Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Víctor Manuel Rivas Santos, tutor del Proyecto Fin de Carrera titulado: Análisis del kit de herramientas Flutter y desarrollo de un prototipo de aplicación multiplataforma, que presenta Adson Henrique Moreira da Silva, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, noviembre de 2020

El alumno: Los tutores:

Adson Henrique Moreira da Silva Victor manuel Rivas Santos

Índice

Tabla de Ilustraciones

1. INTRODUCIÓN

Escribir una introducción

2. EL CAMINO HASTA FLUTTER

En este capítulo será abordado una historia concisa de la rápida evolución de internet y de los dispositivos móviles para entender como llegamos a herramientas como Flutter.

2.1. La rápida ascensión de internet y móviles

Es importante mirar al pasado y constatar que los móviles, para alcanzar el estado que conocemos hoy, tuvieran una rápida evolución. Hoy en día, los móviles tienen muchas más funcionalidades si comparado con antiguamente. Hoy no sirven apenas para llamadas, en verdad es que poca gente lo utiliza para eso, sino que sirve para muchas cosas como banca por internet, juegos, capturar fotos, estar conectado con amigos por redes sociales, mirar noticias en tiempo real, asistir televisión, hacer compras en línea, escuchar músicas y muchas otras. Y esta rápida evolución se debe casi que integralmente a revolución de internet, con la burbuja de las puntocom (medio de los años 2000) y con el suceso de las redes sociales. [1]

Siguiendo con el crecimiento de internet, muchas innovaciones llegaran para los móviles, en los mediados de los años 2000. Fabricantes de móviles empezaron a construir funcionalidades como reproductores de MP3, consola de juegos portátil, entre otros. El mercado de móviles que antes era dominado por Nokia, Sony Ericsson, LG; a principio de los años de 2007, junto con el aparecimiento del termo smartphone, el dominio pertenencia a fabricantes como HTC, BlackBerry y Apple (con el primero iPhone).

Con el suceso de redes sociales como Instagram, Facebook, los móviles adquirieron más funcionalidades. Por ejemplo, Instagram - aplicativo para compartir fotos - lanzado en 2010, trae la preocupación de las fabricantes para producir mejores cameras para sus teléfonos. Cuestiones antiguas como modo de espera, tiempos de llamadas, fueran cambiadas por otras como sistemas operativos, cualidad de las cameras, capacidad de la batería, descargar datos de forma más rápida [2].

Con el pasar de los años, con el mercado de móviles dominado por Apple con su sistema operativo iOS y sus productos, Huawei y Samsung con sus móviles optando por usar el sistema operativo Android desarrollado por Google; otras

fabricantes como BlackBerry, Microsoft fueran perdiendo el mercado de los móviles y consecuentemente de sistemas operativos móviles. [3]

Adicionalmente, en 2017 Android se torna el sistema operativo más utilizado en el mundo, superando a Windows (sistema operativo para ordenadores de mesa), conforme muestra la pesquisa, de la empresa de estadísticas web *StatCounter*. [4]

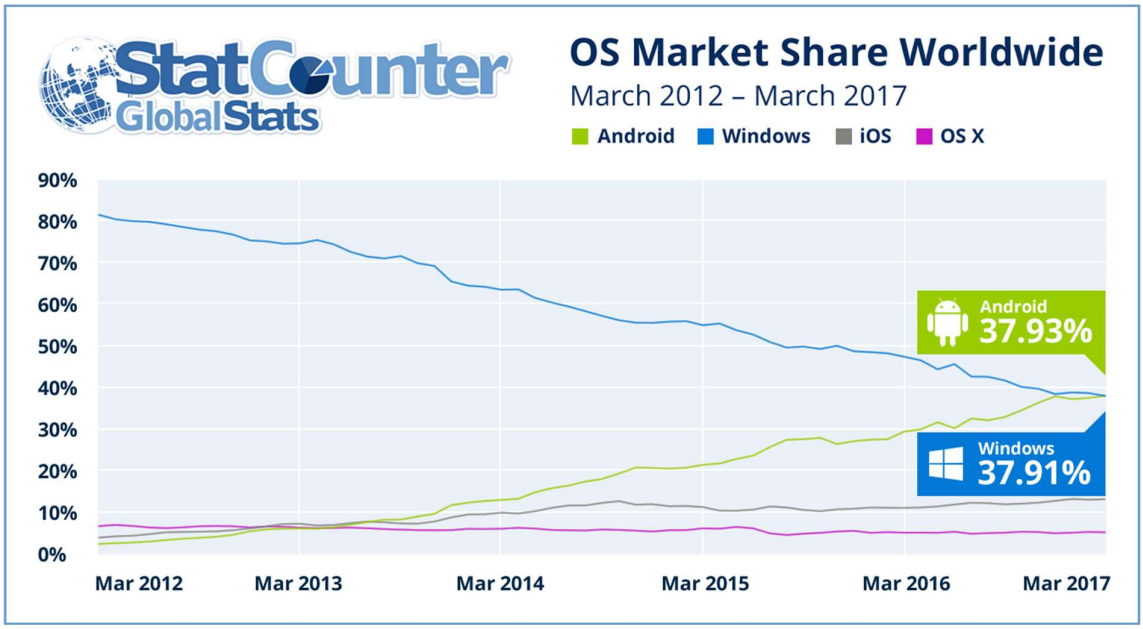

Ilustración 1. Sistemas operativos más usados en el mundo hasta 2017

Tomada de [4]

Según se evidencia en la gráfica abajo, Android sigue superando a Windows en el sistema operativo más utilizado en el entorno global.

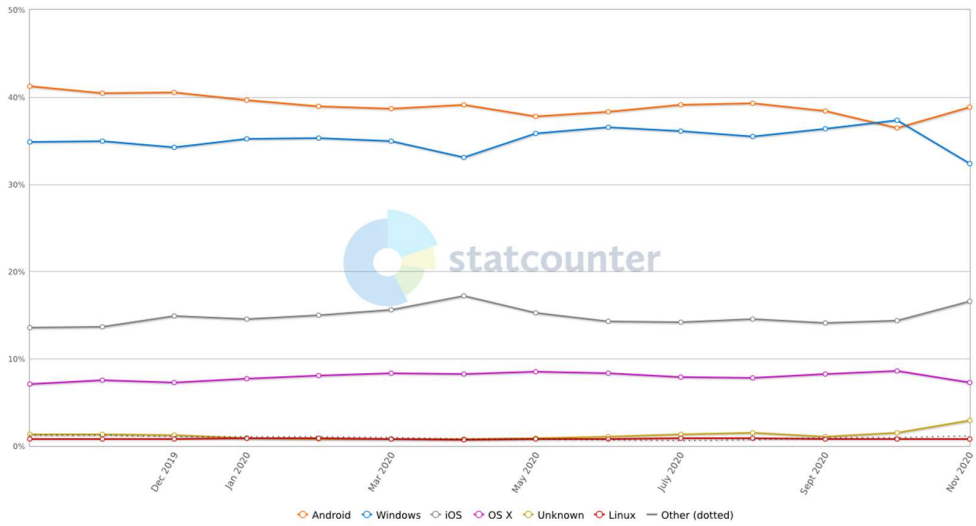

Ilustración 2. Sistemas operativos más usados en el mundo hasta noviembre de 2020

Tomada de [5]

La siguiente imagen, muestra en resumen lo sucedido a lo largo del tiempo con la revolución de internet y la tecnología móvil.

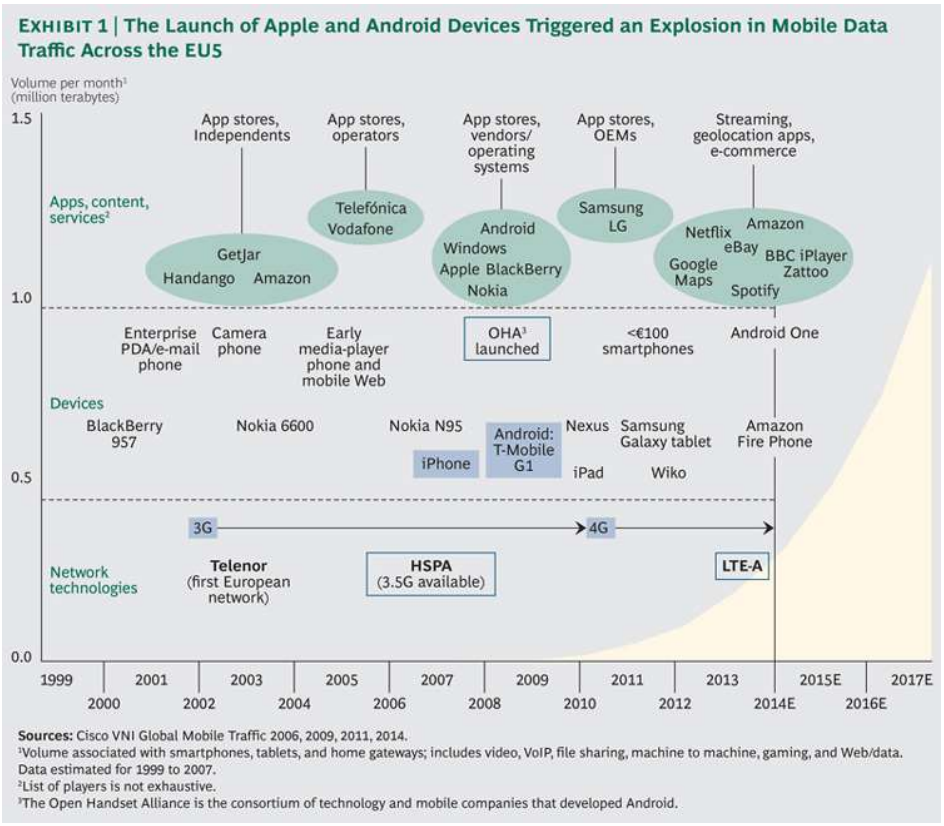

Ilustración 3. Evolución de los móviles y de la internet

Tomada de [6]

