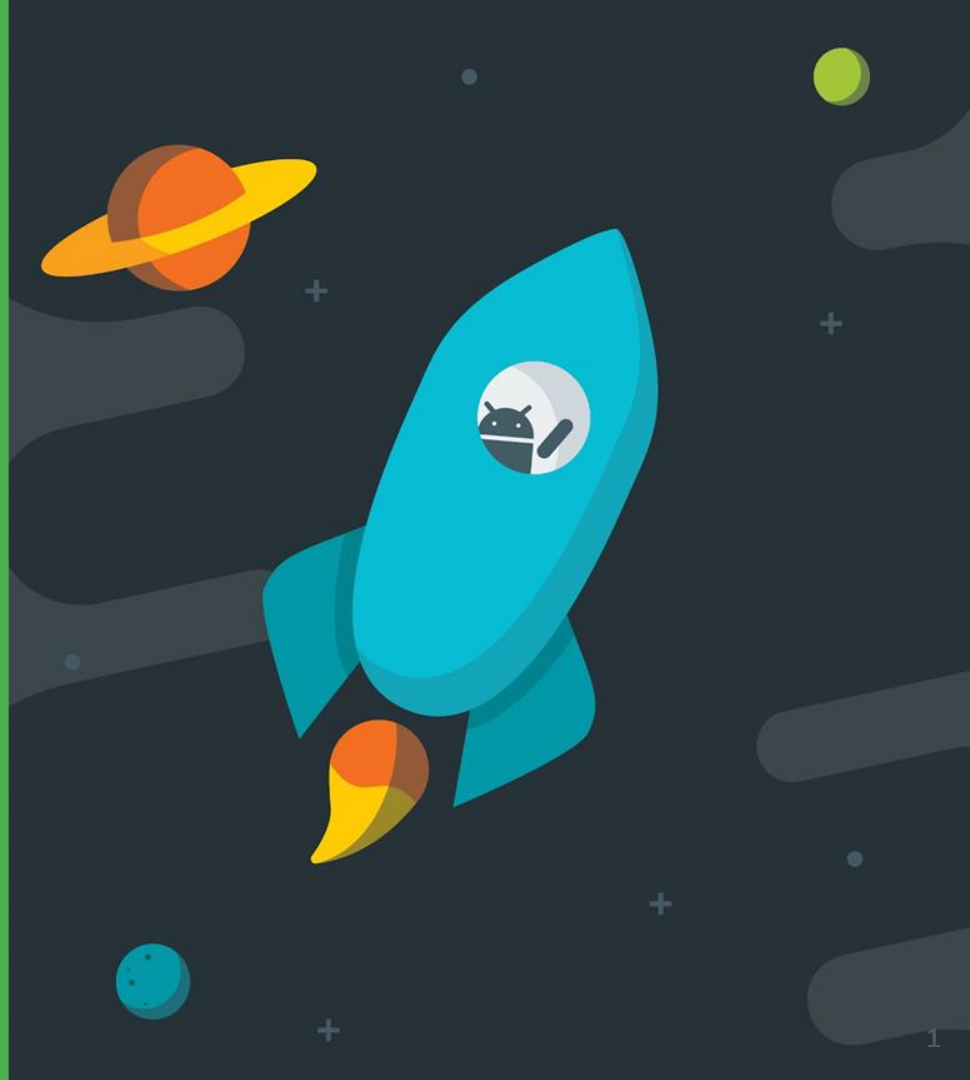
UNITÉ D ’ENSEIGNEMENT (UE) :

DÉVELOPPEMENT MOBILE

CHAPITRE I :

Introduction au Développement d’Applications Mobiles & ANDROID



État de l’ Art des Applications Mobiles

Etat de l’ Art des Applications Mobiles

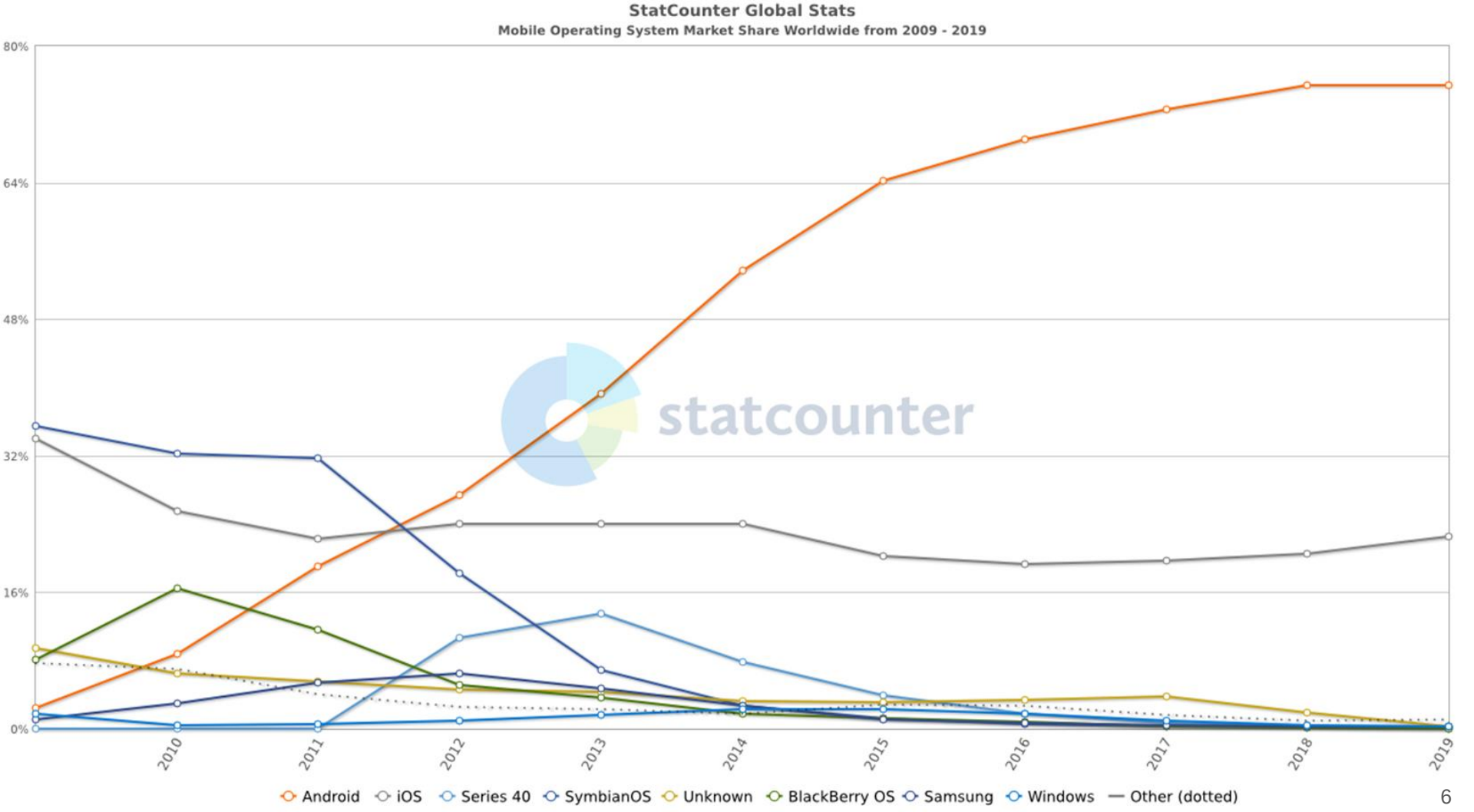
- Utilisation des mobiles
  - 6,9 milliards de terminaux vendus fin 2014
  - Taux de couverture atteint environ 95% de la population mondiale
- Smartphones
  - 1,76 milliards d’utilisateurs de smartphones
    - Augmentation de 23% par rapport à 2013
  - 395 millions de Go de données transférées
    - Augmentation de 48% par rapport à 2013
- En Tunisie
  - Plus de 12,63 millions de cartes SIM vendues (115% de taux de pénération)
  - 36% des Tunisiens abonnés à la téléphonie mobile ont des smartphones (2015)

