4 (4.7", 768 x 1280: xhdpi)
- Target: Android 4.2.2 - API Level 17
- CPU/ABI: ARM (armeabi-v7a)
- Keyboard: Hardware keyboard present (checked)
- Skin: Display a skin with hardware controls (checked)
- Front Camera: None
- Back Camera: None
- Memory Options: RAM: 1907, VM Heap: 64
- Internal Storage: 200 MiB
- SD Card: Size: 100 MiB (radio button selected), File: (Browse...)
- Emulation Options: Snapshot (checkbox), Use Host GPU (checkbox)
- Override the existing AVD with the same name (checkbox)

At the bottom of the modal are 'Cancel' and 'OK' buttons. To the right of the modal is the 'Android Virtual Device Manager' window, which displays a table of existing devices and a set of actions for each row.

AVD Name	Target Name	Platform	API Level	CPU/ABI
Virtual_Nexu	Android 4.2.2	4.2.2	17	ARM (armeabi-v7a)

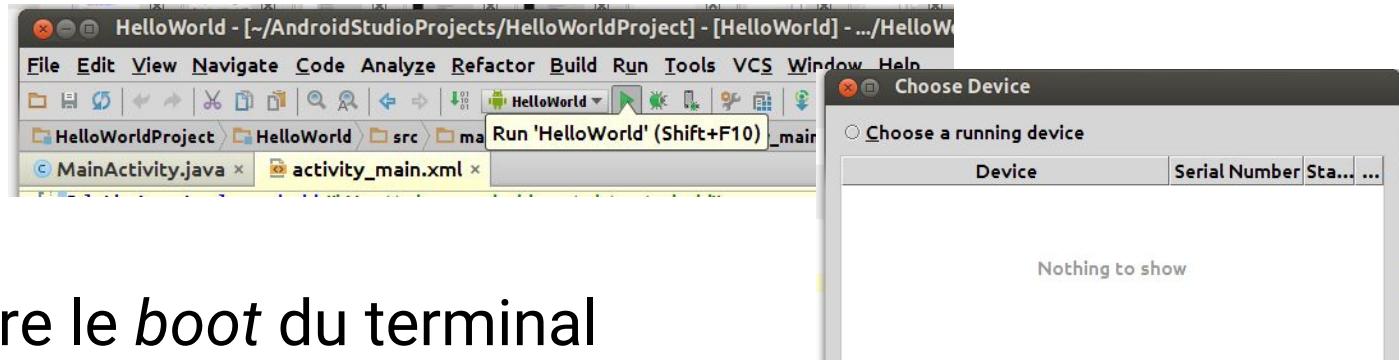
The right side of the window contains several buttons: New..., Edit..., Delete..., Repair..., Details..., Start..., and Refresh. Below the table, status messages are displayed:

- ✓ A valid Android Virtual Device.
- ✗ A repairable Android Virtual Device.
- ✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

