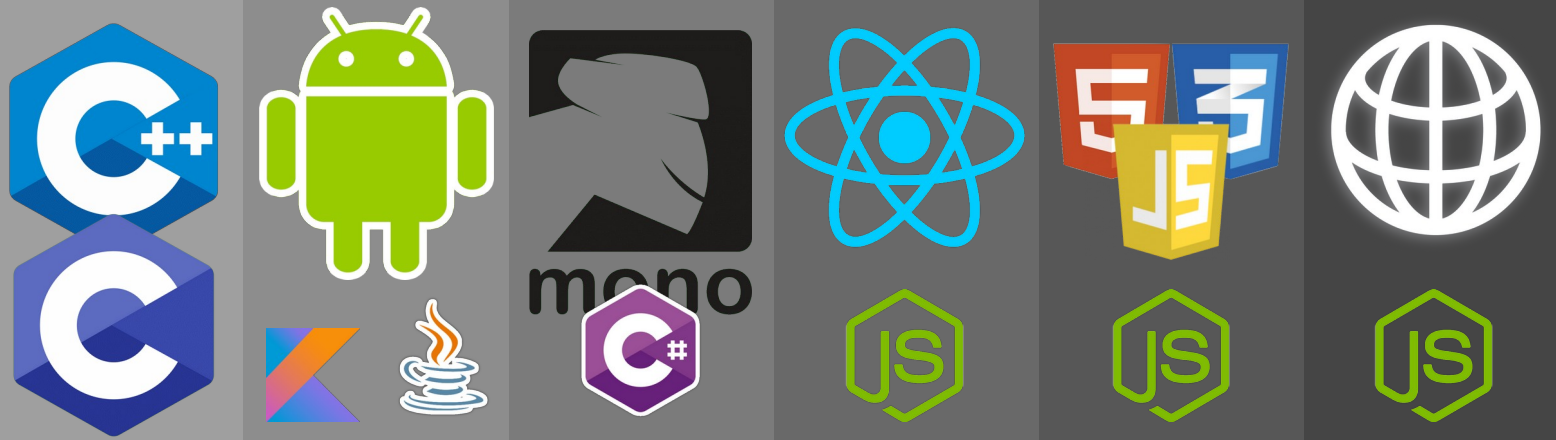
50* Sfumature di “native”

* in realtà più 6 che non 50



Titoli di testa

Federico Di Gregorio

- fog@dndg.it
- github.com/fogzot

Il talk

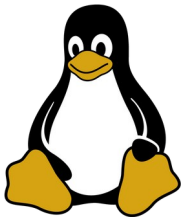
- Linux e Android
- Nativo o non nativo
- Esempio di codice

