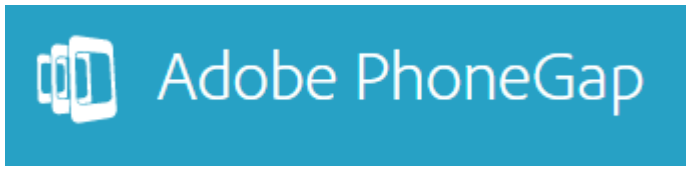
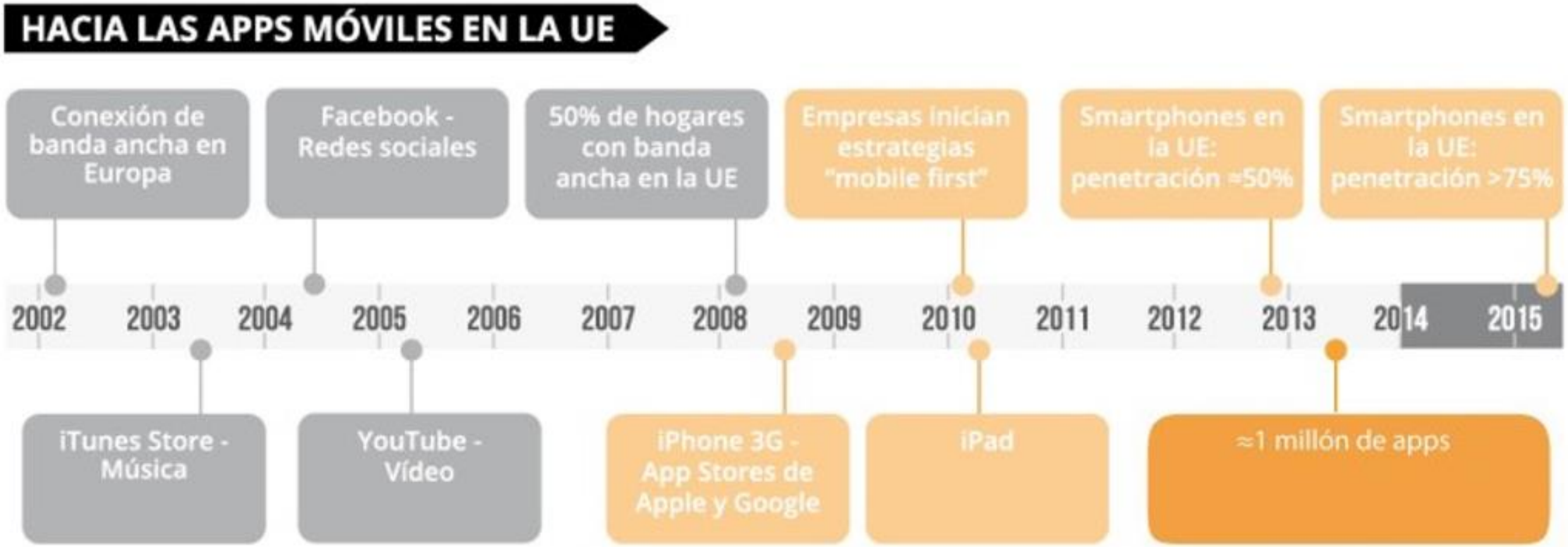
Desarrollo de aplicaciones multiplataforma



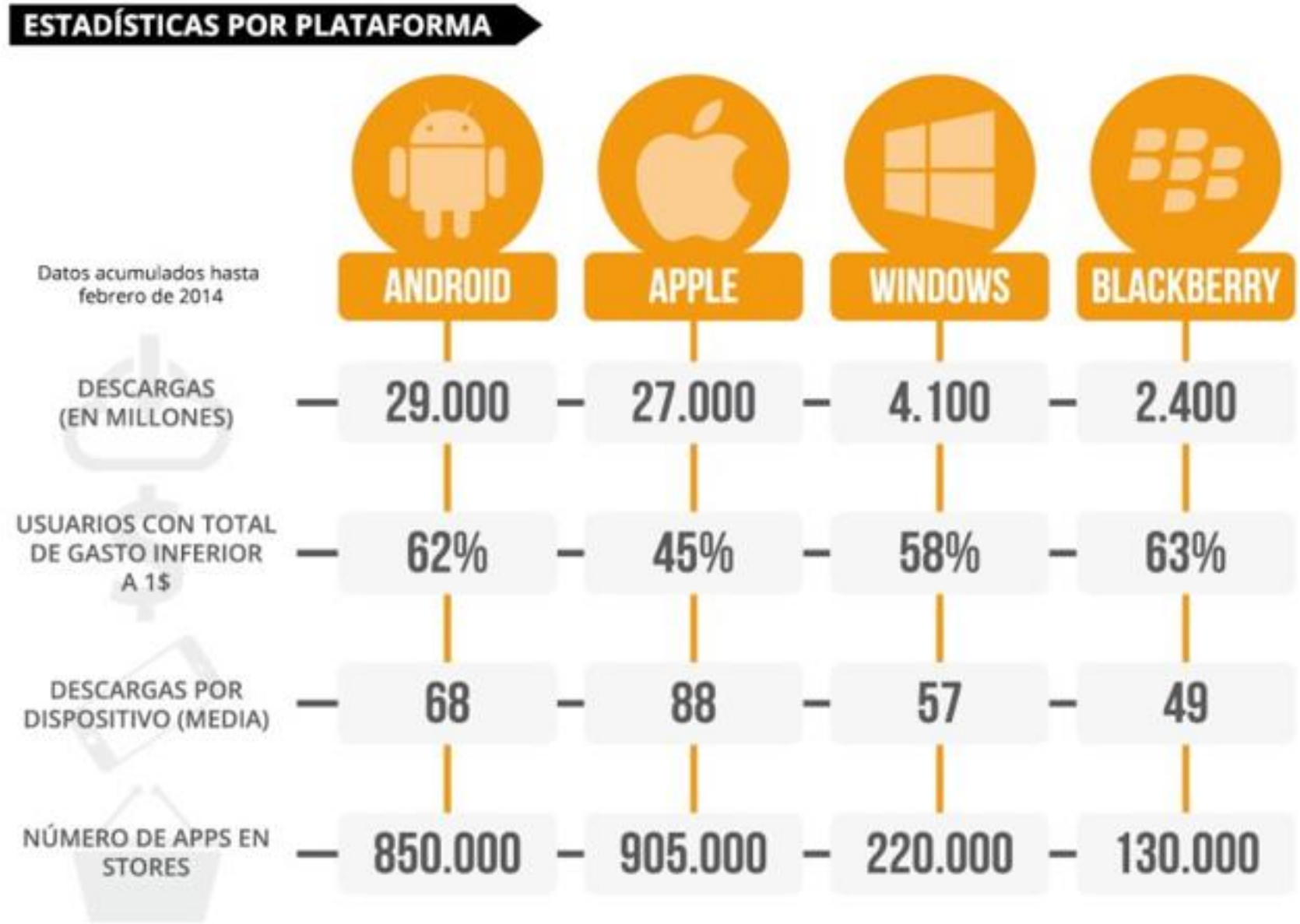
Desarrollo de aplicaciones móviles

Consiste en el conjunto de procesos y procedimientos involucrados en la escritura de software para pequeños dispositivos inalámbricos, tales como smartphones o tablets.

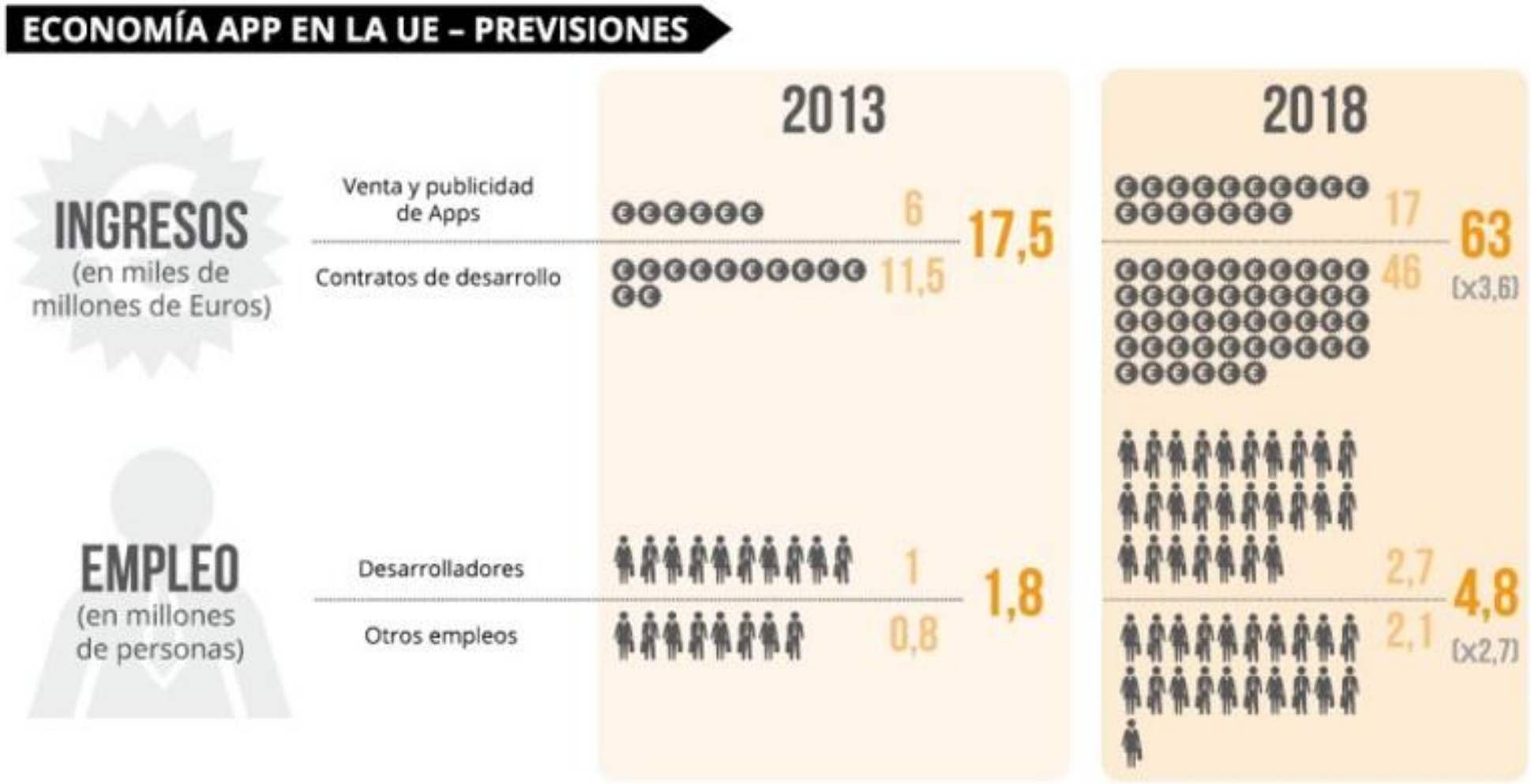
- **Desarrollo de aplicaciones móviles**
  - Desarrollo **similar** al desarrollo de **aplicaciones web**.
  - **Diferencia** fundamental: las aplicaciones (apps) móviles a menudo se escriben específicamente para **aprovechar** las **características únicas** que ofrece un dispositivo móvil en particular.
    - Ejemplos:
      - Uso de acelerómetro en un juego.
      - Uso de GPS para posicionar.
  - Para asegurar que las aplicaciones muestren un **rendimiento óptimo** en un dispositivo determinado se desarrolla la aplicación de forma nativa en ese dispositivo.