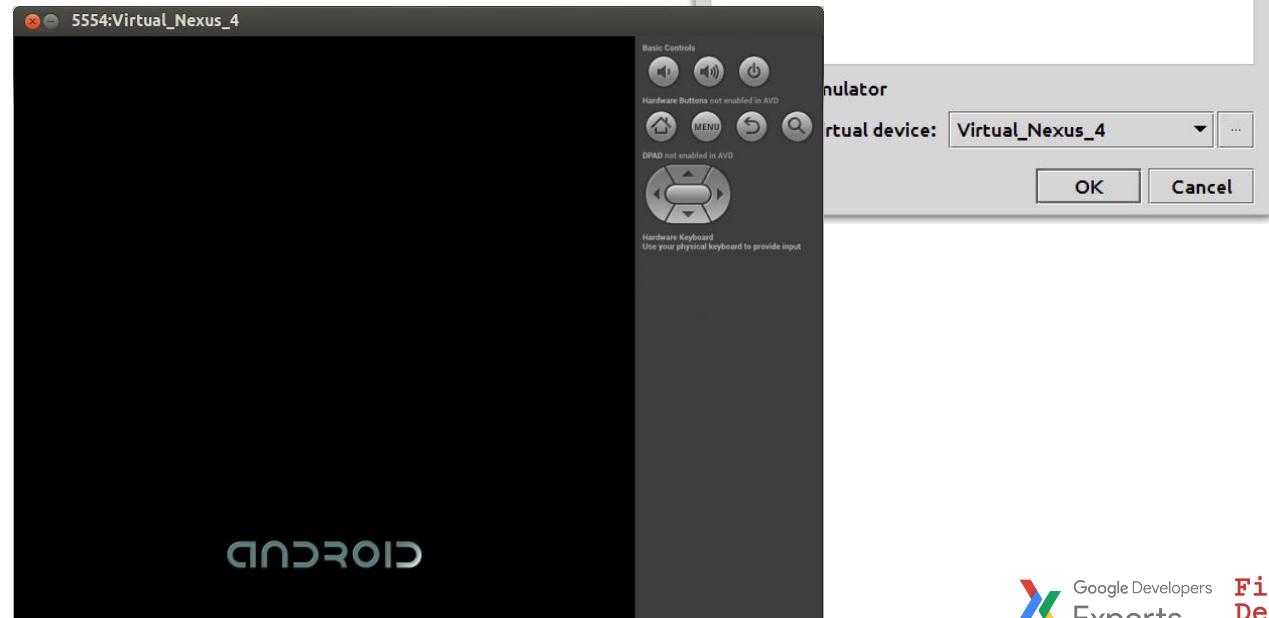


# LiveCoding : Hello world!

## 6. Exécuter le programme



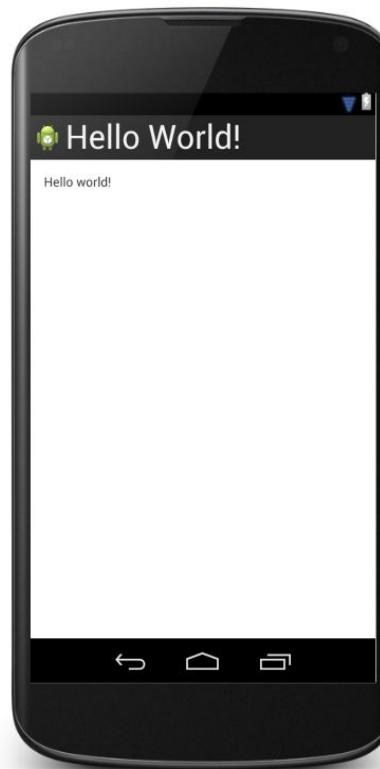
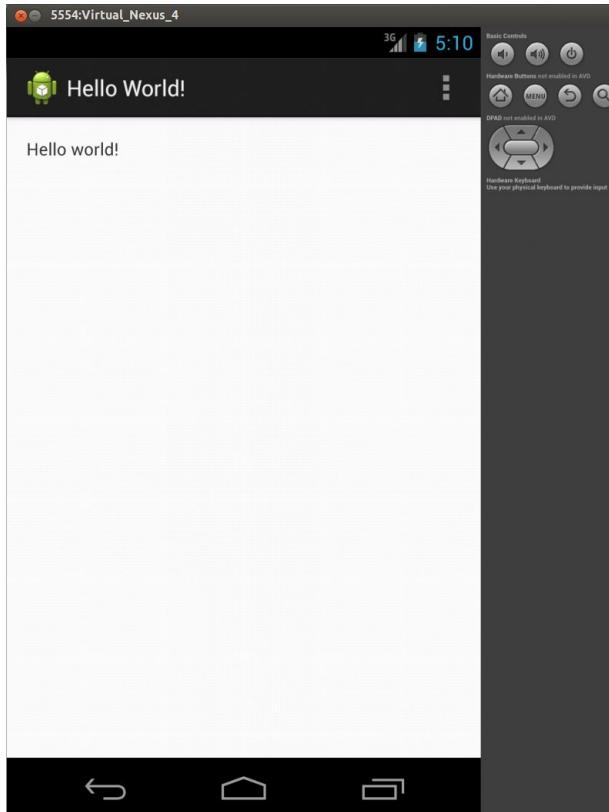
## 7. Attendre le boot du terminal