2.2. Android y iOS dominan el mercado

Con 5.19 mil millones de usuarios únicos en dispositivos móviles lo que representa 67% de la población mundial y con 53% de las peticiones a páginas de internet hechas por móviles, según la investigación de *We are Social*; los smartphones están en una ascensión sin precedentes. [7]

Según la misma investigación, el sistema operativo Android es responsable por 74% de las solicitudes a páginas de internet originadas de dispositivos móviles, mientras iOS es responsable por 25% y el 1% restante es compartido con otros sistemas operativos. [7]

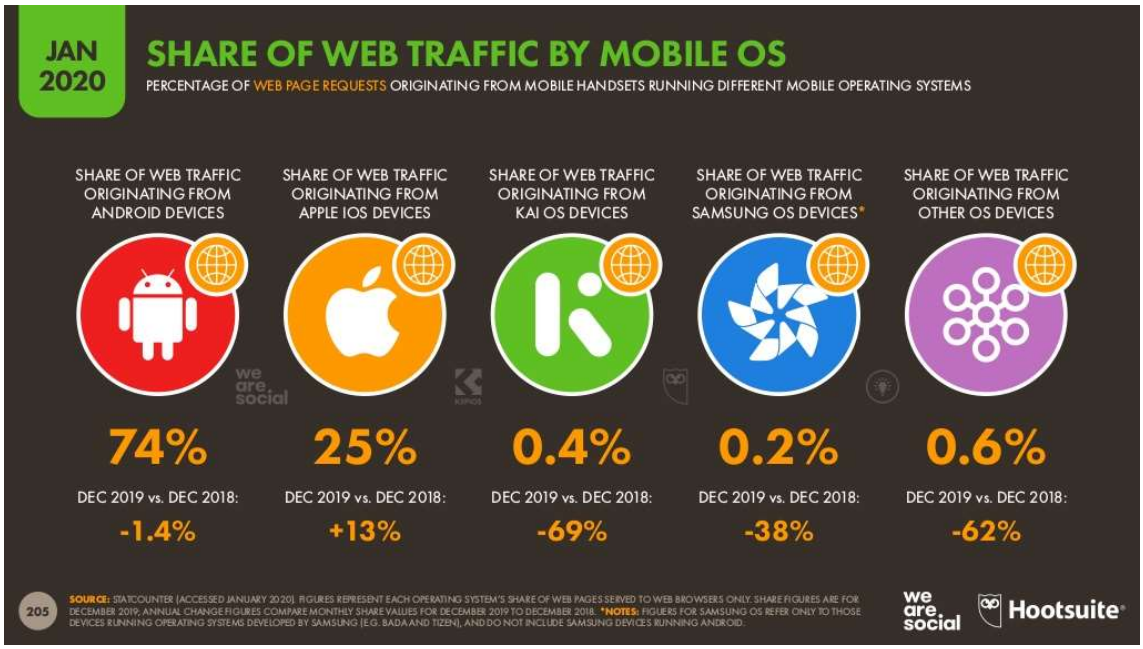

Ilustración 4. Requisiciones a páginas de internet a partir de dispositivos móviles

Tomada de [7]

Se puede evidenciar que Android y iOS dominando 99% del mercado. De un lado Android, de la gigante Google, sistema elegido por casi todas las fabricantes de móviles debido al código abierto - donde todas pueden descargar, modificar y redistribuir de forma gratuita. De otro lado iOS, de Apple, un sistema privado y que solo ella lo utiliza y lo modifica para un de sus principales productos, iPhone.

2.3. El mercado de aplicaciones

Las aplicaciones están presentes en la vida de todos que tienen un móvil, sea para usar redes sociales, hacer compras, jugar, entretenimiento, escuchar canciones, mapas. Hoy en día es muy difícil que no haya alguna actividad en la cual ya no haya una aplicación que te permite hacerla a través de un móvil.

Según el informe anual de *We are Social*, pasamos en media 3 horas y 40 minutos por día usando el móvil y ese valor crece 10% a cada año. De ese total de tiempo, pasamos 91% de ello en aplicaciones que descargamos o que vienen en el proprio móvil. Los otros 9% del tiempo pasamos usando los navegadores de internet presentes en el móvil. [7]

Para si tener una idea, el número de descargas de aplicaciones en 2019 fue de 204 mil millones, ese número tiende a crecer 6% al año. En el mismo año, los usuarios gastaron cerca de 120 mil millones de dólares en aplicaciones móviles y ese valor tiende a continuar creciendo a cada año, con una tasa de 20%.

Sabiendo esto, de todo el capital que los móviles están generando en los últimos años y que la tendencia es el crecimiento al largo de los años, la empresa que no tiene su negocio disponible online; sea por medio de aplicaciones, sea por medio de sitios web está perdiendo dinero y probablemente clientes.

2.4. El mercado de desarrollo de software móvil nativo

Hasta el surgimiento de Android y iOS el mercado de desarrollo de software móvil, en general, era pequeño y tímido, ya que los móviles, hasta entonces, no podrían hacer grandes cosas, eran limitados a hacer llamadas, mensajes de texto, juegos y otras cosas más, pero con mucha limitación si comparados a hoy.

Con el surgimiento de iOS y Android y debido a rápida evolución de los smartphones, al gran valor agregado que los mismos traen para los usuarios, todo este mercado cambió. Ahora con eses sistemas operativos, con la popularización accesible de internet y de los móviles no hay justificación para las empresas no desarrollaren sus proprias aplicaciones y ponerlas a disposición del cliente. Las empresas podrían elegir lanzar sus aplicaciones en una plataforma o en todas y las

que querían lanzar en varias plataformas (sea cualquiera el motivo) tendrían que desarrollar la aplicación muchas veces, una para cada plataforma. Pero mirando la diferencia entre las plataformas y los sistemas operativos, eso no era común.

Enfocando más en las plataformas que dominan el mercado y dado la diferencia entre Android, utilizaba inicialmente para desarrollo el lenguaje de programación Java y iOS inicialmente utilizando Objective-C, muchas empresas y desarrolladores optaban por lanzar sus aplicaciones solo en una plataforma, generalmente donde tenían la mayor parte de sus clientes. Entonces casos de aplicativos como Instagram, que fue lanzado primeramente en la plataforma iOS y solo dos años después lanzada en Android, era muy común.

Desde entonces el mercado de desarrollo móvil ha pasado por grandes evoluciones, expansiones e inversiones [8]. Es difícil encontrar una aplicación que no esté disponible en la mayoría de las plataformas y eso no quiere decir que el desarrollo nativo individual para cada plataforma ha acabado. La verdad es que las herramientas para el desarrollo nativo de aplicaciones, o más conocidos como SDK's, puestas a disposición por sus creadores han mejorado mucho, posibilitando escribir una aplicación nativamente sin tener mucho esfuerzo como antes.

A parte del mejoramiento de los SDK's nativos, hubo también un gran aumento de técnicas y herramientas de desarrollo no nativos de multiplataformas que permiten escribir una aplicación una vez y usarla casi que da la misma forma en todas las plataformas, lo que ha posibilitado entonces desarrollos rápidos y concisos.

2.5. Herramientas para desarrollo móvil multiplataforma

Con el crecimiento y la evolución de herramientas que permitían desarrollar una aplicación una sola vez y usar esa misma única aplicación en todas o casi todas las plataformas; muchas técnicas, softwares y *frameworks* surgieron con el propósito de facilitar la vida de los desarrolladores y de las empresas. A continuación, se relacionan algunas.