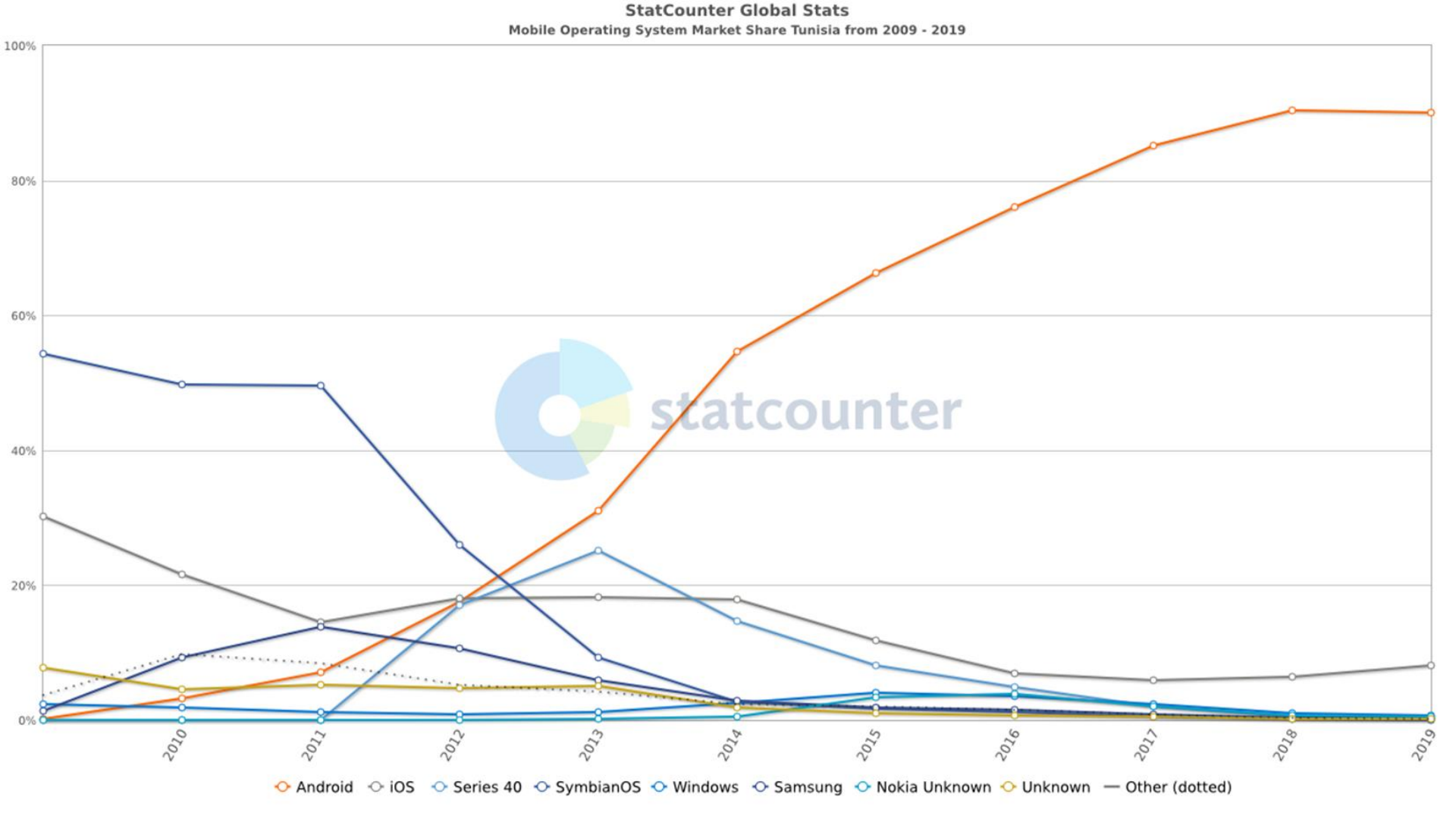
Etat de l’ Art des Applications Mobiles

- Téléphone portable (1983 par Motorola)
- Assistants numériques personnels (PDA) en 1990
  - –Agenda, carnet d’adresses, bloc notes
  - –Synchronisation des données avec un PC
- Smartphone = Téléphone portable + PDA
  - – En 2001 par Sagem
  - – Essor en 2005 avec la sortie de l’iPhone, et le rachat d’Android par Google
  - – Mini-ordinateur portable
  - – Applications téléchargeables

Etat de l’ Art des Applications Mobiles

- Ancienne génération
  - Symbian de Nokia
  - Blackberry OS de RIM
  - Windows Mobile de Microsoft
  - Bada de Samsung
- OS Tactiles
  - iOS de Apple
  - Android de Google
  - Windows Phone de Microsoft


StatCounter Global Stats
Mobile Operating System Market Share Worldwide from 2009 - 2019
80%
64%
48%
32%
16%
0%
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
statcounter
Android
iOS
Series 40
SymbianOS
Unknown
BlackBerry OS
Samsung
Windows
Other (dotted)


StatCounter Global Stats
Mobile Operating System Market Share Tunisia from 2009 - 2019
100%
80%
60%
40%
20%
0%
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
statcounter
Android
iOS
Series 40
SymbianOS
Windows
Samsung
Nokia Unknown
Unknown
Other (dotted)

SPÉCIFICITÉS DU DÉVELOPPEMENT MOBILE : Contraintes Matérielles

- Mémoire limitée / Processeur lent
  - Les smartphones souffrent d'un temps de chargement long
  - Bien faire attention au type de public ciblé
  - Éviter les traitements complexes, gourmands en mémoire
- Capacité de stockage limitée
  - Quelques smartphones très haut de gamme (iPhone) ont une mémoire de 128Go ou plus
  - Mais plusieurs modèles d'entrée de gamme n'ont que 4Go (majorité de 16G0)
  - Penser à s'orienter plus vers le stockage sur le cloud.

SPÉCIFICITÉS DU DÉVELOPPEMENT MOBILE : Contraintes Matérielles

- Autonomie
  - Éviter les applications gourmandes en énergie (RA,…)
- Taille d'écran réduite et variable
  - Réduction du contexte de l'application par rapport à un écran
  - Tailles varient d'un appareil à un autre (smartphone, tablette, phablette,...)
  - Exploiter tout l'espace fourni de manière optimale
  - S'orienter dans le développement vers le responsive-design
  - Penser à des choix d'IHM qui facilitent la navigation (un header fixe, utilisation des icônes à la place des mots...)

SPÉCIFICITÉS DU DÉVELOPPEMENT MOBILE : Contraintes Matérielles

- Problèmes de Connectivité
  - Problèmes de connectivité dus à la mobilité
  - Utilisation des réseaux 3G/4G, donc payants
  - Penser à un mode offline pour vos applications/sites
  - Attention aux mises à jour automatiques
- Téléchargement plus lent
  - Connexions internet plus lentes, latence réseau et mémoire et processeur limités
  - 80% des utilisateurs ne veulent pas utiliser leur téléphone pour surfer sur le web
  - Utiliser des technologies qui facilitent le chargement des pages, réduire la taille des images, le nombre de fichiers et les traitements côté client

SPÉCIFICITÉS DU DÉVELOPPEMENT MOBILE : Contraintes Matérielles

- Manipulation plus délicate
  - Utilisation délicate du clavier tactile, sujette à beaucoup d'erreurs de frappe
    - Minimiser le nombre de champs de texte
    - Favoriser les champs préremplis
    - Fonctionnalités d'auto-complete, correction d'orthographe...
- Clics invalides, à cause de l'utilisation des doigts (FAT FINGER)
  - Attention à la taille et proximité des éléments cliquables
- Absence de l'effet de survol (hover)
  - Besoin de plus de liens et de boutons
  - Utiliser des conventions de conception mobile, comme balayer l'écran (swipe) ou secouer le téléphone (shake)

SPÉCIFICITÉS DU DÉVELOPPEMENT MOBILE : Monétisation

Possibilité pour un éditeur de vendre son application via les plateformes de téléchargement (App Store, Google Play..)