HACIA LAS APPS MÓVILES EN LA UE
Conexión de banda ancha en Europa
Facebook - Redes sociales
50% de hogares con banda ancha en la UE
Empresas inician estrategias "mobile first"
Smartphones en la UE: penetración ≈50%
Smartphones en la UE: penetración >75%
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
iTunes Store - Música
YouTube - Vídeo
iPhone 3G - App Stores de Apple y Google
iPad
≈1 millón de apps

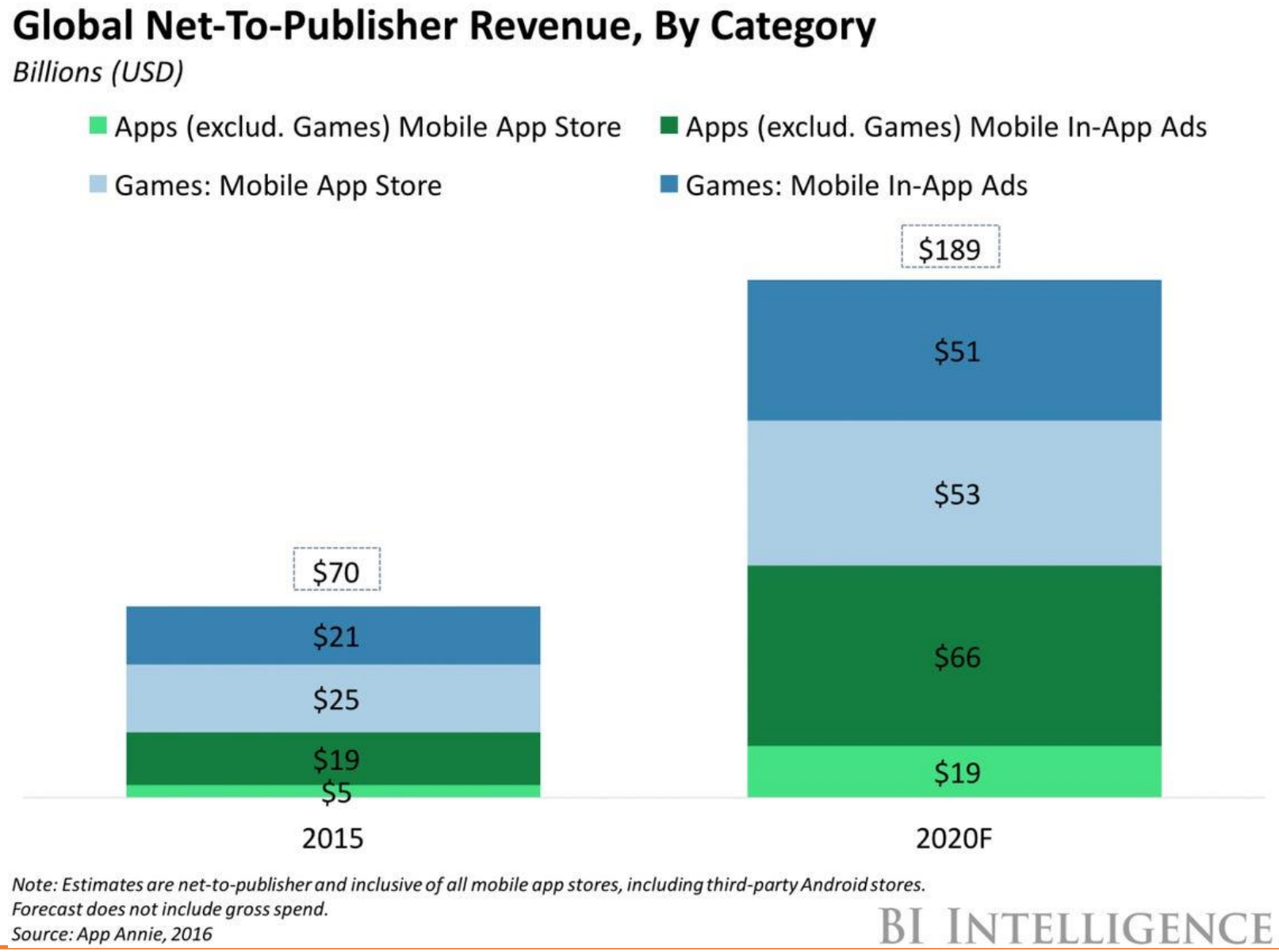
Desarrollo de aplicaciones multiplataforma



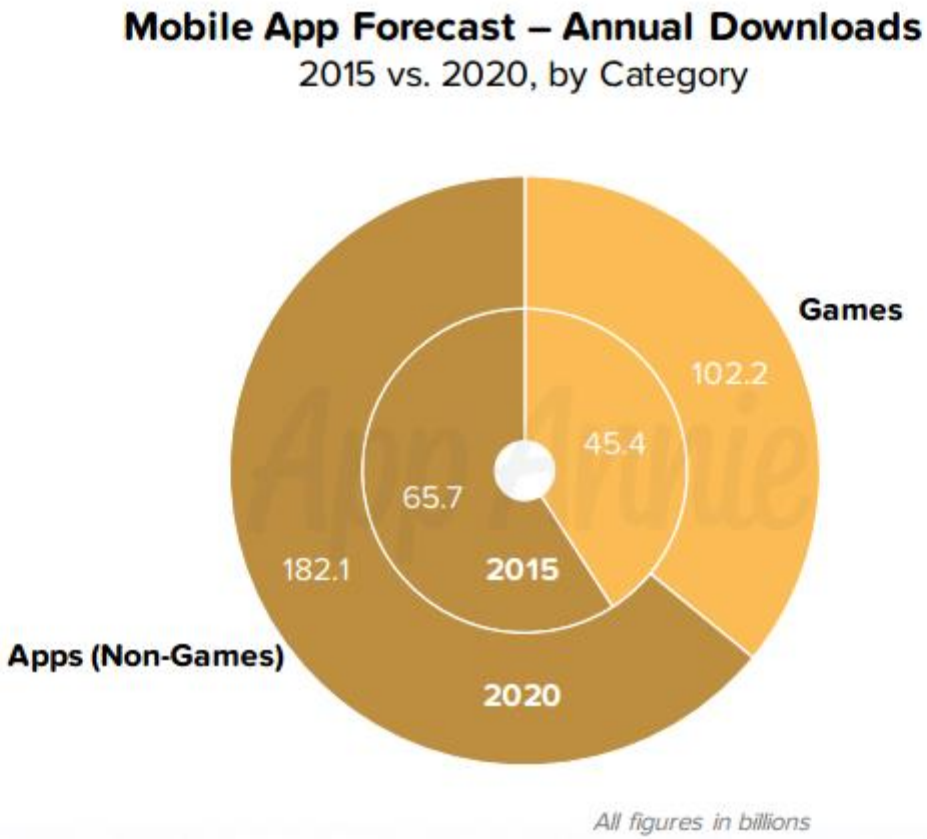
Desarrollo de aplicaciones multiplataforma



Desarrollo de aplicaciones multiplataforma

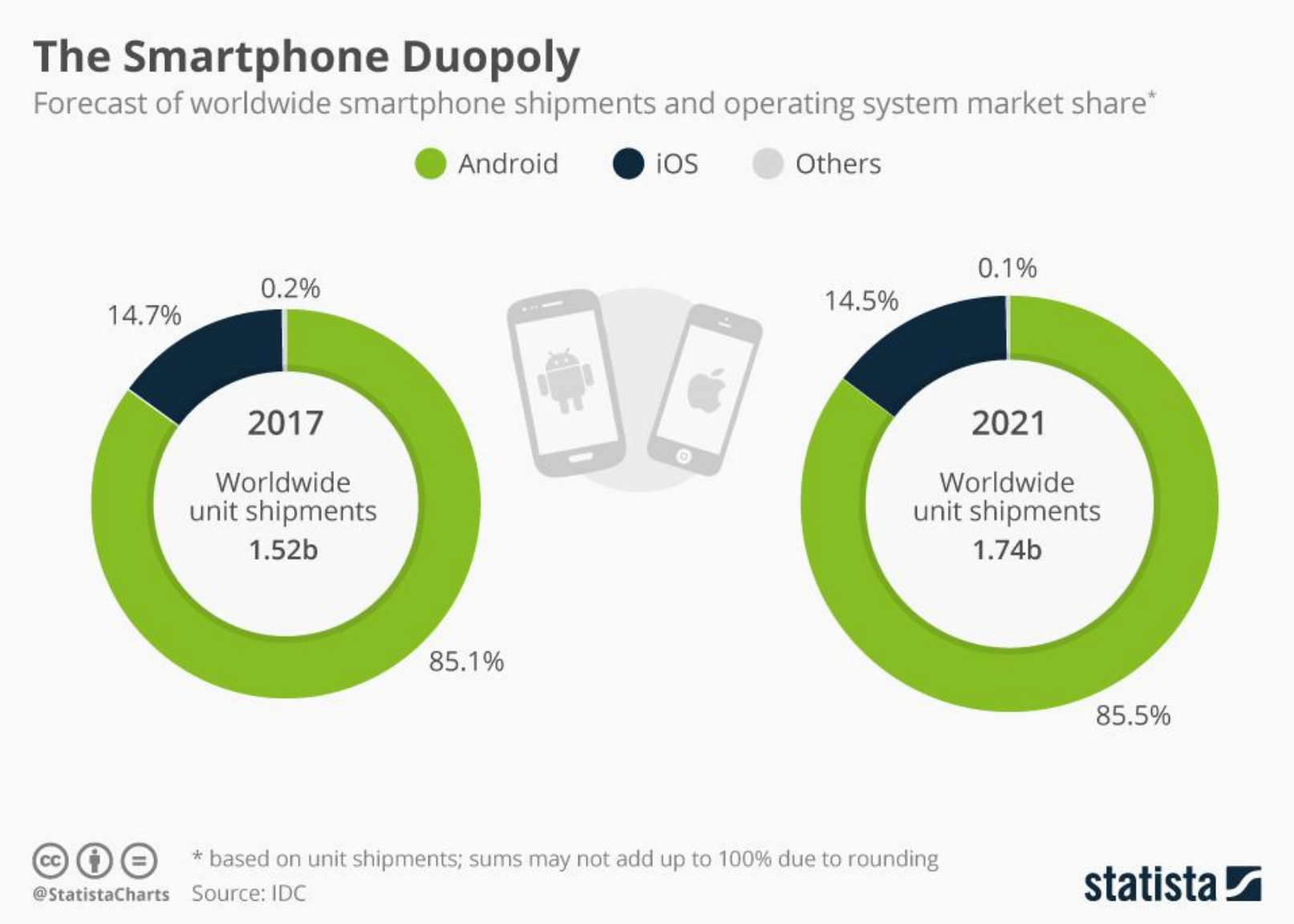


Desarrollo de aplicaciones multiplataforma

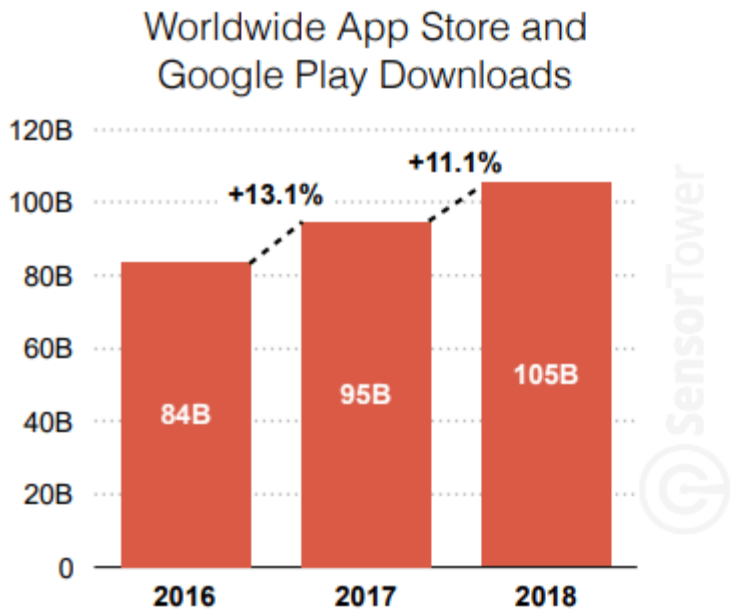


Non-game app downloads are projected to grow at a five-year CAGR of 23%, exceeding 182 billion in 2020.

