Développement mobile : **Android**

Cours 1 — Introduction et préparation • 23/09/2026

Andrew O’SHEI

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Qui suis-je ?

Poste actuel :

Innovation Architect | AI & Robotic - Sogeti Labs France

Éducation :

Master Informatique - Université Paris 8
Conduite de Projets Informatique

Licence Informatique - Université Paris 8
Micro-Informatique et Systèmes Embarqués

Bachelor of Art - Arizona State University
Interdisciplinary Art and Sciences

OBJECTIF DU SEMESTRE

9 séances pour publier votre app

Projet fil rouge

Interface, navigation et état, caméra, API et BD Room.

Comprendre l'OS

Organisation du code, gestion du cycle de vie, notifications et services.

Dev avec l'IA

Demander, lire et vérifier le code tout en gardant la maîtrise technique.

Préparer la sortie

Tests via GitHub Actions et publication sur Play Console.

ORIGINES & ÉCOSYSTÈME

Les origines d’Android

2003 — Création d’Android, Inc. Le projet vise initialement les appareils photo numériques.

2005 — Google rachète Android, Inc. et poursuit le développement de la plateforme mobile.

2006 — Les premiers prototypes explorent notamment des téléphones à clavier, sans écran tactile.

2007 - Apple lance la première génération d'iPhone.

2007 — Annonce d’Android et de l’Open Handset Alliance en novembre ; publication d’un premier SDK pour les développeurs.

2008 — Le HTC Dream / T-Mobile G1 devient le premier téléphone Android commercialisé.

2011 - Android devient le système d'exploitation pour smartphones le plus vendu au monde.

Aujourd’hui - Un écosystème de téléphones, tablettes, montres, téléviseurs et voitures.

ARCHITECTURE

Pourquoi Android ?

Noyau Linux

Fondation technique robuste assurant la gestion matérielle et la sécurité.

Modèle AOSP

Socle entièrement open source, distinct des services propriétaires Google.

Personnalisation

Totale liberté d'adaptation pour les fabricants et les développeurs.

Le passé : Java & XML

Android 1.0 (2008)

- ✗ Code verbeux et répétitif.
- ✗ Interfaces déclarées séparément en XML.
- ✗ Risque élevé de NullPointerException.

Aujourd'hui : Kotlin & Compose

Approche « Kotlin-first » (2019)

- ✓ **Syntaxe concise :** réduction du code boilerplate.
- ✓ **Jetpack Compose :** UI 100% Kotlin unifiée.
- ✓ **Null-safety intégrée :** détection à la compilation.

LE LANGAGE

Pourquoi Kotlin ?

Interopérabilité Totale

Appel de code Java depuis Kotlin, et inversement, sans friction.

Syntaxe Expressive

Réduit drastiquement le code inutile ("boilerplate").

Sécurité Avancée

Distinction stricte entre types nullables et non nullables.

Asynchronisme (Coroutines)

Simplifie l'exécution des tâches lourdes hors du thread principal.

SYNTAXE DE BASE

Variables : val vs var

val (Lecture seule)

Réassignation interdite après initialisation. Privilégié par défaut.

```
val name = "Alice"
name = "Bob" // ❌ Erreur de compilation
```

var (Mutable)

Valeur réassignable à tout moment. Utile pour les compteurs ou états.

```
var age = 35
age = 55 // ✅ OK
```

Inférence de type : Kotlin déduit le type automatiquement (ex: Int pour 35). Spécification explicite possible : var age: Int = 35

SÉCURITÉ

Types et Nullabilité

Types Non Nullables (Par Défaut)

Prévient la redoutée **NullPointerException** dès la compilation.

```
var strNonNullable: String = "Alice"
strNonNullable = null // ❌ Erreur de compilation
```

Types Nullables

Ajoutez un **?** après le type pour autoriser explicitement la valeur **null**.

```
val strNullable: String? = null // ✅ OK
```

CONTRÔLE DE FLUX

if / else & when

Classique : if / else

```
val score = 85

if (score ⩾ 90) {
    println("Grade A")
} else if (score ⩾ 80) {
    println("Grade B")
} else {
    println("Grade C")
}
```

Puissant : when

```
val label = when {
    score < 0      → "Négatif"
    score == 0     → "Zéro"
    score in 1..10 → "Entre 1 & 10"
    else           → "Plus grand que 10"
}
println(label)
```

Architecture Android

Comprendre les composants essentiels d'une application

COMPOSANTS

Fondations Visuelles & Navigation

Activity

- Héberge l'interface utilisateur.
- Peut contenir plusieurs écrans Compose.
- Soumise à un cycle de vie strict.

