

# Introduzione ad Android

Storia, Architettura, concetti base

Lezione 2



Google

Image capture: Jan 2019

Images may be subject to copyright.

[Italy](#)

[Terms](#) [Privacy](#)



Velenson

Photo - Jan 2022



Image capture: Jan 2022 Images may be subject to copyright.

[Italy](#) [Terms](#) [Privacy](#)

# Perché Android?

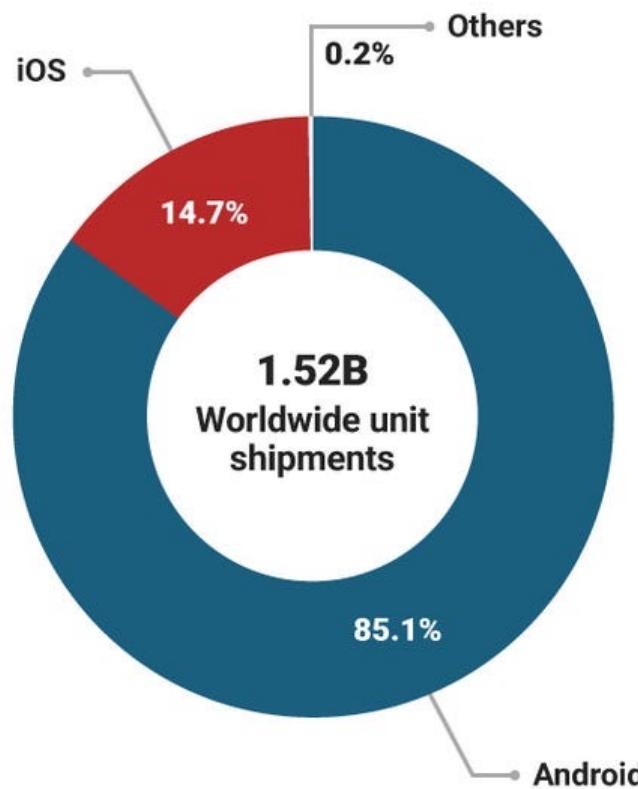
---

- Android è il principale sistema operativo per il mobile al mondo
- Stando alle statistiche, occupa circa l'80% del mercato mobile (a seconda della statistica da 75% a 85%)
- Non si è imposto solo su smartphone e tablet ma anche in
  - Wear OS per gli indossabili
  - Android TV per le televisioni intelligenti
  - Android for Cars per la realizzazione di app integrate con le autovetture
  - Android Things dedicato all'IoT (Internet of Things),
    - fino alla possibilità di distribuire app Android su sistemi Chrome OS come i Chromebook

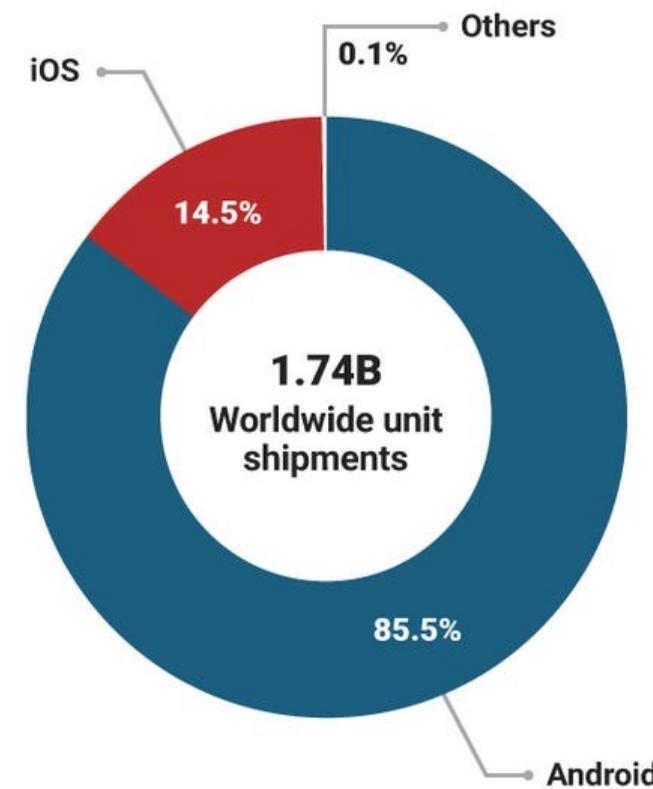


## GLOBAL MOBILE OPERATING SYSTEM MARKET SHARE

2017



2021



SOURCE: IDC Based on unit shipments, sums may not add to 100% due to rounding

statista | BUSINESS INSIDER

# Qual è il più diffuso sistema operativo al mondo?

---

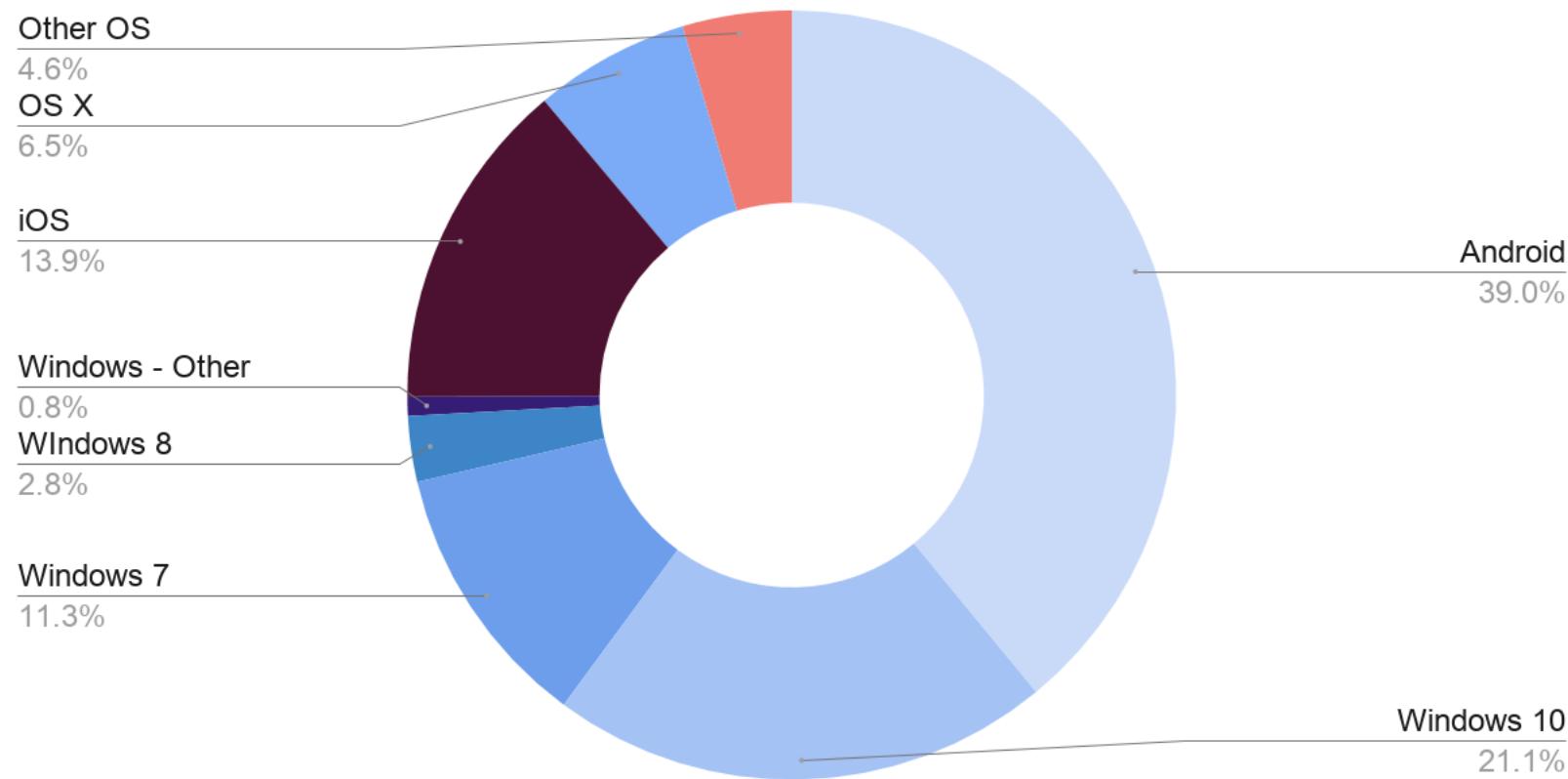
Considerando sia desktop che mobile

# Qual è il più diffuso sistema operativo al mondo?

Operating System Market Share, Install Base, 2019

www.T4.ai

Considerando sia desktop che mobile



<https://www.t4.ai/industry/operating-system-market-share>

# Un po' di storia

---

- Vi dicono qualcosa queste facce?



Andy Rubin



Rich Miner



Nick Sears



Chris White

# Un po' di storia

---

- Nel **2003**, Andy Rubin, Rich Miner, Nick Sears e Chris White fondarono **Android Inc.** il cui scopo era quello di sviluppare “*dispositivi cellulari più consapevoli della posizione e delle preferenze del loro proprietario*”
- Nel **2005**, con l’acquisizione da parte di **Google** per una cifra vicina ai 50 milioni di dollari, Android inizia a diventare un sistema operativo per dispositivi mobili, con un kernel Linux
- Dall’acquisizione da parte di Google ci vollero ancora oltre due anni di sviluppo, partendo da un kernel Linux, da parte del team di Rubin, rimasto nel progetto insieme a Miner e White, prima della **presentazione ufficiale avvenuta il 5 novembre 2007** da parte della Open Handset Alliance (un consorzio che include oggi 84 aziende per sviluppare open standard per dispositivi mobile), nata proprio in quell’occasione. Partners: Google, Motorola, Samsung, Vodafone, T-Mobile, ecc.

# Un po' di storia

---

- il 22 ottobre 2008 venne lanciato sul mercato G1 HTC Dream (il primo smartphone Android)
- Da allora è iniziato un lunghissimo processo di aggiornamenti e miglioramenti che hanno portato Android ad essere il sistema operativo più diffuso al mondo, con aggiornamenti costanti che negli ultimi anni hanno raggiunto una cadenza annuale, con minor release rilasciate tra una versione e l'altra
- In maniera simile a quanto accade per Linux, anche Android ha adottato una convenzione alfabetica per identificare le varie versioni. Per le versioni precedenti alla 1.0 vennero utilizzate solamente delle sigle e la prima release ad avere un nome in codice ufficiale è stata la **1.5**, denominata **Cupcake** (2009). Le versioni 1.0 e 1.1 sono state in seguito definite Alpha e Beta, o Apple Pie e Banana Bread, ma in realtà non hanno mai avuto una denominazione ufficiale

# G1 HTC Dream (2008)



# Versioni Android

---

- A partire dalla versione 1.5, Google ha deciso di abbinare il nome di un dolce ad ogni versione

# Versioni di Android



Apple Pie 1.0



Cupcake 1.5



Donut 1.6



Eclair 2.0/ 2.1



Froyo 2.2



Gingerbread 2.3.x



Honeycomb 3.x



Ice Cream Sandwich 4.0.x



Jelly Bean 4.1/4.2/4.3



KitKat 4.4



Lollipop 5.0



Marshmallow 6.0



Nougat 7.0



Oreo 8.0

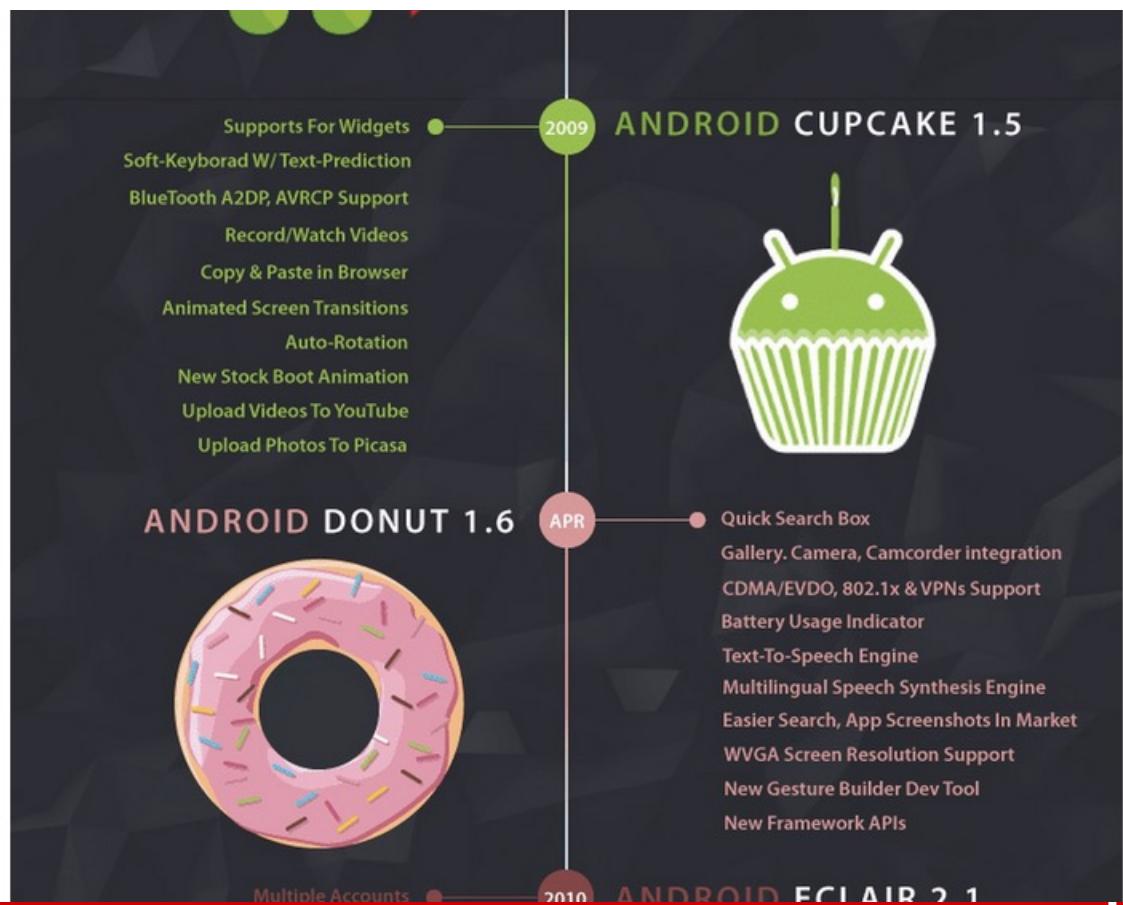


Pie 9.0

- E Versione dopo??

# Infografica versioni Android

- <https://earthlymission.com/android-version-history-a-visual-timeline/>



# Versioni di Android

---

- Dopo un calo di fantasia con la versione 9.0 di Android, chiamata Android 9 Pie, il colosso californiano ha deciso, nell'agosto del **2019**, di cambiare drasticamente rotta. Basta nomi dei dolci, difficili da comprendere per alcuni Paesi e culture. Al loro posto una soluzione molto più semplice, con il solo numero di versione -> Android 10 (chiamata inizialmente Q), e le versioni successive saranno contrassegnate esclusivamente dal numero

# Versioni di Android



Apple Pie 1.0



Cupcake 1.5



Donut 1.6



Eclair 2.0/ 2.1



Froyo 2.2



Gingerbread 2.3.x



Honeycomb 3.x



Ice Cream Sandwich 4.0.x



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

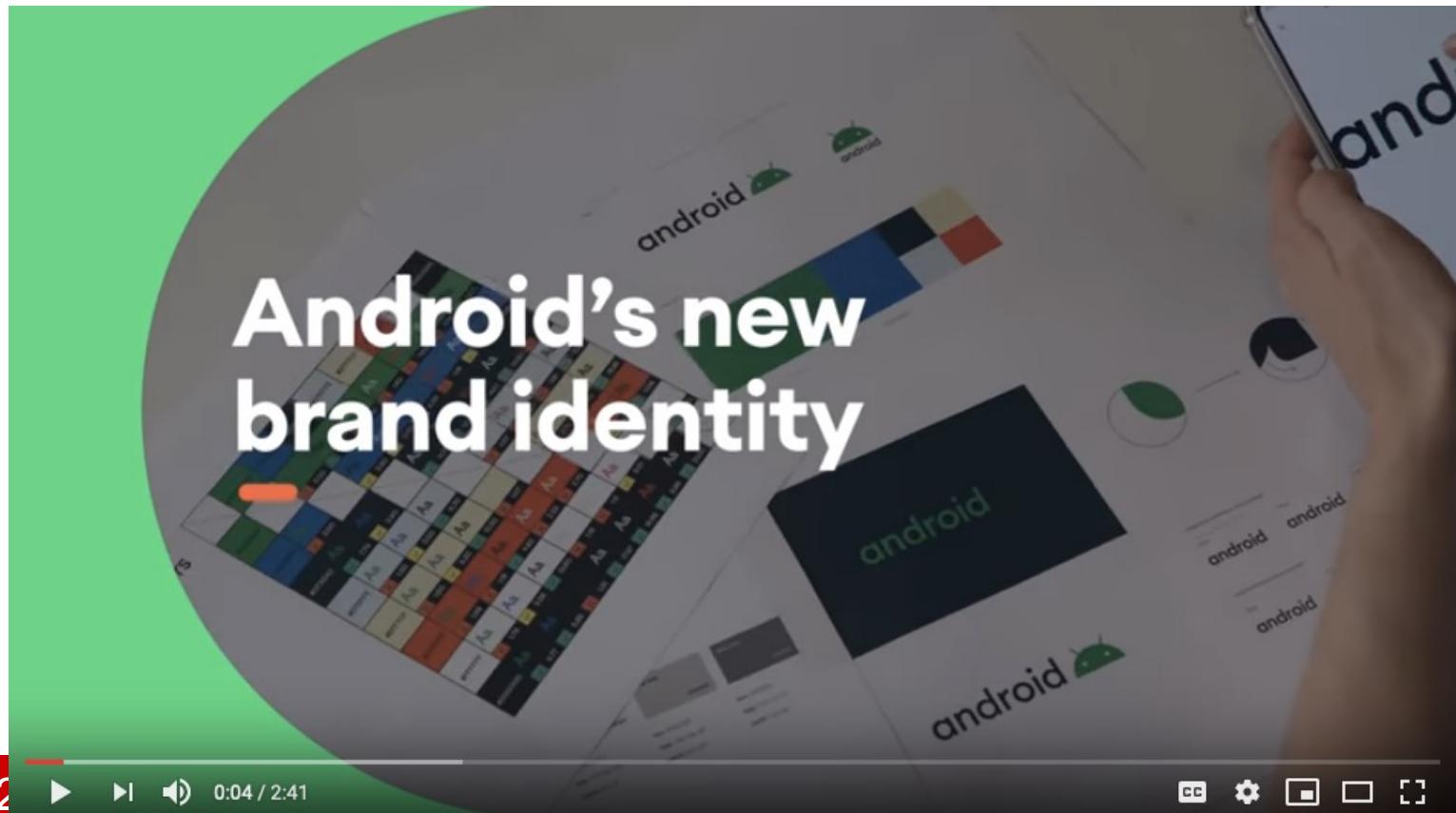
- Android 10, 3 settembre 2019  
(a gennaio 2021, 43.13% dei dispositivi Android ha Android 10, rendendola la versione più usata)