2.5.1. Xamarin

Xamarin es una plataforma de aplicación de código abierto y gratuita de Microsoft para la construcción de aplicaciones iOS y Android modernos y de alto desempeño usando los lenguajes de programación C# y .NET. [9]

2.5.2. PhoneGap

Desarrollado por el equipo de Apache Cordova, PhoneGap es un *framework* de código abierto que fue descontinuado. Utiliza lenguajes de programación usadas en el desarrollo web (HTML, CSS y Javascript) para desarrollar aplicaciones iOS, Android y otras. [10]

2.5.3. Titanium

Es un ambiente de desarrollo que te permite crear aplicaciones iOS, Android. Mantenida por la empresa AppCelerator, utiliza lenguajes de programación usadas en el desarrollo web como PhoneGap, con una pequeña diferencia que Titanium añade una capa de abstracción a más. [11]

2.5.4. React Native

React native es un *framework*, desarrollado por Facebook, para construcción de aplicaciones iOS y Android que utiliza el *framework* React – herramienta creada también por Facebook para creación de interfaces web. React native es gratuito y tiene el código abierto. Actualmente es una de las herramientas más utilizadas para el desarrollo multiplataforma. [12]

2.6. Herramientas multiplataformas.

Antes de empezar a entender el problema por detrás de esas herramientas que permiten el desarrollo rápido para que aplicaciones sean lanzadas em muchas plataformas, es importante entender la clasificación de las herramientas. Podemos clasificar todas esas herramientas citadas anteriormente y todas las otras que no fueran citadas en cuatro categorías: aplicaciones móviles web, aplicaciones web hibridas, aplicaciones interpretadas y aplicaciones generadas por multiplataformas. [13]

2.6.1. Aplicaciones móviles web

Esas aplicaciones son proyectadas para ser ejecutadas en un navegador web (*webview*) y son independentes de la plataforma de la cual están ejecutando. Es decir, ellas no requieren ninguna adaptación a medida que la plataforma cambia. Por otro lado, el tiempo de respuesta de esas aplicaciones son mayores si comparadas con aplicaciones construidas de forma nativa (usando SDK's fornecidos por las plataformas). Además, esas aplicaciones por razones de seguridad tienen permiso limitado para acceder a recursos nativos de un móvil, como la cámara o sensores.

PhoneGap es un ejemplo de esa categoría. Esta proyectado con tecnologías web para ser ejecutado en una *webview* (aplicación que ejecuta en un navegador web), conforme el ejemplo en las siguientes imágenes.

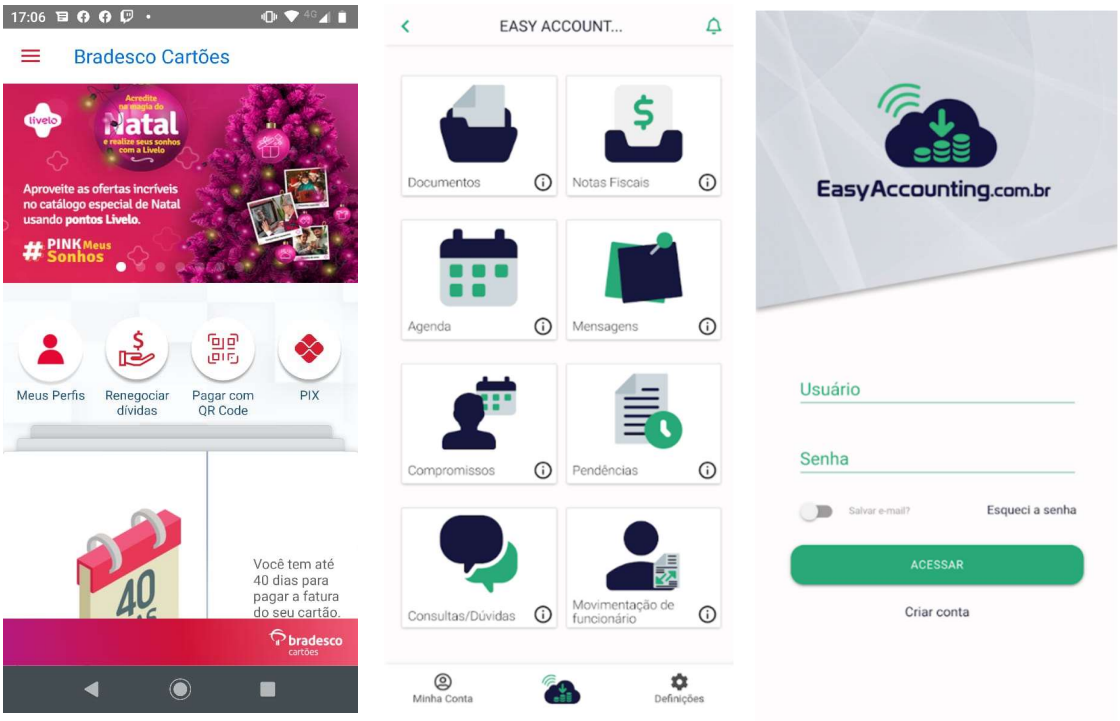

Ilustración 5. Ejemplos de aplicaciones móviles web

Elaboración propia

Por mucho que parezcan aplicaciones nativas, no son. Son aplicaciones brasileñas que son desarrolladas utilizando herramientas de desarrollo web dentro de una *webview.*

2.6.2. Aplicaciones web hibridas

Las aplicaciones web hibridas utilizan el mismo enfoque de la categoría anterior, pero ahora no ejecuta en una *webview.* Ejecutan en un *webcontainer* del dispositivo, que tiene mayor acceso a recursos nativos del dispositivo por medio de una interfaz de programación de aplicación, también conocida como API.

Aplicaciones hibridas ofrecen grandes ventajas porque te permiten reutilizar todo el código para varias plataformas. Pero también ofrecen desventajas como ejecución lenta debido al *webcontainer* y la experiencia del usuario al usar la aplicación es comprometida al no utilizar todos los recursos nativos disponibles.

La siguiente imagen muestra un ejemplo de una aplicación hibrida.

Ilustración 6. Ejemplo de aplicación web híbrida
Elaboración propia

Como se puede observar en la imagen anterior, ella usa el componente nativo del iPhone, *FaceID*, para autenticar el usuario. Esto permite una mejor integración del usuário con el aplicativo.

2.6.3. Aplicaciones interpretadas

Aplicaciones interpretadas es un proyecto en que casi todo el código escrito es traducido para el código nativo de la plataforma. En otras palabras, todo el código fuente de la aplicación desarrollada es implantada en el dispositivo, donde es interpretada usando algún tipo de mecanismo (generalmente un SDK de la herramienta elegida).

La herramienta Titanium citada anteriormente se encaja en esa categoría, pues cuando la aplicación es instalada en el dispositivo, todo el código es implantando en el dispositivo y es interpretado usando algún tipo de motor Javascript. En el caso de Titanium, Mozilla's Rhino es usado en el Android y JavascriptCore en iOS. [11]

La ventaja de este tipo de aplicación es que es independiente de la plataforma, consigue llegar a un nivel muy parecido de una experiencia nativa. Mientras las desventajas son la total dependencia del ambiente de desarrollo.

En la siguiente imagen se hay un ejemplo de una aplicación interpretada, donde en termos de interfaz de la aplicación, es casi imposible notar la diferencia entre ella y una aplicación nativa.

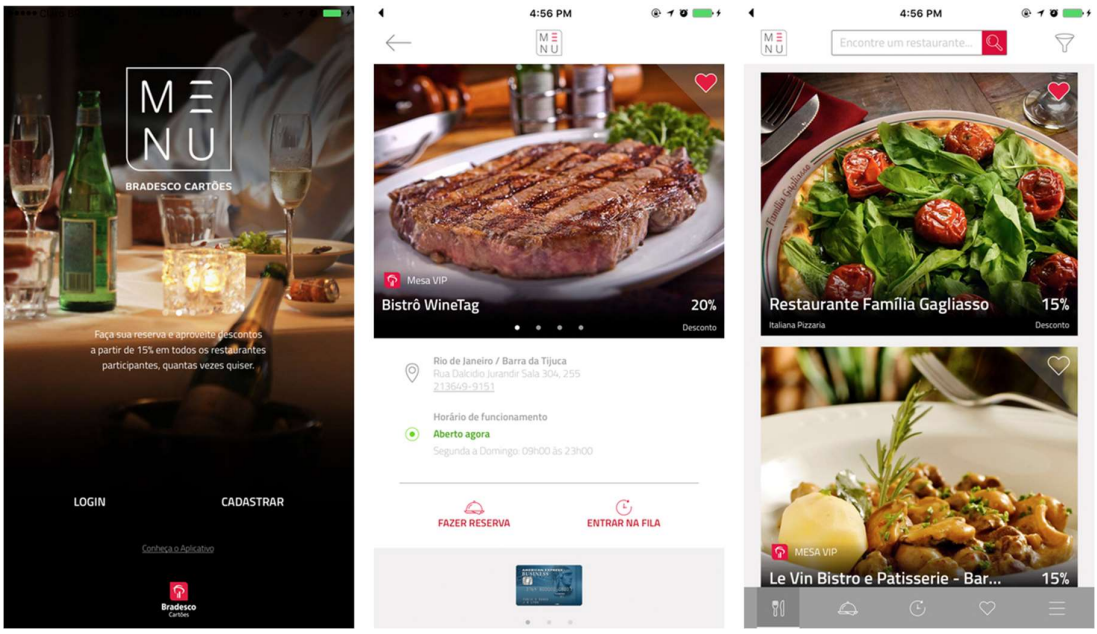

Ilustración 7. Ejemplos de aplicaciones interpretadas

Elaboración propia

2.6.4. Aplicaciones generadas por multiplataformas

Esa categoría es muy parecida con la anterior, la única diferencia es que en lugar del código ser interpretado, el código es compilado. Las aplicaciones son compiladas nativamente, creando una versión específica para cada plataforma de destino. Por ejemplo, antes de lanzar una aplicación, la herramienta hace una especie de "traducción" de la aplicación para la plataforma destino. Observe que no es el desarrollador que tiene ese trabajo, sino que la herramienta elegida. El desarrollador sigue desarrollando la aplicación una sola vez, pero compilando para las plataformas de destino.