- Application Payante
  - Applications peu chères ont un grand succès
  - En général 70% pour le développeur, 30% pour la plateforme
  - Le prix varie selon l'OS. Mais, Certains utilisateurs sont réticents à acheter des applications
- Version d'Essai
  - Fournir une version limitée, qui donne une idée à l'utilisateur de la qualité de l'application
  - Bien adapté pour les jeux
  - Problème de maintenabilité pour les développeurs

SPÉCIFICITÉS DU DÉVELOPPEMENT MOBILE : Monétisation

- Sponsoring
  - Incite les utilisateurs à effectuer une action en échange d'un bien virtuel
  - Bien adaptée aux jeux
  - Public bien ciblé, donc chances de gagner des revenus plus grandes
- Publicité
  - Sous forme de bandeau
  - Profitable pour les développeurs qui veulent se faire connaître et avoir des revenus
  - Les publicités rémunèrent en général au clic à difficile d'estimer le revenu
  - La publicité peut altérer l'UX (lenteur, appui par erreur...)
  - La publicité doit être bien ciblée

SPÉCIFICITÉS DU DÉVELOPPEMENT MOBILE : Monétisation

- Freemium
  - Proposer du contenu virtuel contre de l'argent
  - Peuvent être implémentés sur une applications payante ou gratuite
  - Une application de qualité donne envie aux utilisateurs d'y revenir et d'acheter du contenu
  - Le contenu virtuel doit valoir son prix
  - L'achat doit être simple et rapide

APPS MOBILES :
VENTES ET REVENUS PAR PLATEFORME

Conférence AppDays, les 7 et 8 novembre 2013

Revenu moyen par éditeur

21 000 $
Apple App Store

6 000 $
Google Play

2 222 $
Microsoft Windows Store

Revenu moyen par app

Microsoft Windows Store
625 $

Apple App Store
4 000 $

Google Play
1 125 $

Revenu par téléchargement

Microsoft Windows Store
0.15 $

Apple App Store
0.10 $

Google Play
0.01 $

Nombre de téléchargements par app

Nombre moyen d'apps publiées pour chaque éditeur

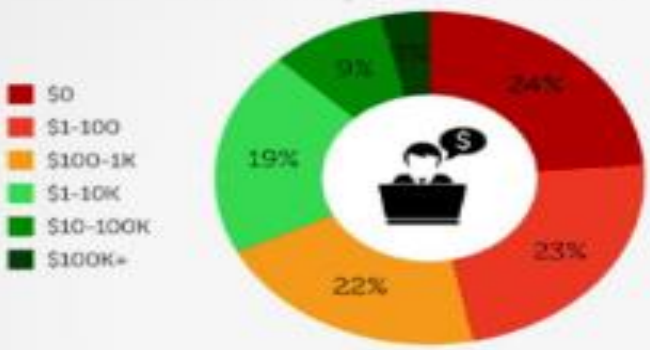
THE APP ECONOMY IS A WINNER TAKES ALL GAME

How many app businesses are sustainable?

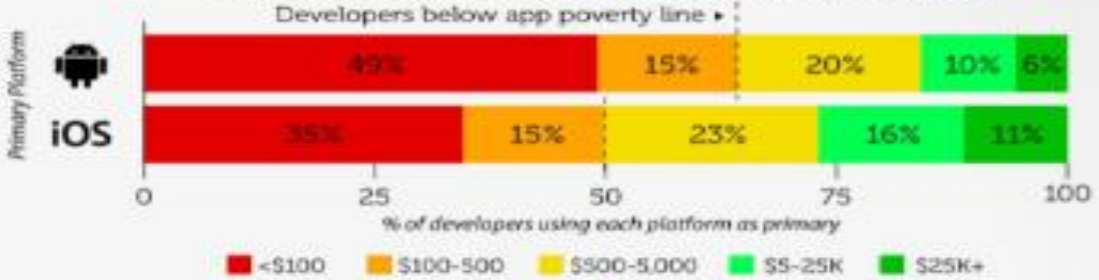
LARGEST EVER 10,000+ APP DEVS APP DEV SURVEY

Monthly App Revenues

% of all developers interested in revenues



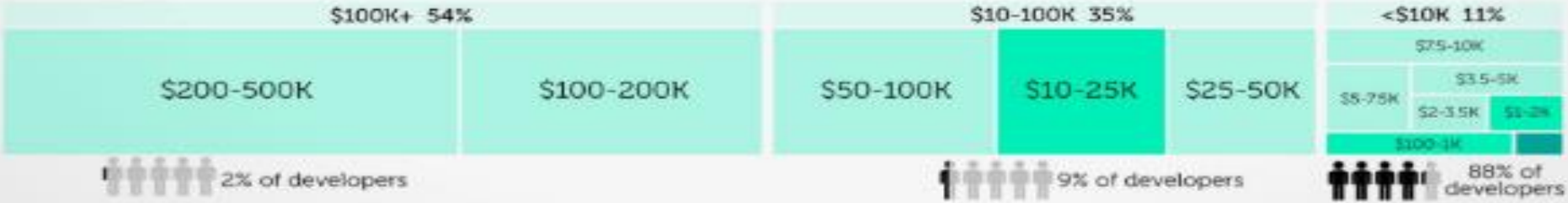
Revenue Distributions - Android vs. iOS First



1.6% of developers have an app earning >$500k per month. Together they earn multiples of the other **98.4%** combined.

How App Revenues are Split

Here's how revenues are split amongst those earning < $500k per month. Box area is proportional to the fraction of total revenues earned by each group.



Source: Developer Economics: State of Nation Q3 2014 | www.DeveloperEconomics.com/go |

Catégories d'Applications Mobiles

- Applications Natives
  - Applications écrites dans un langage de programmation spécifique à une plateforme particulière.
  - Performance accrue et haut degré de fiabilité
  - Ont accès aux fonctionnalités du téléphone (caméra, liste de contacts...)
  - Utilisables sans connexion internet
  - Mais, plutôt chères à développer, car associée à un OS, donc besoin de dupliquer les versions pour d'autres plateformes

Catégories d'Applications Mobiles

- Applications Web
  - Sites web adapté pour une utilisation sur appareil mobile
  - Accessibles via le navigateur web de l'appareil mobile
  - Fonctionnement en ligne, sans installation
  - Modification à un seul endroit, et tous les utilisateurs le voient
  - Besoin de rendre votre site web « mobile-friendly » pour un meilleur usage
  - Utilisation de HTML, JavaScript, CSS...

Catégories d'Applications Mobiles

- Applications Hybrides
  - Exposer le contenu de sites web existants sous forme d'application
  - Enlever le navigateur de l'expérience utilisateur: installée comme toute application native
  - Ont plus accès aux fonctionnalités du téléphone que les applications purement web, mais restent un peu limités
  - Développement en HTML, JavaScript et CSS, puis enveloppées dans une application native
  - 2 majeures plateformes:
    - Phone Gap et sa version open source Cordova
    - Appcelerator Titanium

Android Ecosystem

Qu'est-ce qu'Android ?