Desarrollo de aplicaciones multiplataforma

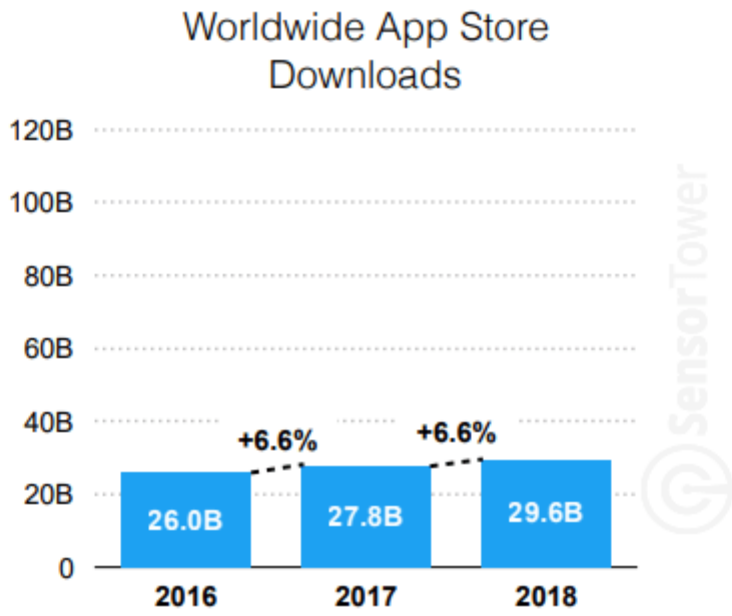


Desarrollo de aplicaciones multiplataforma



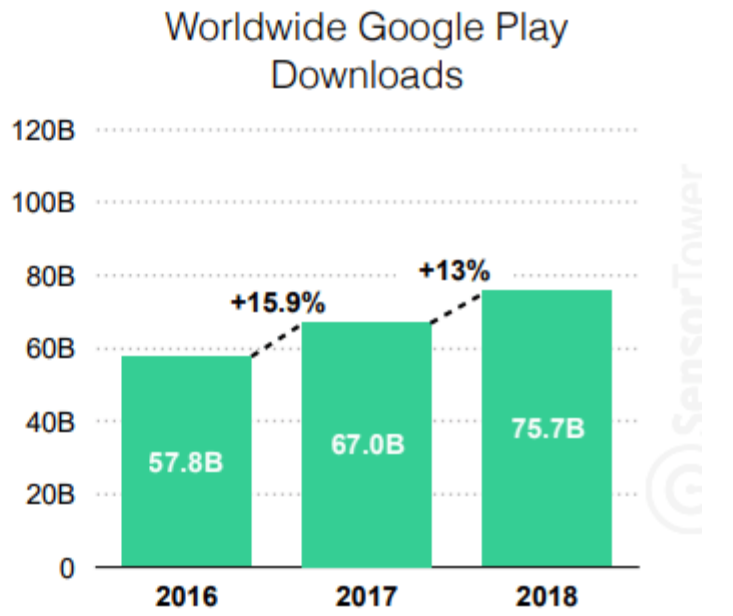
105 Billion

App Store + Google Play Downloads



29.6 Billion

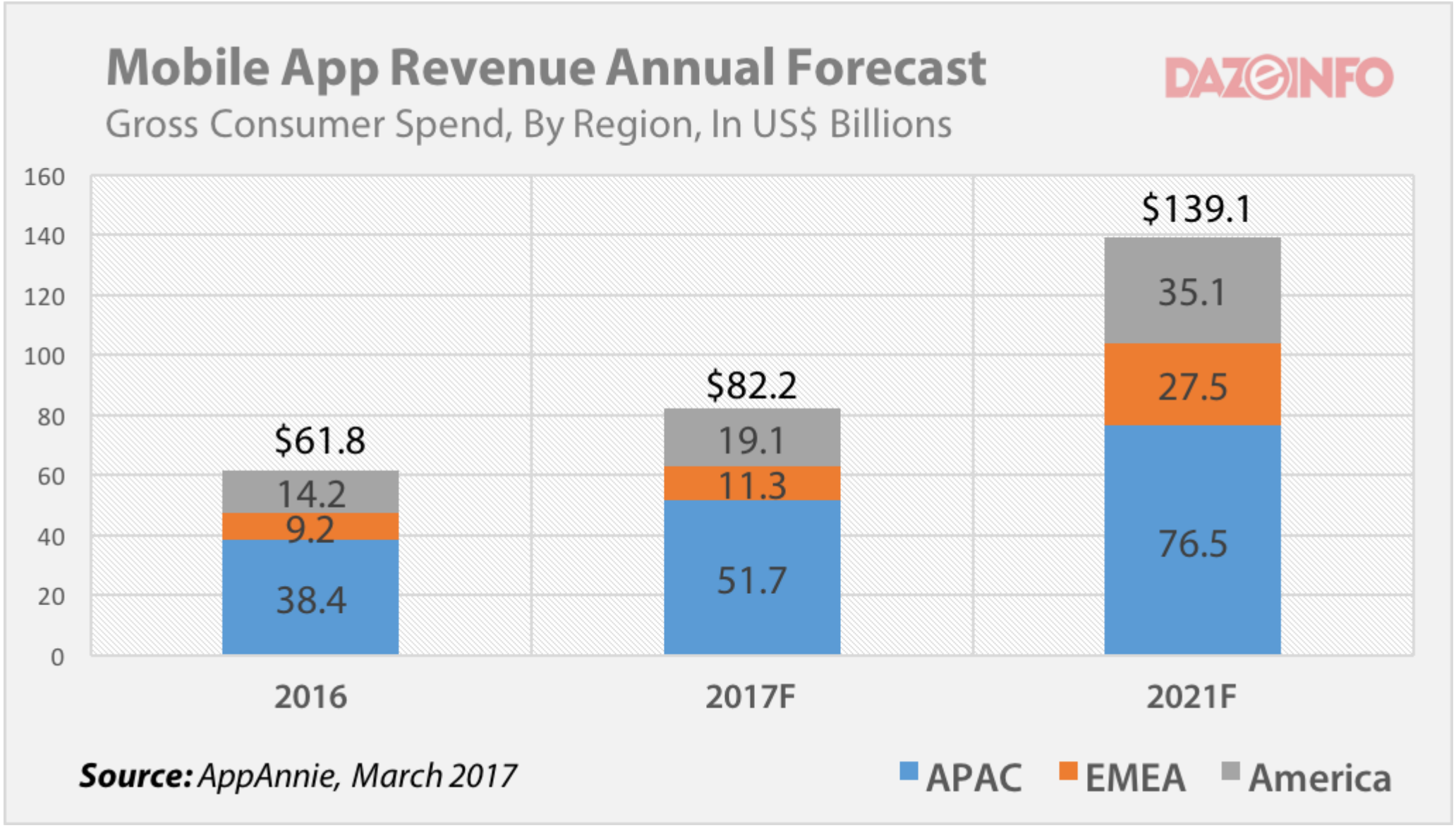
App Store Downloads



75.7 Billion

Google Play Downloads

Desarrollo de aplicaciones multiplataforma



Desarrollo de aplicaciones multiplataforma

- Aspecto clave en la toma de decisiones a la hora de construir aplicaciones para tecnología móvil:
  - Desarrollar una aplicación para cada plataforma.
  - Desarrollar una sola aplicación mediante un framework multiplataforma.
- En la actualidad existen diversas plataformas móviles en el mercado:
  - iOS
  - Android
  - Otras:
    - Windows Phone
    - Blackberry
    - Tizen
    - WebOS
    - KaiOS

- **iOS**:
  - Sistema operativo que sustenta a los iPhone y los iPad de Apple.
  - Se programa en Objective-C, usando el entorno de programación Xcode.
  - Necesitad de un ordenador Mac para poder generar aplicaciones para el sistema.
  - Totalmente atado al mundo Apple.
- **Android**:
  - La plataforma más extendida gracias a la variedad de dispositivos.
  - El sistema está basado en Linux y se programa en Java usando:
    - Android Studio.
    - Eclipse.

- **Windows Phone** **(Enero 2020 Fin Microsoft)**:
  - Sistema hoy por hoy abocado a desaparecer, cuota residual.
  - Se programa en C#+XAML o bien con HTML5+JS.
  - El entorno de desarrollo preferido en este caso es Visual Studio.
- **Otras plataformas**:
  - Como **Tizen**, **WebOS, KaiOS, Blackberry** …
  - Tienen una cuota de mercado prácticamente nula y no suelen disponer de muchos desarrollos pero que en un momento dado podríamos necesitar.

Desarrollo de aplicaciones multiplataforma

Top Five Smartphone Operating Systems, Worldwide Shipments, and Market Share, 2014Q2 (Units in Millions)

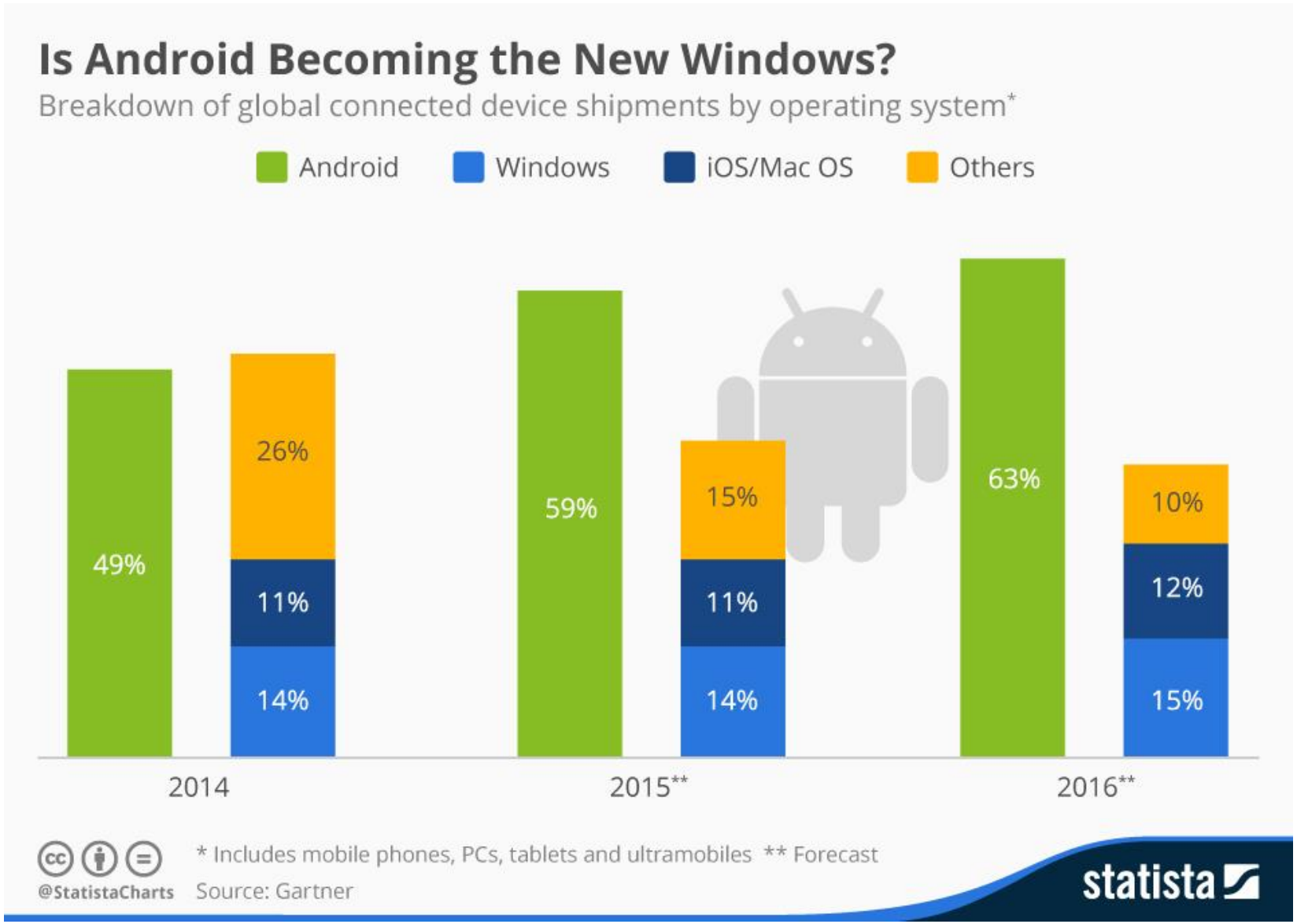
Operating System	2Q14 Shipment Volume	2Q14 Market Share	2Q13 Shipment Volume	2Q13 Market Share	2Q14/2Q13 Growth
Android	255.3	84.7%	191.5	79.6%	33.3%
iOS	35.2	11.7%	31.2	13.0%	12.7%
Windows Phone	7.4	2.5%	8.2	3.4%	-9.4%
BlackBerry	1.5	0.5%	6.7	2.8%	-78.0%
Others	1.9	0.6%	2.9	1.2%	-32.2%
Total	**301.3**	**100%**	**240.5**	**100%**	**25.3%**

Desarrollo de aplicaciones multiplataforma

Worldwide Smartphone Shipments by OS, Market Share, and Annual Growth (shipments in millions)							
Platform	**2016 Shipment Volume***	**2016 Market Share***	**2016 YoY Growth***	**2020 Shipment Volume***	**2020 Market Share***	**2020 YoY Growth***	**5 Year CAGR***
Android	1,228.8	85.0%	5.2%	1,464.7	85.6%	4.2%	4.6%
iOS	206.1	14.3%	-11.0%	243.6	14.2%	2.5%	1.0%
Windows Phone	6.1	0.4%	-79.1%	1.0	0.1%	-19.3%	-48.6%
Others	4.5	0.3%	-50.0%	1.0	0.1%	-7.3%	-35.3%
Total	**1,445.4**	**100.0%**	**0.6%**	**1,710.3**	**100.0%**	**4.8%**	**3.5%**
Source: IDC Worldwide Quarterly Mobile Phone Tracker, November 29, 2016							

* **Table Note:** All figures are forecast projections.

Desarrollo de aplicaciones multiplataforma



- **Los puntos de referencia para el desarrollo multiplataforma**

- Cualquier plataforma de desarrollo transversal requiere escribir código para una aplicación y puesta a punto para cada sistema operativo.
- La **escritura de código nativo** para Android, iOS y Windows **requiere** un equipo de **programadores** que tengan **fluidez** en Java, Objective-C y C#.
- Los **desarrolladores** pueden **ahorrar tiempo** e implantar sus aplicaciones en el mercado más rápidamente mediante el uso de **herramientas de desarrollo cruzado**,
- Principales Inconvenientes:
  - Utilizando un **enfoque de mínimo común denominador** → pierde la ventaja de las capacidades de cada dispositivo.
  - Tiempos más largos para la actualización de aplicaciones.
  - Los **posibles retrasos** en la carga de bibliotecas.
  - Capas de abstracción más grandes.
- Las herramientas de desarrollo como Xamarin/MAUI, y Córdova/Ionic han simplificado el proceso.