# Android 10: New brand identity

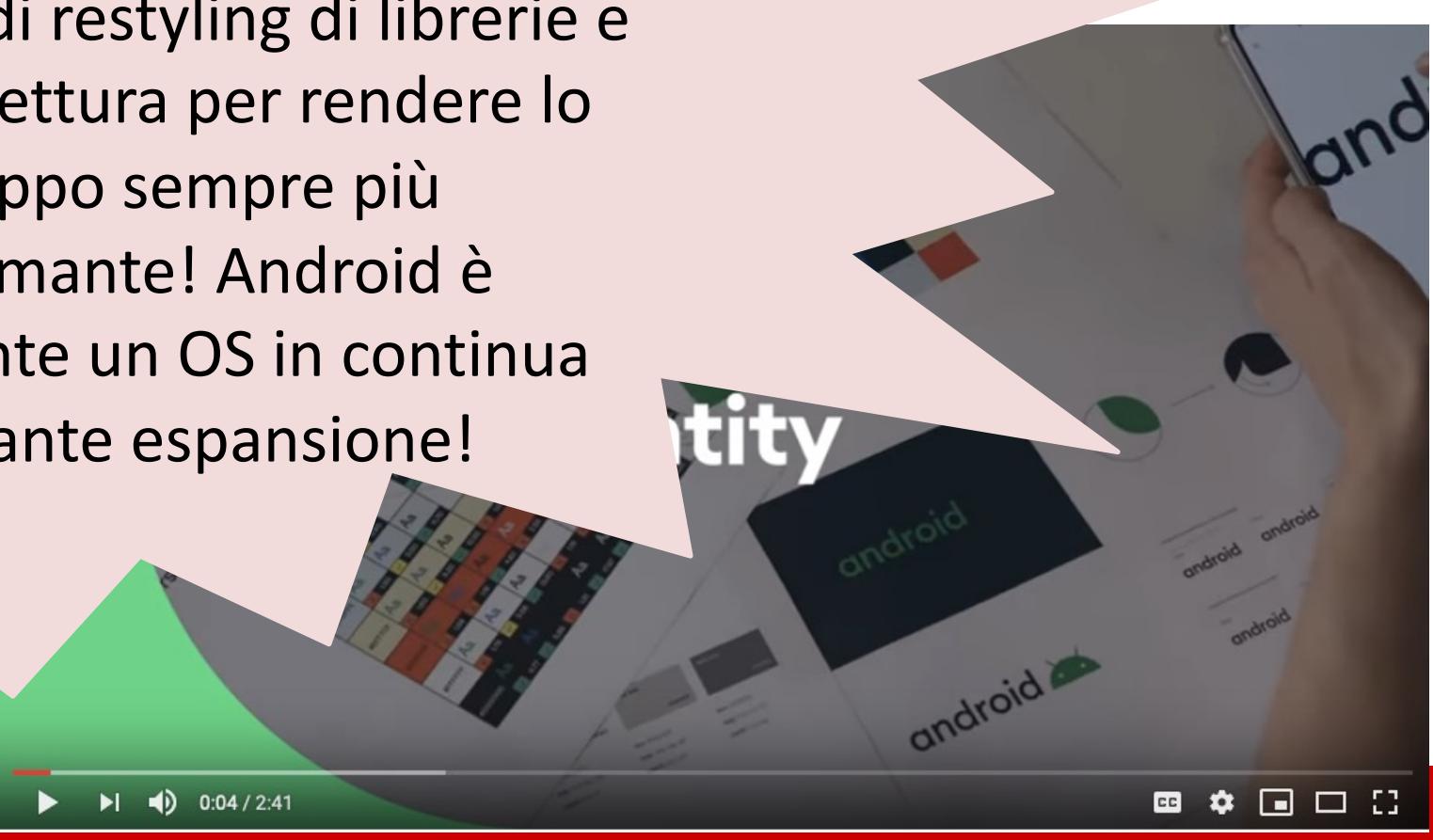
---

- [https://www.youtube.com/watch?time\\_continue=28&v=-2w7RKAIKTs&feature=emb\\_logo](https://www.youtube.com/watch?time_continue=28&v=-2w7RKAIKTs&feature=emb_logo)



# New brand identity

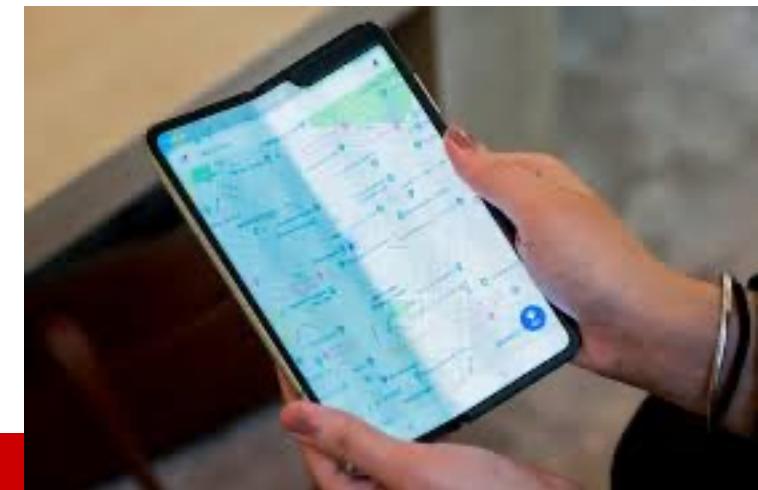
Ma non solo! Si è messa in atto un'attività di restyling di librerie e dell'architettura per rendere lo sviluppo sempre più performante! Android è sicuramente un OS in continua constante espansione!



# Android 10: Maggiori Novità

---

- Foldables
  - Basandosi su un solido supporto multi-finestra, Android 10 estende il multitasking tra le finestre delle app e offre continuità dello schermo per mantenere lo stato dell'app mentre il dispositivo si piega o si apre
  - Android 10 aggiunge una serie di miglioramenti a onResume e onPause per supportare il ripristino multiplo e avvisare la tua app quando è attiva
  - Modifica anche il funzionamento dell'attributo manifest resizableActivity, per aiutarti a gestire la modalità di visualizzazione della tua app su schermi pieghevoli e di grandi dimensioni



# Android 10: Maggiori Novità

---

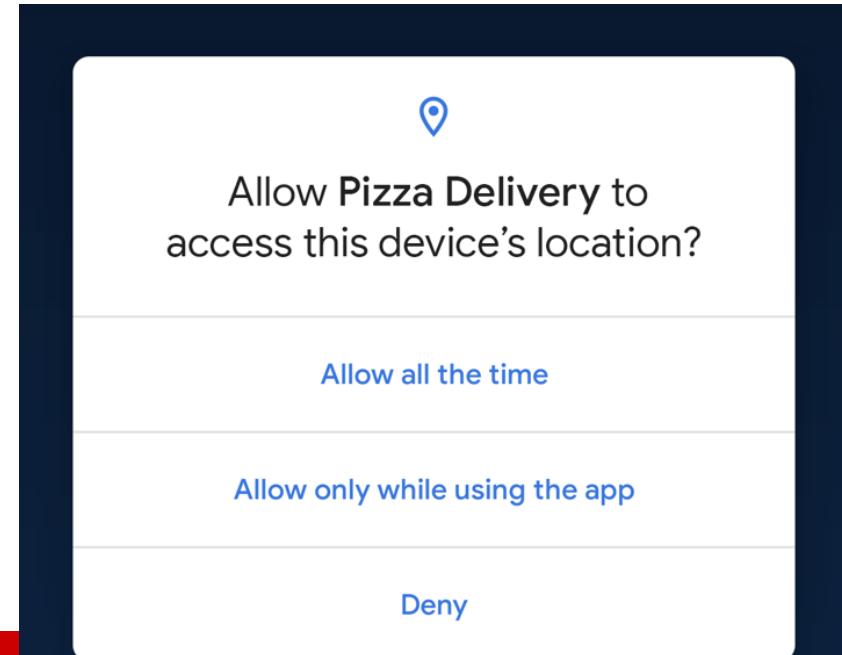
- 5G network
  - Il 5G promette di offrire velocità costantemente più elevate e una latenza inferiore, Android 10 aggiunge il supporto della piattaforma per il 5G ed estende le API esistenti per aiutarti a trarre vantaggio da questi miglioramenti
  - È possibile utilizzare le API di connettività per rilevare se il dispositivo ha una connessione a larghezza di banda elevata



# Android 10: Maggiori Novità

---

- Focus su **privacy** e security
  - Maggiore controllo sui dati di posizione
    - nuova opzione di autorizzazione - ora possono consentire a un'app di accedere alla posizione solo mentre l'app è effettivamente in uso (in esecuzione in primo piano)
  - Protezione del tracking del device
    - Randomized MAC addresses
  - Protezione dei dati degli utenti nella memoria esterna



# Android 10: Maggiori Novità

---

- Focus su **privacy e security**
  - Crittografia dello spazio di archiviazione
    - per crittografare i dati degli utenti e per renderlo più efficiente, Android 10 include Adiantum, nuova modalità di crittografia
  - Biometria migliorata
    - Android 10 estende il framework BiometricPrompt per supportare metodi di autenticazione passiva come il volto e l'aggiunta di flussi di autenticazione impliciti ed esplicativi. Nel flusso esplicito, l'utente deve confermare esplicitamente la transazione durante l'autenticazione. Il flusso implicito è progettato per un'alternativa più leggera per le transazioni con autenticazione passiva

# Android 10: Maggiori Novità

---

- ART (Android Runtime) optimizations
  - I miglioramenti nel runtime ART aiutano le app ad avviarsi più velocemente, consumano meno memoria ed funzionano in modo più fluido, senza richiedere alcun intervento
  - I profili ART forniti da Google Play consentono a ART di precompilare parti dell'app prima ancora che vengano eseguite

# Android 10: Maggiori Novità

---

- Thermal API
  - Quando i dispositivi si surriscaldano, possono rallentare la CPU e / o la GPU e ciò può influire su app e giochi in modi inaspettati
  - Ora in Android 10, app e giochi possono utilizzare un thermal API per monitorare i cambiamenti sul dispositivo e agire per aiutare a ripristinare la temperatura normale
    - Ad esempio, le app di streaming possono ridurre la risoluzione / la velocità in bit o il traffico di rete, un'app per fotocamera potrebbe disabilitare il flash o il miglioramento intensivo delle immagini o un gioco potrebbe ridurre la frequenza dei fotogrammi o la tassellatura dei poligoni

# E poi? Android 11

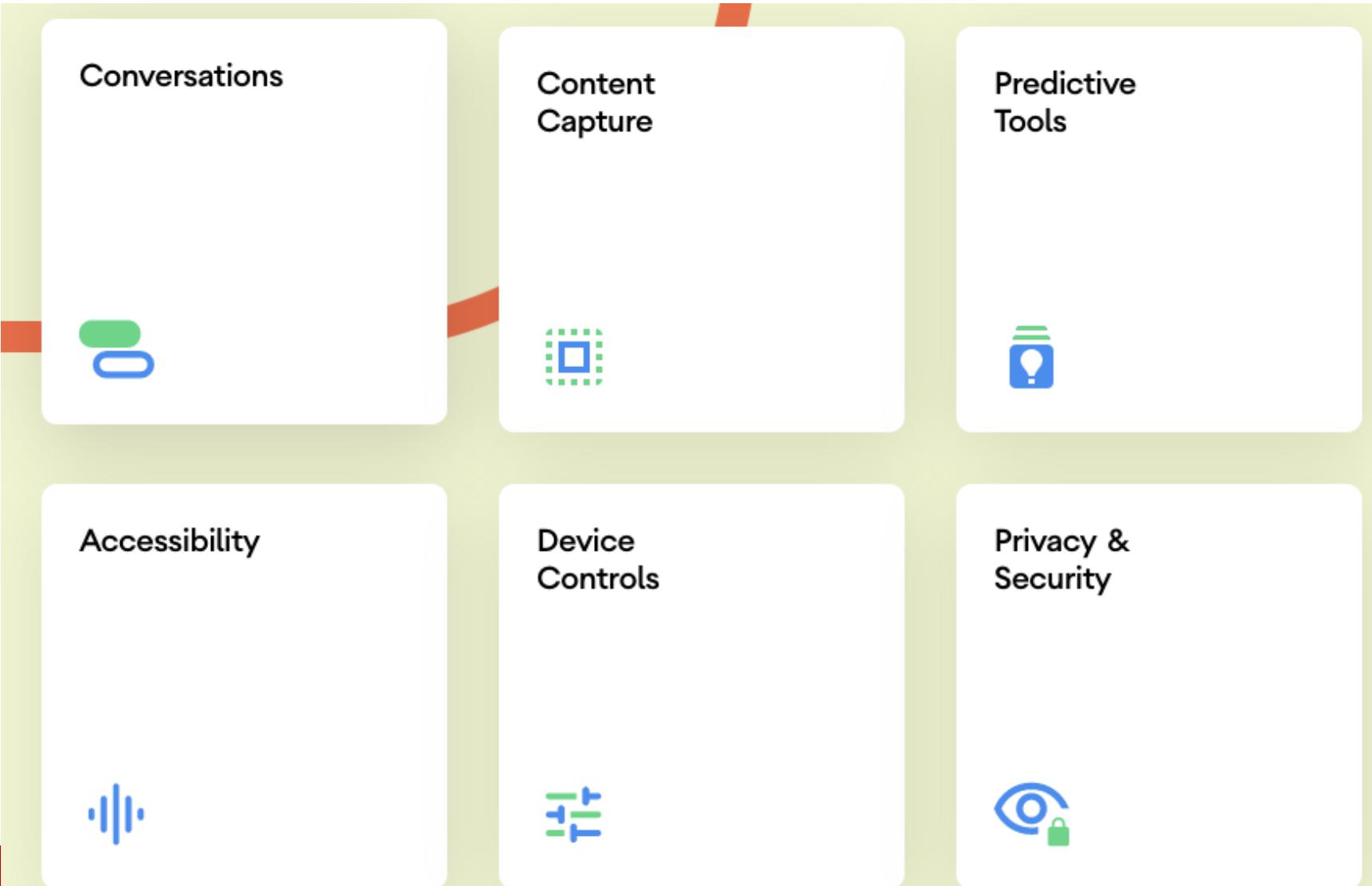
---

- “**The OS that gets to what’s important**”
  - “Go straight to the stuff that matters most. Because Android 11 is optimized for how you use your phone. Giving you powerful device controls. And easier ways to manage conversations, privacy settings and so much more.”

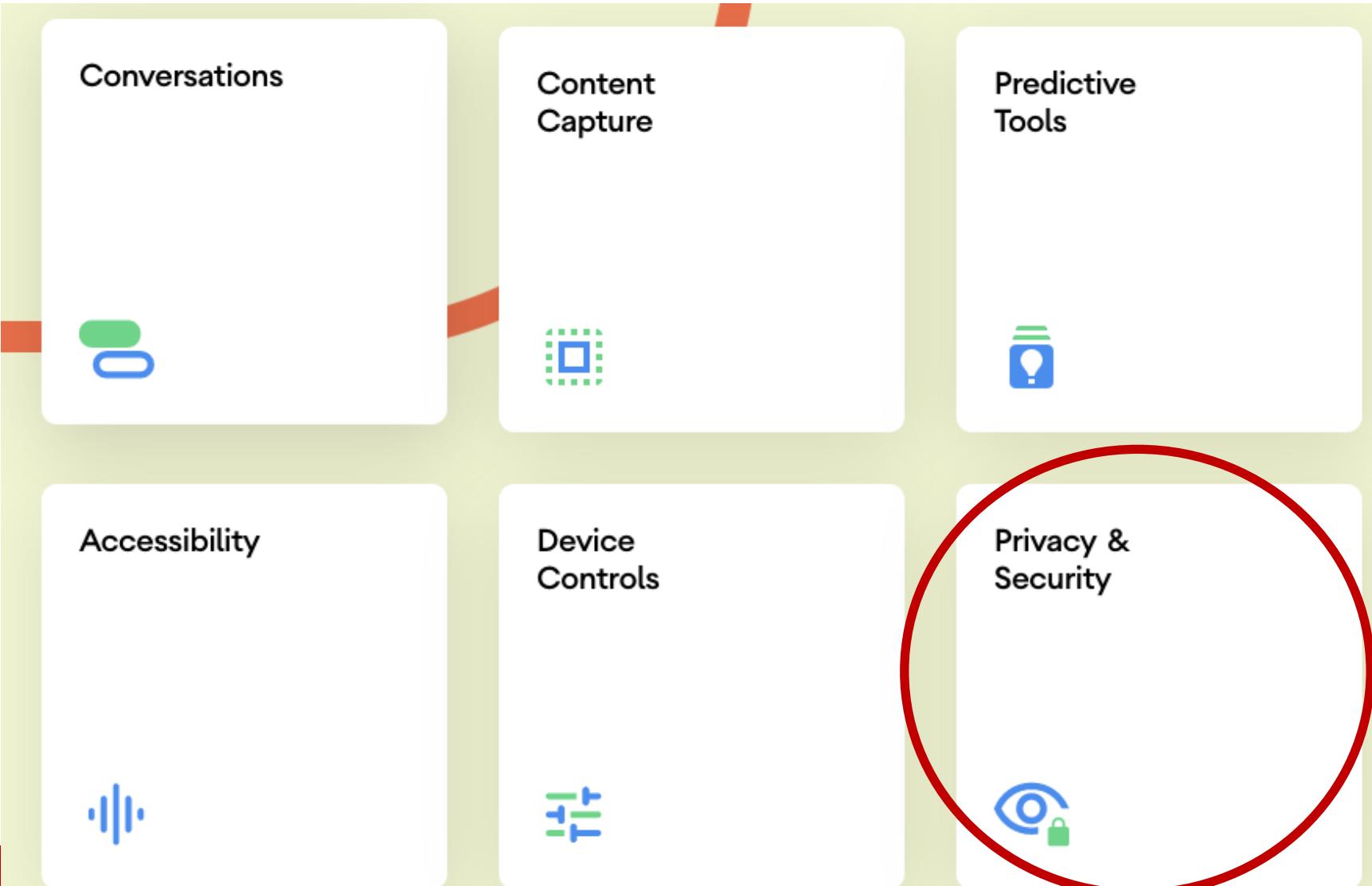
Rilasciato l’8 settembre 2020



# Novità principali



# Novità principali



# Privacy e sicurezza



## One-time permissions

Give one-time permissions to apps that need your mic, camera or location. The next time the app needs access, it must ask for permission again.

## Permissions auto-reset

If you haven't used an app in a while, you may not want it to keep accessing your data. So Android will reset permissions for your unused apps. You can always turn permissions back on.

<https://www.youtube.com/watch?v=vaD-DPI6sgU>

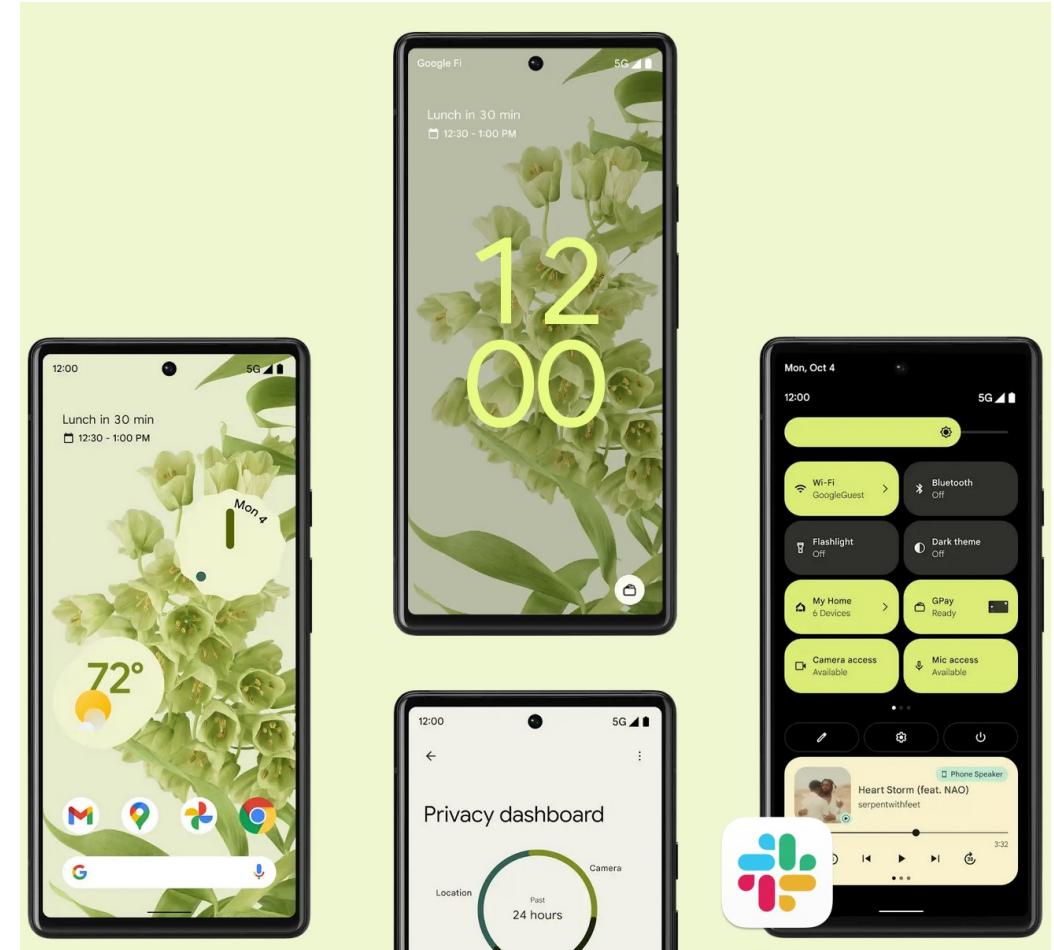
# Android 11: News

---

- <https://www.android.com/android-11/>

# Android 12

- Rilasciato il 4 ottobre 2021
- Android 12 offre esperienze ancora più personali, sicure e effortless sul tuo dispositivo
- Dotato di un'interfaccia utente completamente reinventata solo per te, nuove funzionalità per la privacy progettate per la tua sicurezza e che ti danno il controllo e modi più semplici per accedere direttamente al tuo gioco o persino passare a un nuovo dispositivo.