Intent

- Message asynchrone décrivant une action.
- Ouvre d'autres Activities.
- Exemple : lancer l'appareil photo, partager un lien.

CYCLE DE VIE

États d'une Activity

`onCreate()`	Initialisation de l'Activity.
`onStart()`	L'Activity devient visible à l'écran.
`onResume()`	**L'Activity passe au premier plan, active (focus).**
`onPause()`	Quitte l'état actif, mais peut rester visible.
`onStop()`	L'Activity n'est plus visible pour l'utilisateur.
`onDestroy()`	Destruction (non garanti si le processus est tué).

 ·

Cycle de vie de l'Activity

onCreate()
Initialisation de l'Activity.

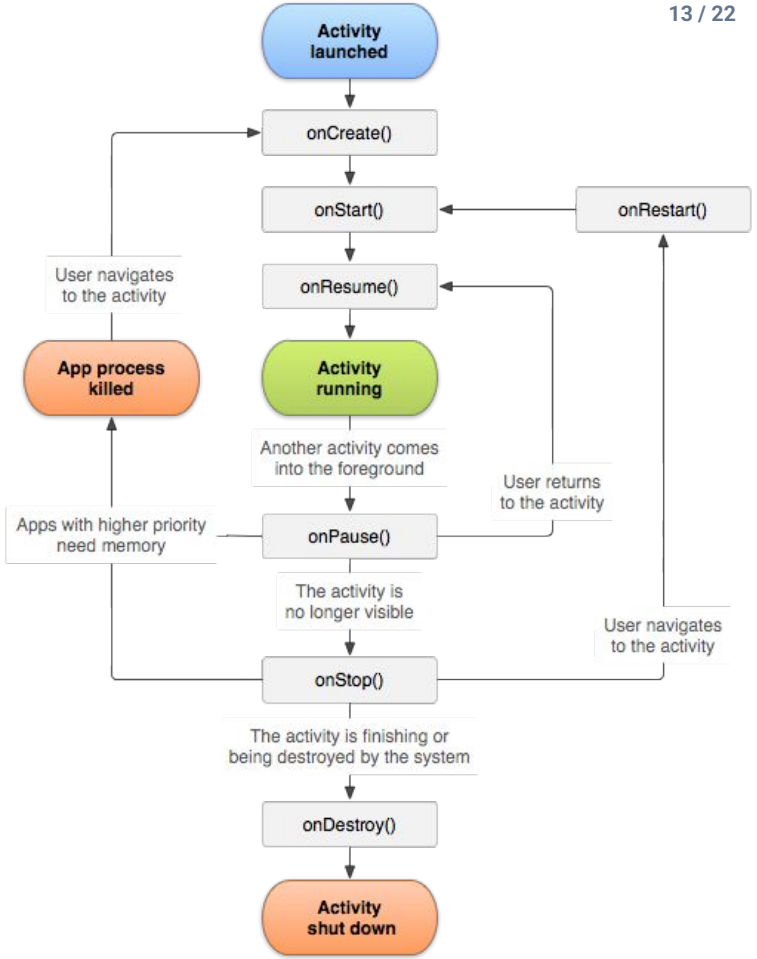
onStart()
L'Activity devient visible.

onResume()
L'Activity passe au premier plan, active.

onPause()
Elle quitte l'état actif ; peut rester visible.