## LiveCoding : Hello world!

8. S'extasier devant sa première application Android





# Initiation à Android



## LiveCoding : Hello world!

The screenshot shows the Android Studio interface with the following details:

- Title Bar:** HelloWorld - [~/AndroidStudioProjects/HelloWorldProject] - [HelloWorld] - .../HelloWorld/src/main/res/layout/activity\_main.xml - Android Studio (I/O Preview) 0.2.7
- Toolbar:** File Edit View Navigate Code An Select Run/Debug Configuration VCS Window Help
- Project Structure:** HelloWorldProject > HelloWorld > src > main > res > layout > activity\_main.xml
- Code Editor:** Shows the XML code for activity\_main.xml:

```
<RelativeLayout 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:paddingLeft="16dp"
    android:paddingRight="16dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello world!"/>

```
- Preview:** Shows a smartphone displaying the text "Hello World!".
- Logcat:** Shows the following error message:

```
gradle.api.internal.externalresource.transport.http.HttpClientHelper.performRequest(HttpClientHelper.java:69)
more
lation completed with 5 errors and 0 warnings in 15 sec
's
ings
gradle.api.internal.artifacts.ivyservice.ModuleVersionResolveException: Could not resolve com.android.tools.build:gradle:0.5.+
gradle.api.internal.resource.ResourceException: Failed to list versions for com.android.tools.build#gradle;0.5.+
gradle.api.internal.resource.ResourceException: Could not list versions using M2 pattern
.maven.org/maven2/[organisation]/[module]/[revision]/[artifact]-[revision](-[classifier]).[ext].
gradle.api.internal.externalresource.transport.http.HttpRequestException: Could not GET
.maven.org/maven2/com/android/tools/build/gradle/`.

5 errors and 0 warnings in 15 sec (a minute ago)
```



# Initiation à Android



# {Intent}



Horacio Gonzalez



## Intents

*L'intent est une description abstraite d'une opération à effectuer*

Documentation Android

- Objectif : déléguer une action à un autre composant
  - une autre application ou une autre Activité de la même application
- Il s'agit d'un objet que l'on passe à la cible, avec :
  - le nom du composant ciblé (facultatif)
  - l'action à réaliser (chaîne de caractères)
  - les données: contenu MIME et URI
  - des données supplémentaires (paires clé/valeur)
  - une catégorie pour cibler un type d'application
  - des drapeaux (informations supplémentaires)



# Lancer une Activité avec un *Intent*

- Démarrer une Activité dans l'application actuelle
  - Premier paramètre : le contexte de l'application
  - Deuxième paramètre : la classe de l'activité à démarrer

```
Intent intent = new Intent(this, MyTargetActivity.class);
intent.putExtra("title", "Hello Bootcamp"); //Des données extra, clé/valeur
startActivity(intent);
```

- Démarrer autre application
  - Premier paramètre : le type d'action demandée
    - On demande une appli capable de faire l'action
    - Le système choisit laquelle
  - Deuxième paramètre : les données à envoyer à la cible

```
Uri number = Uri.parse("tel:5551234");
Intent callIntent = new Intent(Intent.ACTION_DIAL, number);
startActivity(callIntent);
```