La herramienta Xamarin es un ejemplo de esa categoría y te permite escribir el código una sola vez y compilar para la plataforma que esa herramienta soporta. Las ventajas de esa categoría son las mismas de la categoría anterior y las desventajas son cuanto al tamaño de la aplicación que tienden a ser mayores.

En la imagen abajo un ejemplo de una aplicación hecha con Xamarin.

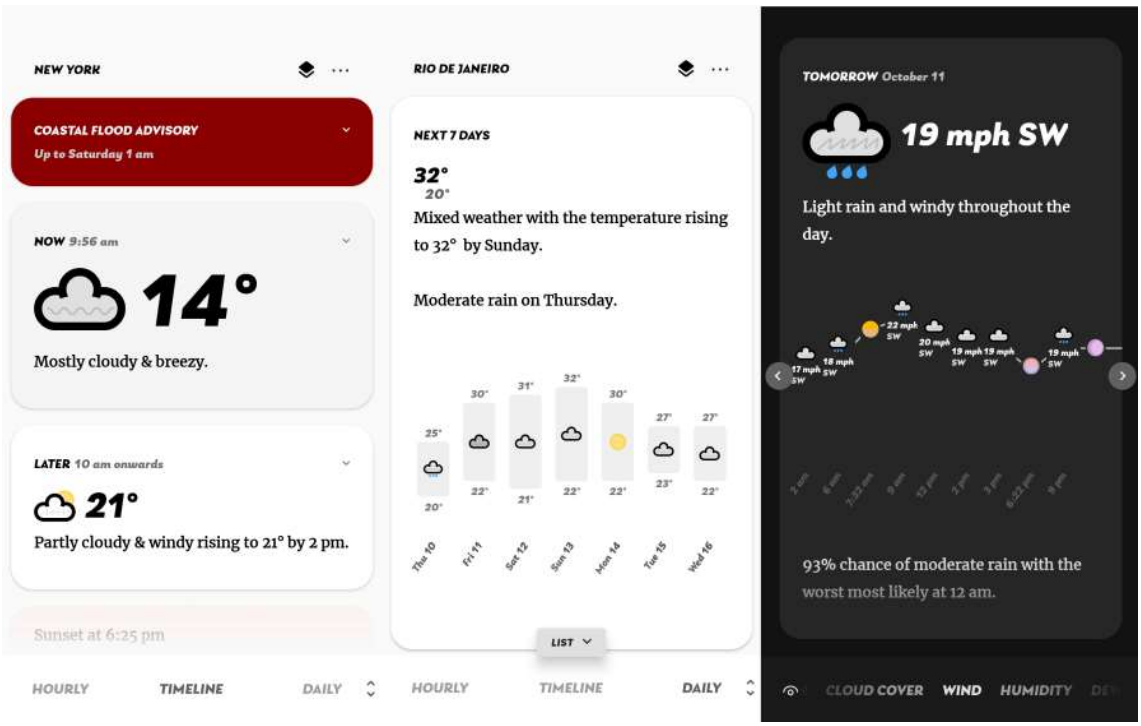

Ilustración 8. Ejemplos de aplicación hecha con Xamarin

Elaboración propia

2.7. El problema de herramientas multiplataformas

Hasta ahora vimos que la mayor parte de las desventajas de las herramientas son la perca de la experiencia nativa, la lentitud, el tamaño de las aplicaciones y la dependencia de ambientes de desarrollo. Eses problemas deben ser llevados en cuenta ya que influyen directamente en el funcionamiento de la aplicación y también en la experiencia del usuario final, lo que puede implicar en una mala evaluación de la aplicación y consecuentemente la perca de clientes.

En contrapartida, las aplicaciones desarrolladas de forma nativa en gran parte no sufren de esos problemas. Pero, desarrollo de aplicaciones nativas exigen algunos requisitos, como por ejemplo diferentes equipos para diferentes plataformas (debido a las diferencias entre plataformas, no es común que el mismo equipo trabaje en diferentes plataformas), diferentes bases de código para diferentes plataformas, equipos que deben contar con más personas. Todos estos requisitos influyen directamente en el coste de la construcción de la aplicación y en el tiempo para el desarrollo de esta.

De esta manera nasce Flutter - framework multiplataforma donde se desarrolla “una” vez, permitiendo agilidad, eficacia y un modelado de diseño único para cada aplicación, en donde sin importar la plataforma, esta se integrará para brindar la mejor experiencia de usuario posible - con el objetivo de agregar lo mejor de los dos mundos. En otras palabras, acabar o mitigar la mayor parte de los problemas de las herramientas multiplataformas y nativas.

3. FLUTTER

Flutter es una creación de la gran compañía Google. Según la propria compañía Flutter es el conjunto de herramientas de interfaz de usuario (UI) para crear hermosas aplicaciones compiladas de forma nativa para dispositivos móviles, web y de escritorio a partir de una única base de código. [14]

Flutter ha ganado vida en 2015 con el nombre de *Sky*. Al principio, por si tratar de una herramienta de Google, solo ejecutaba en dispositivos con el sistema operativo Android, pero después fue portado para iOS y hoy en día sigue cubriendo las dos plataformas que dominan este nicho de mercado.

Después de muchas versiones lanzadas desde el primer lanzamiento inicial en 2015, en diciembre de 2018 acontece el primer lanzamiento estable de Flutter, eso significa que la herramienta esta lista para que la comunidad (desarrolladores, compañías, dentro muchos otros) pueda empezar a utilizarla para crear y poner en marcha aplicaciones reales.

3.1. ¿Para qué sirve Flutter?

Flutter ha nacido con el intuito de una herramienta que agilice el proceso de desarrollo de software, es decir, a través de una base de código, ser posible compilar para multiplataformas, sin tener la necesidad de tener una base de código para cada plataforma.

Pero Flutter surge también con la necesidad de acabar con los problemas comunes a estés tipos de herramientas. Y sabiendo de eso, uno de los principales objetivos de Flutter es ser capaz de renderizar interfaces a una constancia de 120 FPS. Lo que posibilita bastante fluidez en las aplicaciones, donde realmente hay una experiencia nativa y la lentitud no es perceptible.

3.2. Partes principales de Flutter

Flutter es constituido de cuatro partes principales. [8]

3.2.1. Flutter *Engine*

El motor de Flutter es casi todo desarrollado en el lenguaje de programación C++. Su base de código utiliza el mecanismo gráfico Skia, que es una biblioteca gráfica de código abierto también escrita en C++ patrocinada y mantenida por Google, para hacer la renderización. [15]

3.2.2. Foundation Library

Flutter también provee una interfaz sobre los SDK's nativos (de ambas las plataformas – iOS y Android). Eso quiere decir que no es necesario se preocupar en como inicializar el aplicativo de cámara en iOS (programáticamente hablando) y como hacer eso también en Android, por ejemplo. Todo lo que tiene que saber es como llamar el aplicativo de cámara usando Flutter y entonces Flutter se encarga de hacer funcionar en la plataforma que cuya aplicación estará ejecutando.

3.2.3. Dart

Diferente de Android, Google ha elegido para Flutter el uso del lenguaje de desarrollo Dart; que también fue creada por Google en 2011 y tuvo su primer lanzamiento estable en noviembre de 2013, cerca de dos años antes del lanzamiento de Flutter.

