15-Applicazioni Mobili Cross-Platform

Promessa

- **WORA (Write Once, Run Anywhere)**: Creare un singolo codice sorgente per costruire e distribuire un'applicazione su diverse piattaforme. Tuttavia, questo slogan non è sempre veritiero poiché è necessario testare e fare il debug su ogni piattaforma.

Punti di Forza del Cross-Platform

- **Riduzione dei costi e del tempo di sviluppo**: Consente di ridurre il costo e il tempo necessario per sviluppare un'applicazione.
- **Riduzione delle competenze richieste**: Le competenze necessarie per lo sviluppo sono ridotte poiché il framework gestisce gran parte del lavoro.
- **Uniformità delle funzionalità per gli utenti finali**: Gli utenti finali possono ottenere la stessa funzionalità su dispositivi diversi, dove ciò è fattibile.

Debolezze del Cross-Platform

- **Funzionalità limitate** a quelle comuni a tutte le piattaforme.
- **Necessità di imparare nuovi linguaggi/framework**: I linguaggi e i framework utilizzati devono essere neutri rispetto al sistema operativo.
    - Difficoltà nell'integrare metodologie/librerie esistenti.
    - Dipendenza da terze parti (il framework).
- **Difficoltà nell'ottimizzare le prestazioni**: L'approccio neutrale può rendere difficile l'ottimizzazione delle prestazioni, rischiando di creare una UX insoddisfacente.
    - **Test e debug su più piattaforme**: Assicurare la qualità del risultato richiede test e debug su tutte le piattaforme.

Differenze

Ci sono principalemente due approcci per realizzare applicazioni:

- Web based
- **Native** → si basano sul sistema operativo con differenze sostanziali

Differenze di Architettura

Le applicazioni native sono basate su sistema operativo con importanti differenze.

- **Modello di esecuzione**: Processi liquidi in Android contro processi monolitici tradizionali in iOS.
    - **Concorrenza:** può essere diversa
- **Comunicazione tra processi**: ApplicationExtension (plugins) vs Intents & Binder.
- **Gestione della memoria**: Garbage collector vs ARC (Automatic Reference Counting).
    - Librerie dinamiche: Gestite diversamente.

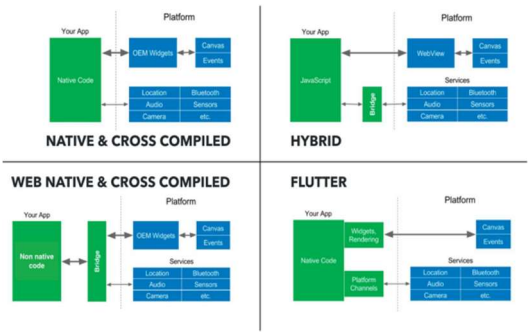
Differenze di SDK (Softare development kit)

- **Linguaggi di programmazione**: Java/Kotlin per Android, Objective-C/Swift per iOS. C++ è disponibile su entrambe le piattaforme ma con vincoli.
- **Componenti simili ma non identici**: Widget elementari, algoritmi di layout, eventi touch/tastiera, animazioni, servizi di piattaforma (localizzazione, sensori, batteria, rete, ecc.).

Differenze di UX

- **Identità e quota di mercato**: Ogni OS propone il proprio stile e temi, pattern di interazione e un set di servizi di supporto integrati (store, mappe e navigazione, archiviazione cloud, riconoscimento vocale, ecc.).
- **Elementi di navigazione e interazione**: NavigationBar vs ActionBar, Tabs vs Segmented controls, Floating action button vs "call to action" button, Hamburger menu vs tabbed menu.

Tecnologie Cross-Platform

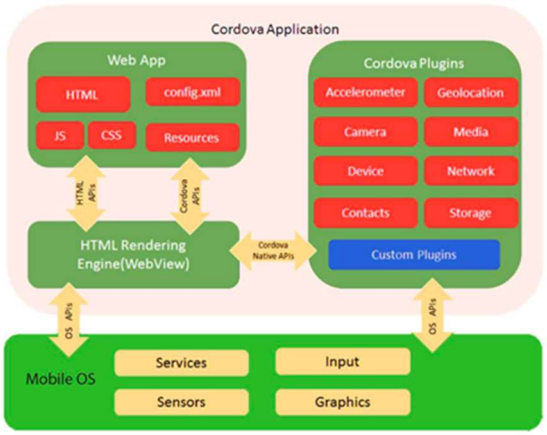


- **WebView-based frameworks**: Cordova, Ionic.
- **Native-widget-based frameworks**: Xamarin, React Native.
- **Custom-widget-based frameworks**: Unity, Flutter.