# Composants d'un *Intent*

- Action : on déclare le type d'action qu'on veut réaliser
  - Le système choisit l'application qui va faire l'action
  - Exemple : `Intent.ACTION_VIEW` pour aller voir une URL
    - Par défaut le système ouvrira le navigateur sur l'URL
- Catégorie : information additionnelle sur le composant qui devrait répondre à l'*Intent*
  - Plusieurs catégories possibles
  - Catégories définies par défaut :
    - `CATEGORY_DEFAULT`
    - `CATEGORY_LAUNCHER`
      - L'Activité cible est Activité principale d'une application
    - `CATEGORY_BROWSABLE`
      - L'Activité cible peut être invoqué depuis le navigateur pour gérer des données référencés par une URI
    - `CATEGORY_GADGET`
      - L'Activité cible peut être insérée dans une Activité hôte
    - `CATEGORY_HOME`
      - L'activité Cible montre l'écran d'accueil du terminal
    - `CATEGORY_PREFERENCE`
      - L'Activité cible est un panneau de préférences





# Composants d'un *Intent*

- Données : une URL et un type MIME
  - l'URL et le type MIME des données qu'on passe à la cible
  - Les URI de type **content** : indiquent des données sur le terminal
    - Contrôlées par un *Content Provider*
- Extras
  - Des données additionnelles en, forme clé/valeur
- Drapeaux
  - Ajoutent des informations sur comment l'*Intent* doit être géré
    - Par exemple si l'Activité doit démarrer dans une Tâche nouvelle



# Comment répondre aux *Intents*

- Chaque application déclare ses *Intent Filters*
  - Déclarés dans le Manifeste
  - Indiquant à quels *Intents* elle peut répondre
  - Plusieurs niveaux de filtrage
    - Action, Catégorie et Données
    - Pour répondre à un Intent, les trois niveaux doivent être ok

```
<intent-filter . . . >
    <action android:name="com.example.project.SHOW_CURRENT" />
    <action android:name="com.example.project.SHOW_RECENT" />
    <action android:name="com.example.project.SHOW_PENDING" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    <data android:mimeType="video/mpeg" android:scheme="http" . . . />
    <data android:mimeType="audio/mpeg" android:scheme="http" . . . />
</intent-filter>
```



## Broadcast Intent

- *Intent* diffusé à l'ensemble des applications du système
  - Pour transmettre des informations à destination d'autres applications
    - demander la réalisation d'actions spécifiques
    - pour fournir des informations sur l'environnement
      - appel entrant, réseau Wi-Fi connecté...

```
Intent broadcast = new Intent(MyBroadcastReceiver.VIEW);
broadcast.putExtra("extra", "Hello World!");
sendBroadcast(broadcast);
```

- Un *Broadcast Received* permet de recevoir ces *Intents*

```
public final class MyBroadcastReceiver extends BroadcastReceiver {
    public static final String VIEW ="org.gdgrennes.intent.action.VIEW";
    @Override
    public void onReceive(Context context, Intent intent) {
        // Code de traitement de l'Intent ici.
    }
}
```



# Initiation à Android



{Events}



Horacio Gonzalez



## Gestion d'Événements

- *Event Listeners*

- Mettent un composant à l'écoute d'un événement
- Le composant reçoit un Event Listener
  - Qui implémente la méthode de réponse à l'événement

Bouton déclaré dans un Gabarit

```
<Button  
    android:text="Hello!"  
    android:id="@+id	btnHello"  
    android:layout_width=  
        "wrap_content"  
    android:layout_height=  
        "wrap_content" />
```

Code utilisant un *Event Listener* sur le click

```
Button b = (Button) findViewById(R.id.btnHello);  
b.setOnClickListener(new OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        Toast.makeText(v.getContext(),  
            "Hello World!",  
        Toast.LENGTH_LONG).show();  
    }  
});
```



# Initiation à Android



[LiveCoding]

[Hello World!]