# Android 12

---

**More personal,  
safe and  
effortless than  
ever before.**

# Android 12

---

- <https://www.android.com/android-12/>
- Video promozionale
  - <https://youtu.be/UHQPdP8qgrk>

# Android 12: accessibility

ACCESSIBILITY IMPROVEMENTS

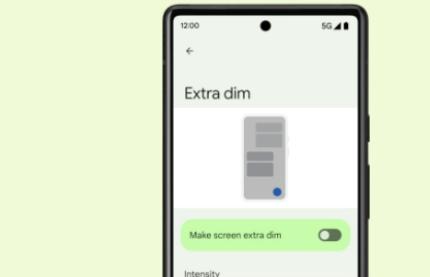
## Built for accessibility.

Android 12 is designed to be even more accessible with new visibility features, including:



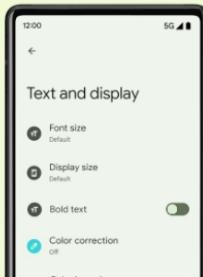
### Area magnification

A new window magnifier lets you zoom in on a part of your screen without having to lose context on the rest of the screen content.



### Extra dim

Make your display extra dim for night-time scrolling or situations when even the lowest brightness setting is too bright.



### Bold text

See text more clearly with the ability to switch the font to bold across the whole phone.



### Grayscale

Adjust how colors display on your device to grayscale.



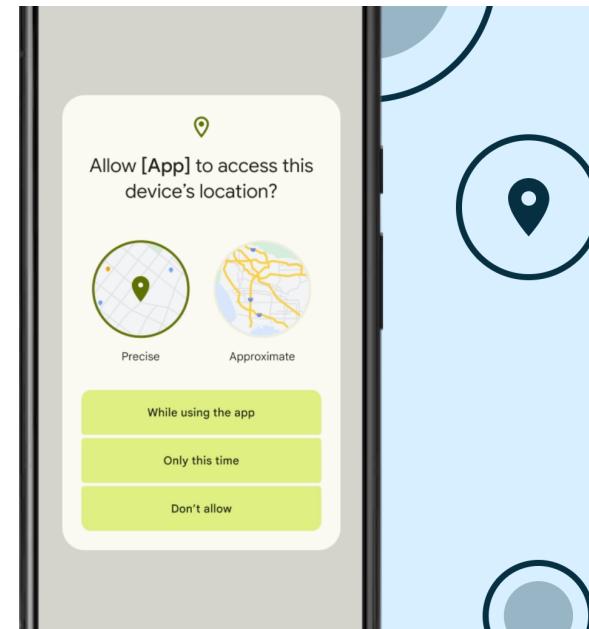
# Android 12: privacy

SAFE

**Private by design so you're in control.**

MIC & CAMERA INDICATORS AND TOGGLERS

**Stronger mic and camera access controls.**



**Keep your precise location private.**

While some apps need precise location information for tasks like turn-by-turn navigation, many other apps only need your approximate location to be helpful. With Android 12, you can choose between giving apps access to your precise location or an

[BACK TO TOP](#)

# E poi?

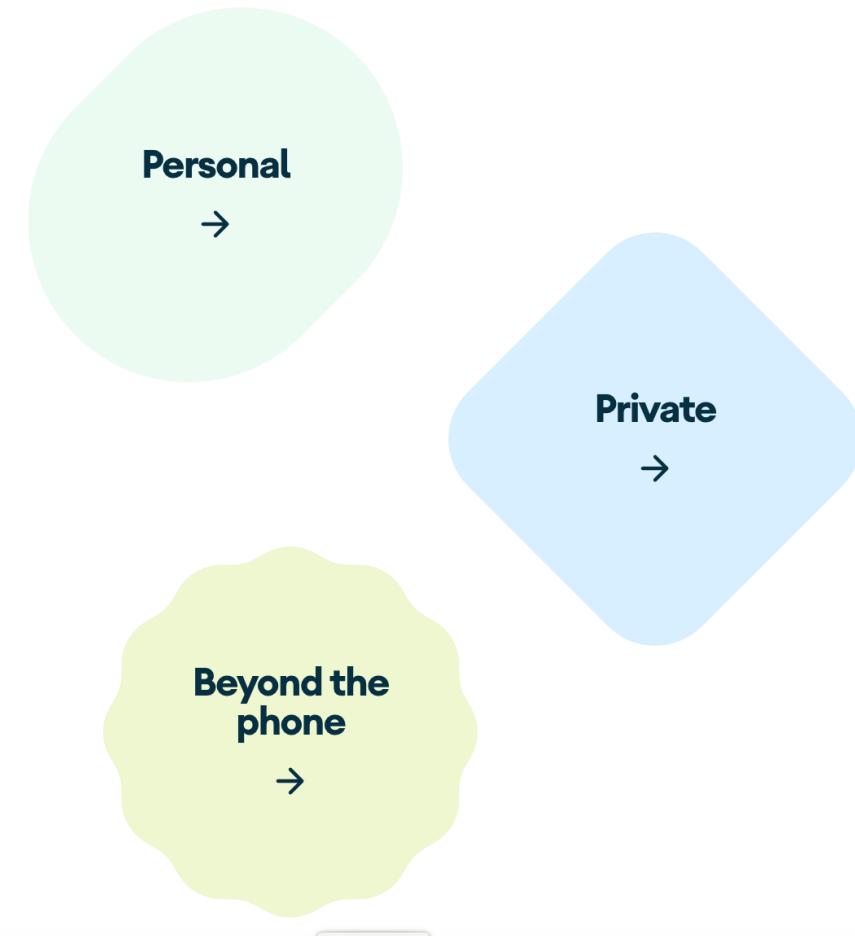
---

Android 13 (rilasciato il 15 agosto 2022)

**Your phone, tablet and more—all on your terms.**

- «Build for user privacy with photo picker and notification permission. Improve productivity with themed app icons, per-app languages, and clipboard preview. Build for modern standards like Bluetooth LE Audio. Deliver a better experience on tablets and large screens»
- <https://www.android.com/android-13/>

# Android 13 Highlights



# Android 13: Security

**Security  
that  
keeps  
your data  
protected  
all day.**

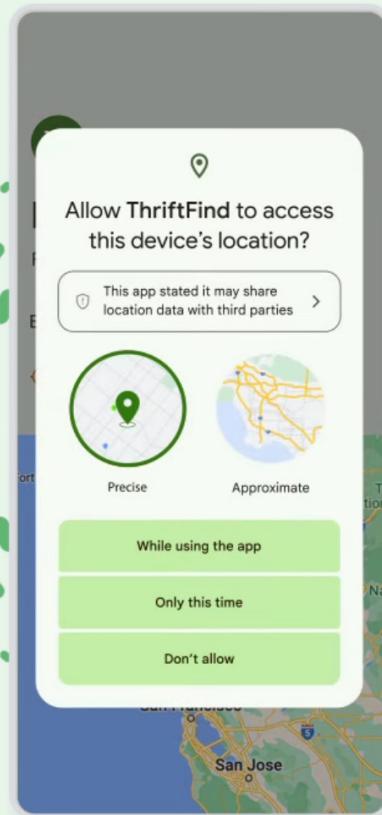
From the moment you turn on your device, Android works to keep your data safe and secure. With Android 13, you have more control over what information apps can and can't access—including specific photos, videos and clipboard history.

# e oggi?

---

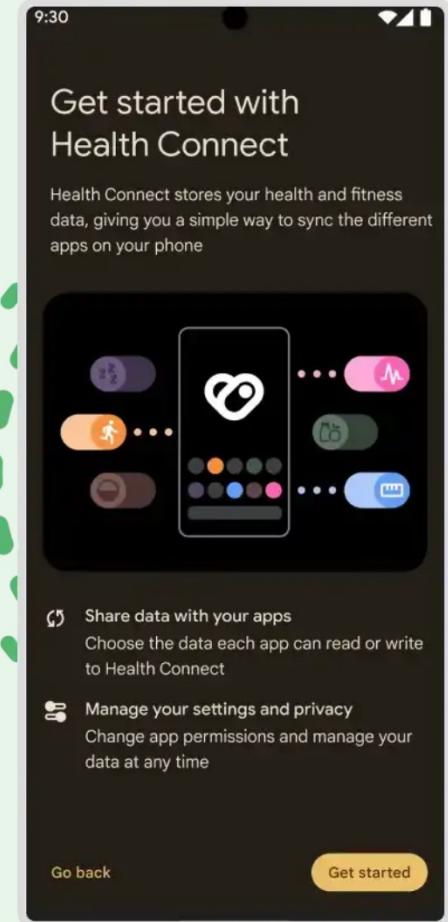
- Android 14 (<https://www.android.com/android-14/>, October 4, 2023)
  - **Make it yours.**
    - Make your device more personal, protected and accessible with the newest OS upgrade. Improved photo quality, new themes and AI generated wallpapers. Privacy updates for your health, safety and data. And expanded accessibility features.

# Spotlight on your protection.



## Data sharing details at a glance.

Android 14 proactively gives you easy-to-understand info about apps' data permissions<sup>6</sup>. Choose what access you grant to apps to keep you safe<sup>7</sup>.



**Get started with Health Connect**

Health Connect stores your health and fitness data, giving you a simple way to sync the different apps on your phone

Share data with your apps  
Choose the data each app can read or write to Health Connect

Manage your settings and privacy  
Change app permissions and manage your data at any time

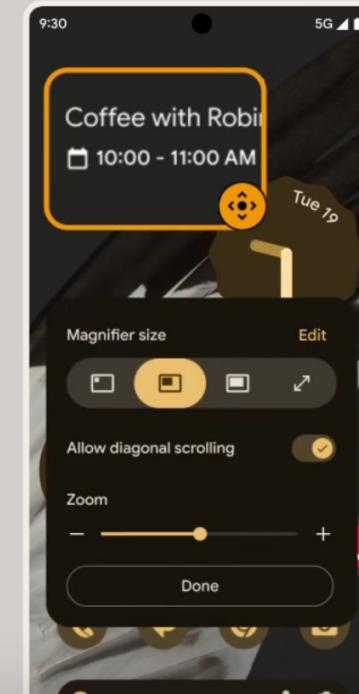
Go back      Get started

**Connect your health data easily.**

Health Connect is now right in your Settings, giving you a central way to connect your favorite health and fitness apps, see your health and fitness data in one place and stay in control of your privacy<sup>8</sup>.

# Improved sight and sound accessibility.

Pinch-to-zoom and set your magnification preference for use across apps.



# Riproviamo...e oggi?

16 February 2024

## The First Developer Preview of Android 15



<https://android-developers.googleblog.com/2024/02/first-developer-preview-android15.html>

# Riproviamo...e oggi?

16 February 2024

The First Developer Pre

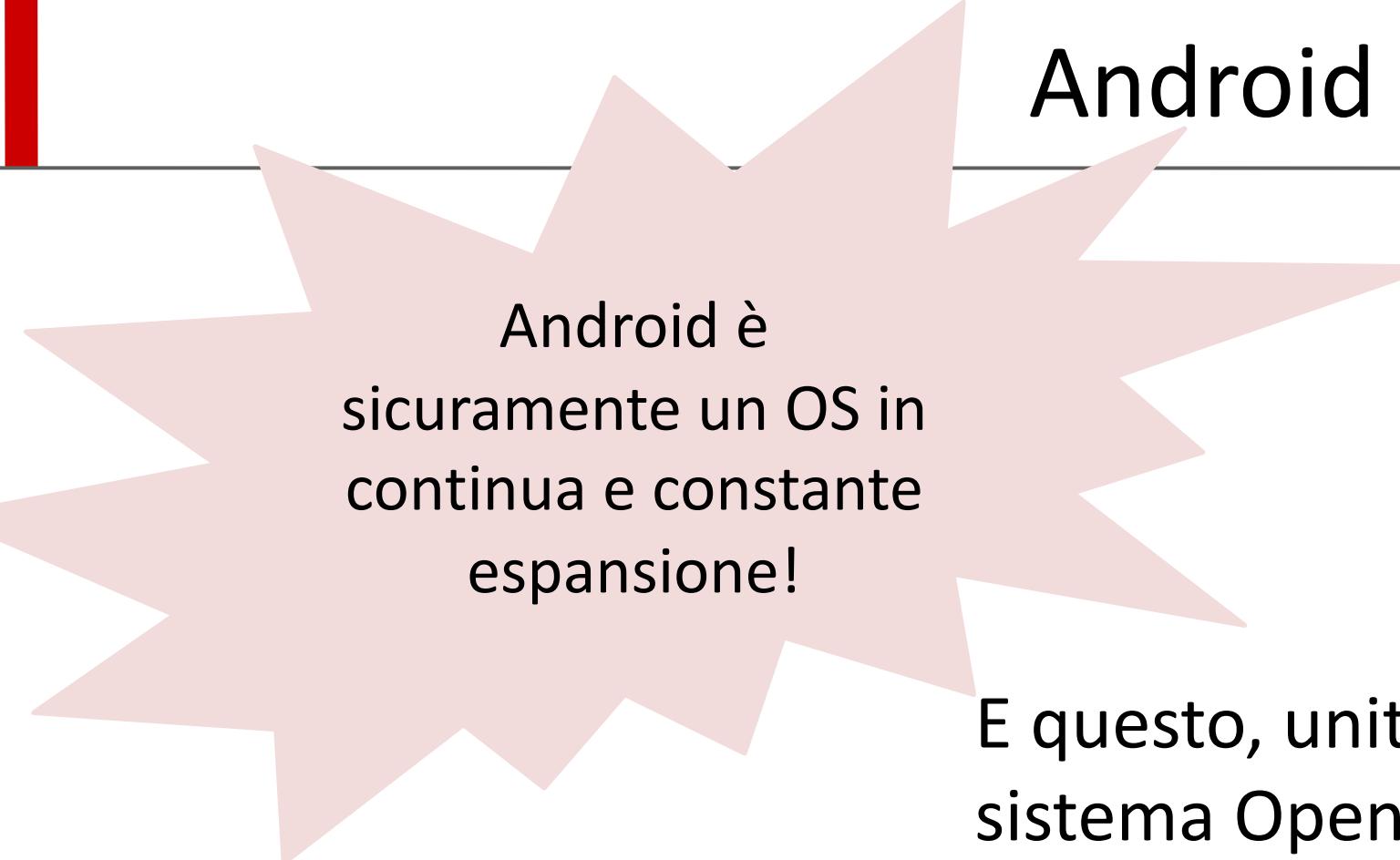
Android 15

Ovviamente, noi non lo  
vedremo!!!



<https://android-developers.googleblog.com/2024/02/first-developer-preview-android15.html>

# Android



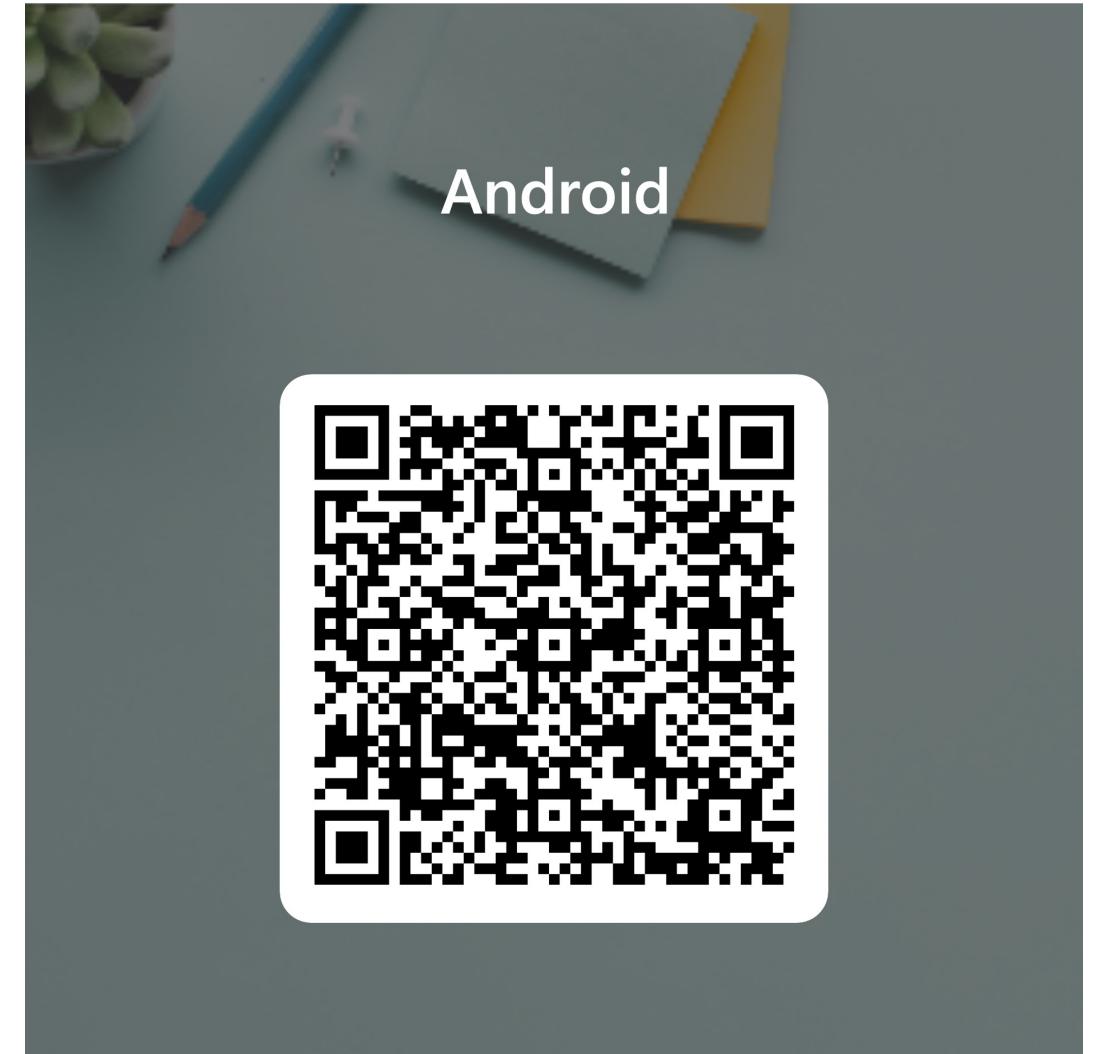
Android è  
sicuramente un OS in  
continua e costante  
espansione!