- Système d'exploitation mobile basé sur le noyau Linux
- Interface utilisateur pour écrans tactiles
- Utilisé sur plus de 80% des smartphones
- Alimente des appareils tels que des montres, des téléviseurs et des voitures
- Plus de 2 millions d'applications Android dans Google Play Store
- Hautement personnalisable pour les appareils / par les fournisseurs
- Open source

Interaction avec l'utilisateur

- Gestes tactiles : balayage, tapotement, pincement
- Clavier virtuel pour les caractères, les chiffres et les emoji
- Prise en charge du Bluetooth, des contrôleurs USB et des périphériques

Android et capteurs

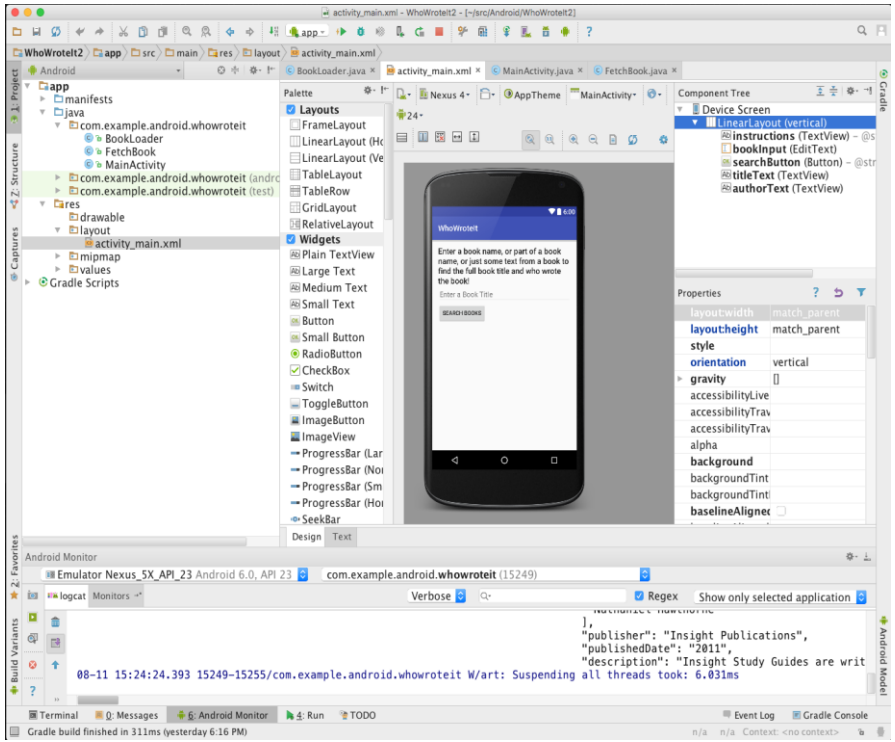
Les capteurs peuvent découvrir l'action de l'utilisateur et réagir

- Le contenu de l'appareil pivote selon les besoins
- Durant une marche ajustement de la position sur la carte
- L'inclinaison dirige une voiture virtuelle ou contrôle un jouet physique
- Se déplacer trop vite désactive les interactions de jeu
- …

Android Software Developer Kit (SDK)

- Outils de développement (débogueur, moniteurs, éditeurs)
- Bibliothèques (cartes, wearables)
- Périphériques virtuels (émulateurs)
- Documentation (developers.android.com)
- Exemple de code

Android Studio



- IDE Android officiel
- Développer, exécuter, déboguer, tester et empaqueter des applications
- Moniteurs et outils de performance
- Appareils virtuels
- Vues du projet
- Éditeur de mise en page visuelle

Architecture de la plate-forme 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

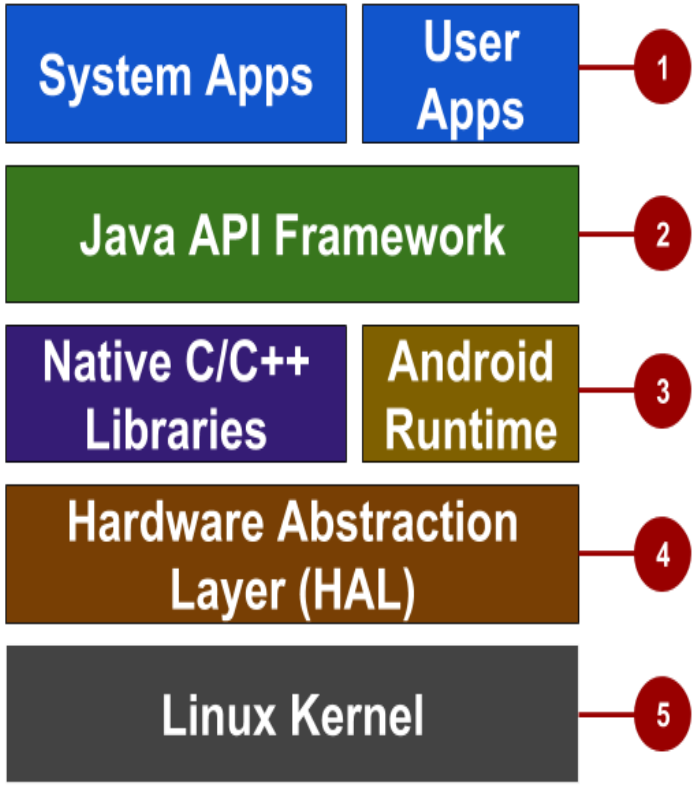
Pie
9.0

10

android

Pile Android

1. Applications système et utilisateur
2. API du système d'exploitation Android dans le framework Java
3. Exposez les API natives ; exécuter des applications
4. Exposer les capacités matérielles de l'appareil
5. Noyau Linux



Applications système et utilisateur

- Les applications système n'ont pas de statut spécial
- Les applications système offrent des fonctionnalités clés aux développeurs d'applications
- Exemple:

Votre application peut utiliser une application système pour envoyer un message SMS.

Java API Framework

L'ensemble des fonctionnalités du système d'exploitation Android est à votre disposition via des API écrites en langage Java.

- Afficher la hiérarchie des classes pour créer des écrans d'interface utilisateur
- Gestionnaire de notifications
- Gestionnaire d'activité pour les cycles de vie et la navigation

Android runtime

Chaque application s'exécute dans son propre processus avec sa propre instance d'Android Runtime.

C/C++ libraries

- Les bibliothèques C/C++ donnent accès aux principaux composants et services natifs du système Android.

Hardware Abstraction Layer (HAL)

- Interfaces standard qui exposent les capacités matérielles de l'appareil sous forme de bibliothèques
- Exemples : appareil photo, module Bluetooth

Linux Kernel

- Threading et gestion de la mémoire de bas niveau
- Fonctions de sécurité
- Pilotes

Développement d'applications

Qu'est-ce qu'une application Android ?

- Un ou plusieurs écrans interactifs
- Écrit en utilisant le langage de programmation Java (ou KOTLIN) et XML
- Utilise le kit de développement logiciel (SDK) Android
- Utilise les bibliothèques Android et Android Application Framework
- Exécuté par la machine virtuelle d'exécution Android (ART)

Les défis du développement Android

- Plusieurs tailles et résolutions d'écran
- **Performance** : rendez vos applications réactives et fluides
- **Sécurité** : protégez le code source et les données utilisateur
- **Compatibilité** : fonctionne bien sur les anciennes versions de la plate-forme
- **Marketing** : comprenez le marché et vos utilisateurs.

KOTLIN

- Depuis mai 2017
- L'interopérabilité : du code Kotlin peut coexister avec du code Java dans un même projet
- Concis et plus moderne que Java, des syntaxes beaucoup plus agréables.
- Moins de code à écrire : moins 20% par rapport à l'équivalent en Java
- Une courbe d'apprentissage très douce
- Sûreté et sécurité
- Coût zéro pour l'adoption du langage grâce à l'existence de convertisseurs de code Java vers Kotlin.