[avec Intents et Events]





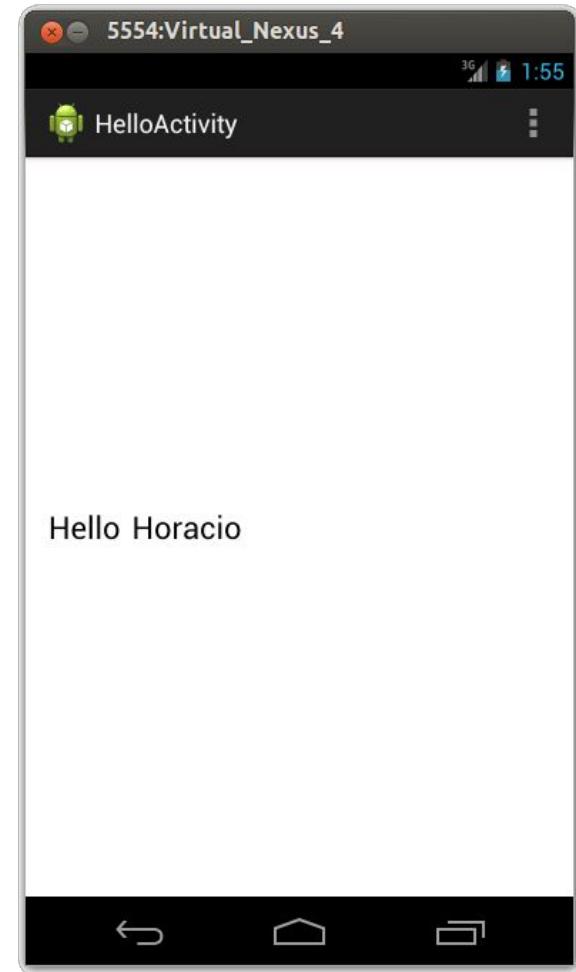
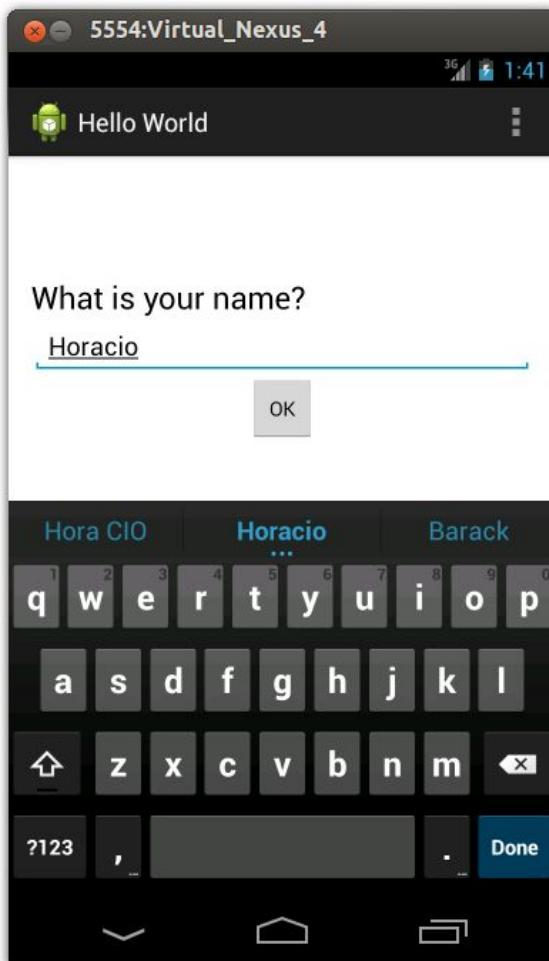
## LiveCoding : Hello World avec Intents et Events

- Objectif : Première application interactive
  - Un premier écran avec un champ pour saisir son nom et un bouton qui amène au deuxième écran
  - Un deuxième écran avec un Hello \$NOM personnalisé
- Concepts à voir
  - Création d'une application avec plusieurs Activités
  - Création d'un Gabarit à plusieurs composants
  - Utilisation des *Event Listeners* pour écouter des Évenements
  - Utilisation des *Intent* pour changer d'Activité

# Initiation à Android



## LiveCoding : Hello World avec Intents et Events



# Initiation à Android



## LiveCoding : Hello World avec Intents et Events

- Code du LiveCoding : dépôt GitHub

<https://github.com/LostInBrittany/Android-TB-2017-HelloWorldWithIntents>

Module Android Télécom Bretagne (Octobre 2016) - Hello World With Intents

[Edit](#)