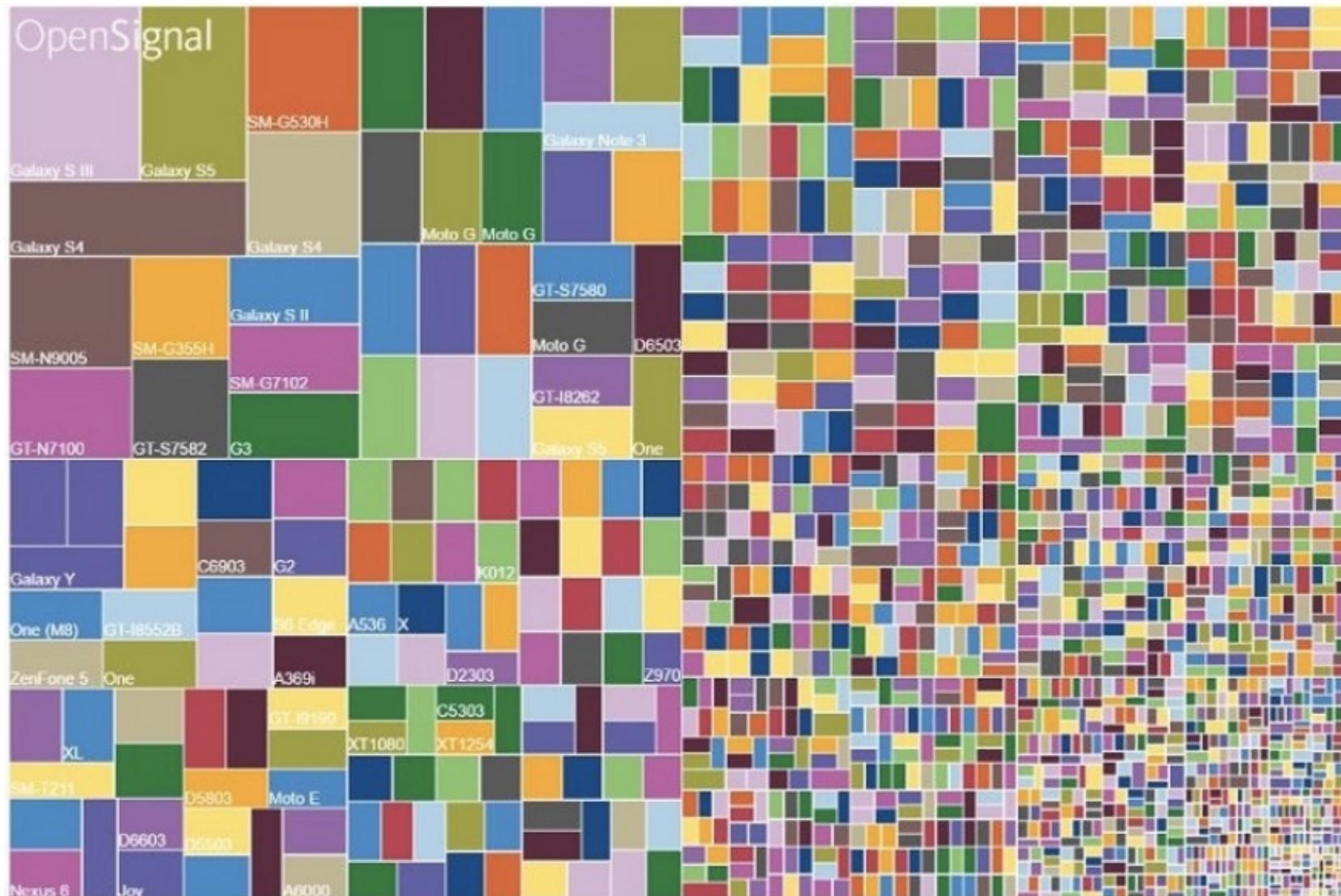
E questo, unito al fatto che è un  
sistema Open Source, porta  
sicuramente moltissimi vantaggi  
ma anche.... Problematiche!!

# Quiz

- Avete Android?  
Se sì, che versione avete?

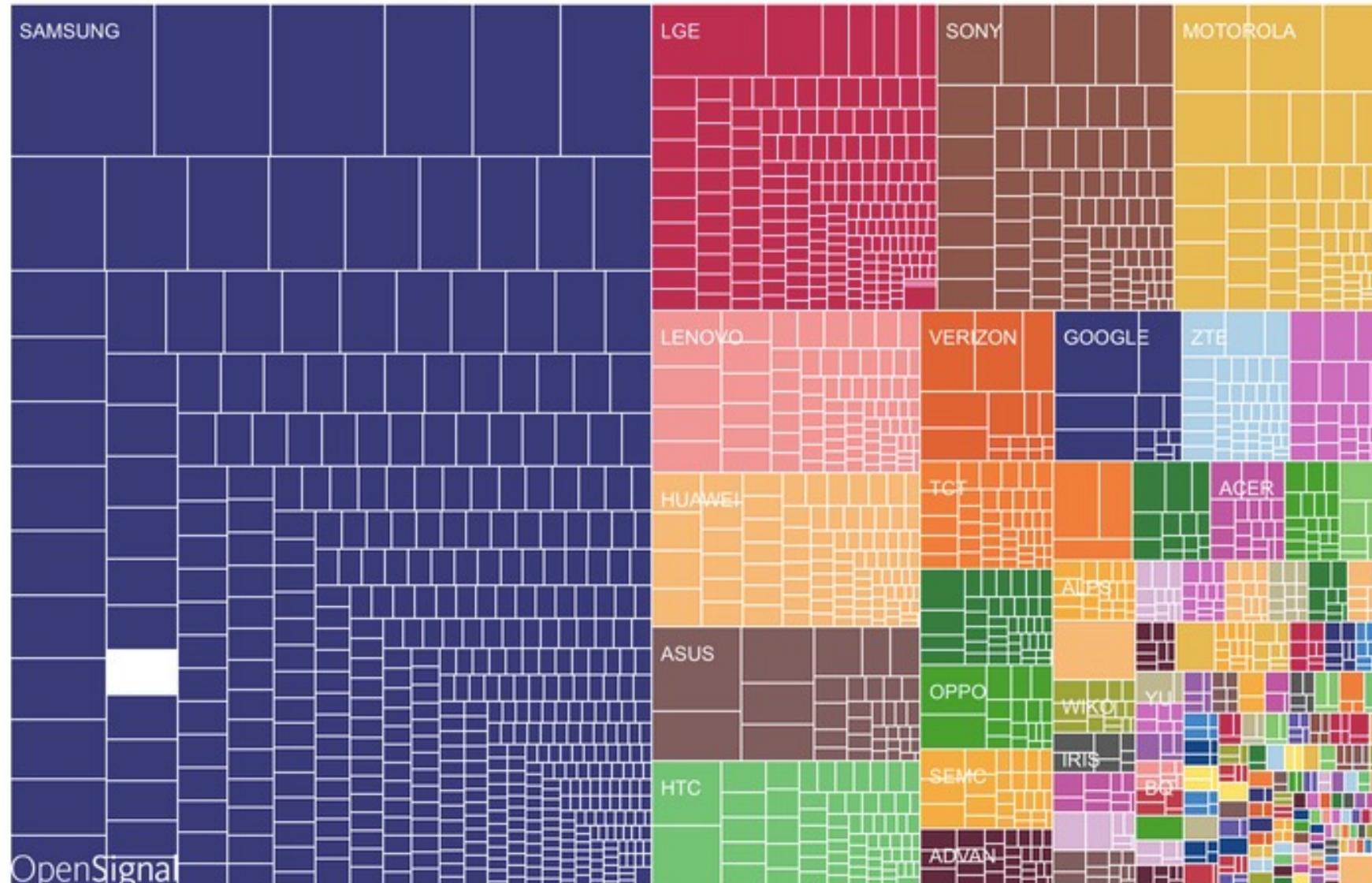


# Frammentazione dei device con Android

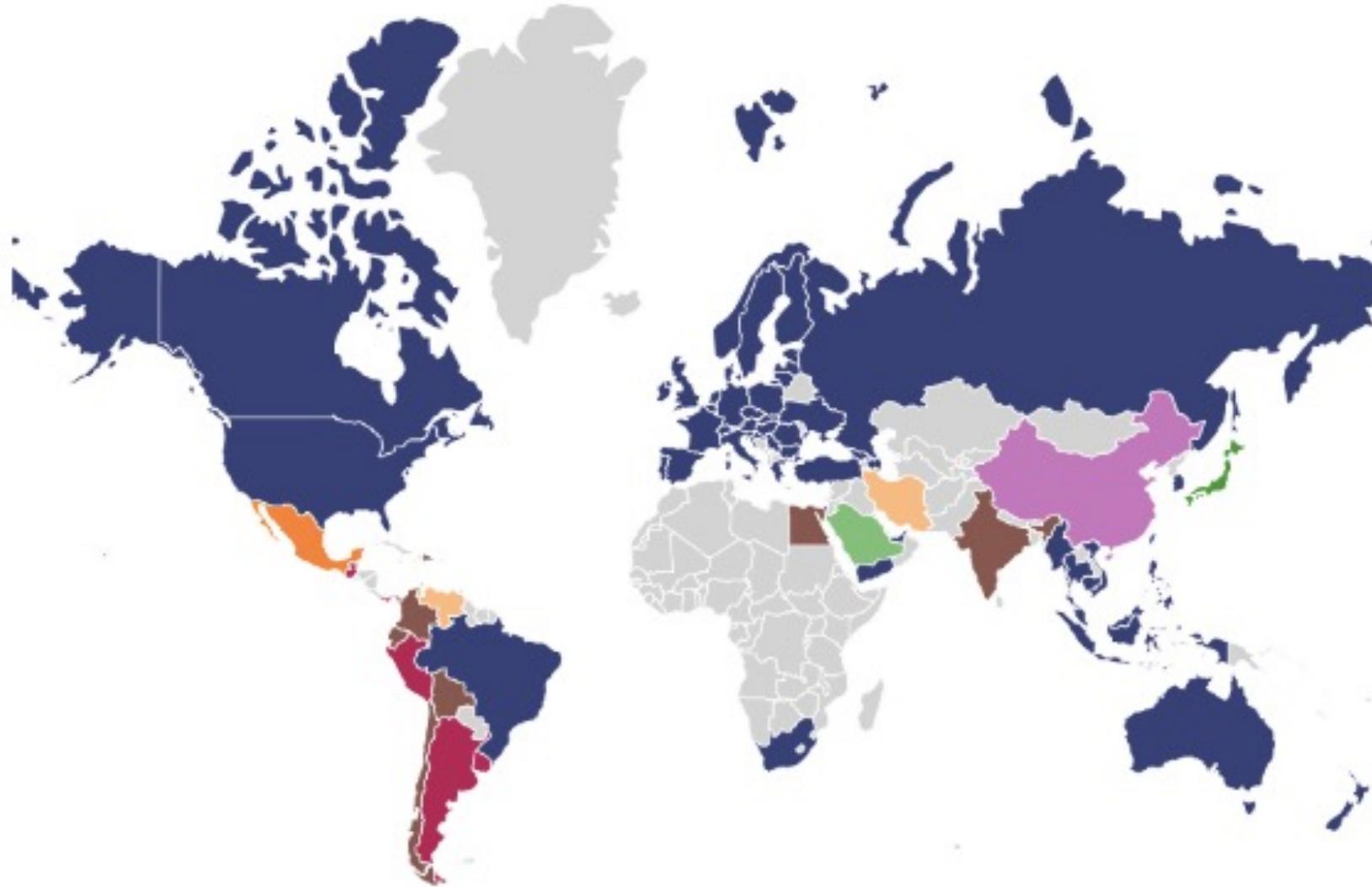


2015

## BRAND FRAGMENTATION



2015

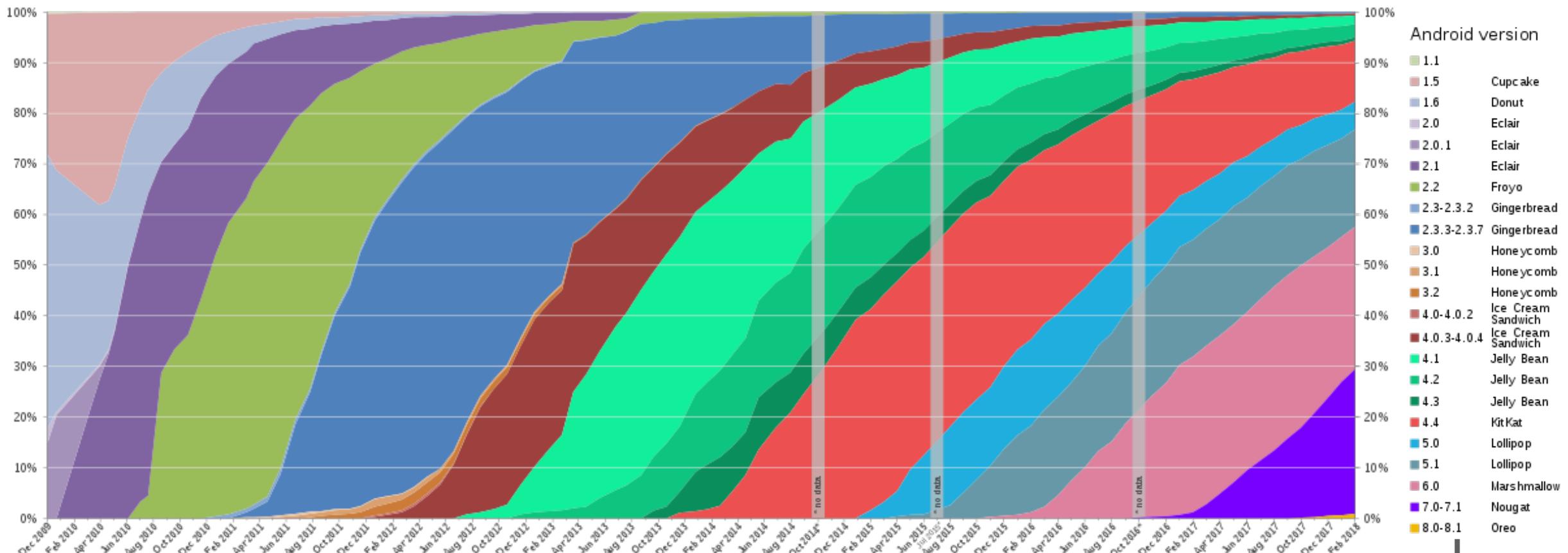


2015



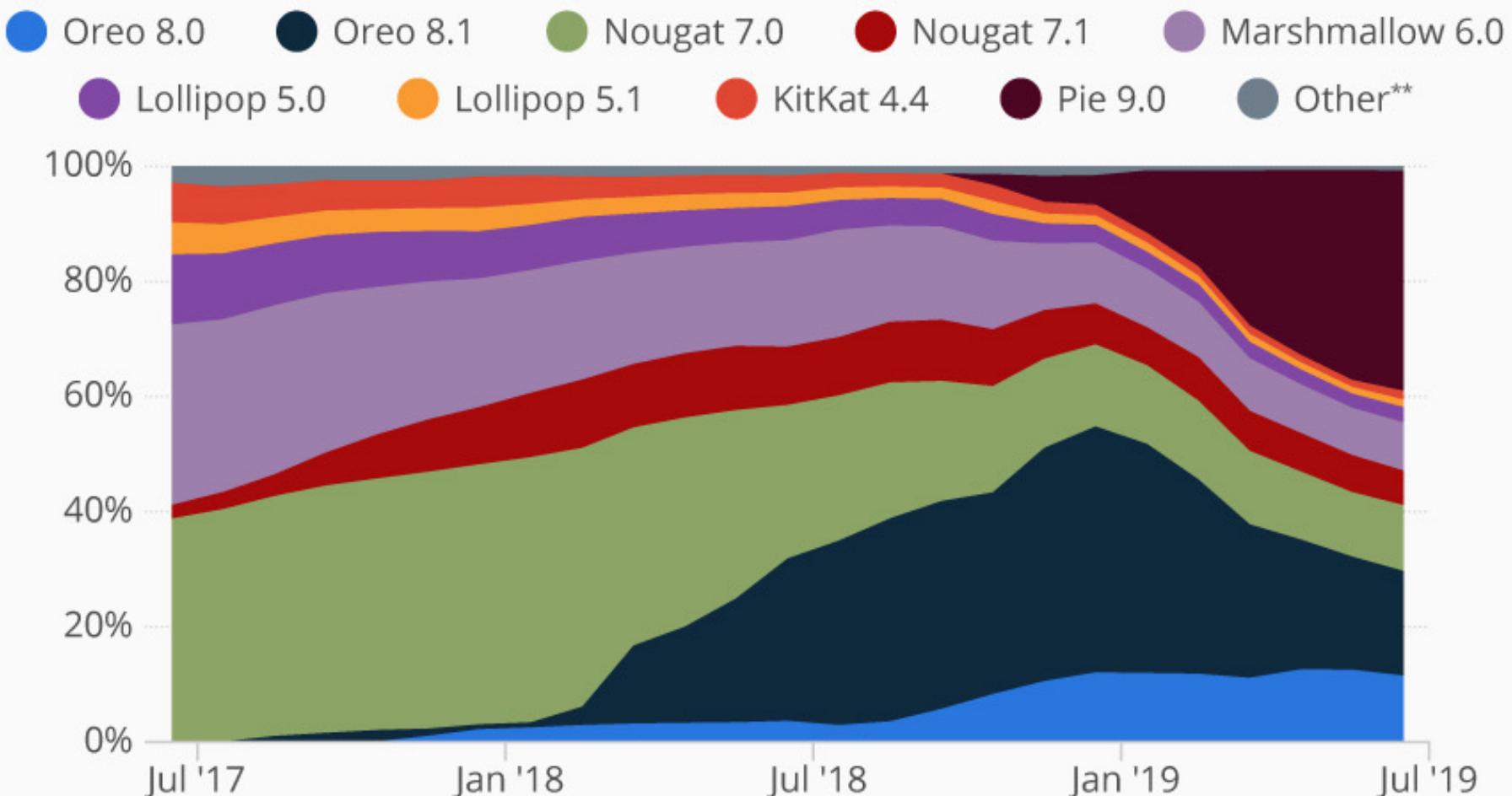
# Frammentazione versioni Android

- Dal 2009 al 2018



# Android OS Still a Fragmented Market

Mobile Android operating system market share by version in the United States 2017-2019\*

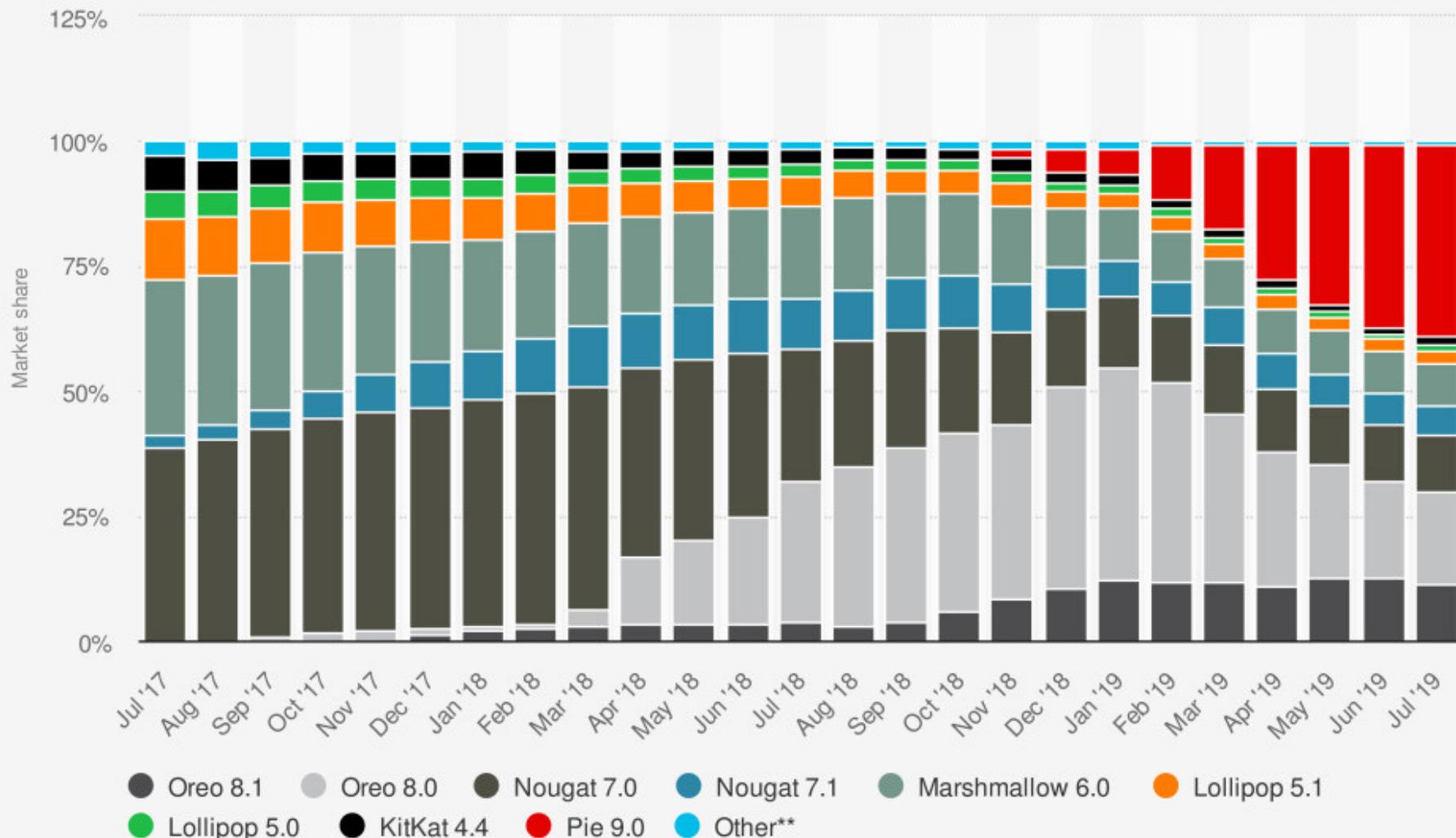


\* Based on aggregate data collected by StatCounter on a sample exceeding 10 billion pageviews per month collected from across the StatCounter network of more than 2 million websites.