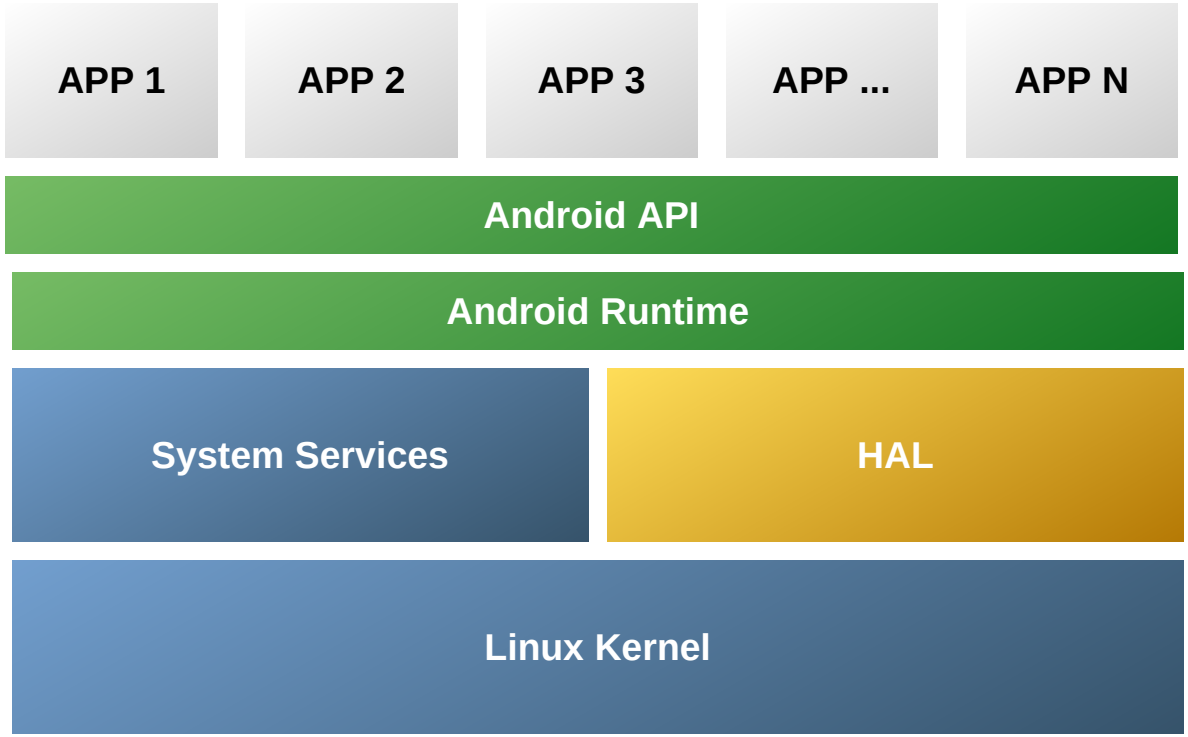
Linux e Android

Android Open Source Project

- Kernel Linux + alcune patch (C)
- Servizi di sistema (e.g., wpa_supplicant)
- Hardware Abstraction Layer (C/C++)
- Android runtime e servizi applicativi (C/C++)
- Android API (Java)
- Applicazioni di base (Java/Kotlin)

Architettura Android





Restrizioni

Sandbox applicazioni

- Ogni applicazione vive in una sandbox separata
- Comunicazione limitata dalle API

Tutte le API pubbliche sono basate sul C++ (basso livello) o Java (UI e servizi).

Cosa significa “native”

Native fa riferimento ad una piattaforma:

- “...Gimp è la versione nativa di PS per Gnome...”
- “...usa le API native di Windows...”
- “...è una applicazione nativa per MacOS X...”

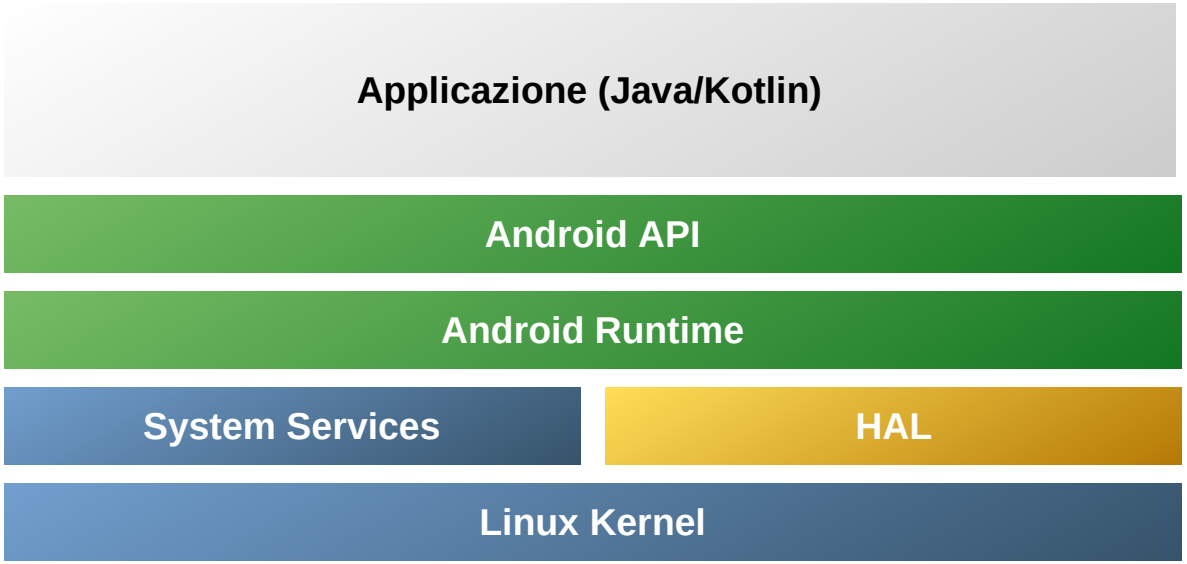
In alcuni casi la distinzione tra nativo e non-nativo è sottile.