Antes de Flutter, pocas personas conocían dicho lenguaje. Pero debido a Flutter, Dart está evolucionando muy rápido desde el lanzamiento. La siguiente imagen muestra un gráfico sobre los lenguajes de programación que fueron más relevantes para los desarrolladores en 2019. [16]

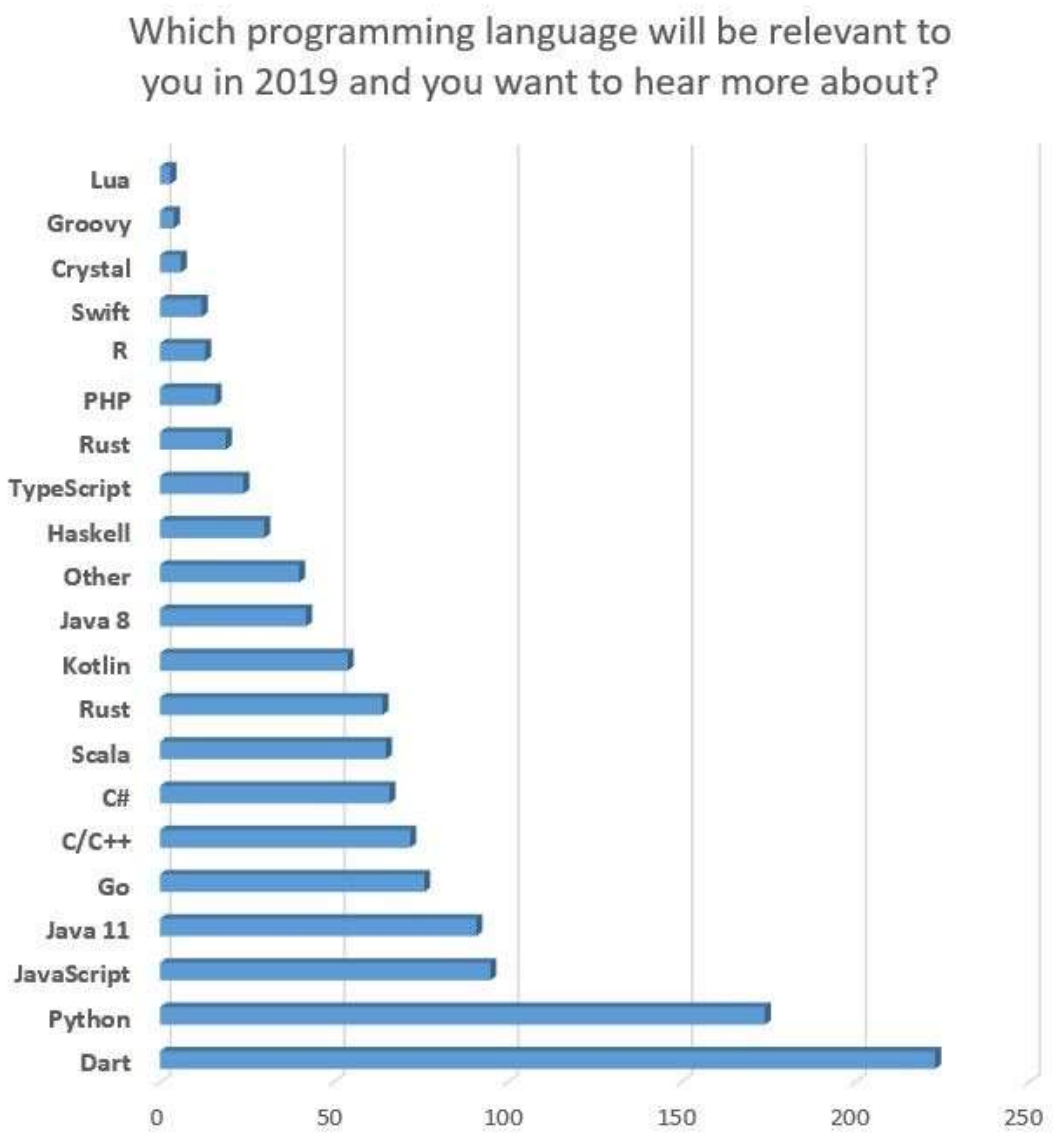

Ilustración 9. Lenguajes de programación más relevantes en 2019 para desarrolladores

Tomada de [16]

Es importante tener en cuenta que Dart no es solo un lenguaje de desarrollo móvil, sino que también para desarrollo IoT (Internet de las Cosas), aplicaciones web, entre otras.

3.2.4. Widgets

Por ultimo y quizás la más importante, los widgets. En Flutter, todo es tratado como un widget y no es un exagero hablar así, porque en Flutter realmente todo es un widget.

Un widget es todo componente dibujado en la pantalla que se puede observar, ver y también los que no se pueden. La mayor de los widgets es dibujable, es decir, son perceptibles en una pantalla. La parte de código que representa el widget también es un widget. [17]

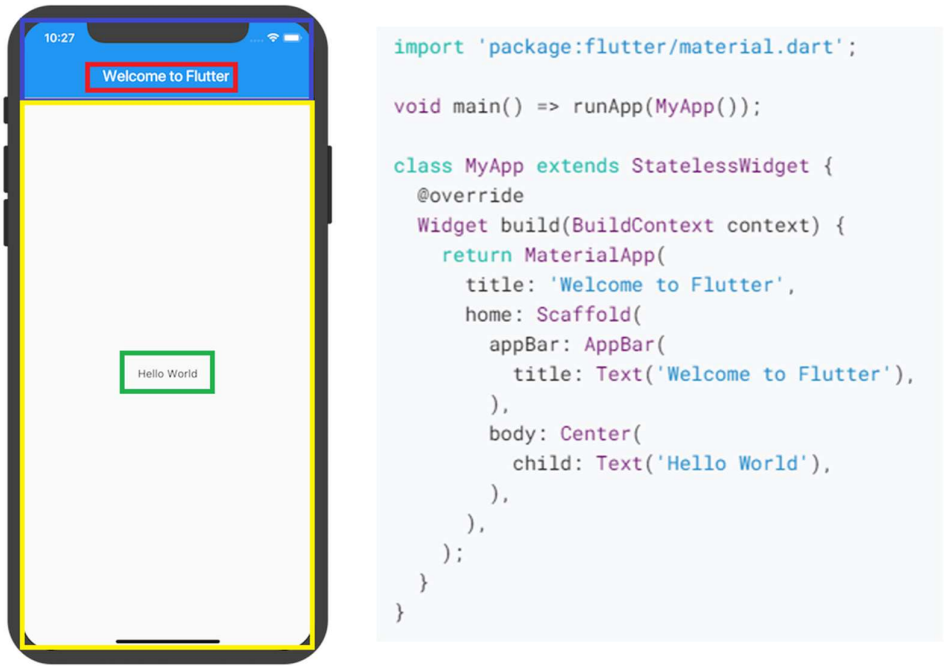

Ilustración 10. Representación de widgets

Elaboración propia

En la imagen arriba, en el móvil a la izquierda cada rectángulo formado por una color es un widet diferente. Al lado derecho, el código que representan cada widget destacado en la izquierda.

En Flutter algunos widgets poden tener hijos; algunos pueden tener uno solo hijo, mientras otros poden tener más de uno y eses hijos poden tener otros hijos y así sucesivamente, lo que resuelta entonces en una gigante hierarquia de widgets, también conocido como árbol de widgets.

3.3. Ventajas

Por ser una herramienta nueva, hasta el momento de la escrita de este documento, Flutter tiene en sus componentes muchas ventajas que son llevadas en cuenta por desarrolladores de softwares. Algunas de ellas, según Zammetti (2019) [8], son:

3.3.1. Hot Reload

Esa ventaja te permite, según Google [18], experimentar y crear interfaces de usuario, agregar funciones y corregir errores de forma rápida y sencilla; esa funcionalidad inyecta archivos de códigos recién actualizados en la máquina virtual de DART en ejecución. Así Flutter reconstruye automáticamente el árbol de widgets sin la necesidad de reiniciar la aplicación a cada modificación.

3.3.2. Multiplataforma de verdad

Según Google, sus aplicaciones Flutter ejecutarán correctamente en ambas las plataformas, con el mínimo de esfuerzo. Debido a Flutter proveer dos grupos de widgets, *Material Design* y *Cupertino* para Android y iOS respectivamente, el desarrollador puede crear aplicaciones con experiencia nativa para cada sistema plataforma distinta.

3.3.3. Dart

Debido a Dart no tener una curva de aprendizaje tan larga como Java, muchos desarrolladores y no desarrolladores pueden empezar con Flutter sin grandes esfuerzos, visto que Dart es similar a lenguajes como Javascript, Java y Objective-C.

Es importante decir también que en la propria documentación de Google sobre el lenguaje Dart, hay un apartado completo de introducción de Dart para desarrolladores Java y también muchas otras herramientas cuyo objetivo es el mejor proceso de aprendizaje. [19]

3.3.4. Widgets

Además de Flutter te permitir crear sus proprios widgets, también tiene un rico conjunto de widgets ya creados disponibles, tanto para iOS como para Android, lo que hace que esa sea una de sus principales ventajas. Sin contar que ofrece muchos más widgets, sin la necesidad de API de terceros, si comparado con React native, por ejemplo.

3.3.5. Herramientas

Flutter te permite empezar con el ambiente desarrollo de forma muy rápida y fácil, porque en Flutter no hay mucha dependencia de ambiente de desarrollo.

Por lo tanto, si un desarrollador quiere empezar con un ambiente más avanzado con la utilización de otras herramientas, Flutter te permite usar muchas de esas herramientas.