- **¿Cómo acometer desarrollos para diferentes arquitecturas de dispositivo móvil?**
  - Crear una versión para cada plataforma, usando sus herramientas nativas.
  - Utilizar algún framework intermedio, como **Xamarin o MAUI**, que funcione en todas ellas y que nos permita reutilizar la lógica y solo desarrollar la interfaz en cada caso.
  - Utilizar HTML+CSS+JavaScript y algún framework multiplataforma, como **Apache Cordova o Ionic**, que nos permita compilar la aplicación para todas las plataformas → Aplicación Híbrida.

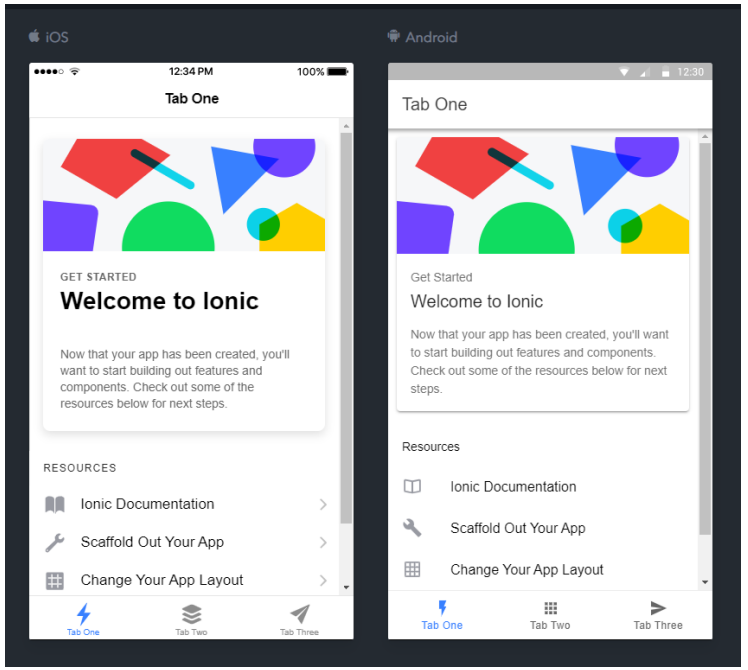
- **Aplicación Híbrida**
  - Nos permite **desarrollar apps** para móviles en base a las **tecnologías web**: HTML + CSS + Javascript.
  - Son como cualquier otra aplicación de las que puedes instalar a través de las tiendas de aplicaciones para cada sistema.
    - Los usuarios finales no percibirán la diferencia con respecto a las aplicaciones nativas.
  - **Ventaja** para desarrolladores con **experiencia** en **tecnologías web**,
    - Los desarrolladores pueden aprovechar sus conocimientos para lanzarse de una manera más rápida en el desarrollo de apps para móviles.
    - Las aplicaciones híbridas se ejecutan en lo que se denomina un ***web view***¿ que no es más que una especie de navegador integrado en el móvil y en el que solamente se ejecuta la app híbrida.

- **Aplicación Híbrida**
  - Interesantes por diversos motivos:
    - Con una **misma base de código** serán capaces de compilar apps para funcionar correctamente en una gran cantidad de sistemas operativos de móviles o tablets.
    - El **coste** del desarrollo es sensiblemente menor, ya que no es necesario contar con varios equipos de desarrollo para cada lenguaje concreto de cada plataforma.
    - El **tiempo** de **desarrollo** también es **menor**, ya que solo es necesario construir la aplicación una vez e inmediatamente la tendremos en todas las plataformas a las que nos dirigimos.
    - Es de más **fácil adaptación** para los desarrolladores que vienen de la web.

- **Aplicación Híbrida**
  - Desventajas respecta a aplicaciones nativas:
    - El **rendimiento** de una aplicación nativa suele ser mejor que el de una híbrida, aunque las híbridas han mejorado mucho en este sentido.
    - Al ejecutarse en un **web view dependemos** de las **tecnologías** disponibles para el desarrollo **web**, que pueden ser menos ricas y potentes que las disponibles en nativo.
    - En las aplicaciones nativas trabajamos directamente con el hardware del teléfono, mientras que en las híbridas dependemos de **plugins** para su acceso → Esto puede derivar en **problemas**, pero afortunadamente cada vez son menores.

- **Aplicación Nativa vs Hibrida**
  - Híbrido no es desarrollar en un lenguaje diferente.
    - Error al pensar si no usamos Swift (iOS) o Java (Android) → desarrollo hibrido.
    - Si requieren una capa adicional para ejecutarse como un navegador o un contenedor, son híbridas.
      - Apache Cordova.
      - Ionic.
      - Unity.
    - Si se ejecutan directamente en el SO, son nativas:
      - ReactNative.
      - NativeScript.
      - Xamarin.
      - .Net MAUI

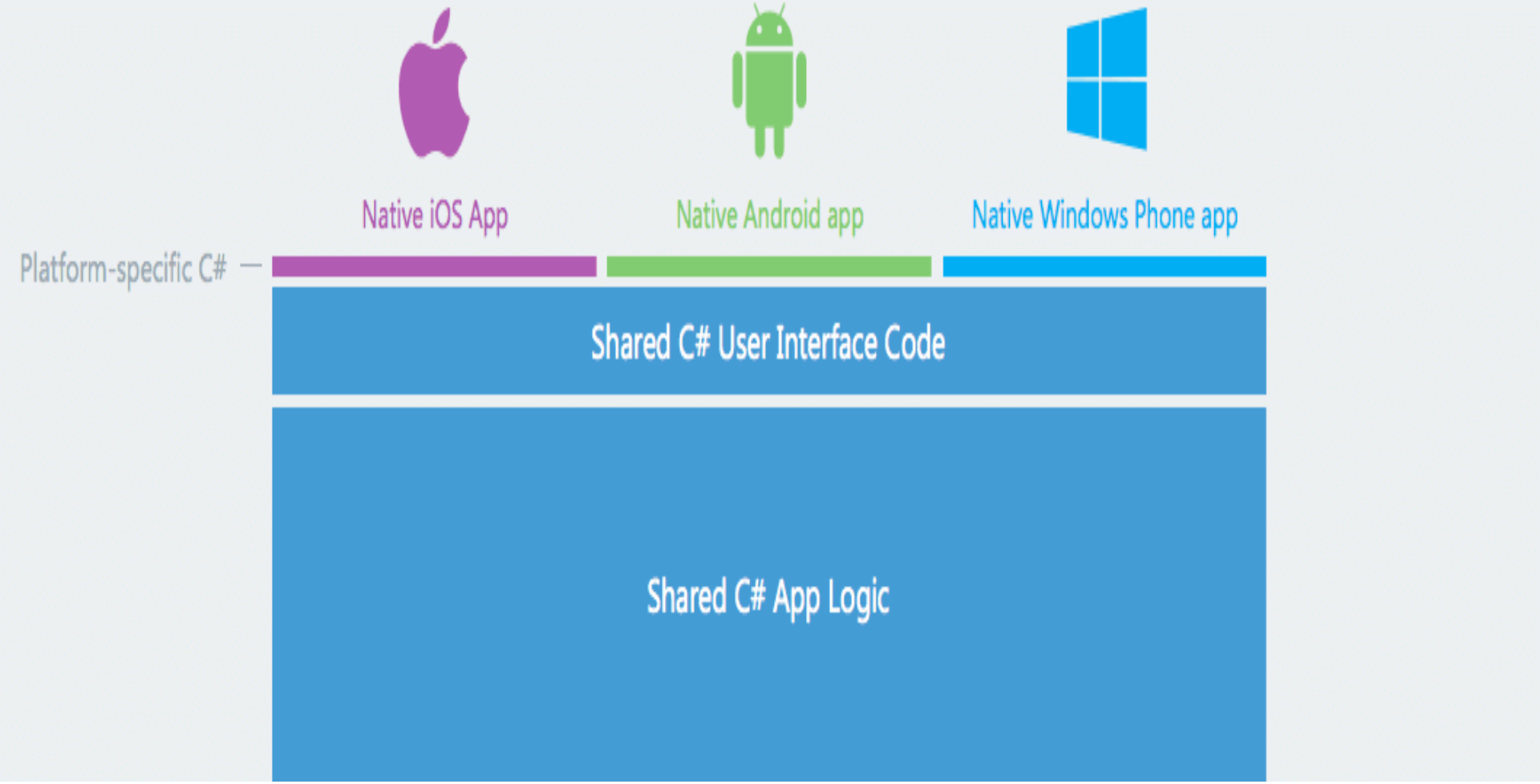
- **Aplicación Nativa vs Hibrida**
  - Aplicaciones Hibridas = Rendimiendo reducido
    - Los primeros desarrollos con phoneGap presentaban problemas de rendimiento.
    - En la actualidad con frameworks como Ionic, presentan apps con *UI* propias de los SO y conexión con HW muy amplia.



- **¿Qué es Xamarin?**
  - Xamarin es una herramienta para los desarrolladores de **aplicaciones móviles**, y la novedad de esta herramienta es la capacidad que tiene para que el desarrollador escriba su app en lenguaje C# y el mismo código sea traducido para ejecutarse en iOS, Android y Windows Phone.
  - C# es el "lenguaje preferido" de **Microsoft**, por lo tanto es muy popular dentro de los programadores .NET, pero aunque este lenguaje sea muy popular existe una desventaja al momento de **crear aplicaciones móviles**.
  - Para **crear aplicaciones iOS** se necesitan escribir código en Objetive-C y para **crear Apps Android** necesitamos conocer Java → Xamarin viene a unificar estas diferencias con su IDE Xamarin Studio, ya que solamente necesitamos dominar C# para **crear aplicaciones iOS, Android y Windows Phone**.

Desarrollo de aplicaciones multiplataforma

Xamarin

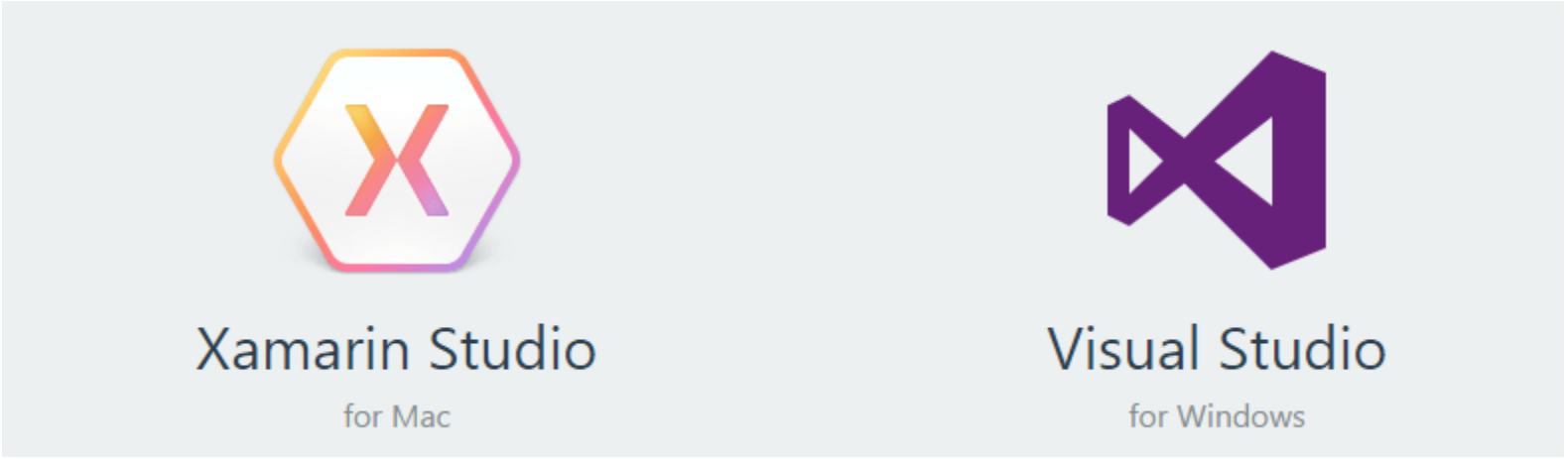


- Creación 2011.
- Hasta 2016 → framework de pago, contaba con una versión gratuita

- En Febrero del 2016, Microsoft compra xamarin → integrado en visual studio 2015/2017/2019.
- https://www.xamarin.com
- Fin Soporte 1 de Mayo del 2024.