Blocs d'applications

- Ressources : layouts, images, strings, couleurs sous forme de fichiers XML et multimédias
- Composants : activités, services et classes d'assistance en tant que code Java
- Manifest : informations sur l'application pour le runtime
- Configuration de build : versions d'APK dans les fichiers de configuration Gradle

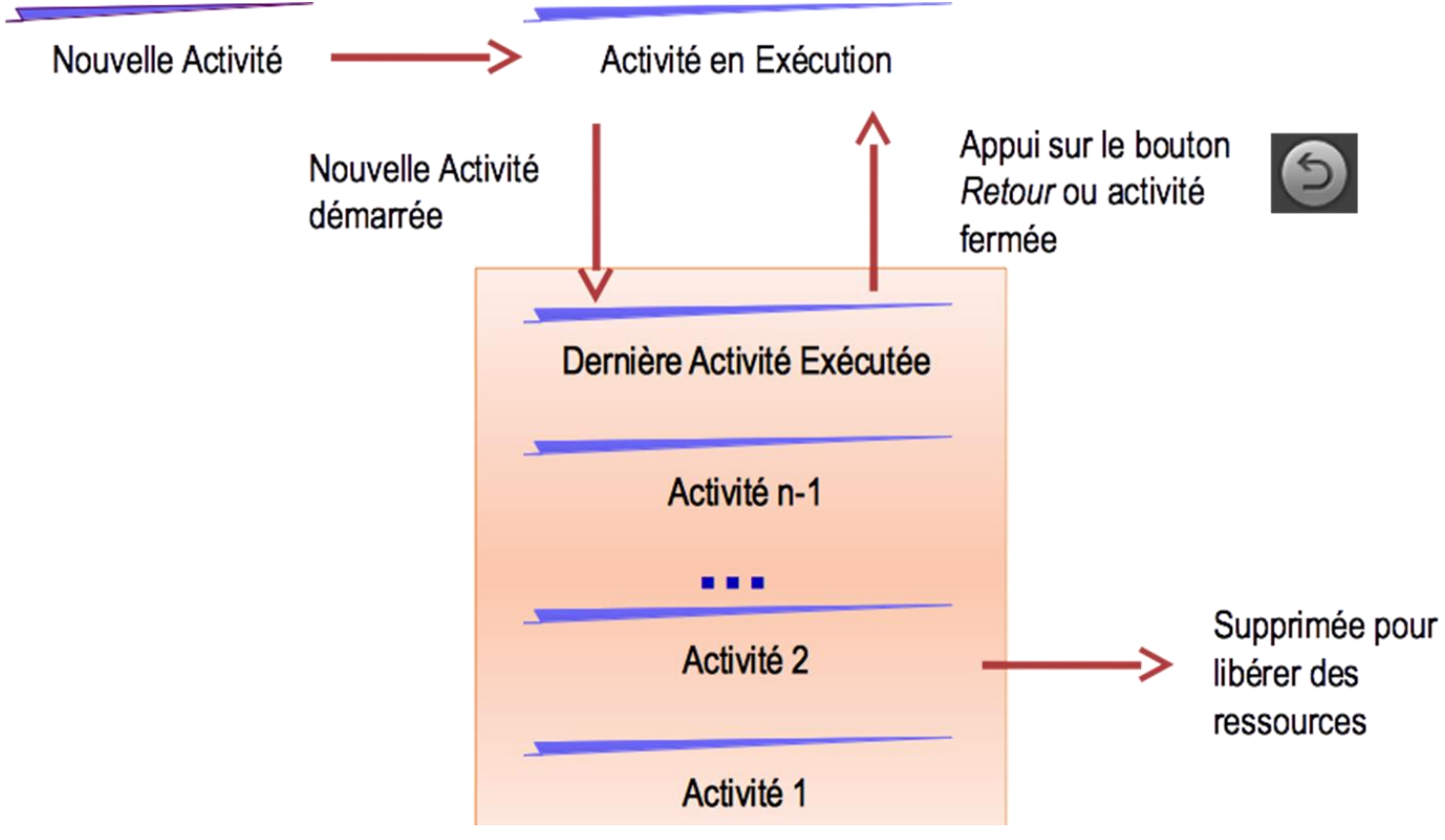
Cycle de Vie d'une Application Android

- Les composants (vues) d'une application ont un cycle de vie
  - Un début quand Android les instancie pour répondre aux Intents
  - Une fin quand les instances sont détruites
- Entre les deux, où ils peuvent être:
  - Actifs ou inactifs
  - Visibles ou invisibles

Cycle de Vie d'une Application Android

- Les activités dans une application sont gérées sous forme de Pile
- Quand une nouvelle activité démarre, elle est placée en haut de la pile et devient l'activité en exécution
- L' activité précédente reste en dessous dans la pile
- Elle ne revient au premier plan que si la nouvelle activité est fermée
- Si l'utilisateur clique sur le bouton Retour du téléphone l'activité suivante dans la pile devient active