Add topics

4 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

	LostInBrittany Slides	Latest commit 128aa31 on Oct 5, 2016
	idea Slides	a year ago
	app Hello World With Intents	a year ago
	gradle/wrapper Initial commit	a year ago
	.gitignore Initial commit	a year ago
	2016%2F10 - Initiation Android - 0- Les bases.pdf Slides	a year ago
	2016%2F10 - Initiation Android - 1- Utiliser le reseau.pdf Slides	a year ago
	build.gradle Initial commit	a year ago
	gradle.properties Initial commit	a year ago
	gradlew Initial commit	a year ago
	gradlew.bat Initial commit	a year ago
	settings.gradle Initial commit	a year ago

Help people interested in this repository understand your project by adding a README.

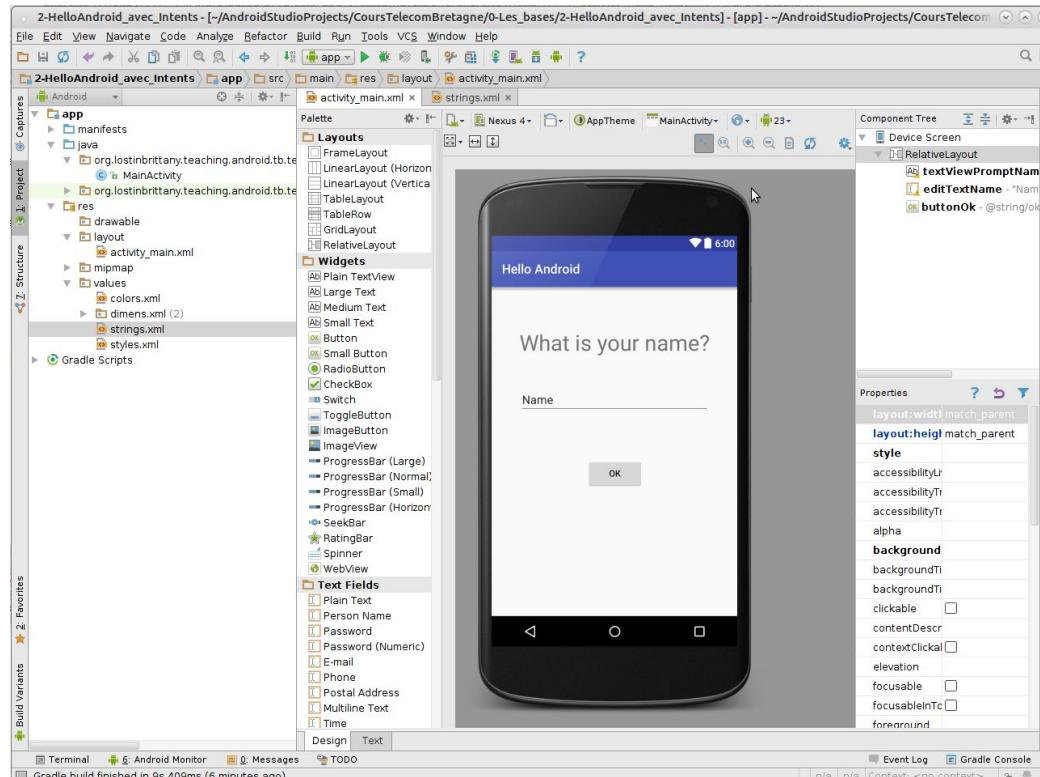
Add a README

# Initiation à Android



## Step-1 : Layout de l'Activity principale

- En utilisant le Designer d'Android Studio



Mais l'i18n, c'est où ?