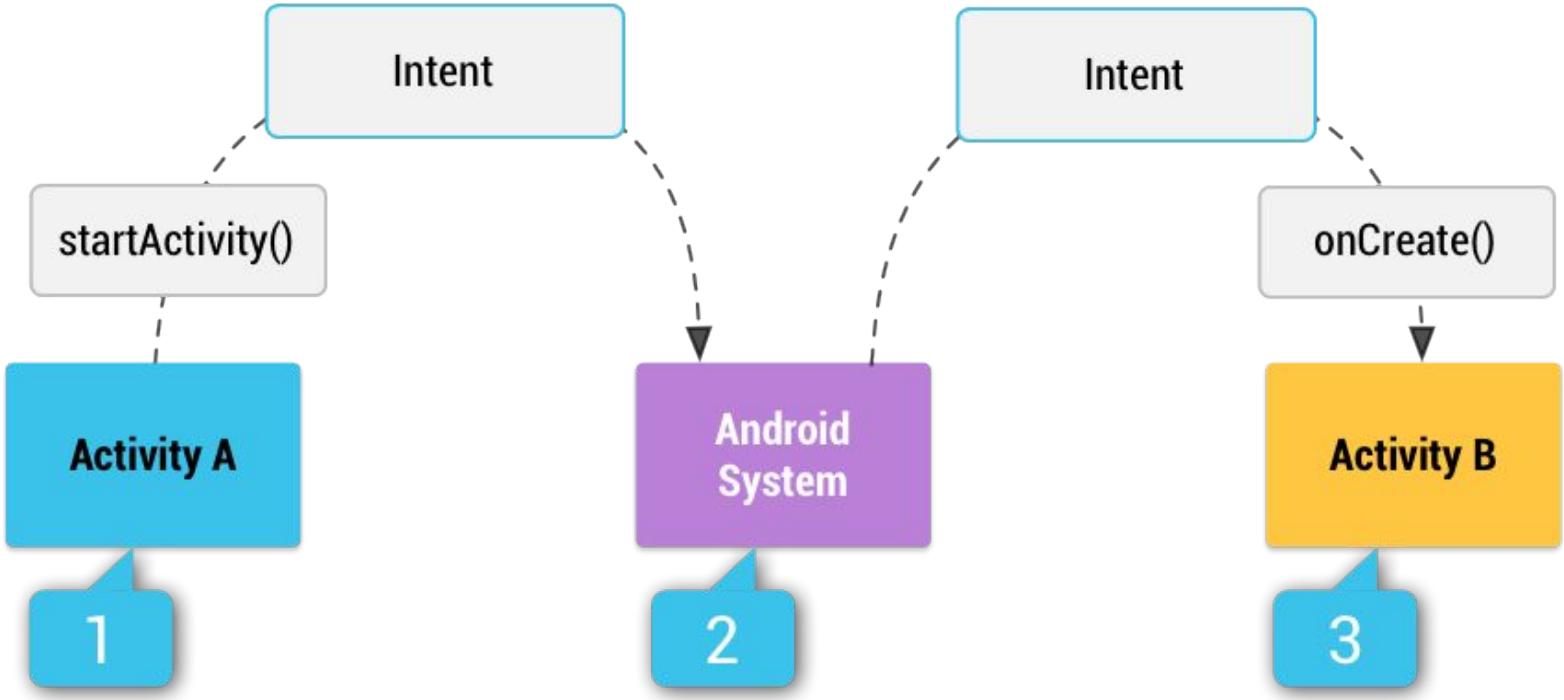
onStop()
L'Activity n'est plus visible.

onDestroy()
Destruction ; non garanti si le processus est tué.



Source: https://developer.android.com/guide/components/activities/activity-lifecycle