\*\* Other includes all versions with a share of less than one percent

Source: StatCounter

# Mobile Android operating system market share by version in the United States from July 2017 to July 2019\*



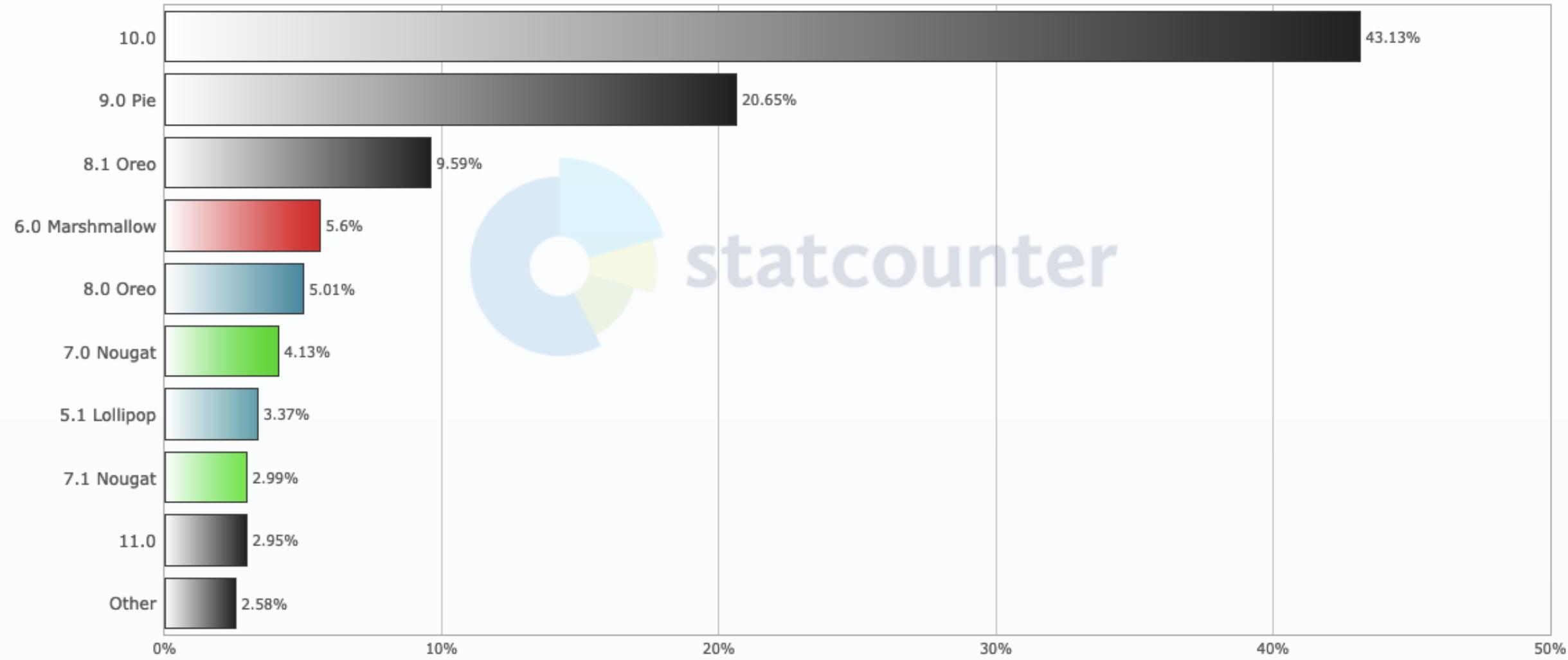
Source  
StatCounter  
© Statista 2019

Additional Information:  
United States; StatCounter; 2017 to 201

# Mobile & Tablet Android Version Market Share Worldwide

[Edit Chart Data](#)

Jan 2021



- 
- Per che versione conviene sviluppare?
    - Versione target



| ANDROID PLATFORM VERSION | API LEVEL | CUMULATIVE DISTRIBUTION | Marshmallow  |  |
|--------------------------|-----------|-------------------------|--|--|
| 4.1 Jelly Bean           | 16        |                         | Security<br>Fingerprint Authentication<br>Confirm Credential                                   | Wireless & Connectivity<br>Hotspot 2.0               |
| 4.2 Jelly Bean           | 17        | 99.9%                   | System<br>App Linking  | Improved Bluetooth Low Energy                        |
| 4.3 Jelly Bean           | 18        | 99.7%                   | Adoptable Storage Devices<br>Multimedia  | Scanning   |
| 4.4 KitKat               | 19        | 99.7%                   | 4K Display Mode<br>Support for MIDI  | Android for Work                                     |
| 5.0 Lollipop             | 21        | 98.8%                   | Create digital audio capture and playback objects<br>APIs to associate audio and input devices | Controls for Corporate-Owned, Single-Use devices     |
| 5.1 Lollipop             | 22        | 98.4%                   | List of all audio devices<br>Updated video processing APIs                                     | Silent install and uninstall of apps by Device Owner |
| 6.0 Marshmallow          | 23        | 96.2%                   | Flashlight API<br>Reprocessing Camera2 API   | Silent enterprise certificate access                 |
| 7.0 Nougat               | 24        | 92.7%                   | Updated ImageWriter objects and Image Reader class   | Auto-acceptance of system updates                    |
| 7.1 Nougat               | 25        | 90.4%                   | User Input<br>Voice Interactions   | Delegated certificate installation                   |
| 8.0 Oreo                 | 26        | 88.2%                   | Assist API<br>Bluetooth Stylus Support   | Data usage tracking                                  |
| 8.1 Oreo                 | 27        | 85.2%                   | User Interface   | Runtime permission management                        |
| 9.0 Pie                  | 28        | 77.3%                   | Themeable ColorStateLists  | Work status notification                             |
| 10. Q                    | 29        | 62.8%                   |  | Last updated: August 4th, 2022                       |
| 11. R                    | 30        | 40.5%                   |  |  |
| 12. S                    | 31        | 13.5%                   |  |  |

<https://developer.android.com/about/versions/marshmallow/android-6.0.html>

Cancel

OK

# Nota: Generic System Image

---

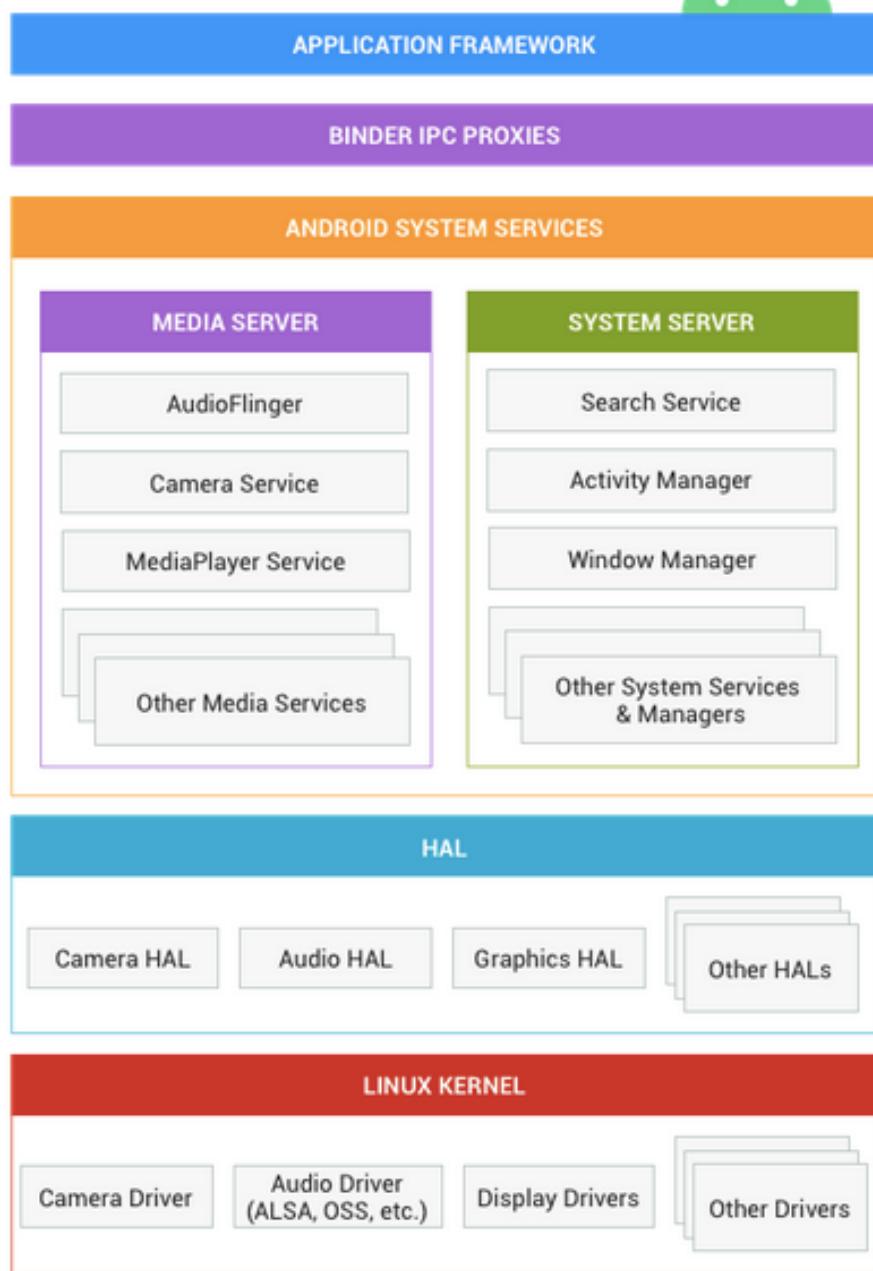
- Un'immagine di sistema generica (Generic System Image - GSI) è un'implementazione Android pura con codice AOSP (Android Open Source Project) non modificato



# Android.. In dettaglio!

---

- Android è
  - open source
  - Linux-based software stack



# Architettura

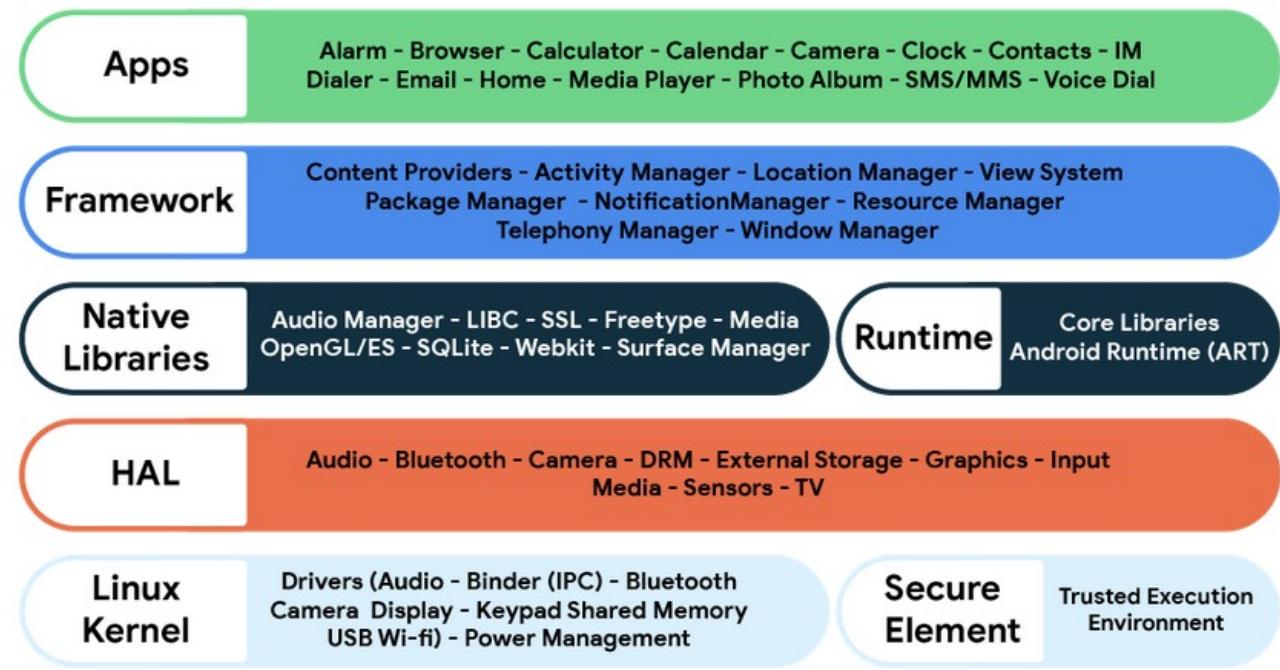
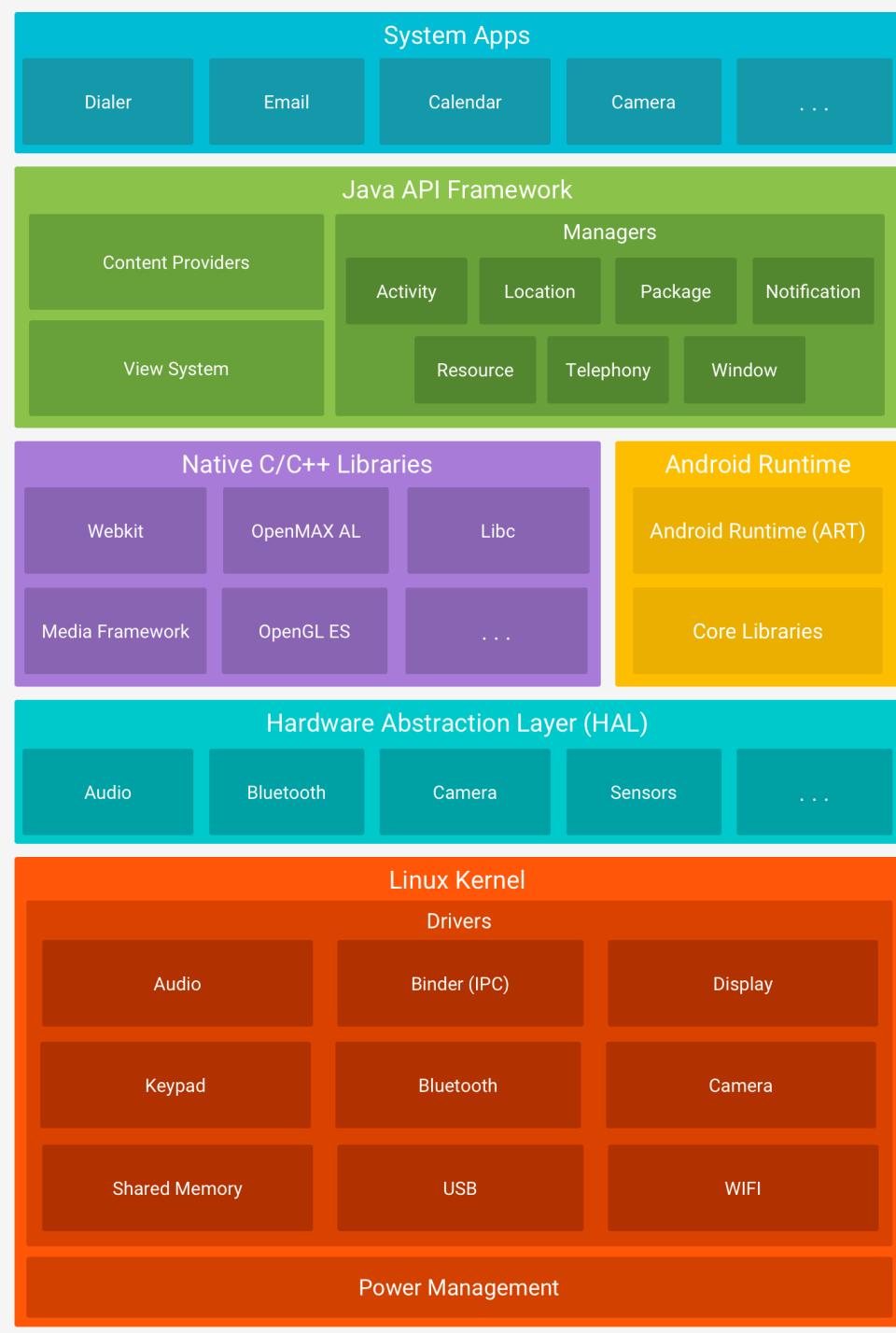


Figure 1. Android stack

# Il Linux Kernel



- Il fondamento della piattaforma Android è il kernel Linux
  - Ad esempio, Android Runtime (ART) si affida al kernel Linux per funzionalità sottostanti come il threading e la gestione della memoria di basso livello
- L'uso di un kernel Linux consente ad Android di sfruttare le principali funzionalità di sicurezza e consente ai produttori di dispositivi di sviluppare driver hardware per un noto kernel