3.4. Desventajas

Así como cualquier herramienta, Flutter tiene sus puntos negativos que necesitan ser evaluados por quién desea utilizar ese *framework* y así juzgar si Flutter es realmente la mejor opción.

3.4.1. Herramienta nueva

Por si tratar de una herramienta nueva, hay muchas dudas cuanto a su credibilidad. Por más que pertenezca a una gran compañía como Google, lo que trae un cierto nivel de seguranza; eso no es suficiente.

Herramientas nuevas traen por sí solas muchas dudas y inseguridad, no solo si tratando de la empresa que las ha creado, sino que también de muchos errores iniciales, no tener disponible una gran comunidad. Y en Flutter no es diferente, por más que el interés en Flutter ha crecido en los últimos años desde su lanzamiento, como puede ser observado en la siguiente imagen, que fue retirada de una búsqueda en Google Trends. [20]

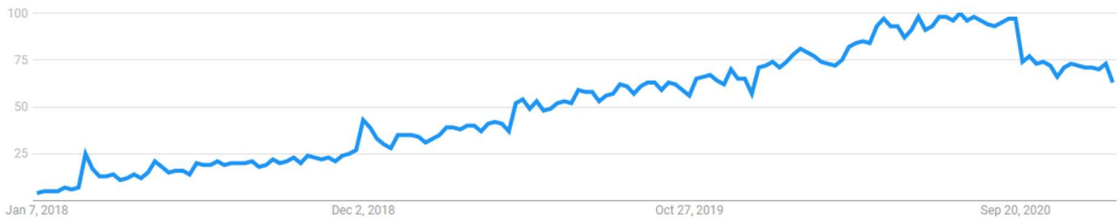

Ilustración 11. Crecimiento de la búsqueda de Flutter en Google (2018 - 2020)
Tomada de [20]

3.4.2. Mezcla de código

Diferentemente de otros ambientes de desarrollo como, por ejemplo, desarrollo web; en Flutter hay mezcla de código de lógica y estilización. Para ejemplificar mejor esa desventaja, en desarrollo web, por ejemplo, en la mayoría de los casos la estilización es separada de la parte lógica. HTML, CSS y JS para la estilización de

interfaz y algún otro lenguaje de desarrollo para la parte lógica (PHP, JS, Java, entre otras).

En Flutter todo es hecho con Dart. Entonces es común tener el código de lógica junto con el código de estilización. Claro que para muchos desarrolladores eso no llega a ser una desventaja, es una cuestión de costumbre con la herramienta.

La siguiente imagen exhibí un ejemplo de código en Flutter, donde hay una mezcla de código de la parte lógica (rectángulo rojo) y código de estilización (rectángulo verde). [21]

```
class MyButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: () {
        print('MyButton was tapped!');
      },
      child: Container(
        height: 36.0,
        padding: const EdgeInsets.all(8.0),
        margin: const EdgeInsets.symmetric(horizontal: 8.0),
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(5.0),
          color: Colors.lightGreen[500],
        ),
        child: Center(
          child: Text('Engage'),
        ),
      ),
    );
  }
}
```

Ilustración 12. Fragmento de ejemplo de la mezcla de código en Flutter

Tomada de [21]

Oficialmente, Google no ha lanzado ninguna forma o actualización que solucione dicho “problema”, hasta el momento. Pero la comunidad ha creado sus proprias soluciones; la más común es la creación de funciones para las partes lógicas en archivos distintos y entonces solo hacen la llamada de esas funciones.

3.4.3. Google

Hay personas que consideran el hecho de que Flutter pertenece a Google una ventaja y hay sentido en eso. Pero para Zammetti (2019) [8] eso hace todo menos confortable, ya que Google controla muchas cosas en la internet hoy en día. Como mencionado anteriormente, Flutter, Dart y hasta Skia fueran creadas por Google o son

mantenidas por él. Entonces con todo ese control Google puede dictar normas y descontinuar la herramienta cuando quiera, así como ya ha pasado antes con otras tecnologías.

3.4.4. Árbol de Widgets

Los widgets en si son una maravilla, pero cuando se trata de la junción de ellos, la jerarquía que generan entre sí tiende a ser un terror para muchos desarrolladores.

La profundidad que puede llegar una jerarquía de árbol de widgets en Flutter, muchas veces es mucho mayor que en HTML, por ejemplo; lo que torna el desarrollo en Flutter un poco asustador. Claro que la comunidad de Flutter ya tiene soluciones para eses problemas, como herramientas para identificación y comentarios en el código para que el desarrollador no se perca dentro de la jerarquía de widgets.

3.4.5. Tamaño de las aplicaciones

Aplicaciones desarrolladas con Flutter tienden a ser un poco mayores que aplicaciones desarrolladas con SDK's nativos. Eso se debe porque las aplicaciones necesitan incluir en su estructura el motor de Flutter (Flutter *Engine*), el framework de Flutter, librarías de soporte y algunos recursos más.

3.4.6. Programación reactiva y gerenciamiento de estado

Flutter sigue el paradigma de programación reactiva, así como React native. Pero a diferencia de React Native, en Flutter cada widget tiene un requisito casi que obligatorio, que es el método *build.*

El método *build* retorna una representación visual que incorpora el estado actual del widget. En otras palabras, suponiendo la siguiente imagen, donde tenemos un texto en el centro de la página que conta el número de veces que el botón fue presionado. El texto nada más que un widget que tiene como estado actual el numero cero indicando que el botón aún no fue presionado. Pero a medida que el botón es presionado, este widget de texto empieza a reaccionar al clic en el botón. El método *build* entonces es llamado para reconstruir nuevamente el widget de texto para que muestre el nuevo valor. [22]

Ese paradigma puede ser muy atractivo no solo para Flutter como para otras herramientas como React native, pero algunas veces puede tornar cosas fáciles más difíciles do que deberían ser. En conjunto con eso entra el tema de gerenciamiento de estado en lo cual no existe una forma en concreto, sino que inúmeras formas de manipulación del estado de la aplicación y de los widgets y cada forma contiene sus ventajas y desventajas.

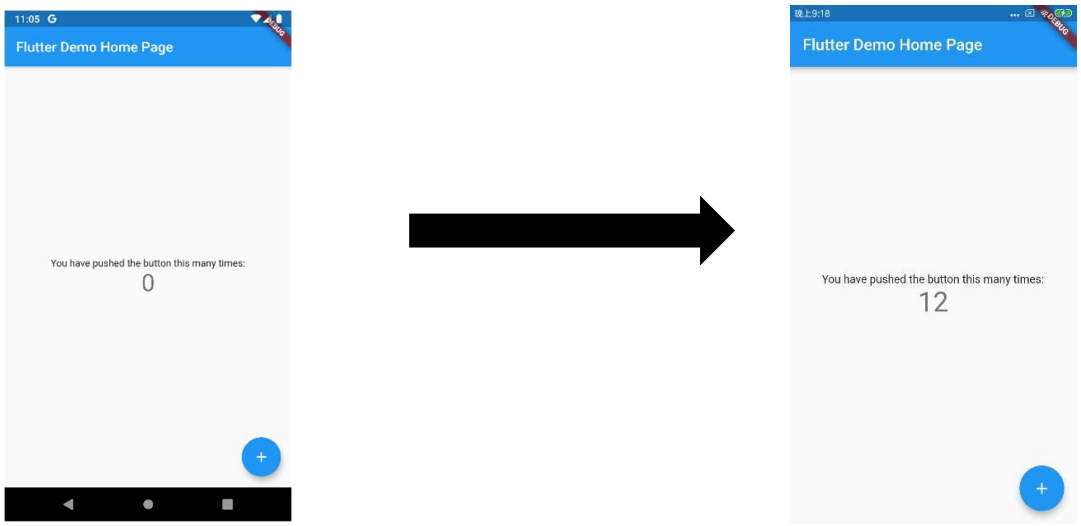

Ilustración 13. Demonstración del paradigma reactivo de programación y el gerenciamiento de estados

Elaboración propia

3.5. ¿Porque utilizar Flutter?

Flutter a comparación de la mayor parte de las otras herramientas disponibles, renderiza sus proprios componentes de interfaz y no usa componentes nativos, como la mayoría de las herramientas hace. Por ejemplo, cuando un desarrollador escribí un código para dibujar un botón en la pantalla, Flutter no hace llamada para el sistema operativo para renderizar el botón, sino que Flutter renderiza el dicho botón por sí solo.

La siguiente imagen muestra como React native hace la renderización de un componente de botón, debido a esa forma de trabajo, no solo de React Native, la medida que la aplicación crece acaba se tornando más lenta. Luego se nota que Flutter tiene una ventaja sobre esas herramientas.