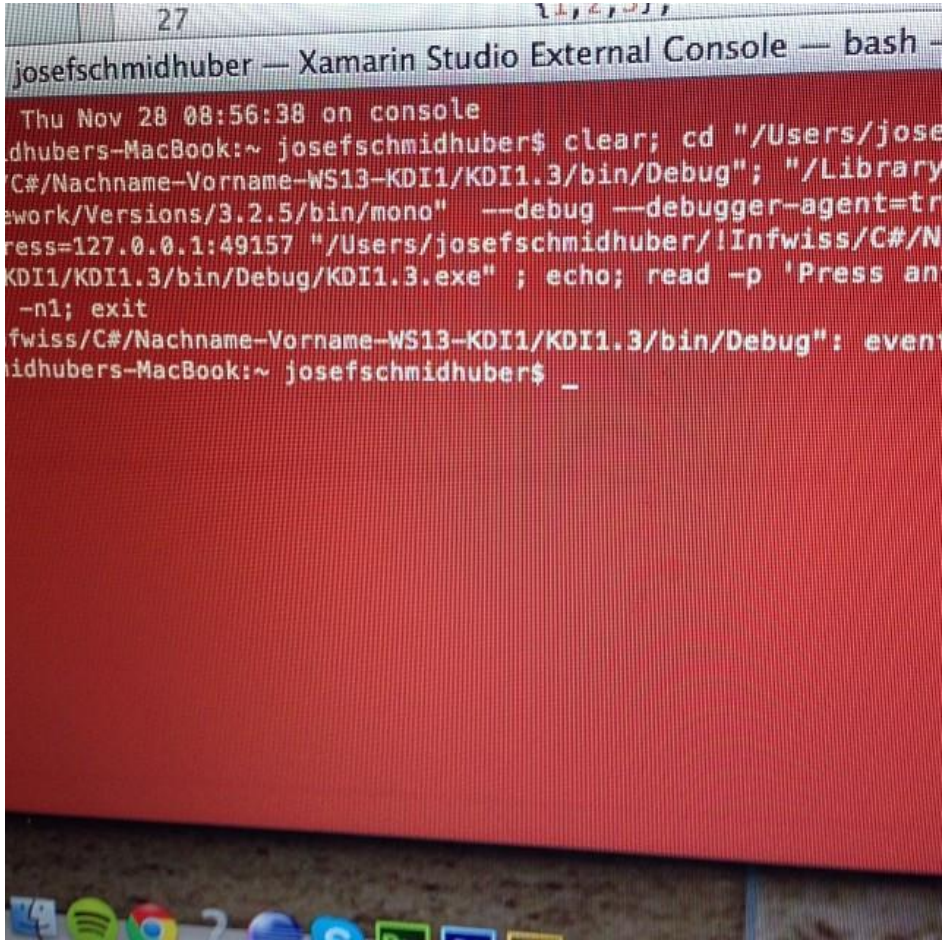
Xamarin

- Cuenta con IDE propio e integración con Visual Studio 2015/2017/2019



Xamarin

- **¿Desarrollando para Android/iOS/Windows Phone?**
  - Algo que hay que saber en cuanto al **desarrollo de aplicaciones** para diversas plataformas, es que **Xamarin Studio** utiliza los recursos nativos de cada plataforma:
    - Si deseas crear aplicaciones para iOS uno de los requisitos previos es que **cuentes** con un **sistema Mac OS X**, tu Mac se enlazará con Windows para compilar el proyecto que estés creando.
  - Esta es una de las desventajas que aún se tienen, todavía no existe una herramienta para Windows que permita el desarrollo independiente de aplicaciones iOS.



Apache Cordova

- Plataforma para la construcción de aplicaciones móviles nativas utilizando HTML, Javascript y CSS.
- Apache Cordova es un conjunto de APIs de dispositivos que permiten a un desarrollador de aplicaciones móviles acceder a las nativa del dispositivo:
  - La cámara
  - El acelerómetro
  - …
- Combinado con un framework de UI, como:
  - jQuery Mobile
  - Dojo Mobile
  - Sencha Touch
  - Bootstrap
  - Ionic
  - Onsen UI
- Y un framework Javascript:
  - Jquery
  - Angular
  - React

Apache Cordova

- Cuando se utiliza la API de Cordova, una aplicación se desarrolla sin ningún código nativo (Java, Objective-C, etc).
  - En su lugar, se utilizan tecnologías web, y todo se encuentra ubicado en la propia aplicación a nivel local.
- Ya que las API de **JavaScript** son **consistentes** a través de múltiples plataformas de dispositivos y están construidas siguiendo los **estándares** web → la aplicación debe ser portable a otras plataformas de dispositivos con un cambio **mínimo** o **ningún cambio**.
- Finalmente las Apps que utilizan Cordova se empaquetan como aplicaciones que utilizan los SDK de la plataforma destino, listas para la instalación en la tienda de aplicaciones de cada dispositivo.

Apache Cordova

- Córdova ofrece un conjunto de bibliotecas de JavaScript uniformes que se pueden invocar, con el código de respaldo nativo del dispositivo.
- Córdova está disponible para las siguientes plataformas:
  - iOS
  - Android
  - Blackberry
  - Windows Phone
  - Palm WebOS
  - Bada
  - Symbian.
- En la URL https://cordova.apache.org/plugins , podemos encontrar infinidad de plugins.

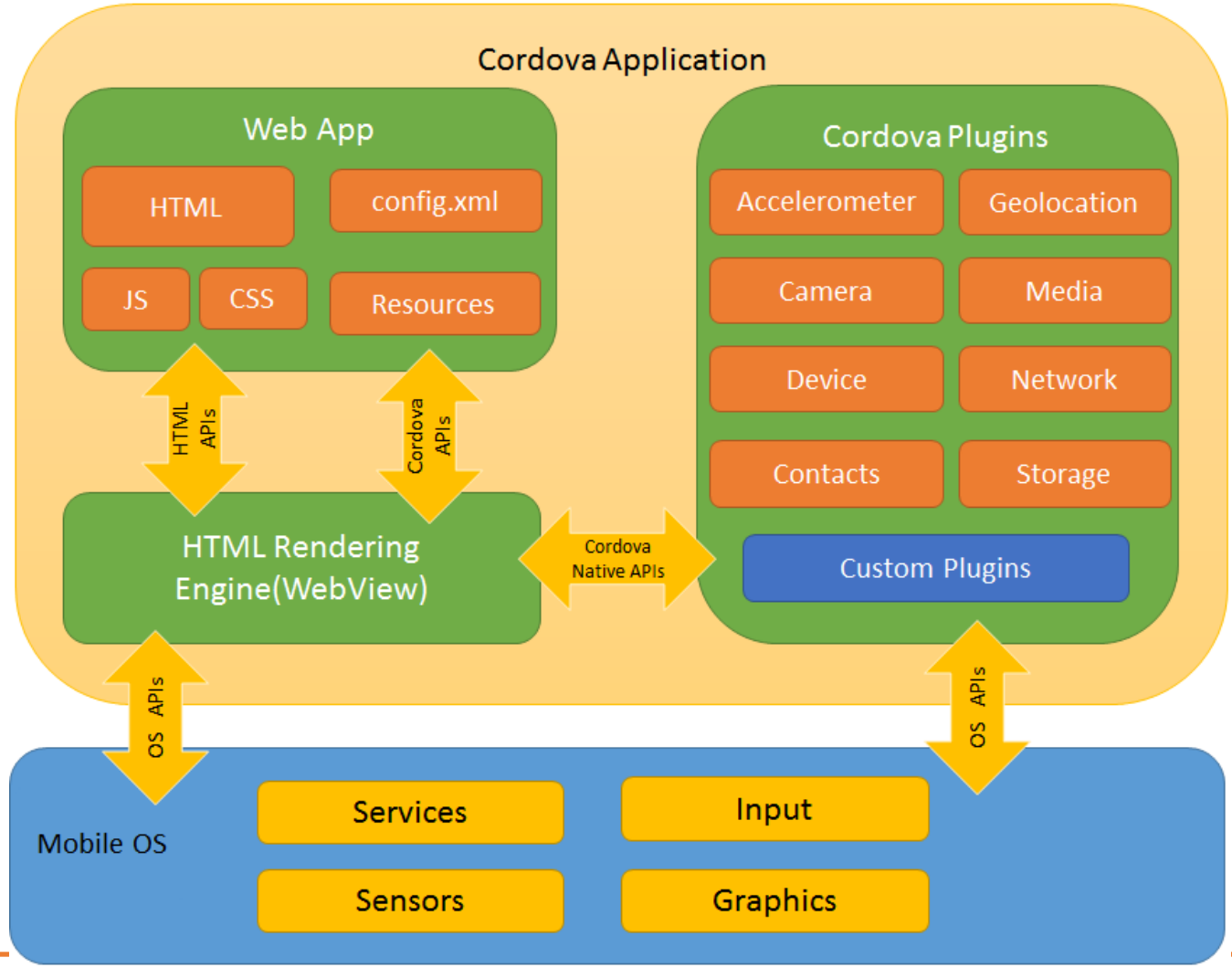
Apache Cordova

- Apache Cordova viene con un conjunto de plugins pre-desarrollado que dan acceso:
  - Cámara del dispositivo
  - GPS
  - Sistema de archivos
  - Etc.
- Como los dispositivos móviles evolucionan, añadir soporte para hardware adicional es simplemente una cuestión de desarrollar nuevos plugins.

Apache Cordova

- Las aplicaciones Cordova se instalan como aplicaciones nativas.
  - La construcción de su código para iOS producirá un archivo IPA
  - Para Android un archivo APK
  - Para Windows Phone produce un archivo XAP.
- Si se pone suficiente esfuerzo en el proceso de desarrollo, los usuarios no distinguirán si están utilizando una aplicación nativa o híbrida.

Apache Cordova

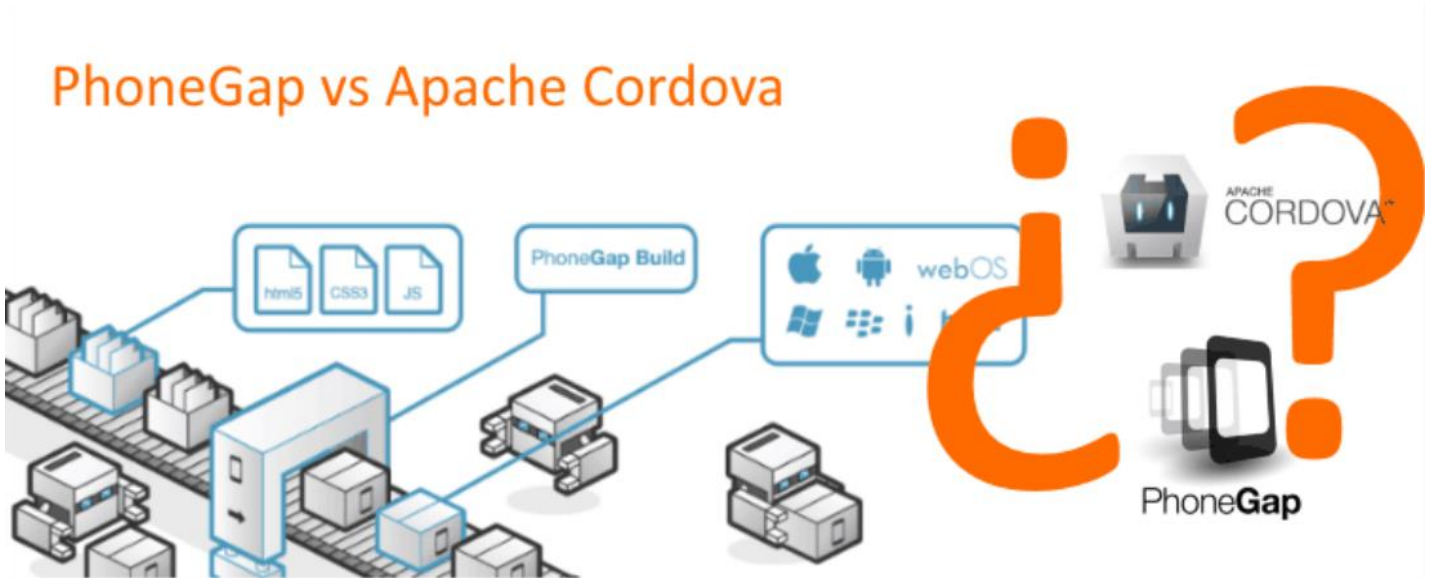


PhoneGap vs **Apache Cordova**

Adobe PhoneGap

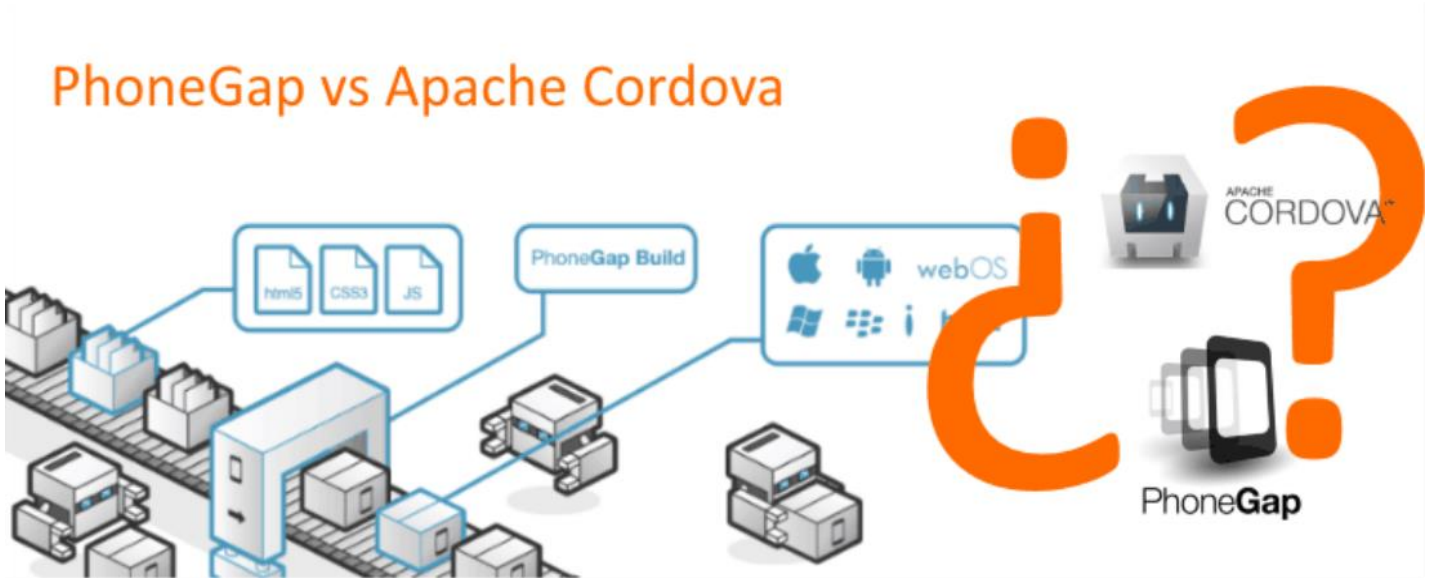
- ¿se trata de la misma herramienta?
- ¿son distintas?
- ¿tienen algo que ver entre ellas?
- ¿Qué relación existe entre ambas?