# Il Linux Kernel

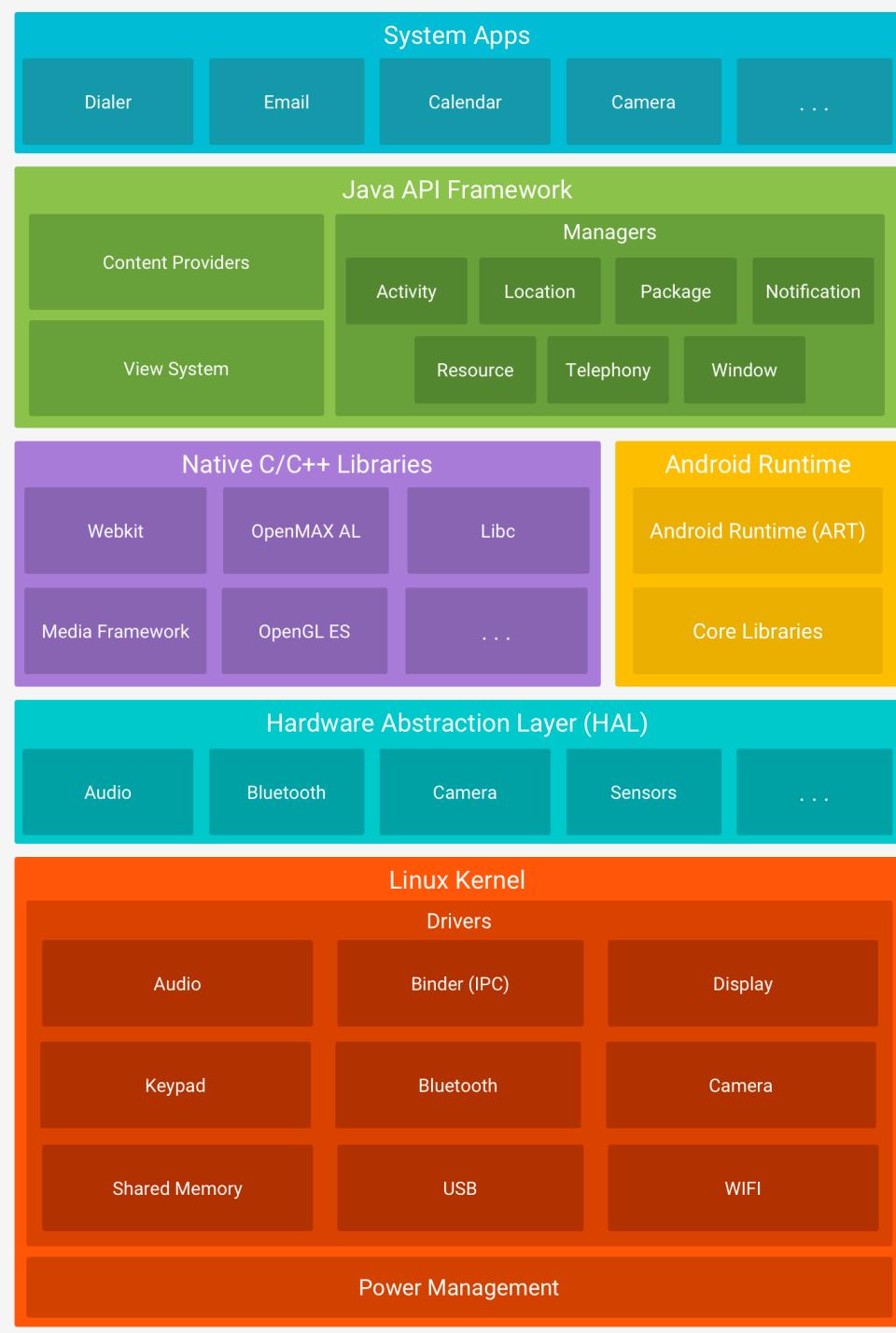
---

- Vantaggi
  - Portabilità (ovvero facile da compilare su diverse architetture hardware)
  - Sicurezza (ad es. Ambiente multi-processo sicuro)
  - Power Management
  - ART (Android Runtime) si basa sul kernel per thread e gestione della memoria
  - Produttori (Manufacturers) costruiti su un kernel affidabile

# Sicurezza del kerner

---

- Modello di permessi user-based
- I processi sono isolati
- Inter-process communication (IPC)
- Le risorse sono protette dagli altri processi
- Ogni applicazione ha il proprio User ID (UID)
- Sandbox
- Boot verificato



# Hardware Abstraction Layer (HAL)

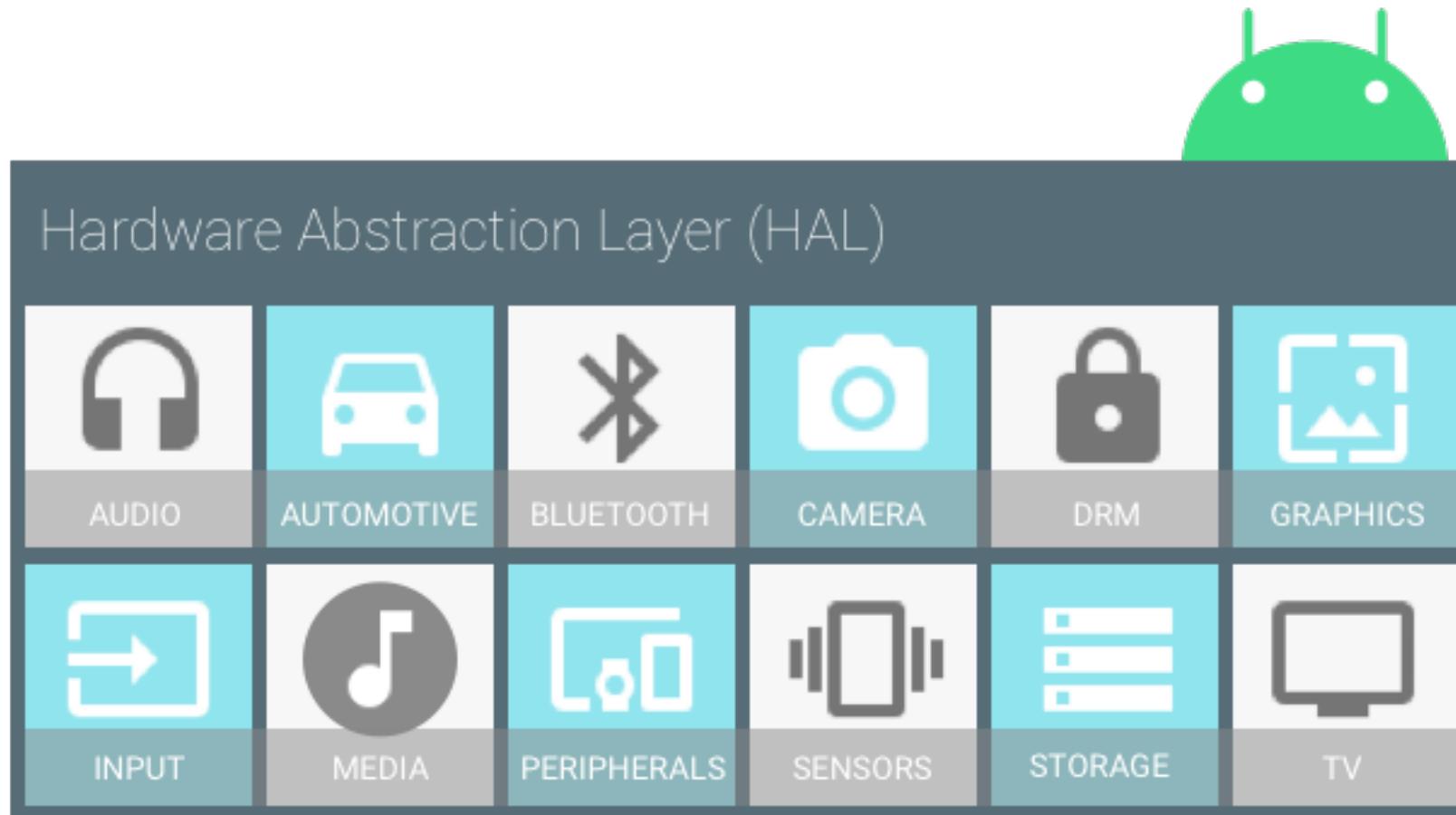
- Il livello di astrazione hardware (HAL) fornisce **interfacce standard** che espongono le capacità hardware del dispositivo al framework API Java di livello superiore
- L'HAL è costituito da più moduli di libreria, ognuno dei quali implementa un'interfaccia per un tipo specifico di componente hardware, come il modulo videocamera o bluetooth
- Quando un'API framework effettua una chiamata per accedere all'hardware del dispositivo, il sistema Android carica il modulo della libreria per quel componente hardware
  - Le implementazioni HAL sono raggruppate in moduli e caricate dal sistema Android al momento opportuno

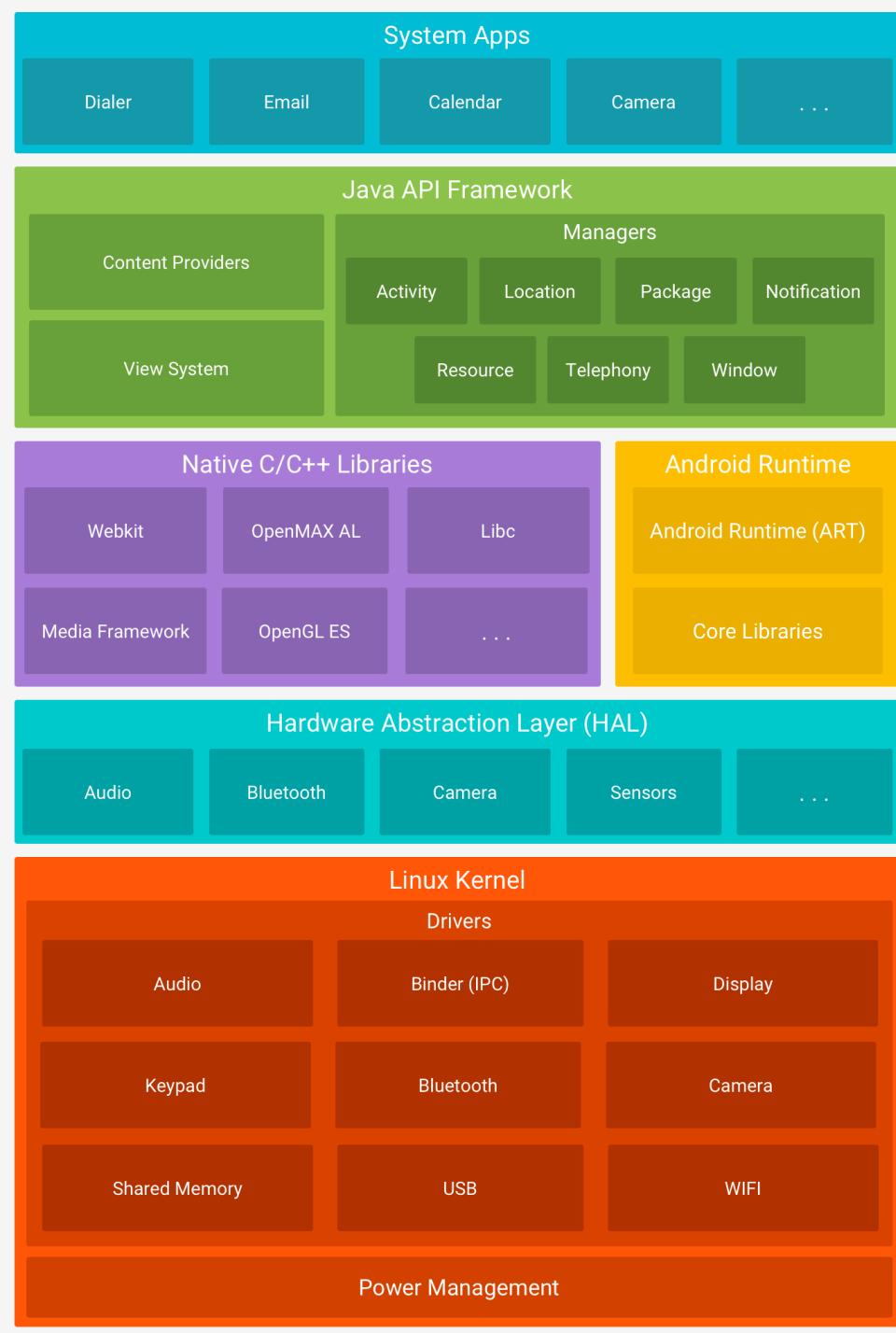
# Hardware Abstraction Layer (HAL)

---

- Vantaggi
  - Nasconde il vero device
  - Interfacce standard per esporre funzionalità di livello inferiore ad API di livello superiore
    - Ci sono interfacce standard che i manufacturers devono implementare (Android è indipendente dalle implementazioni di driver di livello inferiore)
    - Gli sviluppatori di applicazioni si basano su API comuni
    - A seconda dell'hardware, vengono caricate le librerie appropriate
  - L'uso di un HAL consente di implementare funzionalità senza influire o modificare il sistema di livello superiore

# Hardware Abstraction Layer (HAL)





# Android Runtime (1/2)

- Per i dispositivi che eseguono Android versione 5.0 (livello API 21) o superiore, ogni app viene eseguita nel proprio processo e con la propria istanza di Android Runtime (ART)
- ART è scritto per eseguire più macchine virtuali su dispositivi a memoria insufficiente eseguendo file DEX, un formato bytecode progettato appositamente per Android ottimizzato per un minimo ingombro di memoria
- Build toolchains, come Jack (NOTA: Jack toolchain è deprecato. Veniva usato come build toolchain per Android 6.0–8.1), compilano i sorgenti Java in bytecode DEX, che può essere eseguito sulla piattaforma Android

# Android Runtime (2/2)



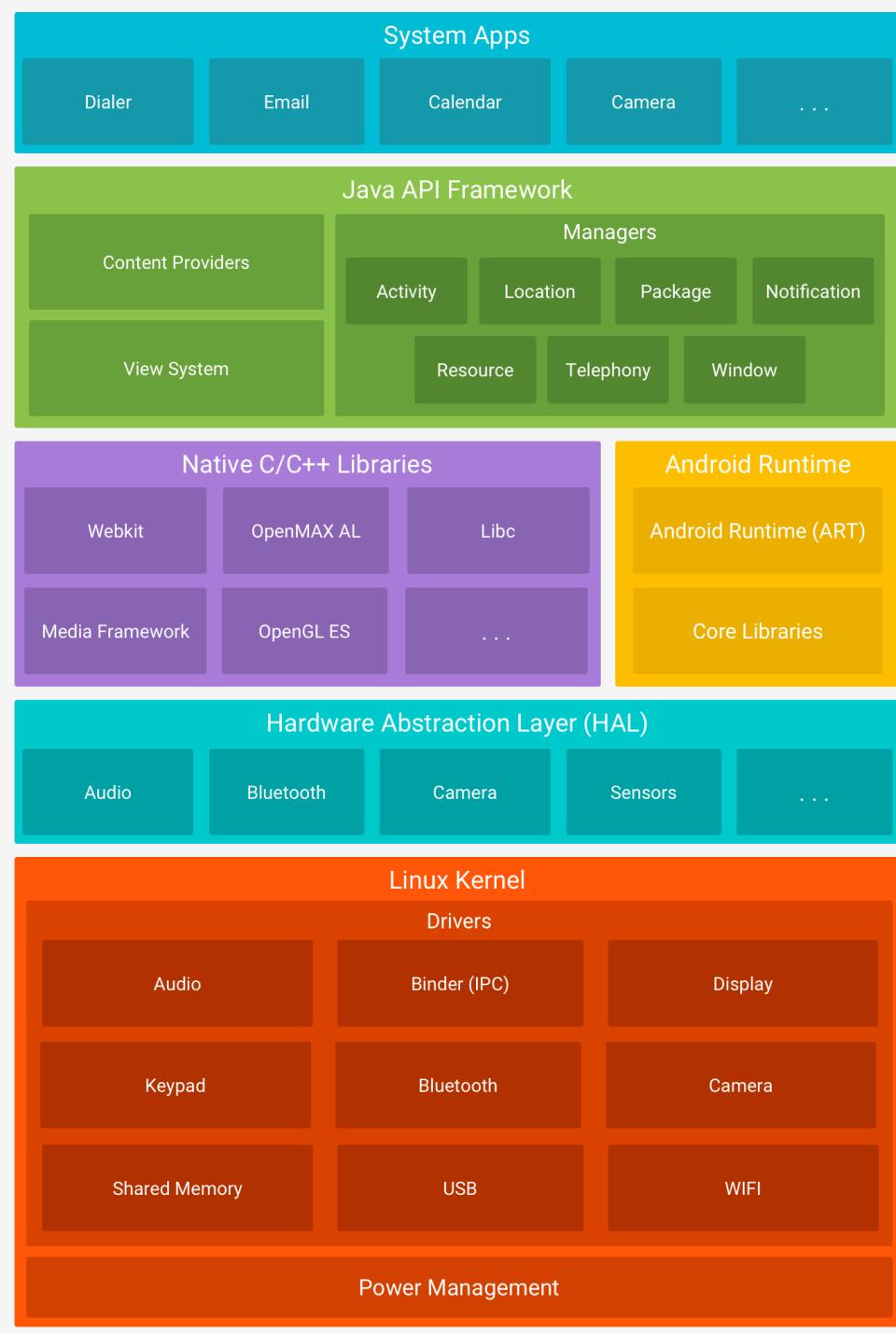
- Alcune delle principali caratteristiche di ART includono:
  - Compilazione anticipata (AOT) e just-in-time (JIT)
  - Garbage Collection ottimizzata (GC)
  - Su Android 9 (livello API 28) e versioni successive, conversione dei file in formato eseguibile Dalvik (DEX) di un pacchetto di app in un codice macchina più compatto
  - Migliore supporto per il debug, incluso un profiler di campionamento dedicato, eccezioni diagnostiche dettagliate e report sugli arresti anomali e la possibilità di impostare punti di controllo per monitorare campi specifici
- Android include anche una serie di librerie core di runtime che forniscono la maggior parte delle funzionalità del linguaggio di programmazione Java, incluse alcune funzionalità del linguaggio Java 8, che utilizza il framework API Java



# Native C/C++ Libraries

- Molti componenti e servizi del sistema Android di base, come ART e HAL, sono costruiti da codice nativo che richiede librerie native scritte in C e C++
- La piattaforma Android fornisce API del framework Java per esporre le funzionalità di alcune di queste librerie native alle app
  - Ad esempio, si può accedere a OpenGL ES tramite l'API OpenGL Java del framework Android per aggiungere supporto per il disegno e la manipolazione di grafica 2D e 3D nella tua app
- Se si vuole sviluppare un'app che richiede codice C o C++, si può utilizzare Android NDK per accedere ad alcune di queste librerie della piattaforma nativa direttamente dal tuo codice nativo

# Java API Framework



- L'intero set di funzionalità del sistema operativo Android è disponibile tramite API scritte in linguaggio Java
- Queste API costituiscono i mattoni necessari per creare app Android semplificando il riutilizzo di componenti e servizi di sistema modulari e core, che includono quanto segue
  - Un View System ricco ed estensibile che si può utilizzare per creare l'interfaccia utente di un'app, inclusi elenchi, griglie, caselle di testo, pulsanti e persino un browser Web incorporabile
  - Un Resource Manager che fornisce accesso a risorse non di codice come stringhe localizzate, grafici e file di layout
  - Un Notification Manager che consente a tutte le app di visualizzare avvisi personalizzati nella barra di stato
  - Un Activity Manager che gestisce il ciclo di vita delle app e fornisce un back stack di navigazione comune
  - Content Providers che consentono alle app di accedere ai dati di altre app, come l'app Contatti o di condividere i propri dati