# Initiation à Android



## Step-2 : i18n des chaînes

The screenshot shows the Android Studio interface with the project "2-HelloAndroid\_avec\_Intents" open. In the top-left, there are tabs for "activity\_main.xml" and "strings.xml". The "strings.xml" tab is active, displaying the following XML code:

```
<resources>
    <string name="app_name">Hello Android</string>
    <string name="what_is_your_name">What is your name?</string>
    <string name="ok">OK</string>
</resources>
```

The "values" folder under "res" contains "colors.xml", "dimens.xml", and "strings.xml". The "strings.xml" file is selected in the Project Structure view.

In the center, the "Layouts" palette shows various view types like FrameLayout, LinearLayout, etc., and the "Widgets" palette shows Text Fields (Plain Text, Person Name, Password, etc.).

The main area displays a smartphone emulator showing a simple UI with a title bar "Hello Android" and a text input field with the placeholder "What is your name?".

The right side of the screen includes the "Properties" panel (with "layout:width:match\_parent" and "layout:height:match\_parent" selected), the "Component Tree" (listing "Device Screen", "RelativeLayout", and "textViewPromptName"), and the "Android Model" panel.

At the bottom, the status bar indicates "Gradle build finished in 9s 409ms (6 minutes ago)" and "Event Log", "Gradle Console", and "Context: <no context>".

# Initiation à Android



## Step-3 : Crédation de HelloActivity

The screenshot shows the Android Studio interface with the project '2-HelloAndroid\_avec\_Intents' open. The left sidebar displays the project structure, showing files like MainActivity.java, activity\_hello.xml, and strings.xml. The main workspace shows a preview of a smartphone displaying the text 'Hello Android' with a single 'Hello' below it. The properties panel on the right is set to 'activity\_hello.xml' and shows attributes like 'layout:width' and 'layout:height' both set to 'match\_parent'. The bottom status bar indicates a successful Gradle build.





# Initiation à Android



## Step-4 : EventListener basique sur le bouton

The screenshot shows the Android Studio interface with the project "HelloWorldWithIntentsAndEvents" open. The code editor displays `MainActivity.java` which contains Java code for handling button clicks. The emulator window on the right shows a virtual Nexus 4 device with the application running, displaying a text input field and a button.

```
package org.gdgrennes.android.bootcamp;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    protected void onStart() {
        super.onStart();
    }

    Button b = (Button)findViewById(R.id.buttonOK);
    b.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Toast.makeText(v.getContext(),
                    "Hello World!",
                    Toast.LENGTH_LONG).show();
        }
    });
}
```





# Initiation à Android

## Step-4 : Intent Filter & Intent

### Dans le l'Activité principale

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

    Button buttonOk = (Button) findViewById(R.id.buttonOk);
    buttonOk.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            EditText editTextName =
                (EditText) findViewById(R.id.editTextName);
            String name = editTextName.getText().toString();

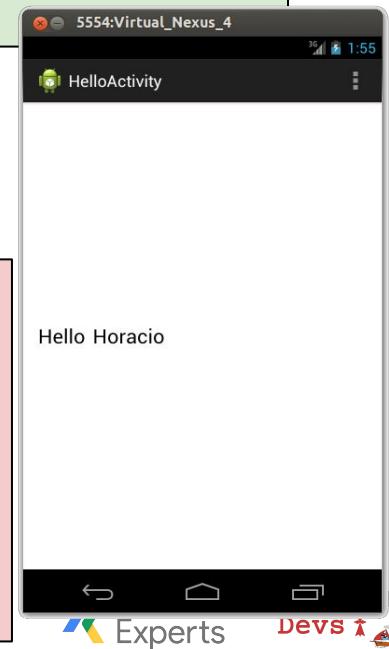
            Intent helloIntent =
                new Intent("org.lostinbrittany.teaching.android.tb.helloandroid_with_intents.HELLO");
            helloIntent.putExtra("name", name);
            startActivity(helloIntent);
        }
    });
}
```

### Dans HelloActivity

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

    Bundle extras = getIntent().getExtras();
    String name = extras.getString("name");

    TextView nameLabel =
        (TextView) findViewById(R.id.textViewHelloName);
    nameLabel.setText(nameLabel.getText() + " " + name);
}
```