Intent



Source: https://developer.android.com/guide/components/intents-filters

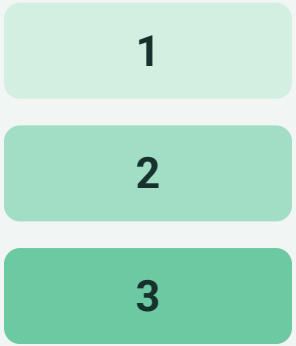
INTERFACE UTILISATEUR

Organiser l’interface avec Compose

Row

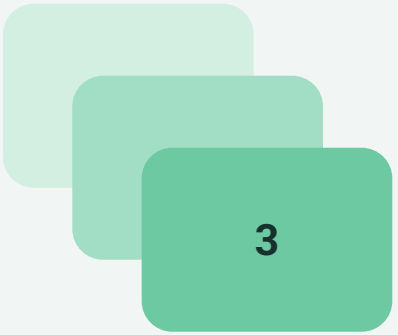
Sur une ligne

Column



En colonne

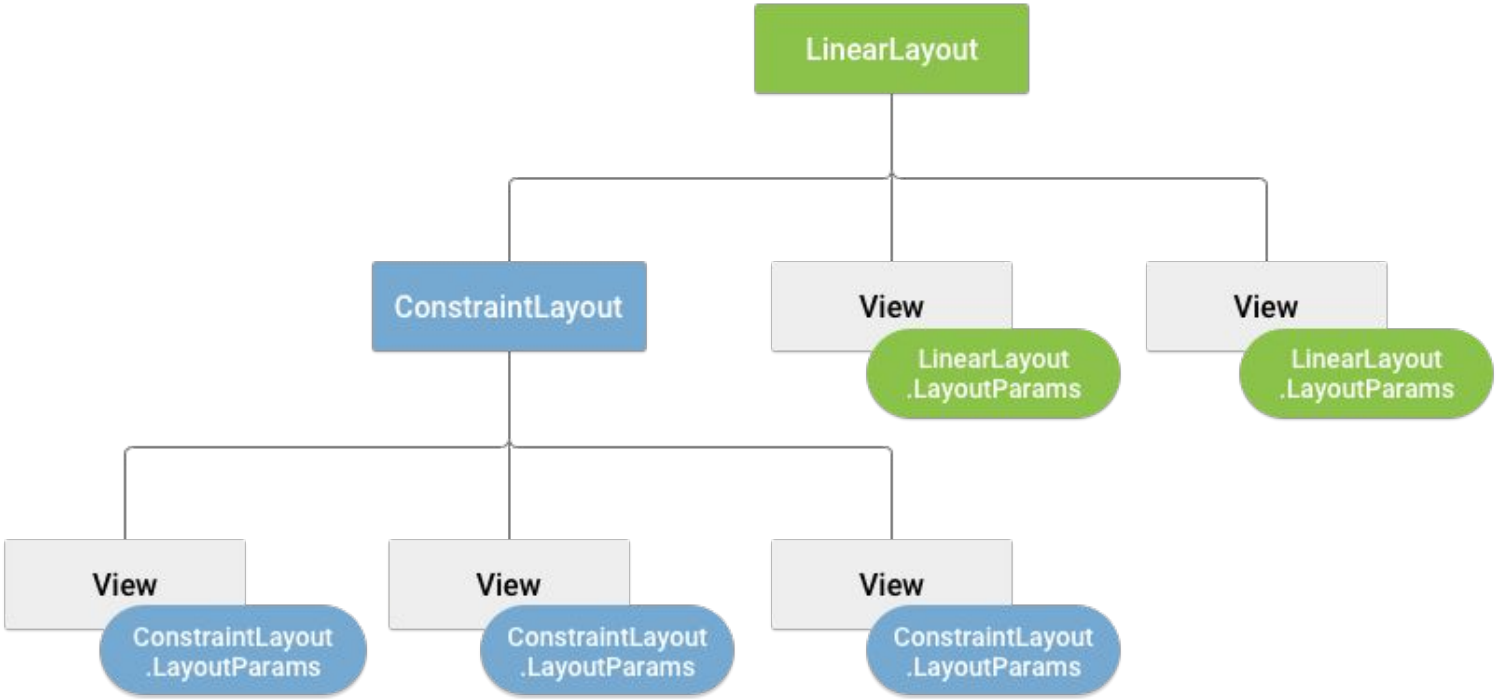
Box



En superposition

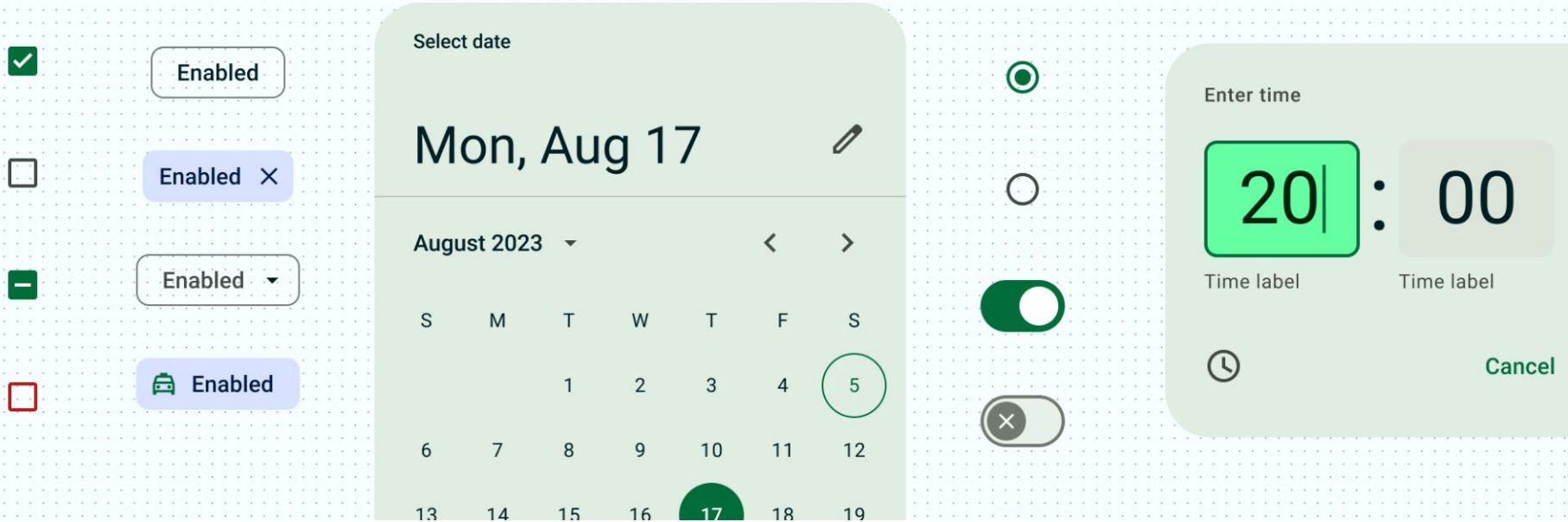
Source : documentation Jetpack Compose

Views et XML : comprendre le modèle historique



Source: https://developer.android.com/develop/ui/views/layout/declaring-layout

Composants visuels Material



Source: https://developer.android.com/design/ui/mobile/guides/components/material-overview

SYSTÈME

Autres composants Android

Services

Composants sans interface, ils ne créent pas automatiquement un thread de travail.
Exemples : lecture audio via un service au premier plan, tâches différées avec WorkManager.

Broadcast Receivers

Réagissent à des événements diffusés par Android ou par une application.
Exemples : événement de batterie faible, réception soumise aux restrictions d'Android.

Content Providers

Exposent un accès structuré à des données, sous contrôle des permissions.
Permettent notamment le partage de données entre applications.
Exemple : l'application Contacts fournit des données de contact à d'autres applications.

Avant la séance 2 : installer Android Studio

Télécharger la version stable d’Android Studio →
https://developer.android.com/studio

Linux

1. Extraire l’archive, puis ouvrir android-studio/bin
2. Lancer ./studio.sh -> suivre le Setup Wizard (SDK et outils)

Windows

1. Lancer le .exe ; suivre le Setup Wizard (SDK et outils)