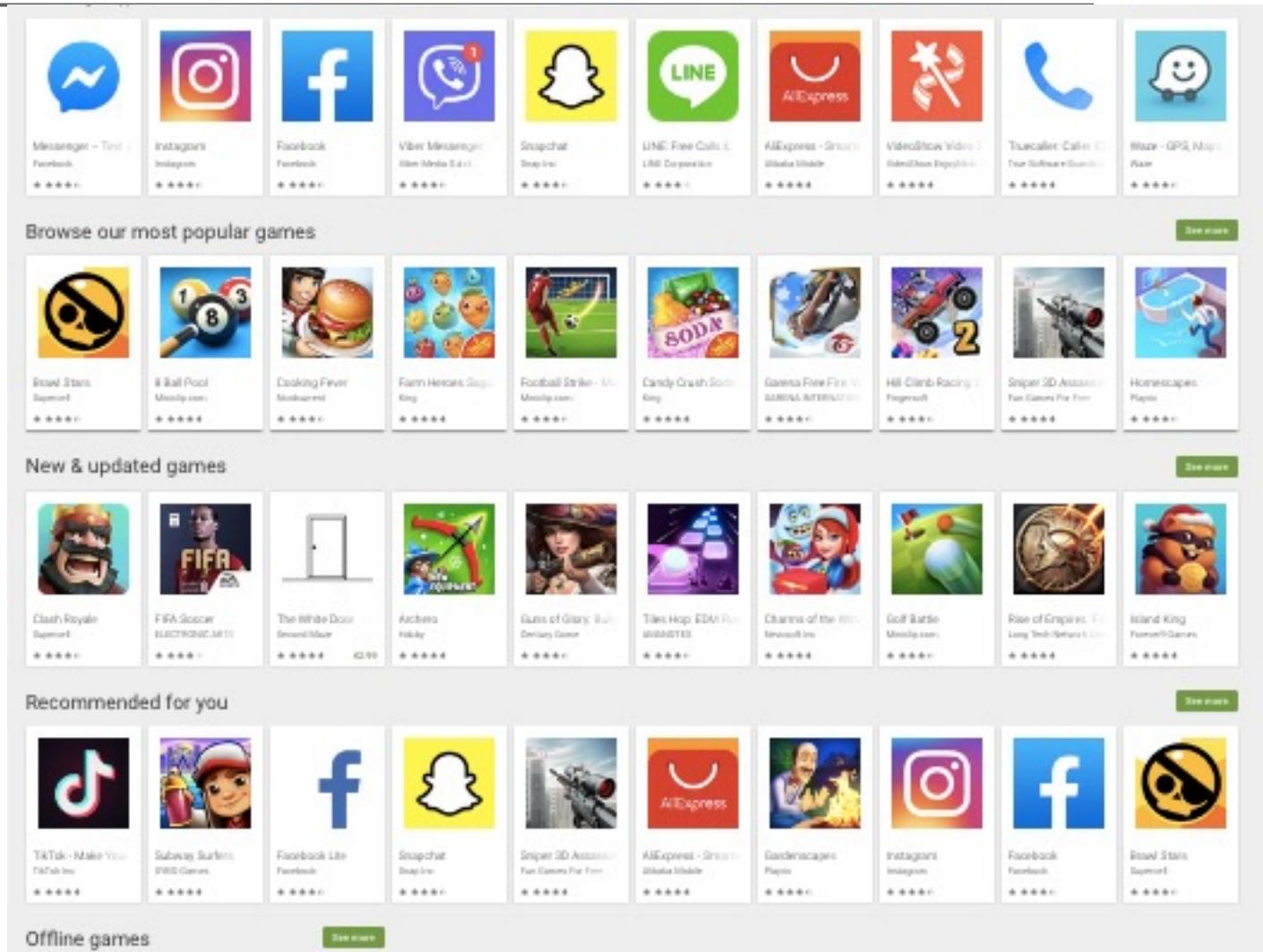
# System Apps

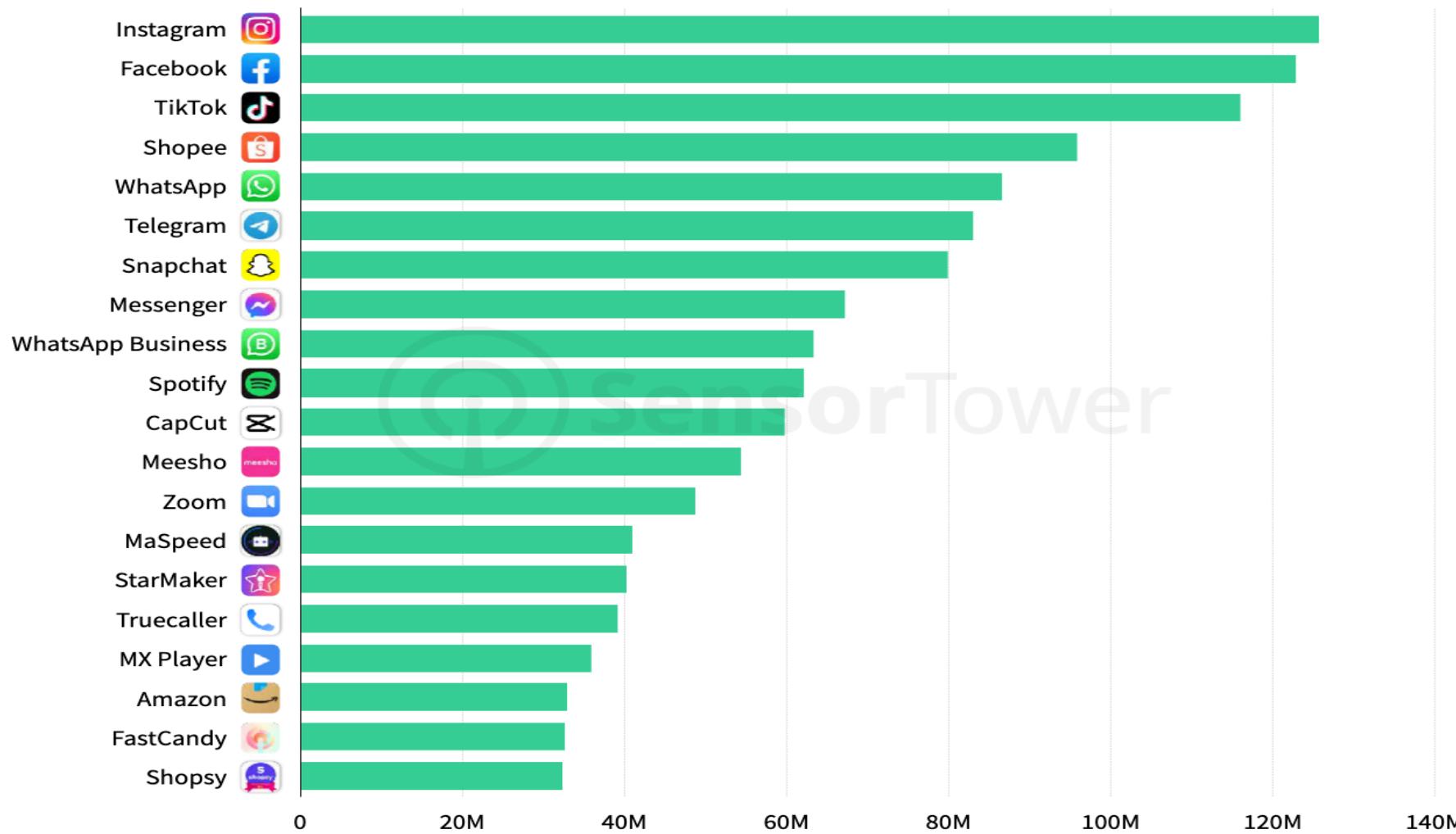


- Android viene fornito con una serie di app principali per e-mail, messaggi SMS, calendari, navigazione in Internet, contatti e altro
- Le app incluse con la piattaforma non hanno uno stato speciale tra le app che l'utente sceglie di installare. Pertanto un'app di terze parti può diventare il browser Web predefinito, il servizio di messaggistica SMS o persino la tastiera predefinita (si applicano alcune eccezioni, come l'app Settings del sistema)
- Le app di sistema funzionano sia come app per gli utenti sia per fornire funzionalità chiave a cui gli sviluppatori possono accedere dalla propria app.
  - Ad esempio, se la tua app desidera recapitare un messaggio SMS, non è necessario creare tale funzionalità da solo: puoi invece invocare qualunque app SMS sia già installata per recapitare un messaggio al destinatario specificato

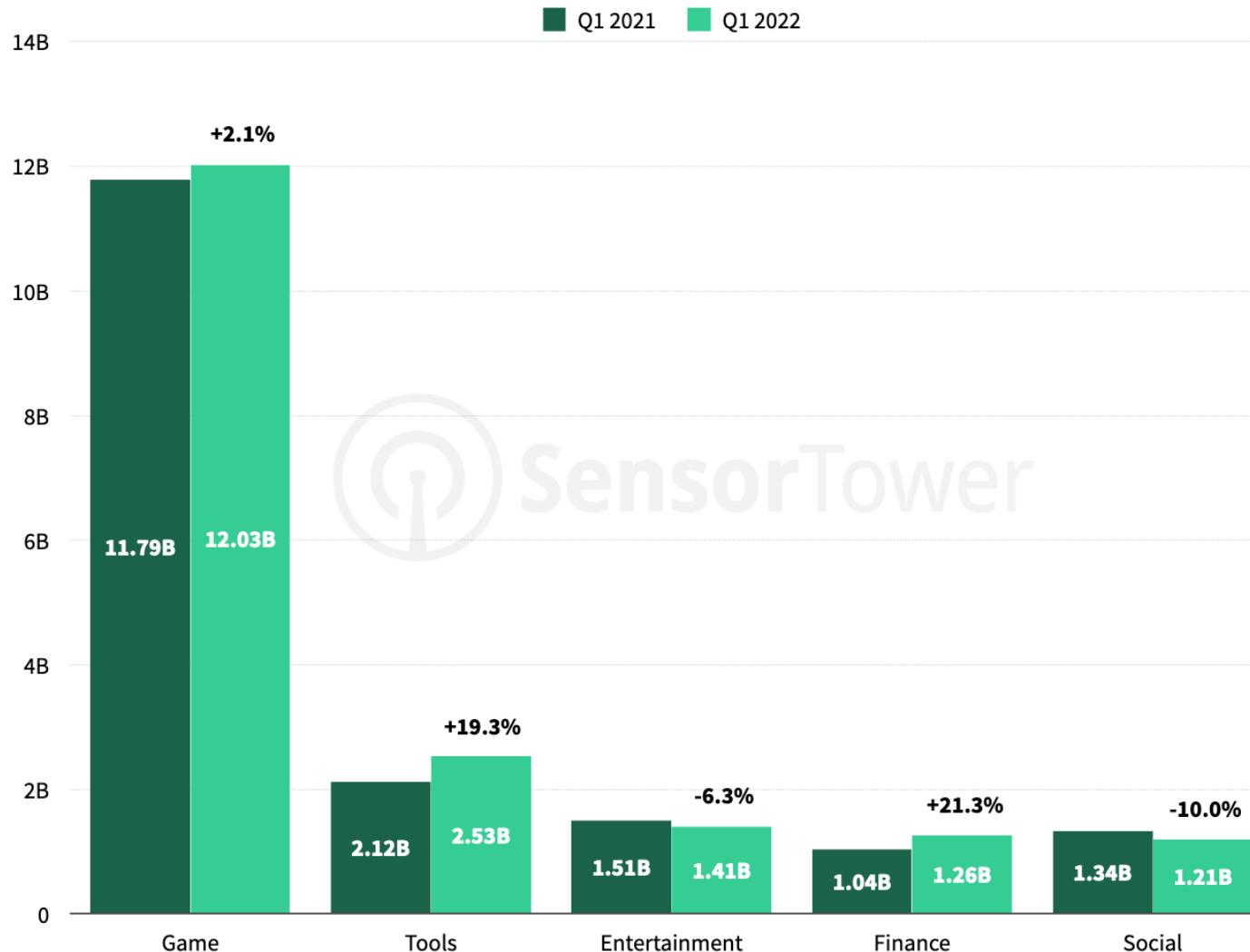
# App (applicazioni mobile)

- Applicazioni software che permettono di accedere ai più svariati servizi, contenuti interattivi e dinamici, giochi, contenuti di lavoro, ecc...





<https://go.sensortower.com/rs/351-RWH-315/images/Sensor-Tower-Q1-2022-Data-Digest.pdf>



<https://go.sensortower.com/rs/351-RWH-315/images/Sensor-Tower-Q1-2022-Data-Digest.pdf>

# Alcuni Fondamenti

---

- Le app Android native possono essere scritte sia in Java che in Kotlin (che in C++)
- Gli strumenti Android SDK compilano il codice insieme a tutti i file di dati e risorse in un APK, un pacchetto Android, che è un file di archivio con un suffisso .apk
  - Un file APK contiene tutto il contenuto di un'app Android ed è il file utilizzato dai dispositivi Android per installare l'app

# Kotlin o Java?

---



# Android's Kotlin-first approach

At Google I/O 2019, we announced that Android development will be increasingly Kotlin-first, and we've stood by that commitment. Kotlin is an expressive and concise programming language that reduces common code errors and easily integrates into existing apps. If you're looking to build an Android app, we recommend starting with Kotlin to take advantage of its best-in-class features.

In an effort to support Android development using Kotlin, we co-founded the [Kotlin Foundation](#) and have ongoing investments in improving compiler performance and build speed. To learn more about Android's commitment to being Kotlin-first, see [Android's commitment to Kotlin](#).



# Develop Android apps with Kotlin

---

Write better Android apps faster with Kotlin. Kotlin is a modern statically typed programming language used by over 60% of professional Android developers that helps boost productivity, developer satisfaction, and code safety.