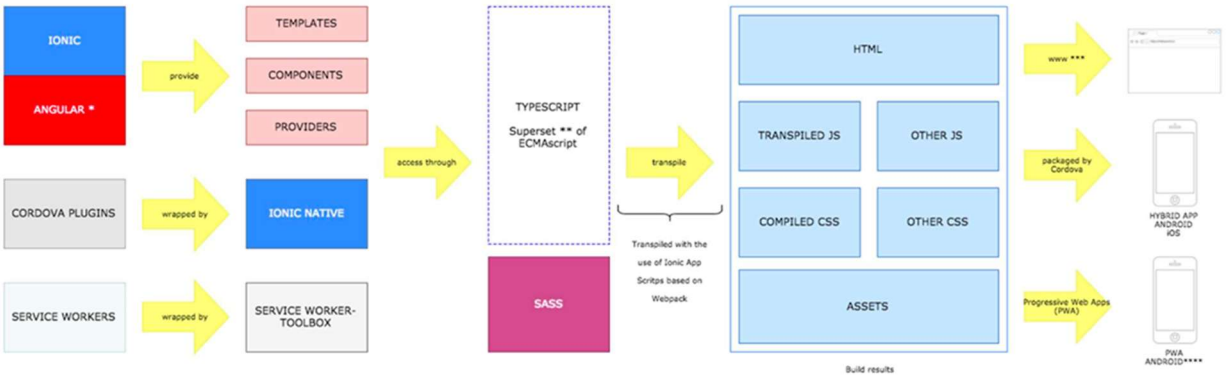
WebView

- Componente software per costruire app usando tecnologie web: HTML, CSS, JavaScript.
- Basato sulla **libreria Webkit**: Utilizzata da Safari e Chrome.
- **Motore di rendering**: Fornisce la superficie di visualizzazione, personalizza l'aspetto e la sensazione, gestisce l'interazione con l'utente e la presentazione multimediale.
- **Estensioni JavaScript personalizzate (plug-in)**: Consentono di importare/esportare dati dalla piattaforma nativa.

1. **Apache Cordova**
   - Framework per costruire applicazioni basate su **WebView**: Fornisce uno scheletro dell'applicazione che carica un set di plugin e mostra una WebView.
   - **Meccanismo di estensione**: Consente al codice dell'applicazione (JavaScript) di accedere alle funzionalità native (sensori del dispositivo, file system, fotocamera).



2. **Ionic**
   - **Ambiente di sviluppo open source**: Basato sia su Cordova che sul framework Angular.
   - Descrizione della struttura dell'app tramite **componenti web modulari**: Ogni componente ha il proprio template UI e logica dell'applicazione.
   - **Supporto per PWA e applicazioni desktop**: Integrato con il framework Electron.

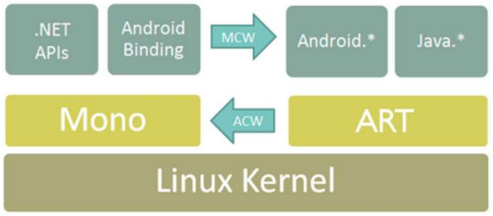


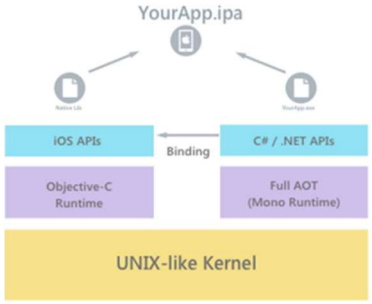
NATIVE WIDGETS:

Si prova a scrivere un'applicazione che descrive il tipo di component e poi generiamo il component in base al device su cui stiamo lavorando.

- Positivo: posso basarmi su linguaggi esistenti
- Negativo: può essere necessario del native code

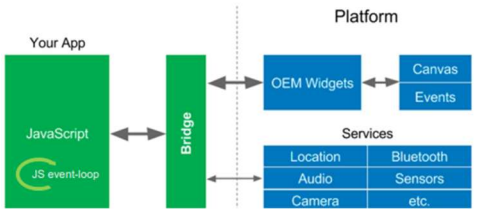
1. **Xamarin**
   - **Piattaforma basata sulla macchina virtuale Mono**: La logica dell'applicazione è scritta in C#.
   - **Esecuzione MSIL Mono**: Richiede direttamente i servizi al kernel quando possibile, altrimenti utilizza un bridge per le funzionalità native.





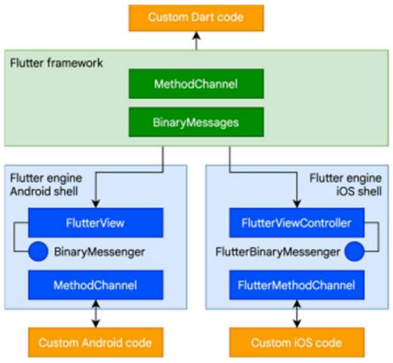
Funziona in maniera completamente diversa in iOS dove Mono non si riesce ad eseguire → dunque Mono viene modificata prima di essere inviata ad iOS in modo tale che ottenga solo c#

2. **React Native**
   - **Framework sviluppato da Facebook**: Basato su componenti UI forniti dalla piattaforma sottostante.
   - **Meccanismi di React**: Componenti semplici combinabili in stile funzionale, utilizza un broker interno (JS bridge) per coordinare lo scambio di dati tra il thread principale e il thread JavaScript.



   - c'è un motore javascript che ha un ponte verso platform dove i widgets sono creati. Inoltre, il Widget connette anche ai servizi come Location, Bluetooth, etc

3. **Flutter**
   - **SDK open source sviluppato da Google**: Supporta Android, iOS, web e piattaforme desktop.
   - **Linguaggio di programmazione Dart**: Applicazioni compilate AOT in codice nativo, utilizza la libreria grafica Skia per disegnare widget direttamente su una canvas.
   - **Canali di comunicazione tra piattaforma nativa e motore Dart**.
   - ha un'architettura differente:
     - comunicazione tra i vari layer usando un Buffer che si basa su dati BinaryMessage



Platform	Xamarin. Forms	React Native	Flutter	Native Android	Native iOS
Start-up time	Slow	Medium	Medium	Fast	Fast
App size	Big	Medium	Medium	Small	Small
Memory usage	Medium	Medium	High	Small	Small
CPU usage	Medium to high	Medium to high	Medium	Medium	Medium
Development experience	Medium	Medium	Very good	Good	Good