macOS

1. Ouvrir le .dmg et glisser Android Studio dans Applications
2. Lancer Android Studio ; suivre le Setup Wizard (SDK et outils)

Avant la séance 2 : Git, GitHub et assistant IA

1. Git & GitHub

Git : installer Git et créer ou vérifier son compte GitHub.

2. GitHub Copilot

Copilot : faire valider son statut dans GitHub Education, puis activer Copilot Student (gratuit, avec limites d'usage).

3. Android Studio

Dans Android Studio : Settings > Plugins > Marketplace ; installer GitHub Copilot et se connecter à GitHub.

Autre option : ouvrir Gemini / Agent dans Android Studio et se connecter avec son compte Google.

À préparer chez vous

Vérifier l'accès à un assistant. La prise en main guidée aura lieu en séance 2.

 ·

À faire chez vous : tester un premier projet

ÉTAPE 01

Nouveau projet

Sélectionner **New Project**

ÉTAPE 02

Type d'activité

Sélectionner **Empty Activity (Kotlin / Compose)**

ÉTAPE 03

Version SDK

Choisir Minimum SDK : **API 30 (Android 11)**

ÉTAPE 04

Nom du projet

Nommer votre projet : **BonjourAndroid**

ÉTAPE 05

Finalisation

Cliquer sur **Finish** - attendre la synchronisation Gradle

ÉTAPE 06

Exécution

Choisir un émulateur ou un téléphone, puis **Run** : vérifier que l'app s'affiche.

 ·

Ressources

- Informations générales sur tout ce qui concerne Android
  - https://developer.android.com/
- Tutoriel - Créer votre première application Android
  - https://developer.android.com/codelabs/basic-android-kotlin-compose-first-app
- Tutoriels Android
  - https://codelabs.developers.google.com/?product=android
- Compilateur Kotlin en ligne
  - https://play.kotlinlang.org/