### Dans le Manifeste

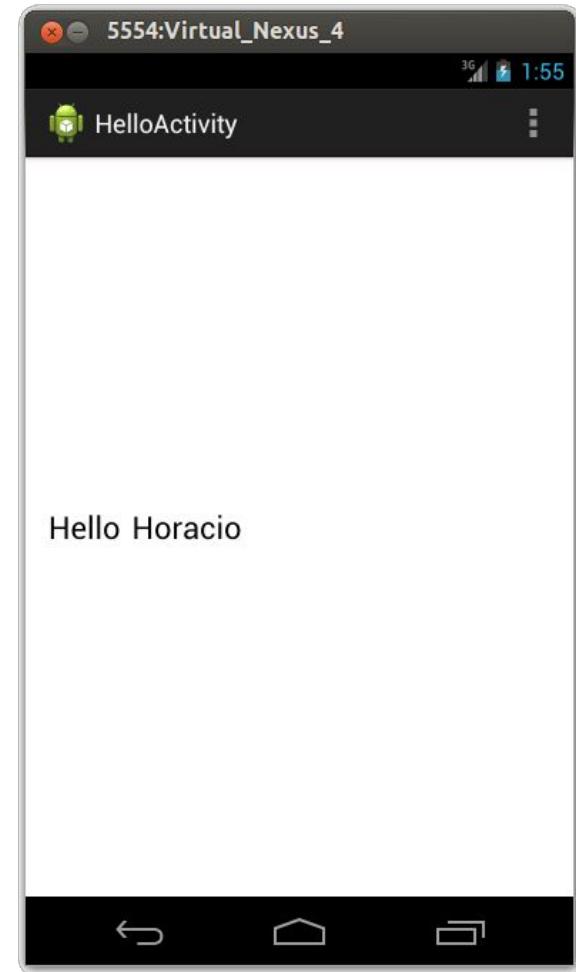
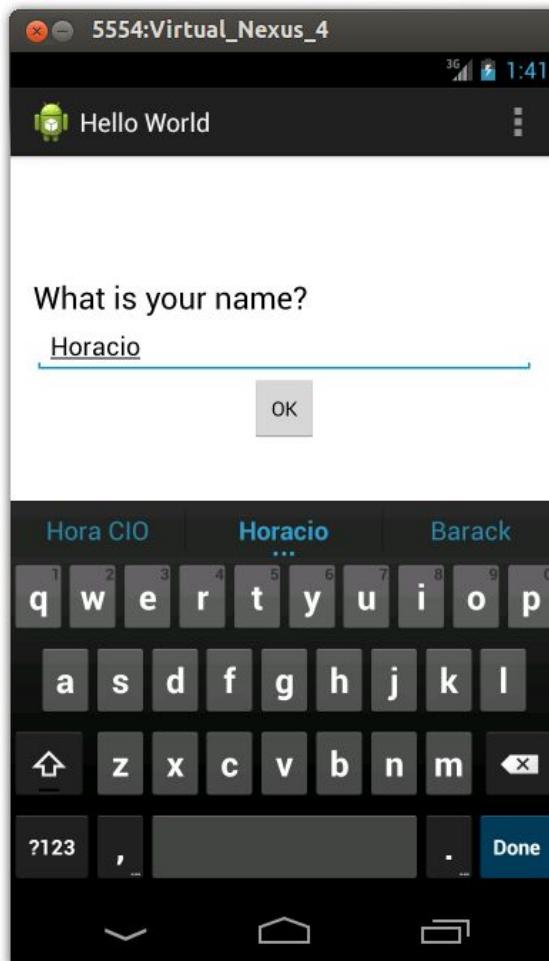
```
<activity android:name=".HelloActivity" >
    <intent-filter>
        <action android:name=
            "org.lostinbrittany.teaching.android.tb.helloandroid_with_intents.HELLO" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```



# Initiation à Android



## LiveCoding : Hello World avec Intents et Events



# Initiation à Android



[MERCI]



Horacio Gonzalez