Sfumature

Dal più nativo al meno nativo:

- Android NDK – C/C++
- Android SDK – Java/Kotlin
- Compilatore/Binding – qualsiasi linguaggio, e.g. C#
- Bridge API – qualsiasi linguaggio, e.g. Javascript
- Embedded Web Application – Javascript
- Progressive Web Application – Javascript

Android SDK


Applicazione (Java/Kotlin)
Android API
Android Runtime
System Services
HAL
Linux Kernel

Android SDK

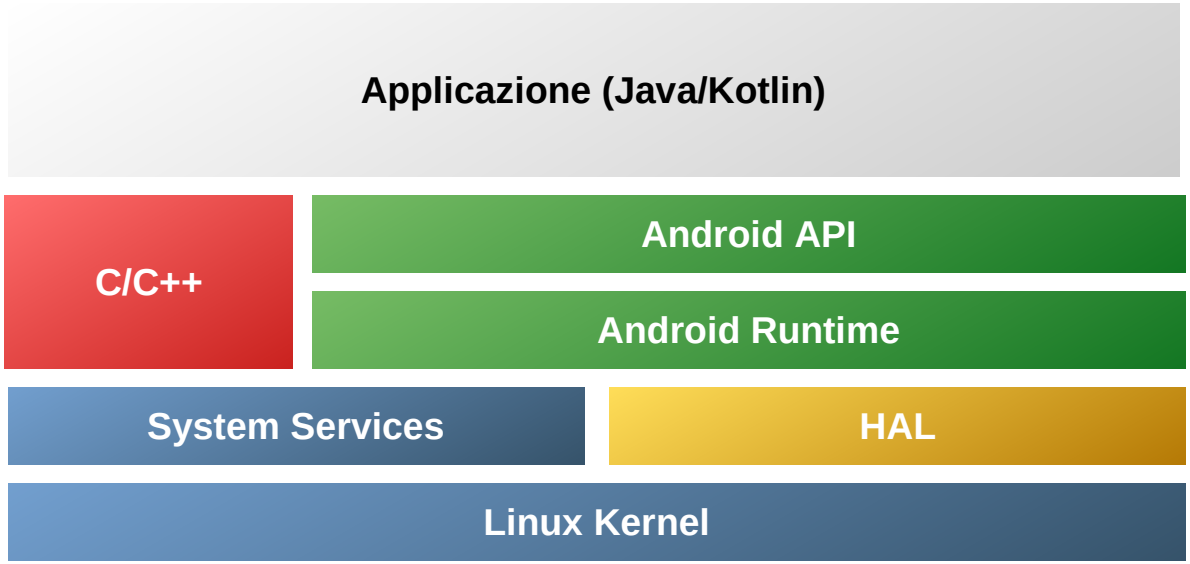
PRO

- Ambiente di sviluppo integrato
- Accesso a tutta la piattaforma via API

CONTRO

- Programmazione in Java o Kotlin