APACHE CORDOVA™



PhoneGap vs **Apache Cordova**

- PhoneGap y Apache Cordova son herramientas **idénticas**, pero tienen diferentes nombres y se descargan desde diferentes sitios.
  - ¿Se diferencia en algo?, ¿iguales?

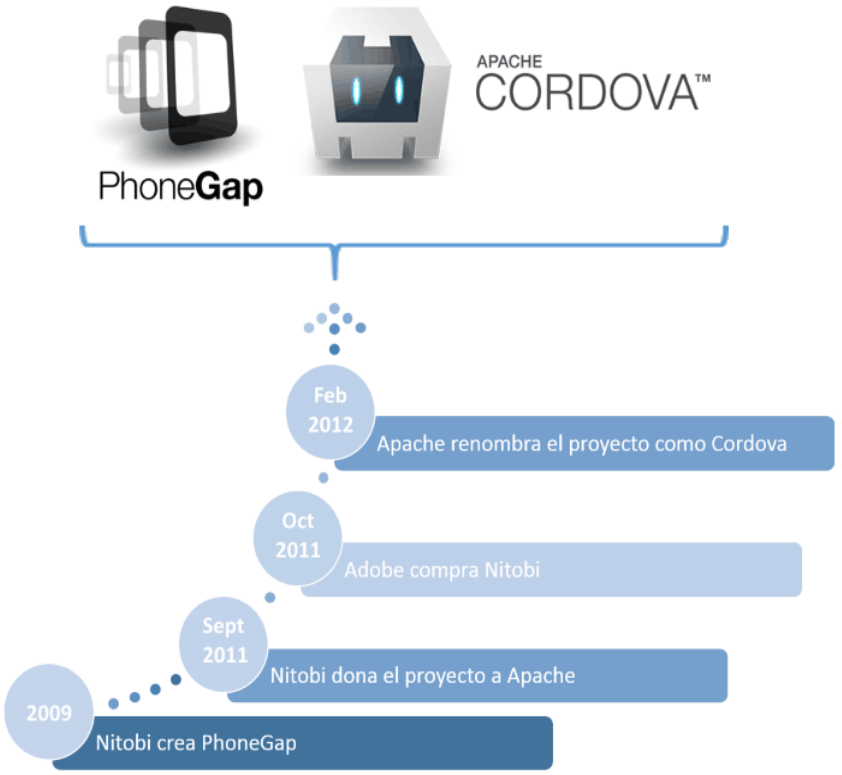


PhoneGap vs **Apache Cordova**

- En 2009 Nitobi crea un framework para desarrollo móvil multiplataforma llamado PhoneGap.
- La idea:
  - Crear aplicaciones orientadas a móviles con **HTML5** y dotarlas de una capa **JavaScript** que permita acceder a las **funciones nativas** de cada sistema.
  - Crear un entorno que permita ejecutarlas en cualquier sistema operativo móvil.

PhoneGap vs **Apache Cordova**

- Mayor Promoción → En septiembre de 2011 deciden donar el código fuente del producto a la fundación Apache → proyecto Open Source → dar un gran impulso.
- En octubre de 2011, Adobe compra Nitobi, y con la empresa se lleva a sus empleados, el producto y la marca PhoneGap.
- A pesar de la compra están de acuerdo en que el código se done a la fundación Apache, y el plan sigue adelante.
- Dado que PhoneGap es una marca registrada de Adobe, para evitar problemas legales Apache decide renombrar en febrero de 2012 el proyecto a Cordova.
  - Este nombre tan peculiar se lo ponen porque esta era la calle en la que estaban las oficinas de Nitobi en Vancouver (Canadá).



Fin Vida Phonegap Octubre 2020

Recomendado el uso de Apache Cordova

PhoneGap vs **Apache Cordova**

- Entonces, **¿Existe alguna diferencia entre ellos? ¿O son lo mismo?**

  - Actualmente no existe ninguna diferencia entre **PhoneGap y Apache Cordova**, a excepción del nombre y desde dónde se descargan.
  - Ambos son gratuitos, Open Source y se utilizan del mismo modo.
  - *PhoneGap no es más que una distribución de Apache Cordova.*
    - Servicio Compilación en la nube.
  - La diferencia entre ambos estriba en que **Adobe** tiene la **potestad** para **extender** si quiere el producto y dotarlo de herramientas o características propias, por las cuáles puede decidir cobrar.

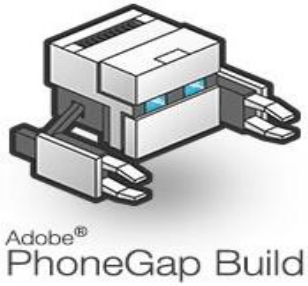
PhoneGap vs **Apache Cordova**

- Entonces, **¿Existe alguna diferencia entre ellos? ¿O son lo mismo?**

  - PhoneGap permite la integración con los servicios de compilación de Adobe.
  - Las actualizaciones de Apache Cordova suelen ser algo más frecuentes, dado que es el producto base.
  - En la práctica, se habla de manera indistinta de uno o de otro, como si se tratara del mismo producto, cuando son prácticamente lo mismo.

PhoneGap vs **Apache Cordova**

Los servicios de compilación de Adobe



- En el caso de usar los servicios de compilación de PhoneGap (llamados Adobe PhoneGap Build), te ahorras la necesidad de tener que instalar los SDK y las herramientas específicas de cada plataforma (es decir, por ejemplo, no tienes que tener un Mac para compilar para iPhone)
- *Este servicio es algo específico de PhoneGap que no tienes con Apache Cordova.*
  - ***"Pero si se tenia con Visual Studio 2015 / 2017"***
- PhoneGap Build compila para las principales plataformas del mercado: iOS, Android, Windows Phone, Blackberry 5/6/7 y webOS.
- Para proyectos Open Source el servicio es gratuito.
- El servicio de compilación solamente te permitirá subir el código a compilar obteniéndolo desde un repositorio de GitHub, máximo de un único repositorio privado para compilar.

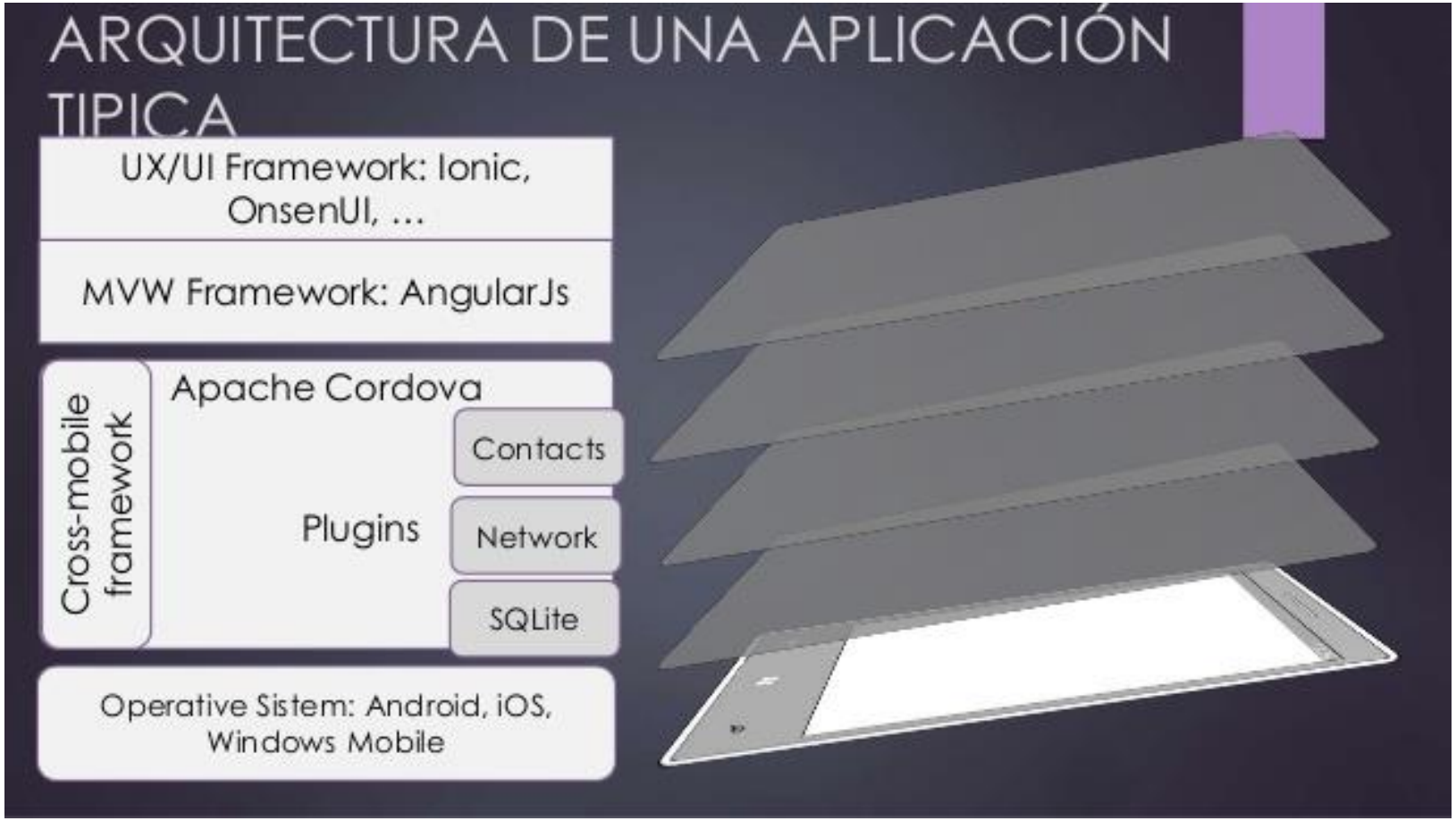
PhoneGap vs **Apache Cordova**
Resumen

- Apache Cordova y PhoneGap son en la práctica lo mismo.
  - A la hora de aprender uno u otro no hay diferencia.
- PhoneGap es una **distribución específica** de Apache Cordova que realiza Adobe.
  - Permite **compilar** los proyectos para múltiples plataformas si quieres utilizar el servicio PhoneGap Build que ofrece la empresa.
- **Ambos** son **gratuitos** y Open Source.
- **Adobe**, la empresa se ha comprometido a que siempre siga siendo gratuito y de código abierto.
  - *La decisión de cuál usar debería basarse solamente en si tienes interés en poder sacar partido alguna vez a los servicios de Adobe o no.*

Ionic

- Ionic es un framework para el desarrollo de **aplicaciones híbridas**, inicialmente pensado para **móviles** y **tablets**, aunque ahora también capaz de implementar aplicaciones web e incluso aplicaciones de escritorio multiplataforma (https://ionicframework.com/docs/deployment/desktop-app).
- Uso conjunto con Angular (podría ser Vue o React), para facilitar el desarrollo de las aplicaciones.
  - Permitirá contar con una excelente estructura de proyecto, trabajar con buenas prácticas, uso de patrones de diseño de software variados y una buena gama de componentes y directivas.
- Permite crear aplicaciones para distintas plataformas móviles con una misma base de código.