Cycle de Vie d'une Application Android



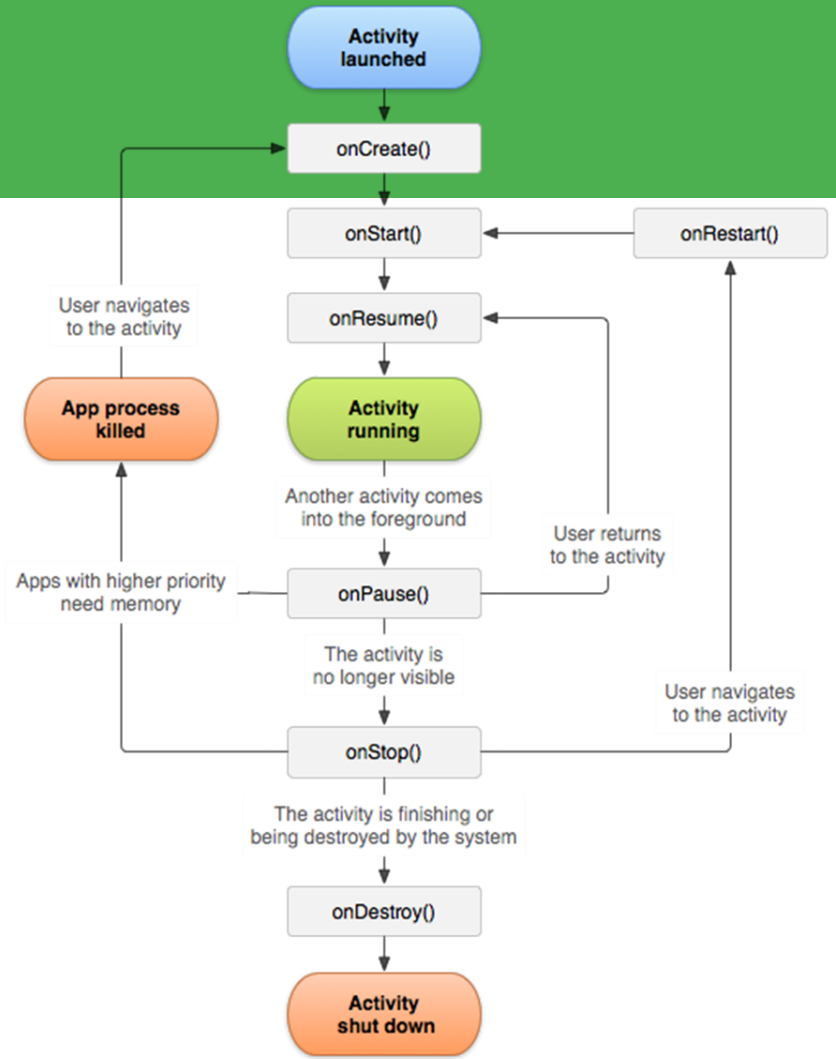
États d'une Activité

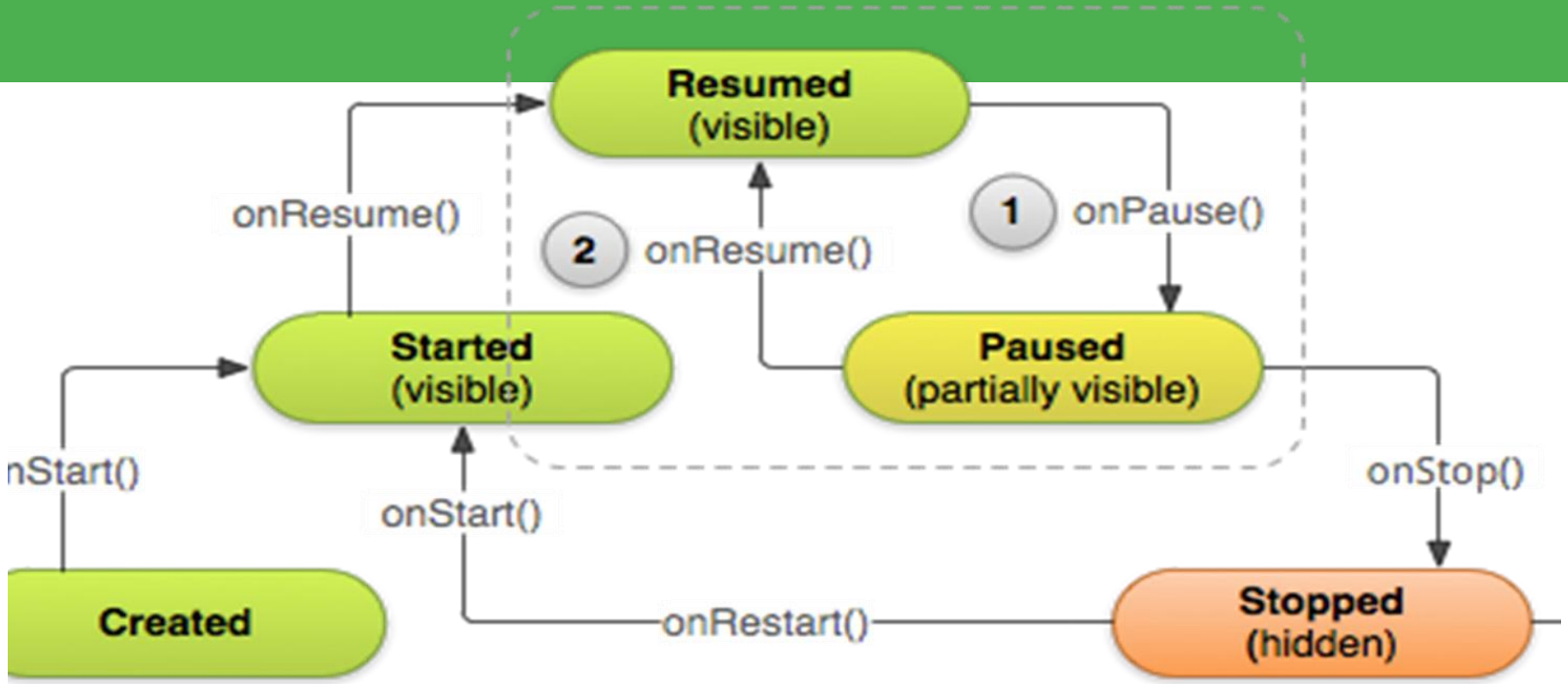
- Active ou en Exécution
  - Elle est au premier plan de l'écran (donc en haut de la pile)
  - C'est l'activité ciblée par les actions de l'utilisateur
- En Pause
  - A perdu le focus, mais est encore partiellement visible
  - Une autre activité est en haut de la pile, mais elle est soit transparente, soit ne couvre pas tout l'écran
- Arrêtée
  - Complètement recouverte par une autre activité
  - Ses informations sont encore chargées, mais elle n'est plus visible
  - Peut être tuée par le système si besoin de mémoire

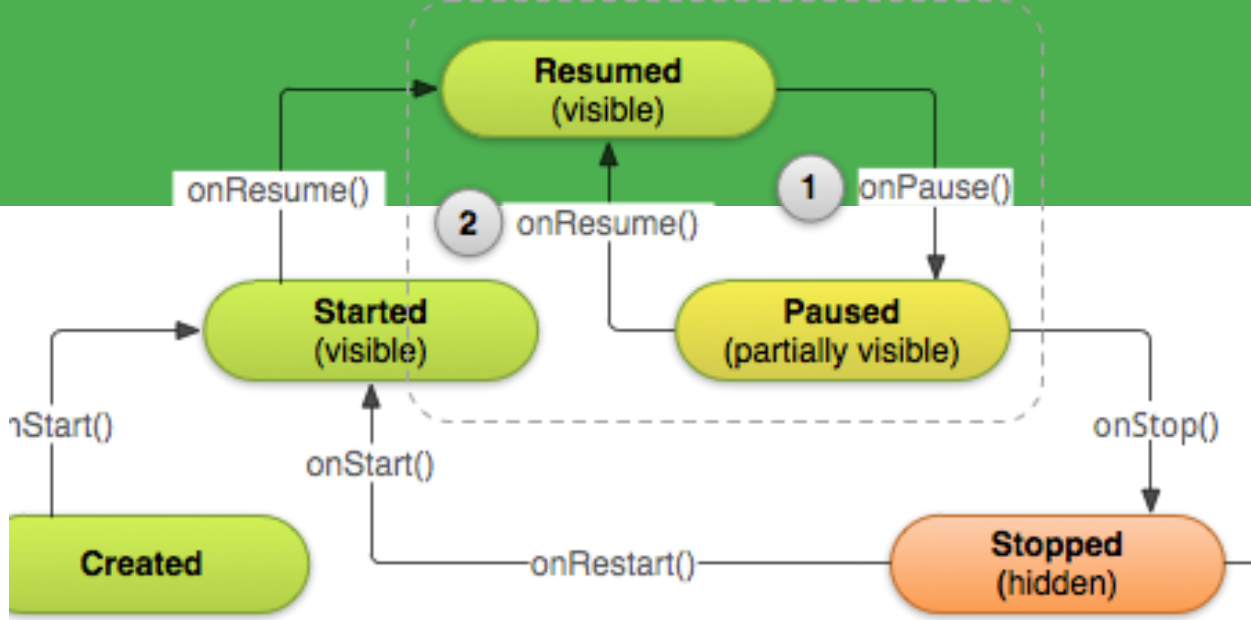
Événements du Cycle de Vie d'une Activité

Quand une activité passe d'un état à un autre, le framework Android appelle les méthodes de transition correspondantes (pattern IoC) :

- void onCreate(Bundle savedInstanceState) : obligatoire
- void onStart()
- void onRestart()
- void onResume()
- void onPause() : recommandée
- void onStop()
- void onDestroy()