- Nessun accesso a basso livello

Native Development Kit



Native Development Kit

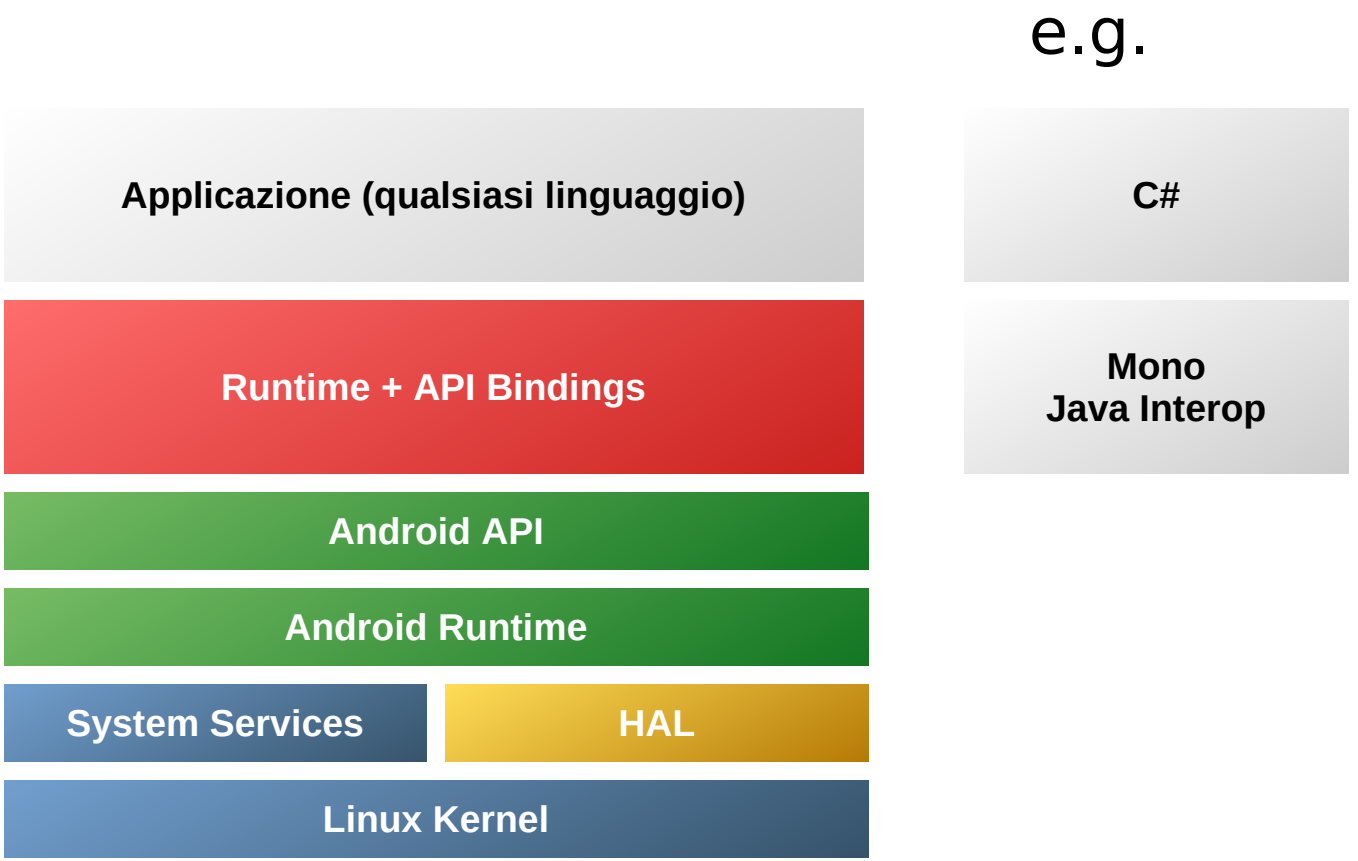
PRO

- Ambiente di sviluppo integrato
- Accesso a basso livello

CONTRO

- Limitato a C e C++
- Necessità di Java/Kotlin per l’UI

Binding API


e.g.
Applicazione (qualsiasi linguaggio)
C#
Runtime + API Bindings
Mono
Java Interop
Android API
Android Runtime
System Services
HAL
Linux Kernel

Binding API

PRO

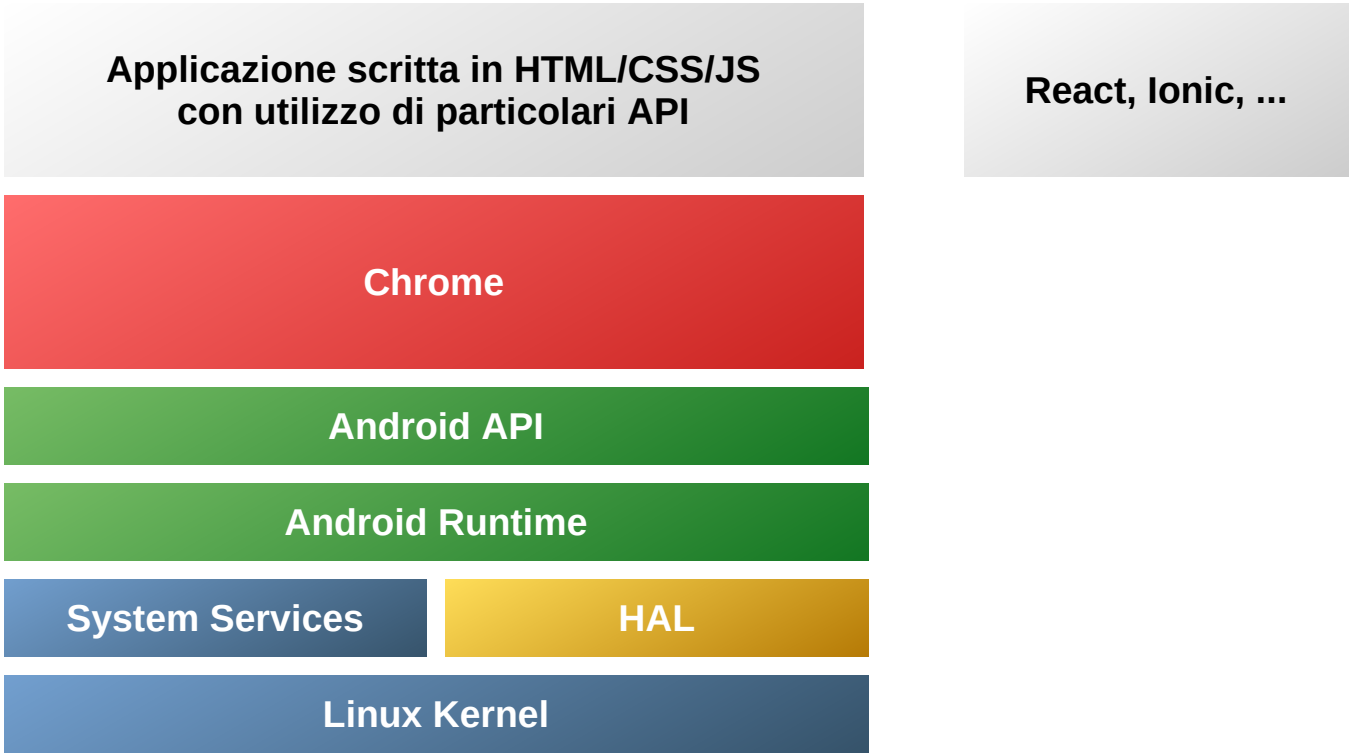
- Libertà nella scelta di ambiente di sviluppo e linguaggio di programmazione