## What does Kotlin code look like?

KOTLIN

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        ...  
        fab.setOnClickListener { view ->  
            Snackbar.make(view, "Hello $name", Snackbar.LENGTH_LONG).show()  
        }  
    }  
}
```

Nullable and NonNull types help reduce NullPointerExceptions

Use lambdas for concise event handling code

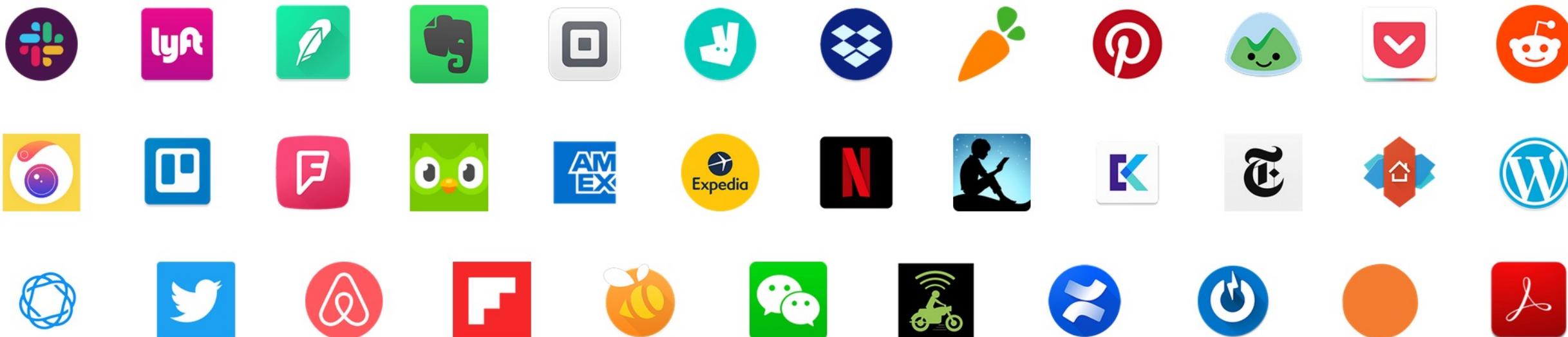
Use template expressions in strings to avoid concatenation

Semicolons are optional

# Apps built with Kotlin

Many apps are already built with Kotlin—from the hottest startups to Fortune 500 companies. Learn how Kotlin has helped their teams become more productive and write higher quality apps.

[See developer stories](#)



# Alcuni Fondamenti

---

- Ogni app Android vive nella propria **sandbox** di sicurezza, protetta dalle seguenti funzionalità di sicurezza Android
  - Il sistema operativo Android è un sistema Linux multiutente in cui **ogni app è un utente diverso**
  - Per impostazione predefinita, il sistema assegna a **ogni app un ID utente Linux univoco** (l'ID viene utilizzato solo dal sistema ed è sconosciuto all'app). Il sistema impone le autorizzazioni per tutti i file in un'app in modo che solo l'ID utente assegnato a quell'app possa accedervi
  - **Ogni processo ha la propria macchina virtuale (VM)**, quindi il codice di un'app viene eseguito separatamente da altre app
  - Per impostazione predefinita, ogni app viene eseguita nel proprio processo Linux. Il sistema Android avvia il processo quando uno dei componenti dell'app deve essere eseguito, quindi interrompe il processo quando non è più necessario o quando il sistema deve recuperare memoria per altre app

# Alcuni Fondamenti

---

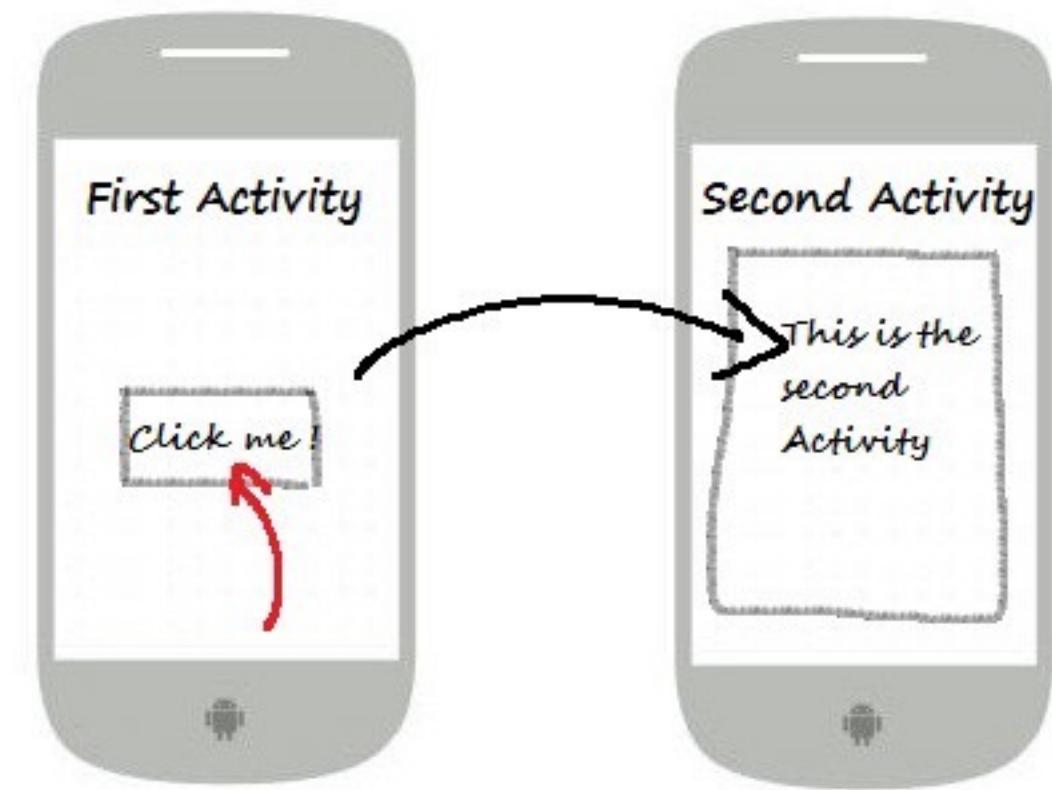
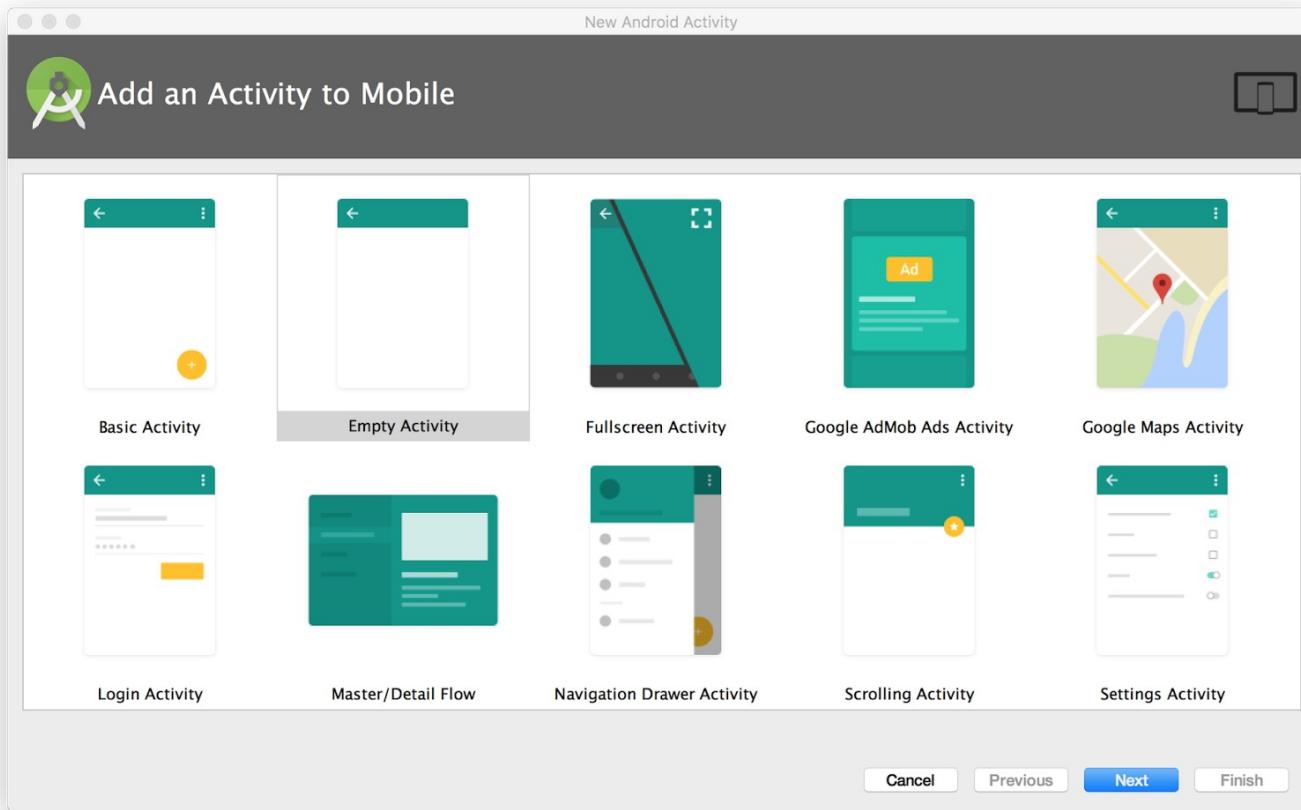
- Il sistema Android implementa il principio del **privilegio minimo**
  - ogni app, per impostazione predefinita, ha accesso solo ai componenti necessari per svolgere il proprio lavoro e non di più. Questo crea un ambiente molto sicuro in cui un'app non può accedere a parti del sistema per le quali non è stata autorizzata
- Tuttavia, un'app può richiedere l'autorizzazione per accedere ai dati del dispositivo come la posizione del dispositivo, la videocamera e la connessione Bluetooth
  - L'utente deve concedere esplicitamente queste autorizzazioni

# App components

- I componenti sono i blocchi/mattoni fondamentali di un'app Android
- Ogni componente è un punto di accesso (entry point) attraverso il quale il sistema o un utente possono accedere all'app
- Alcuni componenti dipendono da altri
- Ci sono 4 diversi tipi di componenti:
  - *Activities*
  - *Services*
  - *Broadcast receivers*
  - *Content providers*



# Activity



# Activities

---

- Un *Activity* è un entry point per l'interazione con l'utente
- Rappresenta una singola schermata con un'interfaccia utente
  - Ad esempio, un'app di posta elettronica potrebbe avere un'attività che mostra un elenco di nuove e-mail, un'altra attività per comporre un'e-mail e un'altra attività per la lettura di e-mail. Sebbene le attività lavorino insieme per formare un'esperienza utente coerente nell'app di posta elettronica, ognuna è indipendente dalle altre. Pertanto, un'app diversa può avviare una di queste attività se l'app di posta elettronica lo consente. Ad esempio, un'app della fotocamera può avviare l'attività nell'app di posta elettronica, activity di scrittura nuova email per consentire all'utente di condividere un'immagine

# Activity

---

- Un *activity* facilita le seguenti interazioni chiave tra sistema e app:
  - Tenere traccia di ciò che l'utente attualmente ha sullo schermo per garantire che il sistema continui a eseguire il processo che ospita l'attività
  - Sapere che i processi utilizzati in precedenza contengono elementi a cui l'utente può tornare (attività interrotte) e quindi dare priorità più alta a mantenere tali processi
  - Aiutare l'app a gestire il suo processo interrotto in modo che l'utente possa tornare alle attività con lo stato precedente ripristinato
  - Fornire un modo per le app di implementare i flussi utente tra loro e per il sistema di coordinare questi flussi (l'esempio più classico è «share»)

Si implementa *un'activity* come sottoclasse della classe **Activity**

# Services

---

- Un Service è un entry point generico per mantenere un'app in esecuzione in background
  - È un componente che viene eseguito in background per eseguire operazioni di lunga durata o eseguire lavori per processi remoti
- Un service non fornisce un'interfaccia utente
  - Ad esempio, un service potrebbe riprodurre musica in background mentre l'utente si trova in un'altra app, oppure potrebbe recuperare dati sulla rete senza bloccare l'interazione dell'utente con un'attività
- Un altro componente, come un'activity, può avviare il service e lasciarlo in esecuzione o associarsi ad esso per interagire con esso
- In realtà ci sono tre tipi di services che dicono al sistema come gestire un'app:
  - Foreground
  - Background
  - Bound services

## Started services

# Foreground Services

---

- Indicano al sistema di mantenerli in esecuzione fino al completamento del loro lavoro, anche se l'utente non sta interagendo con l'app
- Per riprodurre musica anche dopo che l'utente ha lasciato l'app o per le notifiche:
  - La riproduzione musicale è qualcosa di cui l'utente è direttamente a conoscenza, quindi in questo caso il sistema sa che deve fare davvero del suo meglio per mantenere in esecuzione il processo di quel servizio, perché l'utente se ne accorgerà se scompare

# Background services

---

- Sono services che eseguono in background senza che l'utente ne sia direttamente a conoscenza come in esecuzione
- In questo caso, il sistema ha più libertà nella gestione del suo processo
  - Può consentire che venga ucciso (e quindi riavviando il servizio qualche tempo dopo) se ha bisogno di RAM per cose che destano maggiore preoccupazione per l'utente

\*\*\*Note: If your app targets API level 26 or higher, the system imposes restrictions on running background services when the app itself isn't in the foreground. In most situations, for example, you shouldn't access location information from the background. Instead, schedule tasks using WorkManager.

# Bound services

---

- Vengono eseguiti perché un'altra app (o il sistema) ha affermato che desidera usufruire del servizio, chiamando bindService()
- Questo è fondamentalmente il servizio che fornisce un'API a un altro processo
  - Un bound service offre un'interfaccia client-server che consente ai componenti di interagire con il servizio, inviare richieste, ricevere risultati e persino farlo attraverso processi con comunicazione interprocesso (IPC)
- Il sistema quindi sa che esiste una dipendenza tra questi processi, quindi se il processo A è associato a un servizio nel processo B, sa che deve mantenere il processo B (e il suo servizio) in esecuzione per A
  - Un bound service viene eseguito solo finché un altro componente dell'applicazione è associato ad esso. È possibile associare più componenti contemporaneamente al servizio, ma quando vengono svincolati tutti, il servizio viene eliminato

# Services

---

- A causa della loro flessibilità (nel bene e nel male), i services si sono rivelati un elemento davvero utile
  - Sfondi animati, listener di notifiche, screen saver, metodi di input, servizi di accessibilità e molte altre funzionalità di sistema di base sono tutte costruite come servizi implementati dalle applicazioni e il sistema si impegna a gestire la loro esecuzione quando occorre
- Il componente è implementato come una sottoclassificazione di *Service*
- NOTA: dalla versione 5 in poi di Android è meglio usare un job di *JobScheduler* o un worker di Work Manager, per schedulare azioni Ciò permette di ridurre il consumo di batteria

# Broadcast receivers

---

- Un Broadcast receiver è un componente che consente al sistema di inviare eventi all'app al di fuori di un normale flusso utente, consentendo all'app di rispondere agli annunci di trasmissione a livello di sistema
- Poiché i broadcast receiver sono un entry ben definita nell'app, il sistema può fornire broadcasts anche alle app che non sono attualmente in esecuzione
  - ad esempio, un'app può schedulare un allarme per visualizzare una notifica per comunicare all'utente un evento imminente ... e consegnando tale allarme a un *BroadcastReceiver* dell'app, non è necessario che l'app rimanga in esecuzione fino a quando l'allarme si spegne

# Broadcast receivers

---

- Molti broadcast provengono dal sistema
  - ad esempio un broadcast che annuncia che lo schermo è spento, che la batteria è scarica o che è stata scattata un'immagine
- Le app possono anche avviare broadcast
  - ad esempio per far sapere ad altre app che alcuni dati sono stati scaricati sul dispositivo e sono disponibili per l'uso
- Sebbene broadcast receivers non visualizzino un'interfaccia utente, possono creare status bar notification per avvisare l'utente quando si verifica un evento di broadcast

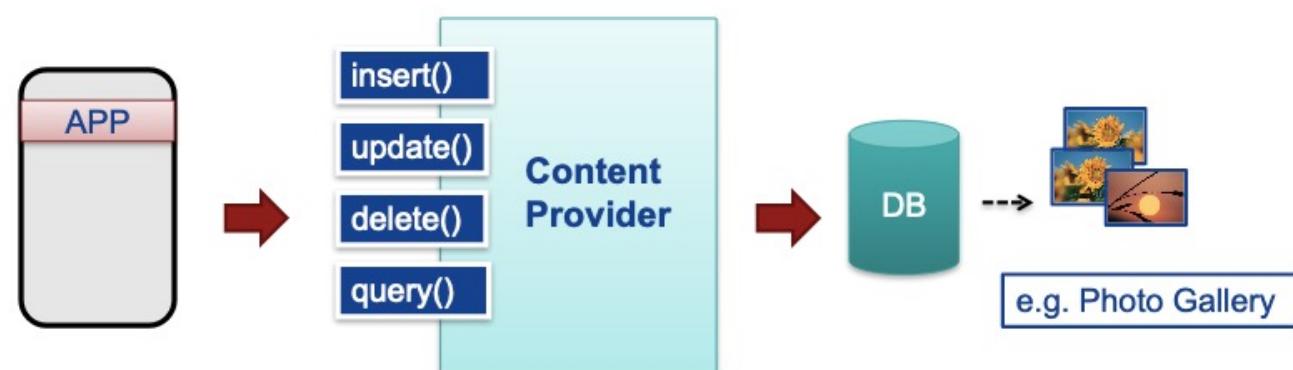
# Broadcast receivers

---

- Più comunemente, tuttavia, un broadcast receiver è solo un gateway per altri componenti ed è destinato a svolgere una quantità minima di lavoro
  - Ad esempio, potrebbe pianificare un job service per eseguire alcuni lavori in base all'evento (con *JobScheduler*)
- Un broadcast receiver viene implementato come sottoclassificazione di *BroadcastReceiver* e ogni trasmissione viene erogata come oggetto *Intent*

# Content providers

- Un content provider gestisce un set condiviso di dati dell'app che è possibile archiviare nel file system, in un database SQLite, sul Web o in qualsiasi persistent storage location a cui l'app può accedere
- Tramite il content provider, altre app possono eseguire query o modificare i dati se il provider di contenuti lo consente
  - Ad esempio, il sistema Android fornisce un content provider che gestisce le informazioni di contatto dell'utente. Pertanto, qualsiasi app con le autorizzazioni appropriate può richiedere al fornitore di contenuti, *ContactsContract.Data*, di leggere e scrivere informazioni su una determinata persona



# Content providers

---

- Si può pensare a un content provider come un'astrazione su un database. Tuttavia, ha uno scopo principale diverso dal punto di vista della prospettiva della progettazione di sistema
- Per il sistema, un content provider è un entry point in un'app per la pubblicazione di dati, identificati da uno schema URI
- I content provider sono utili anche per leggere e scrivere dati privati nella la propria app e non condivisi
- Un Content Provider è implementato come sottoclasse di *ContentProvider* e deve implementare un set standard di API che consenta ad altre app di eseguire transazioni

# App components

---

- Un aspetto unico del design del sistema Android è che qualsiasi app può avviare il componente di un'altra app
  - Ad esempio, se si desidera che l'utente scatti una foto con la fotocamera del dispositivo, probabilmente c'è un'altra app che lo fa e l'app può usarla invece di sviluppare una nuova activity. Ciò può essere fatto semplicemente attivando l'activity nell'app della camera. Appena scattata la foto, la foto viene restituita all'app che si sta creando per essere gestita. Per l'utente è come se la camera fosse parte dell'app che sta usando

# App components

---

- Quando il sistema avvia un componente, avvia il processo per quell'app (se non è già in esecuzione) e crea un'istanza delle classi necessarie per il componente. Ad esempio, se l'app avvia l'attività nell'app della fotocamera che scatta una foto, tale attività viene eseguita nel processo che appartiene all'app della fotocamera, non nel processo dell'app. Pertanto, a differenza delle applicazioni nella maggior parte degli altri sistemi, le app Android non hanno un unico punto di ingresso (non esiste una funzione main ())
- Poiché il sistema esegue ciascuna app in un processo separato con autorizzazioni per i file che limitano l'accesso ad altre app, l'app non può attivare direttamente un componente da un'altra app. Tuttavia, il sistema Android può. Per attivare un componente in un'altra app, recapita un messaggio al sistema che specifica l'intenzione di avviare un componente particolare. Il sistema attiva quindi il componente per te

# *Intent*

---

- Per attivare 3 dei 4 componenti disponibili, ovvero «*activities*, *services*, e *broadcast receivers*» si usa un messaggio asincrono chiamato *intent*
  - che descrive un'azione da eseguire, inclusi i dati su cui agire, la categoria del componente che dovrebbe eseguire l'azione e altre istruzioni
- Gli intenti (*intent*) associano i singoli componenti tra loro in fase di esecuzione. Si possono considerare come i messaggeri che richiedono un'azione a componenti (indipendentemente dal fatto che il componente appartenga alla tua app o ad un'altra)
- Un Intent viene creato con un oggetto *Intent*, che definisce un messaggio per attivare un componente specifico (explicit intent) o un tipo specifico di componente (implicit intent)

# *Intent*

---

- Per **activity** e **service**, un *intent* definisce **l'azione da eseguire** (ad esempio, per visualizzare – *view* - o inviare – *send* - qualcosa) e può specificare l'URI dei dati su cui agire, una tra le cose che un component avviato potrebbe avere necessità di conoscere
  - Ad esempio, un intent potrebbe trasmettere una richiesta per un'attività per mostrare un'immagine o per aprire una pagina Web
- In alcuni casi, è possibile avviare un'attività (**activity**) per ricevere un risultato, nel qual caso l'attività restituisce anche il risultato in un Intent
  - Ad esempio, è possibile emettere l'intenzione di consentire all'utente di scegliere un contatto personale e di restituirlo all'utente. L'intent di ritorno include un URI che punta al contatto scelto
- Per **broadcast receivers**, l'intent definisce semplicemente **l'annuncio da trasmettere**
  - Ad esempio, una trasmissione per indicare che la batteria del dispositivo è scarica include solo una stringa di azione nota che indica che la batteria è scarica

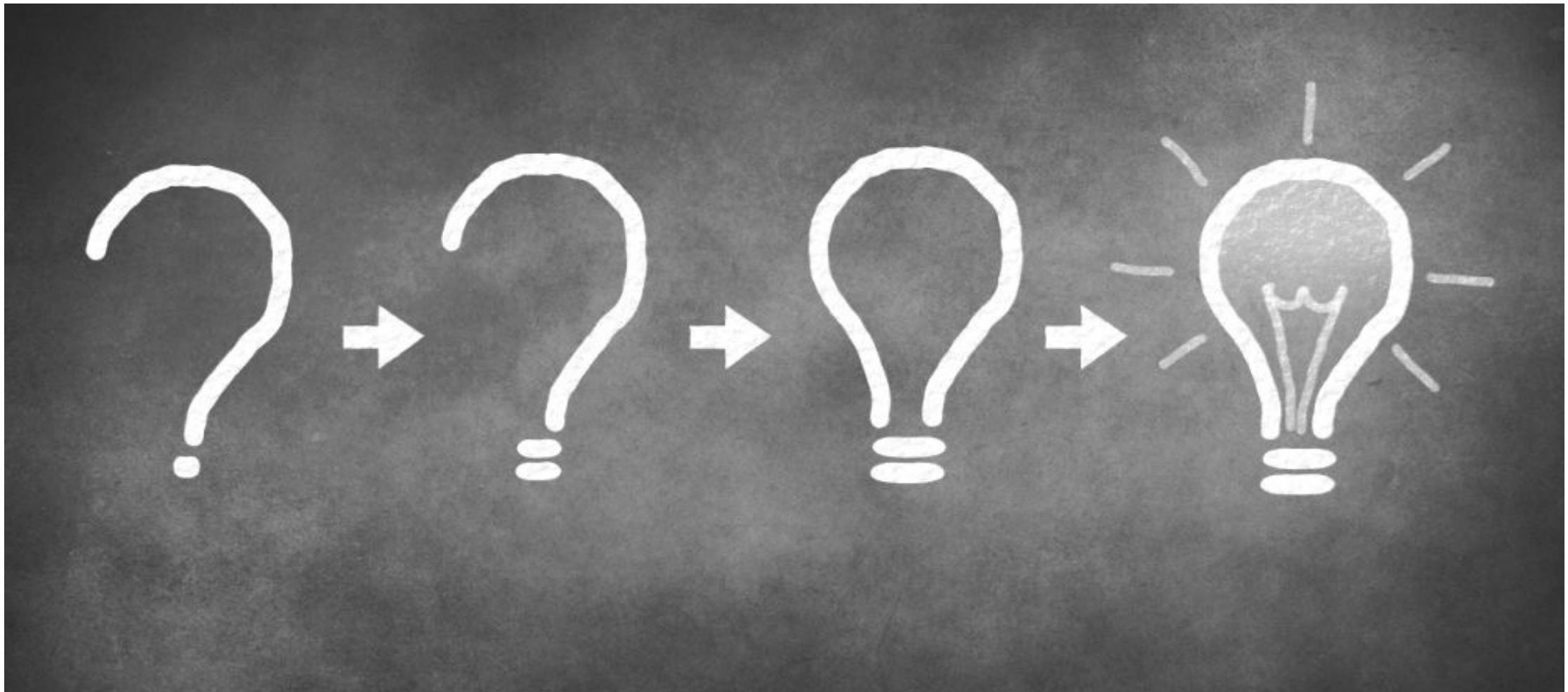
# *Intent e Content providers*

---

- A differenza delle altre tre componenti, i content provider non vengono attivati dagli intent. Vengono invece attivati quando vengono scelti come target da una richiesta di un *ContentResolver*
- Il *ContentResolver* gestisce tutte le transazioni dirette con il content provider in modo tale che il componente che sta eseguendo transazioni con il fornitore non debba farlo ma debba invece solo chiamare metodi sull'oggetto ContentResolver
  - Ovvero lascia uno **strato di astrazione** tra il fornitore di contenuti e il componente che richiede informazioni (per motivi di sicurezza)

# Domande?

---



# Riferimenti

---

- <https://developer.android.com/>
- <https://earthlymission.com/android-version-history-a-visual-timeline/>
- <https://developer.android.com/guide/platform>
- <https://developer.android.com/guide/components/fundamentals>
- <https://android-developers.googleblog.com/2020/02/Android-11-developer-preview.html>
- ...