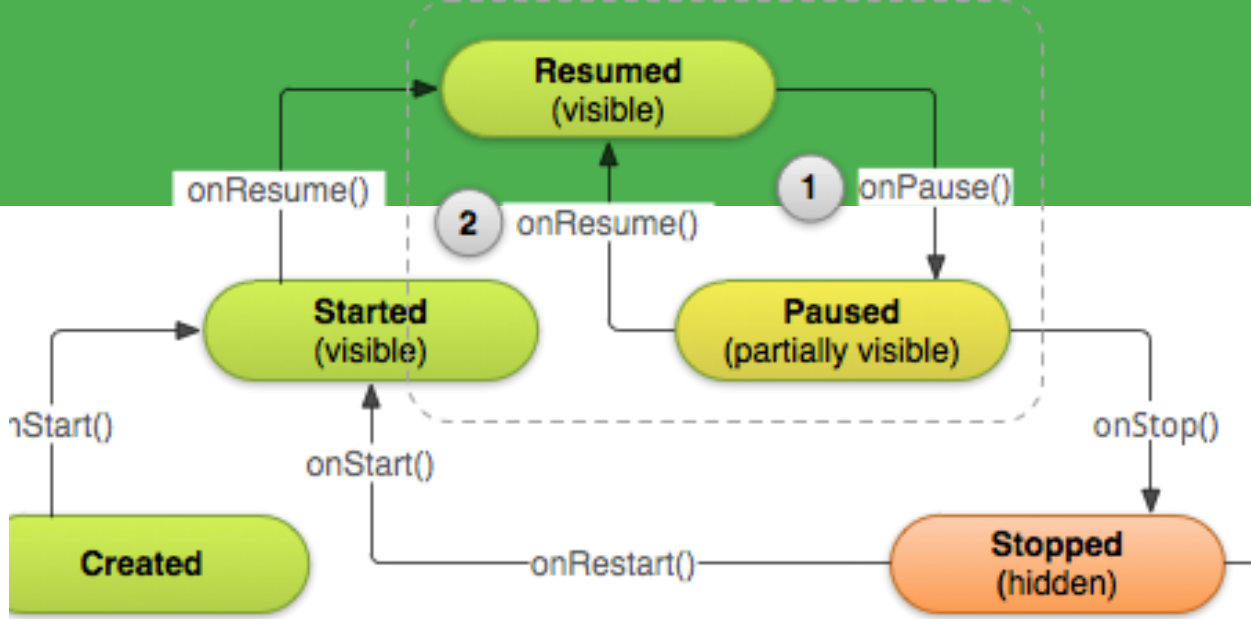

Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down


Resumed
(visible)
onResume()
2
onResume()
1
onPause()
Started
(visible)
Paused
(partially visible)
nStart()
onStart()
onStop()
Created
onRestart()
Stopped
(hidden)



onCreate

- Appelée à la création d'une activité
- Initialisation de tous les éléments
- Un bundle est passé à cette méthode, contenant l'état précédent de l'activité
- Toujours suivie de onStart

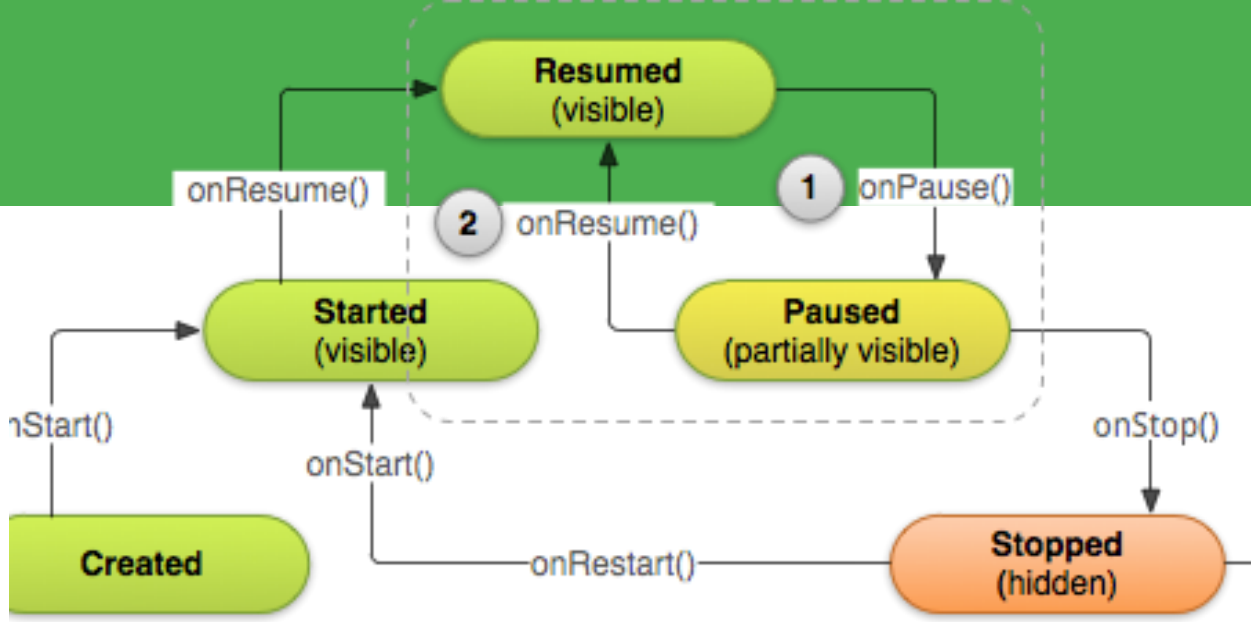


onStart

- Appelée juste avant que l'activité ne devienne visible,
- Suivie de

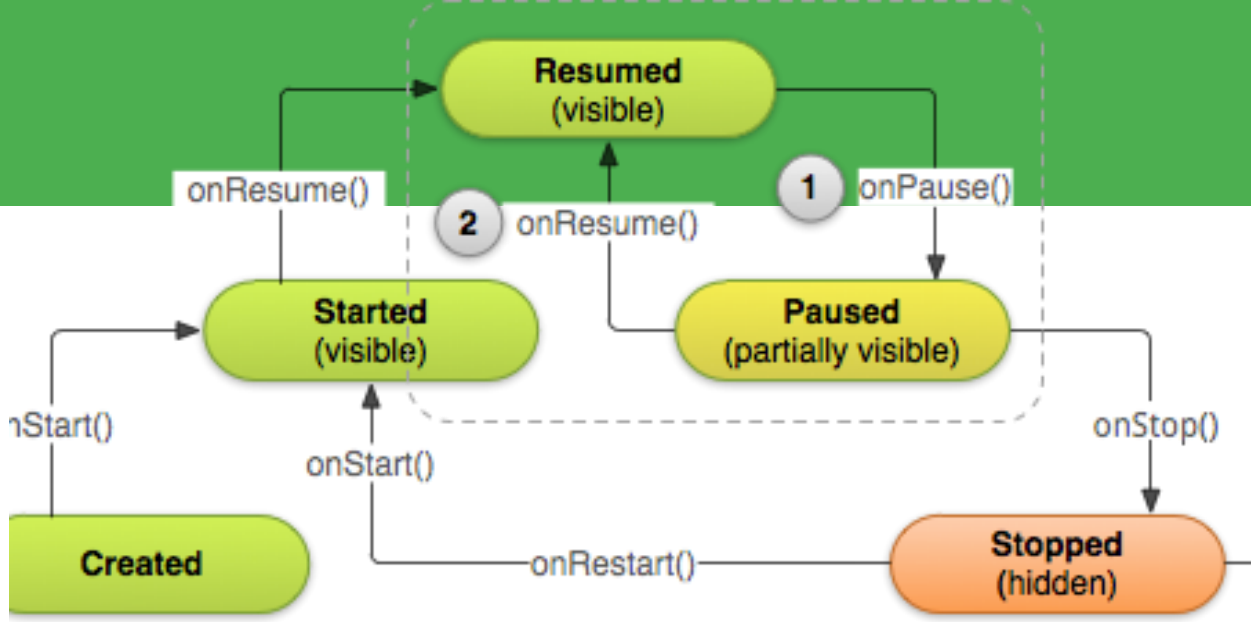
  onResume si l'activité revient en premier plan