CONTRO

- Accesso alle API limitato dai binding
- In genere nessun accesso a basso livello

Progressive Web Application

e.g.



Binding API

PRO

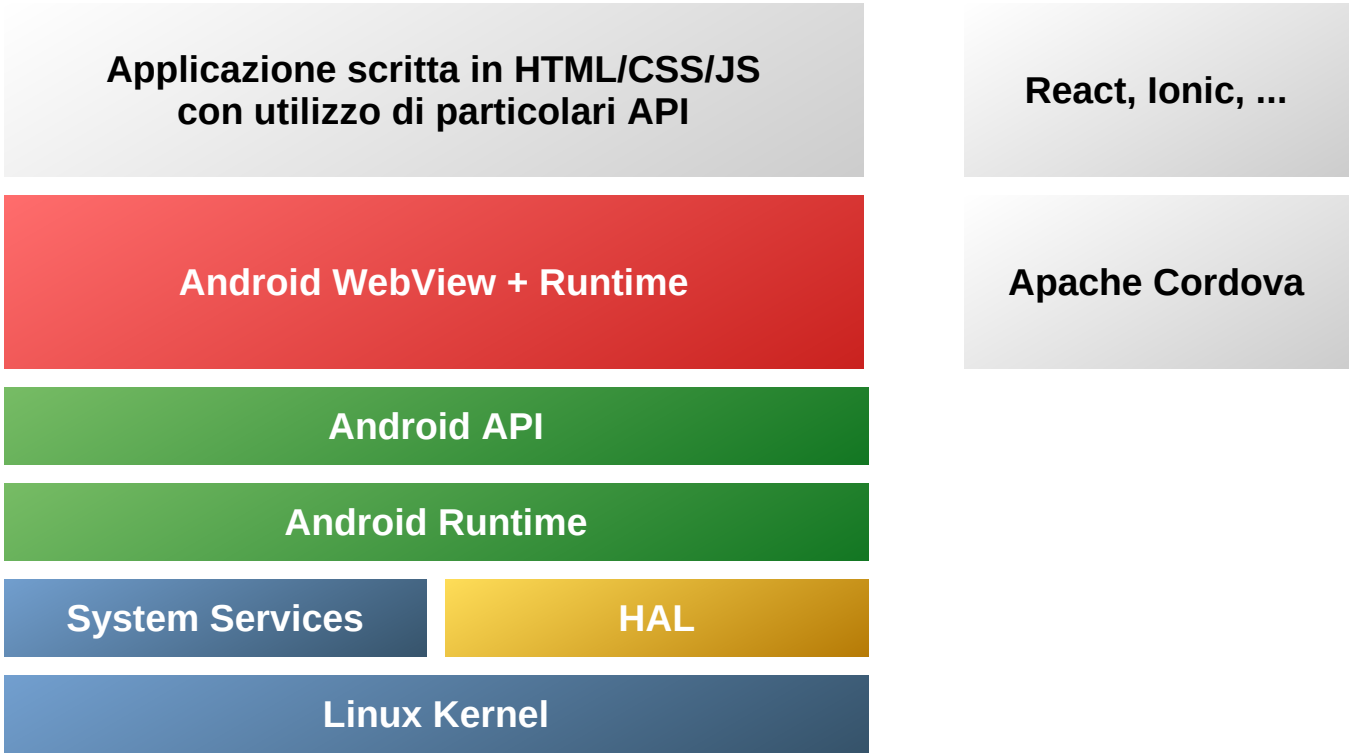
- Sviluppo in HTML/CSS/JS
- Nessuna installazione richiesta