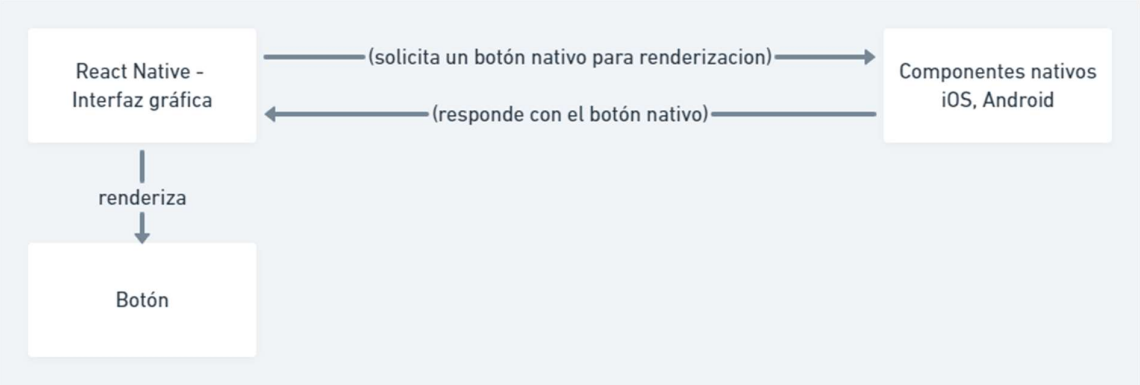

Ilustración 14. Modo de construcción de widget en React Native

Elaboración propia

Debido a su propria forma de renderizar componentes, Flutter puede proyectar sus proprios widgets. En otras palabras, Flutter no está vinculado al sistema operativo en se tratando de widgets. Eso permite a Flutter ofrecer dos kits de conjuntos de widgets: *Material Design* widgets (Android) y *Cupertino* widgets (iOS), lo que brinda una experiencia nativa.

Es importante decir también que Flutter no impone restricciones cuanto a renderización de widgets. En otras palabras, si el desarrollador quiere renderizar un widget que es común en Android, en iOS; lo puede hacer. Así como puede renderizar un widget común en iOS en Android. Claro que influye en una cuestión de experiencia nativa de ambos los sistemas operativos, ya que no es común una persona que tiene un teléfono con Android tener widgets similares al de iOS.

Ilustración 15. Diferencia de widget en Android y iOS

Elaboración propia

La imagen arriba es un ejemplo de la diferencia de un widget entre los dos sistemas operativos.

3.6. ¿Quién usa Flutter y por qué?

Con la rápida ascensión de Flutter en los últimos años desde su lanzamiento y debido a sus puntos positivos en comparación con otras tecnologías existentes del

mismo ramo, Flutter ha atraído la atención de grandes empresas, visto que es mucho mejor tener un solo base de código y la compilación de esta para multiplataformas, que mantener varias bases de códigos, una para cada plataforma distinta.

Multiplataformas no quiere decir solamente iOS y Android, sino que web y aplicaciones para ordenadores personales también. Claro que Flutter ha evolucionado mucho más en se tratando del mundo móvil, pero sigue teniendo muchos avanzos en los otros ramos. Así que, en un futuro próximo, dependiendo de cómo esa tecnología avance, podremos tener una sola base de código y aprovechar esa misma base para hacer aplicaciones para todas las plataformas, con pequeñas modificaciones de una para otra.

Teniendo en cuenta el coste, es mucho más ventajoso para las compañías mantener un solo base de código, ya que, por ejemplo, para una empresa mantener una base de código para cada plataforma, muy probablemente debería tener más de uno equipo de desarrollo, ya que las tecnologías que cada plataforma utiliza, en general, son distintas.

Sabiendo de eso y con todas ventajas que Flutter ha proporcionado desde su lanzamiento, empresas como BMW, NuBank, Ebay, Tencent, Google, Alibaba Group, New York Times entre muchas están a utilizar Flutter. Aplicaciones como Google Ads, Insight Timer, Stadia, Baidu, NuBank, Philips Hue, Ebay ya están hechos con Flutter. [23]

3.7. ¿Cuánto cuesta Flutter?

Hasta el momento, Flutter sigue siendo una herramienta de código abierto lo que significa que cualquier persona puede usarla, modificarla sin tener que pagar nada. Como dicho anteriormente y siguiendo el ritmo del mercado de nuevas tecnologías, su creadora Google tiene la costumbre de ofrecer muchas tecnologías desarrolladas por ella sin la necesidad de pagar, hasta cierto punto.

Es ventajoso para Google seguir con Flutter siendo gratis, pues así la comunidad tiende a crecer mucho y evolucionar cada vez más esta tecnología y quién sabe así tornarse la mayor compañía en este mercado también.

3.8. ¿Como aprender Flutter?

Así como la mayoría de las nuevas tecnologías que surgen, Google no ha hecho diferente con Flutter. En el sitio web oficial de Flutter (https://flutter.dev/), Google ofrece toda una documentación sobre la herramienta. Es importante recordar que aprender Flutter implica también en aprender el lenguaje Dart, ya que es el lenguaje usada por Google en Flutter. Dart también ofrece un sitio web (https://dart.dev/) oficial con toda la documentación necesaria para aprender este lenguaje.

Empezando con la instalación, la documentación de Flutter cobre gran parte, sino todo, que hay en Flutter. En el mismo apartado, hay una parte solo enseñando como configurar el ambiente de desarrollo y un tutorial de como escribir su primera aplicación en Flutter.

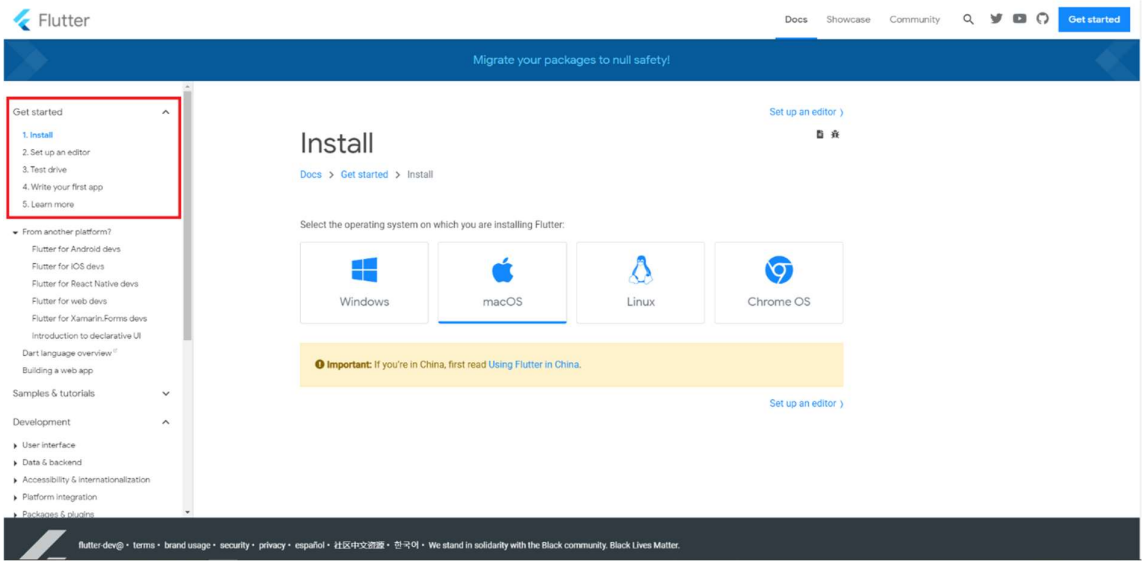

Ilustración 16. Página oficial de instalación de Flutter

Tomada de [24]

En la imagen arriba, marcado con el rectángulo rojo, tiene todas las etapas iniciales para quién desea empezar con Flutter.

Pero claro, si ya tiene experiencia con desarrollo de software con alguna otra tecnología, las cosas deben ser más fáciles. Se puede encontrar en el mismo sitio web un apartado solo para desarrolladores que ya tienen experiencia con otras tecnologías. Este apartado ayuda a desarrolladores a entender Flutter partiendo de tecnologías como Android (nativo), iOS (nativo), React native, tecnologías web o

Xamarin. En otras palabras, si conoces el funcionamiento de alguna de esas tecnologías, la tendencia es que el entendimiento de Flutter sea más fácil.

En el mismo apartado hay una visión general sobre el lenguaje Dart y también de cómo utilizar Flutter y Dart para la construcción de una aplicación web.