  onStop si l'activité est cachée



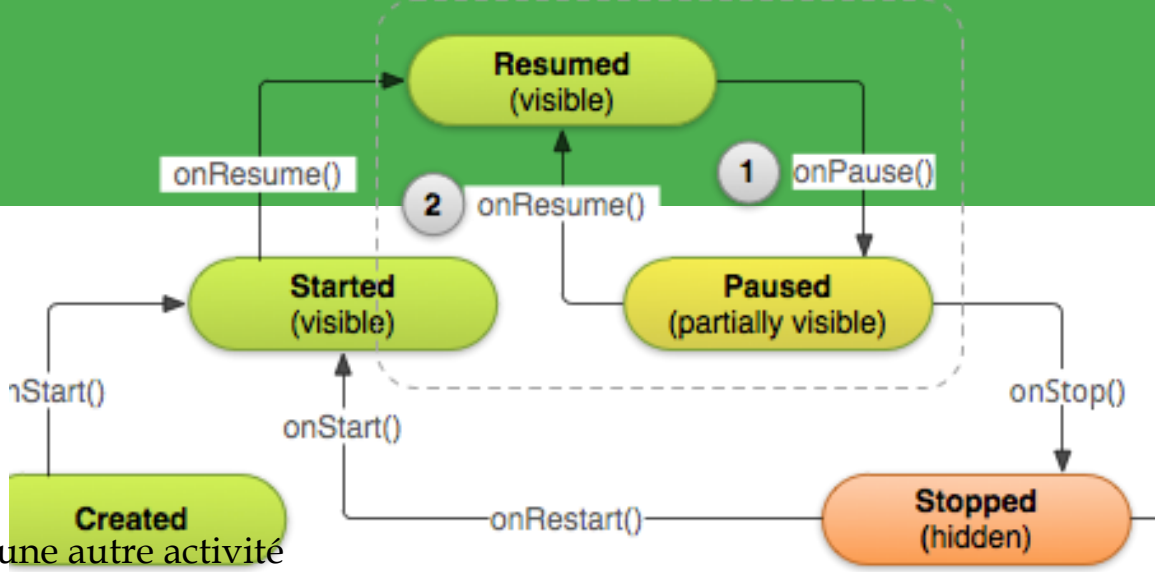
onRestart

- Appelée quand l'activité va redémarrer après avoir été stoppée
- Toujours suivie de onStart



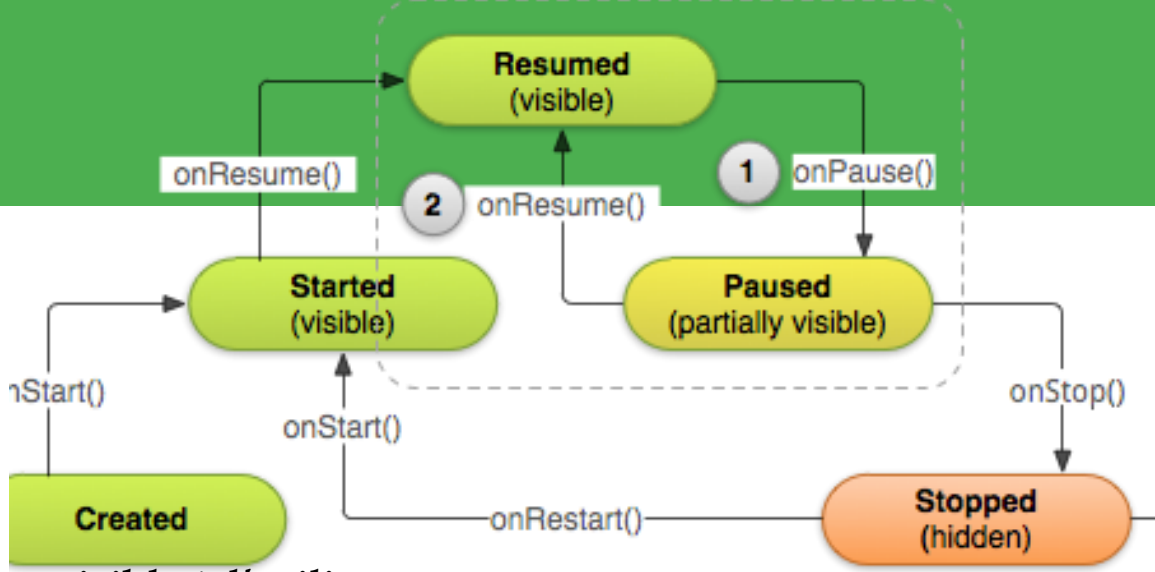
onResume

- Appelée juste avant que l'activité ne commence à interagir avec l'utilisateur
- A ce point, l'activité est en haut de la pile
- Toujours suivie de onPause()



onPause

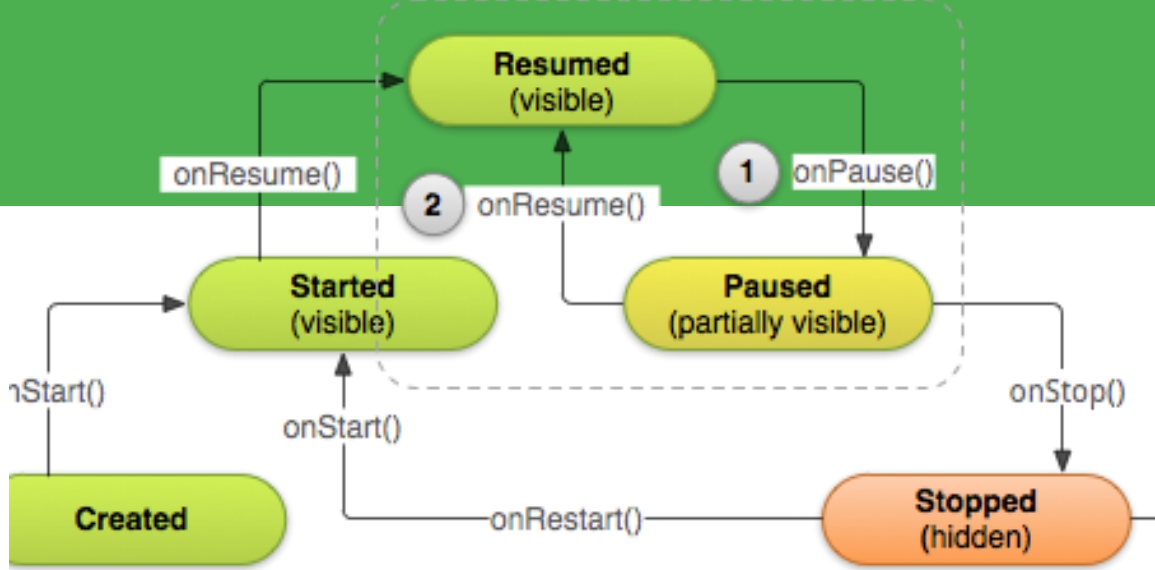
- Appelée quand le système va démarrer une autre activité
- Utilisée typiquement pour
  - Consigner les données non sauvegardées
  - Arrêtées les animations ou tout ce qui consomme de la mémoire…
- Ne doit pas consommer beaucoup de temps, car l'activité suivante ne va démarrer que si cette méthode retourne
- Suivie de :
  - onResume si l'activité est rechargée en premier plan
  - onStop si l'activité devient invisible
- Dans cet état, l'activité peut être tuée par le système



onStop

- Appelée quand l'activité n'est plus visible à l'utilisateur
- Peut arriver si:
  - L'activité est détruite
  - Une autre activité a repris son exécution et l'a recouverte
- Suivie par:
  - onRestart si l'activité recommence à interagir avec l'utilisateur
  - onDestroy si l'activité va disparaître

- Dans cet état, l'activité peut être tuée par le système



onDestroy

- Appelée quand l'activité est détruite
- Dernier appel que l'activité va recevoir
- Peut intervenir si:
  - L' activité se termine (appel de finish)
  - Le système détruit temporairement cette instance de l'activité pour gagner de l'espace
- On peut distinguer entre ces deux scénarios avec la méthode isFinishing()
- Dans cet état, l'activité peut être tuée par le système