Ionic


ARQUITECTURA DE UNA APLICACIÓN TIPICA
UX/UI Framework: Ionic, OnsenUI, ...
MVW Framework: AngularJs
Cross-mobile framework
Apache Cordova
Plugins
Contacts
Network
SQLite
Operative Sistem: Android, iOS, Windows Mobile

Ionic

- Posibilidad de emplear **Cordova** o bien de **Capacitor** (framework propio Ionic) para acceder a las capacidades de hardware de los dispositivos.
- El primer paso para emplear Ionic, consiste en instalar **node.js**.
- Las aplicaciones Ionic se crean y desarrollan principalmente a través de Ionic Cli.
  - Ionic CLI es el método de instalación preferido, ya que ofrece una amplia gama de herramientas de desarrollo.
  - https://ionicframework.com/docs/intro/cli

Ionic

- Typescript:
  - Es un "**superset**" de Javascript.
  - TypeScript es Javascript pero con añadidos pensados para mejorar el trabajo por parte de los desarrolladores, haciéndonos más productivos.
  - La mayor aportación de TypeScript al lenguaje Javascript es la posibilidad de **definición** de **tipos** para las variables.
  - TypeScript requiere una transpilación (***transpiler***) del código, de modo que el código que escribimos en TypeScript se convierta en código Javascript capaz de ejecutarse en cualquier navegador.
    - Esa transpilación la realiza Ionic internamente, por lo que no representa un problema para el desarrollador.

Ionic

- Apariencia adaptada al dispositivo:
  - Los componentes de Ionic ya vienen adaptados al dispositivo de manera estética.
    - Cuando se compila una aplicación para iOS el componente se visualizará de manera diferente que cuando se compila para Android.
    - En Android usará Material Design mientras que en iOS usará las guías de diseño definidas por Apple.
  - Esto es una ventaja en sí, porque las usuarios disfrutarán de aplicaciones con una **experiencia** de **usuario** cercana a la que están **acostumbrados** en su dispositivo móvil y nos evita a los desarrolladores la necesidad de trabajar más para conseguir este efecto.

Progressive Web Apps

- Es una solución basada en la web tradicional, aunque incorpora algunas particularidades que la hacen parecerse a una app nativa.
- Son la intersección entre la facilidad de la Web y la experiencia de uso de las Apps.
- Ventajas:
  - No se necesita entrar a Google Play o Apple Store para descargar nada. Solo se requiere, al principio, una conexión a internet y un navegador.
  - Cualquier usuario puede “instalarla” en la pantalla de inicio de su dispositivo.

Progressive Web Apps

- Características:
  - **Progresiva**: Funciona para todos los usuarios.
  - **Adaptable**: Su funcionalidad se adapta al dispositivo.
  - **Independiente de la conectividad**: Soporta funciones sin conexión.
  - **Estilo app**: Para el usuario una PWA es similar a una App, de las que podría descargar desde una tienda.
  - **Fresca**: Gracias a las estrategias de caching se podrá mantener actualizada cada vez que esté disponible una conexión.
  - **Segura**: Funciona sobre HTTPS.
  - **Descubrible**: Los motores de búsqueda son capaces de indexar la PWA, y la detectarán como App.
  - **Acciones Posteriores**: Va a permitir **Push Notification** tras instalar.
  - **Instalable**: Les permite a los usuarios tener la App disponible en su dispositivo con un icono de acceso como las Apps que descargas desde una tienda.
  - **Vinculable**: Se puede compartir fácilmente con una URL.

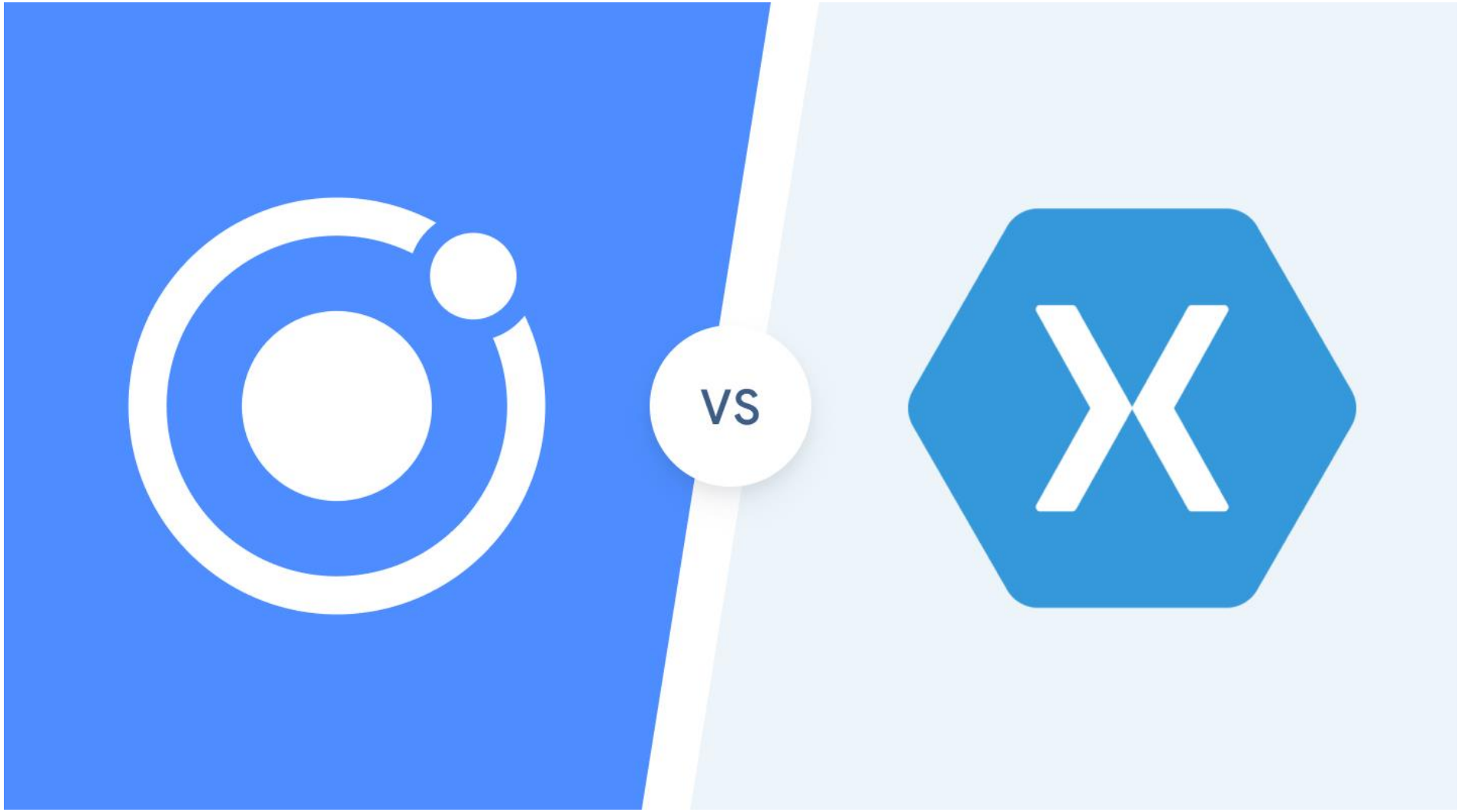
Progressive Web Apps

- Tienen una estructura normal de una WebApp, con un índice y archivos HTML, CSS y JS. Pero además, para que sea considerada una Progressive Web App, debe tener 3 cosas adicionales:
  - **Un archivo de manifiesto**. Un archivo el cual describe la información de tu PWA.
  - **Un icono de App**. Una imagen que será utilizada para ser el icono en los dispositivos móviles al ser instalada.
  - **Service Workers**. Es un archivo Javascript que es registrado en el navegador y es el que va a permitir hacer las tareas de caching y el push notification, entre otras.
- Ionic permite crear PWA:
  - https://ionicframework.com/docs/angular/pwa

Xamarin vs Apache Cordova

- Apache Cordova es un framework fácil de usar para la creación de aplicaciones en HTML5, JavaScript y CSS.
  - No ofrece la flexibilidad y la potencia que proporciona Xamarin.
  - Sin embargo, si los programadores están familiarizados con esos tres lenguajes, puede ser muy interesante decantarse por su uso.
  - En el uso de tecnologías web estándar, Cordova siempre seguirá siendo de código abierto y libre, siempre y cuando se distribuya bajo la Apache Software Foundation.
  - Cuando se compara con Xamarin, Cordova genera problemas que incluyen:
    - API funcionalidad limitada.
    - Mayores posibilidades de tener errores en la interfaz de usuario y API complejas.
    - Complejidad para el desarrollador, al utilizar open frameworks.
    - “Dificultades para hacer que las aplicaciones parezcan nativas”.
  - *Con Ionic todos estos problemas se minimizan.*

Ionic vs Xamarin



Ionic vs Xamarin

- Ionic y Xamarin son dos de las opciones más populares para el desarrollo de aplicaciones móviles.
- **Ionic** es una plataforma que utiliza tecnologías y lenguajes web ampliamente conocidos y basados en estándares para crear experiencias móviles galardonadas, y se basa en el **ecosistema web** y JavaScript más amplio.
- **Xamarin** es un producto de Microsoft y abarca el **ecosistema** de **Microsoft**.

Ionic vs Xamarin

- **Xamarin** permite a los desarrolladores escribir sus aplicaciones en C # y compartir y compilar código para cada plataforma soportada.
- Solución ideal para las empresas que desean acelerar el desarrollo al obtener las características de aceleración de hardware y la interfaz de usuario nativa.
- A pesar de que podría haber una **curva de aprendizaje** para los programadores que no están familiarizados con C # y .NET, el **código resultante** se **comporta** como **aplicaciones nativas** en todas las plataformas.
- El código - entre el 60 y el 100 por ciento - se puede reutilizar para sistemas de servidor y cliente.

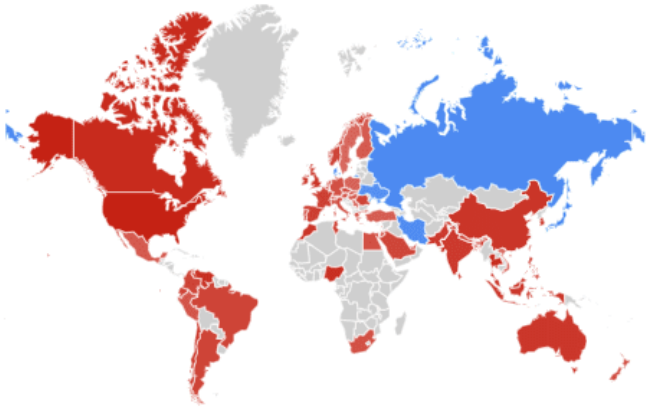
Ionic vs Xamarin

- Popularidad geográfica:

Compared breakdown by region

Region

xamarin ionic



Colour intensity represents percentage of searches LEARN MORE

Sort: Interest for ionic

1	United States	
2	Canada	
3	Philippines	
4	Bangladesh	
5	Thailand	

Include low search volume regions

Showing 1–5 of 64 regions

Ionic vs Xamarin

- **Xamarin**
  - El 14% de los desarrolladores de todo el mundo utilizan Xamarin para el desarrollo de aplicaciones multiplataforma.
  - Hay más de 15.000 aplicaciones creadas con el framework de Xamarin, lo que equivale a más de 2 mil millones de descargas.
  - La cuota de mercado media de las aplicaciones de Xamarin es 1,74%.
- **Ionic**
  - El 86% de los desarrolladores prefieren Ionic para el desarrollo móvil.
  - Casi 5 millones de aplicaciones se crean utilizando el framework Ionic.

Ionic vs Xamarin

- **Xamarin Ventajas**
  - **Desarrollo más rápido** : reduce el tiempo de desarrollo, los desarrolladores solo deben realizar pequeños cambios para lanzar aplicaciones en diferentes plataformas.
  - **Experiencia de usuario nativa** : Aprovecha las API específicas del hardware y del sistema, es casi imposible distinguir entre una aplicación Xamarin y una nativa.
  - La **integración** mediante el IDE **Visual Studio**.

Ionic vs Xamarin

- **Xamarin Desventajas**
  - **Tamaño** de la aplicación **más grande** : agrega de 3 a 5 megabytes para el lanzamiento y 20 megabytes para las compilaciones de depuración, lo que aumenta el tamaño de la aplicación.
  - No es adecuado para **gráficos pesados** : Xamarin no es muy bueno para incorporar animaciones y elementos gráficos ricos.
    - **No** es la mejor opción para **juegos** y otras aplicaciones que involucran gráficos avanzados.
  - **Actualizaciones retrasadas**: las actualizaciones para las últimas versiones de iOS y Android tardan entre 1 y 3 días en integrarse en el ecosistema.

Ionic vs Xamarin

- **Ionic Ventajas**
  - **Framework independiente de la plataforma**: reduce el tiempo, el esfuerzo y los recursos empleados para crear una aplicación multiplataforma a la vez que le da un aspecto y una sensación nativos.
  - Utiliza **capacitor** o **Cordova**: ahorra tiempo de construcción al proporcionar una interfaz sencilla para acceder al SDK nativo y la API nativa en cada plataforma.
  - Apto para desarrolladores: crea una única base de código mediante el uso de bibliotecas y frameworks de Javascript familiares, lo que **reduce la reescritura de código**.