CONTRO

- Accesso limitato alle API native
- UX non completamente in linea con Android

Embedded Web Application

e.g.



Embedded Web Application

PRO

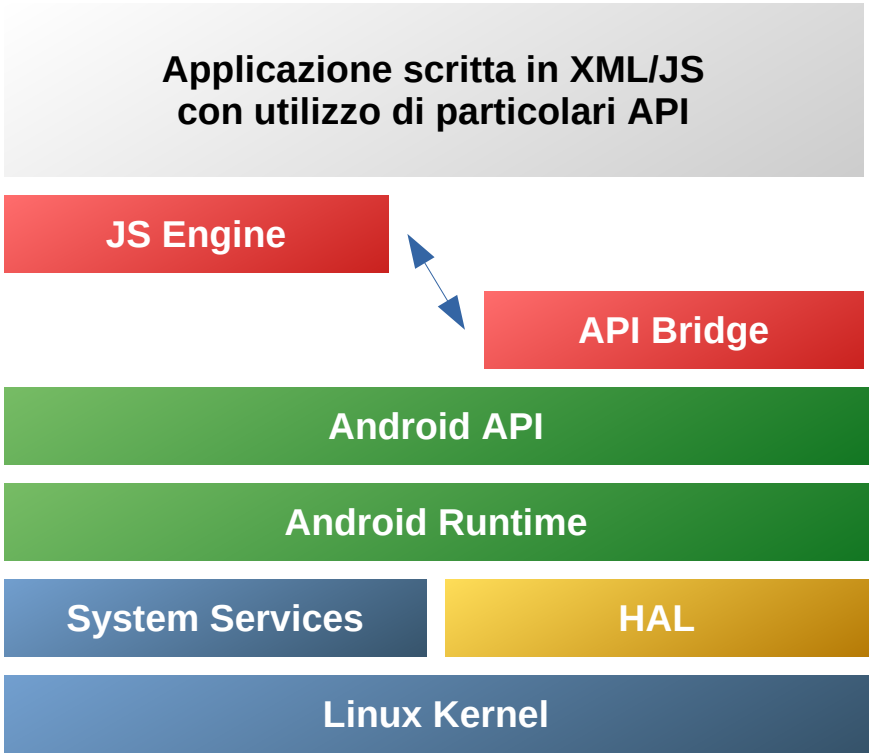
- Sviluppo in HTML/CSS/JS
- Accesso alle API previste dal runtime

CONTRO

- UX non completamente in linea con Android

API Bridge

e.g.



React Native,
Vue Native, ...

React Runtime

API Bridge

PRO

- Libertà nella scelta di ambiente di sviluppo e linguaggio di programmazione (in teoria, vedi CONTRO)
- Accesso alle API previste dal bridge

CONTRO

- “ma è sempre Javascript...”

Copiare, capire, estendere*

I concetti che stanno dietro allo sviluppo di un API bridge sembrano complessi ma si può sviluppare un pre-pre-pre-prototipo in qualche centinaio di righe di Java.

- Motore per eseguire il codice dell'app
- Runtime che pilota la UI di Android
- Canale di comunicazione tra motore e runtime

*versione buona di Embrace, extend, and extinguish

Esempio

- Linguaggio del runtime: Java
- Linguaggio per l’app: Javascript
- Motore di esecuzione: WebView
- Canale di comunicazione: Channel Messaging API

Titoli di coda

Domande?

- Per esempio “ma davvero hai messo un riferimento ad un musical nel codice?”
- Oppure domande più serie…

Codice di esempio e slides su:

- http://github.com/fogzot/linuxday2019