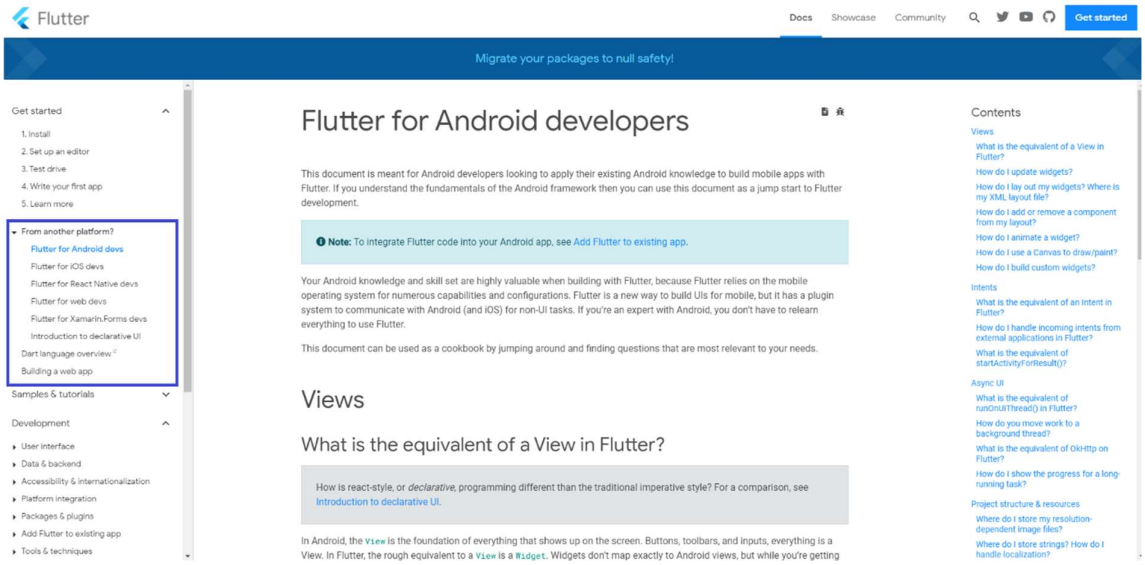

Ilustración 17. Documentación de Flutter para desarrolladores de otras plataformas

Tomada de [25]

Luego en seguida, hay otro apartado en que hay muchas aplicaciones hechas con Flutter para que el interesado pueda observar cómo es una aplicación en tiempo real. Además, también hay tutoriales y enlaces para códigos de terceros.

Más adelante, ya se puede notar una parte más descriptiva y avanzada. En otras palabras, cada tecnología o asunto necesario para desarrollo en Flutter está expuesto detallado e individualmente. Por ejemplo, si quieres saber cómo es el funcionamiento de datos en una aplicación Flutter, ahí hay apartados para ese tema y muchos otros.

La documentación de Flutter, engloba también la parte de testes en aplicaciones, como depurar y también como desplegar su aplicación en las tiendas de aplicaciones oficiales, Google Play (Android) y App Store (iOS).

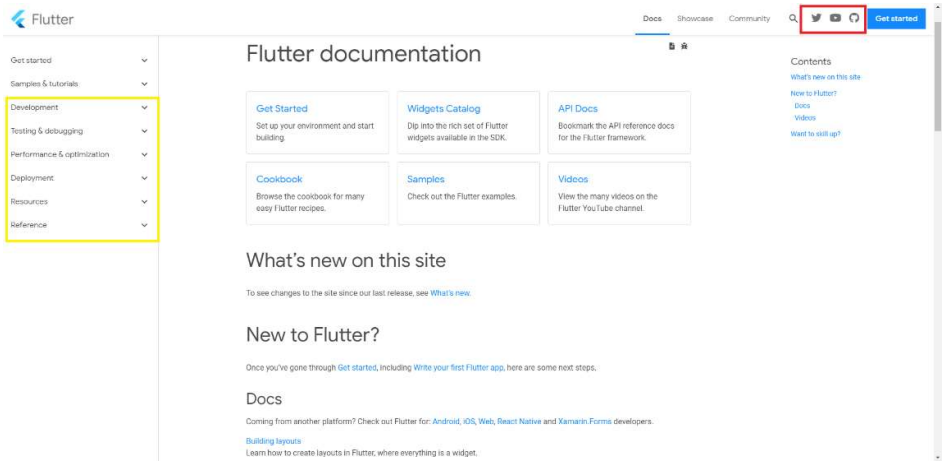

Ilustración 18. Contenidos más avanzados y detallados de Flutter y sus canales de comunicación oficial

Tomada de [26]

La forma de aprender es una elección personal, porque hay muchas formas de aprender Flutter. Actualmente ya hay muchos cursos siendo ofrecidos em muchas plataformas; en YouTube ya es posible encontrar muchos videos tutoriales sobre esa herramienta; además de los canales oficiales de la propria herramienta, en lo cual hay disponibilidad de muchas más informaciones oficiales.

4. Bibliografía

[1] Quora (2011). "What factors led to the bursting of the Internet bubble of the late 1990s?". Recurso digital https://www.quora.com/What-factors-led-to-the-bursting-of-the-Internet-bubble-of-the-late-1990s. [Accedido el 15/12/2020]

[2] Netstar (2020). "How mobile phone technology has changed over the last 40 years". Recurso digital https://www.netstar.co.uk/mobile-phones-years/. [Accedido el 15/12/2020]

[3] PuroMarketing (2011). "Google y Apple consolidan su dominio en el mercado de los smartphones". Recurso digital https://www.puromarketing.com/12/10750/google-apple-consolidan-dominio-mercado-smartphones.html. [Accedido el 16/12/2020]

[4] StatCounter (2017). "Android overtakes Windows for first time". Recurso digital https://gs.statcounter.com/press/android-overtakes-windows-for-first-time. [Accedido el 15/12/2020]

[5] StatCounter (2020). "Operating System Market Share Worldwide". Recurso digital https://gs.statcounter.com/os-market-share. [Accedido el 24/11/2020]

[6] BCG (2014). "The mobile Internet Economy in Europe". Recurso digital https://www.bcg.com/publications/2014/telecommunications-technology-digital-mobile-internet-economy-europe. [Accedido el 02/12/2020]

[7] We are Social (2020). "DIGITAL 2020: 3.8 BILLION PEOPLE USE SOCIAL MEDIA". Recurso digital https://wearesocial.com/blog/2020/01/digital-2020-3-8-billion-people-use-social-media. [Accedido el 14/12/2020]

[8] Zammetti, Frank. (2019). "Practical Flutter: Improve Your Mobile Development with Google's Latest Open-Source SDK". Apress. [Accedido el 16/11/2020]

[9] Microsoft (2020). "What is Xamarin". Recurso digital https://dotnet.microsoft.com/learn/xamarin/what-is-xamarin. [Accedido el 15/12/2020]

[10] Adobe PhoneGap (2020). "Build amazing mobile apps powered by open web tech". Recurso digital https://phonegap.com/. [Accedido el 15/12/2020].

[11] Appcelerator (2020). "Titanium Mobile Development Environment". Recurso digital https://www.appcelerator.com/Titanium/. [Accedido el 15/12/2020]

[12] GitHub (2020). "React Native". Recurso digital https://github.com/facebook/react-native. [Accedido el 15/12/2020]

[13] Delia, Lisandro & Galdámez, Nicolás & Thomas, Pablo & Corbalán, Leonardo & Pesado, Patricia. (2015). "Multi-platform mobile application development analysis". ResearchGate. [Accedido el 15/12/2020]

[14] Flutter (2020). “Flutter”. Recurso digital https://flutter.dev/. [Accedido el 16/12/2020]

[15] Skia (2021). “Skia Graphics Library”. Recurso digital https://skia.org/. [Accedido el 18/01/2021]

[16] Jaxenter (2019). “Results are in: Fully fledged Dart takes first in 2019 pool”. Recurso digital https://jaxenter.com/poll-results-dart-word-2019-154779.html. [Accedido el 18/01/2021]

[17] Flutter (2021). “Write your first Flutter app, part 1”. Recurso digital https://flutter.dev/docs/get-started/codelab. [Accedido el 18/01/2021]

[18] Flutter (2021). “Hot reload”. Recurso digital https://flutter.dev/docs/development/tools/hot-reload. [Accedido el 18/01/2021]

[19] Dart (2021). “Codelabs”. Recurso digital https://dart.dev/codelabs. [Accedido el 18/01/2021]

[20] Google Trends (2021). “Flutter”. Recurso digital https://trends.google.es/trends/explore?date=2018-01-01%202021-01-01&q=%2Fg%2F11f03_rzbg. [Accedido el 18/01/2021]

[21] Flutter (2021). “Introduction to widgets”. Recurso digital https://flutter.dev/docs/development/ui/widgets-intro. [Accedido el 18/01/2021]

[22] Flutter (2021). “Test drive”. Recurso digital https://flutter.dev/docs/get-started/test-drive. [Accedido el 18/01/2021]

[23] Flutter (2021). “Apps take flight with Flutter”. Recurso digital https://flutter.dev/showcase. [Accedido el 18/01/2021]

[24] Flutter (2021). “Install”. Recurso digital https://flutter.dev/docs/get-started/install. [Accedido el 18/01/2021]

[25] Flutter (2021). “Flutter for Android developers”. Recurso digital https://flutter.dev/docs/get-started/flutter-for/android-devs. [Accedido el 18/01/2021]

[26] Flutter (2021). “Flutter documentation”. Recurso digital https://flutter.dev/docs [Accedido el 18/01/2021]