Ionic vs Xamarin

- **Ionic Desventajas**
  - **Sistema dependiente de complementos**: aunque Ionic ofrece una gran cantidad de complementos, es posible que los desarrolladores deban crear algunas funciones muy específicas.
  - **Rendimiento**: no apto para aplicaciones complejas o con uso intensivo de memoria, ya que Ionic usa WebView para representar aplicaciones.

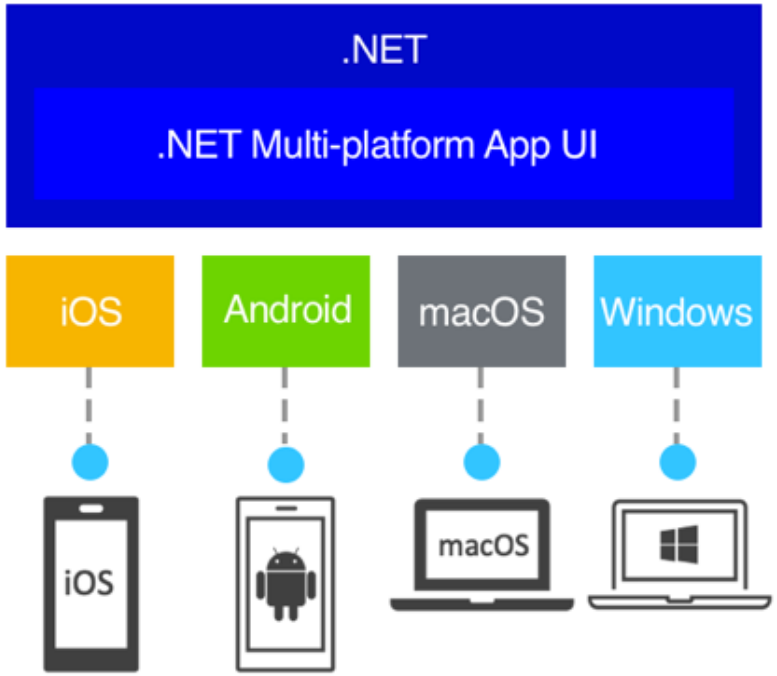
Ionic vs Xamarin

- **Comunidad**
  - Según Statista , el 14% de los desarrolladores de todo el mundo usan Xamarin para el desarrollo de aplicaciones multiplataforma.
    - La comunidad está formada por cerca de 1,4 millones de desarrolladores repartidos en 120 países .
  - La comunidad de desarrolladores de Ionic ha superado los 5 millones de desarrolladores en más de 200 países.

Ionic vs Xamarin

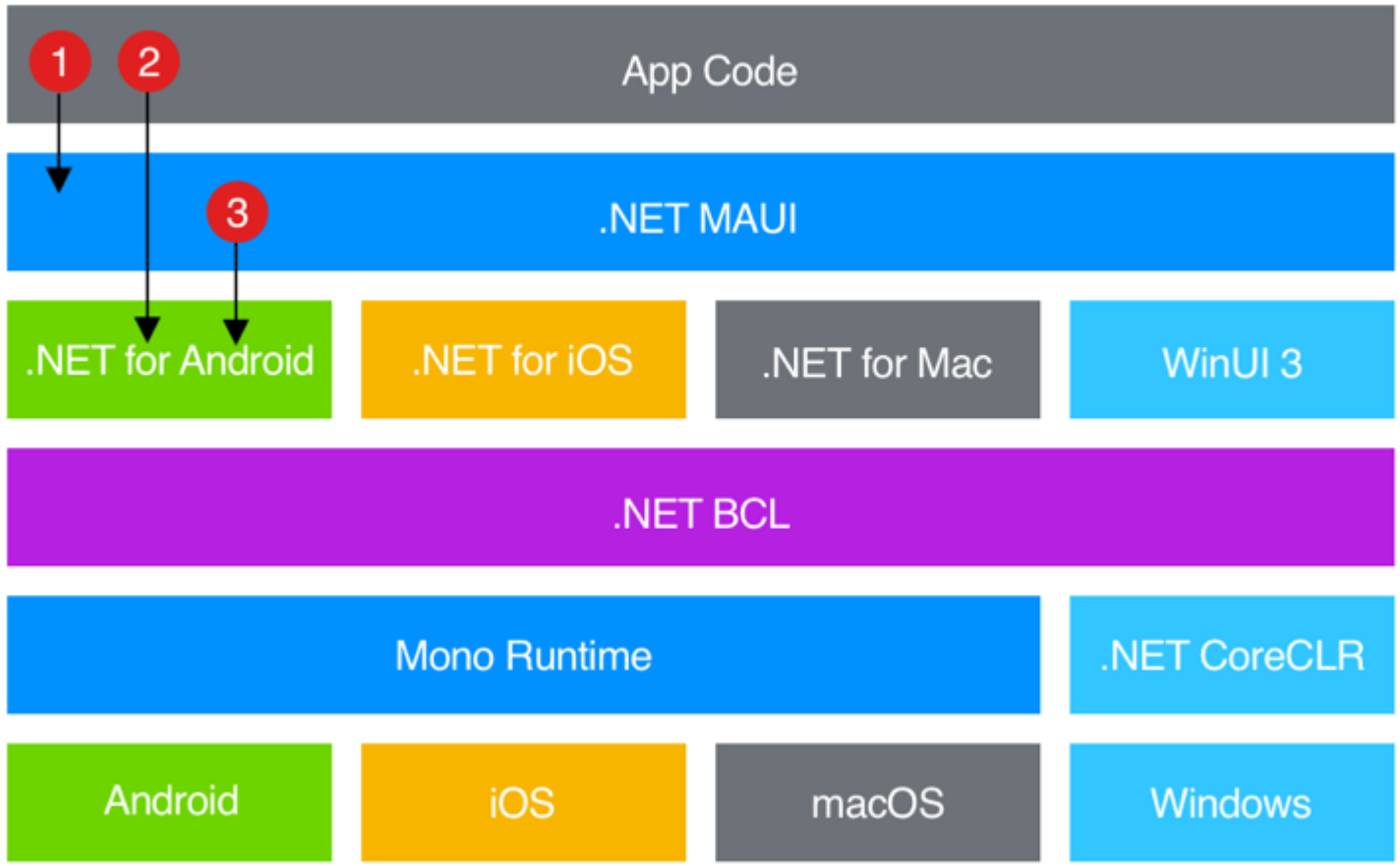
- **Conclusión**
  - Elegiremos **Xamarin** si:
    - Somos conocedores de **C# y .net** framework
    - Queremos crear aplicaciones multiplataforma **eficientes**.
    - No es demasiado importante el tamaño de la aplicación.
  - Elegiremos **Ionic** si:
    - La idea de la aplicación es nueva y queremos desarrolla una App de forma **rápida**.
    - La aplicación es **simple**, no requiere personalizaciones de alto nivel y necesita una experiencia similar a la nativa.
    - Tenemos experiencia con las **tecnologías** basadas en la **web**
    - El tamaño de la App es muy importe.

- **.NET MAUI (mayo 2022)**
  - .NET Multi-platform App UI (.NET MAUI) es un framework multiplataforma capaz de crear aplicaciones móviles y de escritorio **nativas** con C# y XAML.
  - .NET MAUI, permite desarrollar aplicaciones que se pueden ejecutar en Android, iOS, macOS y Windows desde una sola base de código compartida.



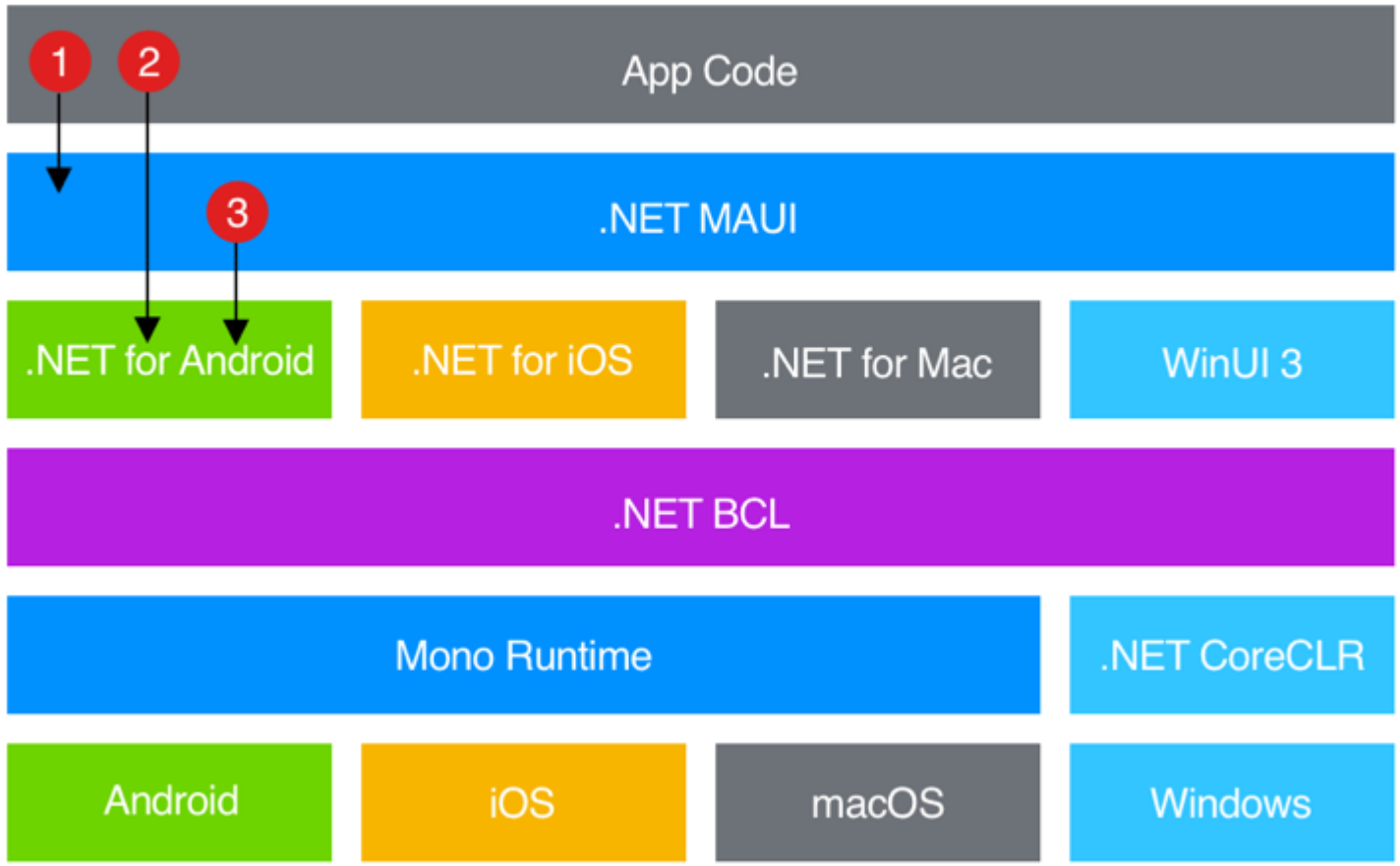
- **.NET MAUI**
  - .NET MAUI, código abierto.
  - Evolución de Xamarin.Forms.
  - Permite crear aplicaciones multiplataforma mediante un único proyecto.
  - Permite desarrollo en cualquier sistema operativo .
  - Aplicaciones multiplataforma en XAML y C#, desde una única base de código compartida en Visual Studio 2022.
  - Permite Compartir el diseño de la interfaz de usuario entre plataformas.
  - Permite Compartir código, pruebas y lógica de negocios entre plataformas.
  - La compilación de aplicaciones para iOS y macOS requiere un equipo Mac. (Al igual que xamarin)
  - Recarga activa de .NET

- **.NET MAUI Arquitectura**



En una aplicación MAUI de .NET, se escribe código que interactúa principalmente con la API MAUI de .NET (1). Después, .NET MAUI consume directamente las API de plataforma nativa (3). Además, el código de la aplicación puede ejercer directamente las API de la plataforma (2), si es necesario.

- **.NET MAUI Arquitectura**



- NET 6 o posterior proporciona una serie de frameworks para crear aplicaciones: biblioteca .NET Android, .NET iOS, .NET macOS y Windows UI 3 (WinUI 3)
- NET BCL abstrae los detalles de la plataforma subyacente del código.
- Para Android, iOS y macOS, mono implementa un entorno de ejecución de .NET. En Windows, .NET CoreCLR proporciona el entorno de ejecución.

- **Flutter**
  - **Flutter es un SDK** (Kit de Desarrollo de Software) desarrollado por Google **que permite crear aplicaciones móviles tanto para Android como para iOS.**
  - Ventajas:
    - Compila en **nativo** tanto en Android como en iOS.
    - La creación de interfaces gráficas es muy flexible, puede combinar diferentes Widgets para crear las vistas.
    - El desarrollo es muy rápido, permite ver el resultado de forma instantánea mientras se escribe código.
    - Permite realizar aplicaciones multiplataforma, utilizando Dart como lenguaje de programación.
    - Permite realizar aplicaciones móviles, web, escritorio, embebidas.