PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO DE SISTEMAS Y COMPUTACIÓN

FRAMEWORKS PARA EL DESARROLLO DE APLICACIONES MÓVILES MULTIPLATAFORMA COMPILADAS DE FORMA NATIVA – ESTUDIO COMPARATIVO Y EJEMPLOS PRÁCTICOS

AUTOR: KEVIN SEBASTIÁN ZURITA NÚÑEZ

DIRECTOR: ING. OSWALDO ESPINOSA

QUITO, DICIEMBRE 2020

Agradecimiento

Gracias a Dios y a la Madre Dolorosa a quien me he consagrado en el colegio y desde entonces siempre me ha acompañado. Madre Querida, guíame siempre, naufragaré sin ti.

Gracias a mis padres Silvana y William, por su amor y su apoyo incondicional que he sentido desde el primer día de vida. Ellos me han formado como la persona que soy ahora y me han brindado la oportunidad de tener una educación de calidad que permite convertirme en el profesional que quiero llegar a ser.

A mi hermano Anthony, por enseñarme que con esfuerzo y dedicación todo es posible, por los innumerables consejos que he recibido de su parte y por ser el ejemplo al que siempre he tratado de imitar.

Al director de este trabajo, Oswaldo Espinosa, quién me ha brindado su apoyo durante este proceso y siempre estuvo dispuesto a resolver cualquier inquietud que se fue presentando. Gracias por el tiempo y la predisposición de guiarme en este último escalón de mi carrera universitaria.

A mis amigos quienes de uno u otra forma me han brindado su apoyo durante el transcurso de la universidad. Gracias por sus consejos, por sus buenos deseos y sobre todo gracias por las risas las cuales estoy seguro todos guardaremos en nuestras memorias.

Finalmente quiero agradecer a todas las personas y docentes que han contribuido con el desarrollo de este trabajo de manera directa o indirecta.

Dedicatoria

Dedico este trabajo a mi madre Silvana, nada de esto fuera posible sin su ayuda y sin el cariño que me lo ha demostrado desde que tengo memoria. Ella me ha enseñado la importancia de ayudar a los demás sin esperar nada a cambio. Sus afectos y sus detalles son los que hacen la diferencia en un mundo que es cada vez es más indolente.

A mi padre William quien me ha enseñado la importancia del trabajo justo y honesto. Quien además me ha demostrado con su ejemplo que hay que enfrentar de cara cualquier tipo de problema y que hay que hacer todo lo posible, siempre y cuando no afecte a los demás, para conseguir cualquier objetivo que tenga planteado.

A mi hermano Anthony, quien además de ser un buen ejemplo es también quién me motiva por ser cada día mejor. No hay ni siquiera un momento en vida en el que recuerde que no esté él. Nuestros logros son el mayor reconocimiento que podemos brindar al sacrificio de nuestros padres.

Resumen

En años anteriores, una de las limitantes que tenían los programadores al momento de desarrollar una nueva aplicación móvil era la capacidad con la que contaban los dispositivos. Actualmente y gracias al avance tecnológico que se ha producido en este campo, esto no representa un mayor problema, de hecho, el problema ahora es escoger una herramienta de desarrollo de entre las tantas que existe. Al tener menos obstáculos con los que lidiar, las personas buscan maneras de desarrollar aplicaciones de forma rápida, eficiente y que se adapten a las características de la gran variedad de dispositivos que existen actualmente.

Como respuesta a la demanda de un mayor número de aplicaciones, con el tiempo se han establecido nuevas maneras de construir desarrollo de forma más eficaz. Para esta problemática, las aplicaciones móviles multiplataforma compiladas de forma nativa, son de las mejores opciones que existen actualmente en el mercado. Esta tecnología permite construir una aplicación móvil con una base de código única y que es adaptable tanto para dispositivos tanto con iOS como Android.

La presente investigación expone un análisis descriptivo y comparativo de cuatro frameworks de desarrollo específicos que son: Flutter de Google, NativeScript de Progress, React Native de Facebook y Xamarin de Microsoft. Estas herramientas son las más utilizadas actualmente por la comunidad de desarrolladores, esto es en parte al soporte que tienen, la popularidad que ha ido adquiriendo, la información con la que cuentan y sobre todo por los lenguajes de programación que utilizan.

El análisis mencionado se lo realizará primero exponiendo la teoría de cada una de las herramientas y su comparación se mostrará por medio de tablas, gráficas ilustrativas o en donde sea más factible observar la información. De esta manera y en base a lo expuesto en el presente trabajo, la tarea de escoger una decisión se vuelva un poco más sencilla o por lo menos que se pueda tener un panorama más claro sobre lo que ofrece cada tecnología, así como las ventajas y desventajas que traería trabajar con cualquiera de estas herramientas.

Como parte complementaria del trabajo de disertación, se presenta el desarrollo de ejemplos prácticos construidos mediante la utilización de los frameworks anteriormente mencionados. En estos, además de mostrar cómo es su codificación, se muestra también las diferencias que presentan en la construcción de una misma tarea. De esta forma, se expondrá dos ejemplos con interacción

hacia el usuario, uno en el que muestre como se manejan las alertas en forma de notificación, y otro donde se observe como se puede acceder a librerías externas que ejecuten una tarea en específico, en ente caso, enviar un correo electrónico.

Índice de Contenido

Índice de Ilustraciones

Índice de Tablas

1. Introducción

1.1. Justificación

Los dispositivos móviles se han convertido en una herramienta indispensable para la mayoría de las personas. Con estos se pueden cumplir con varias tareas que son propias del ser humano y que se lo realizan a diario, estudiar, trabajar, informarse, comunicarse, entre otras. Todo esto es posible gracias a las aplicaciones móviles que se ejecutan en los dispositivos, las cuales han ido en crecimiento, tanto en número como en su capacidad de rendimiento.

Aunque la mayor parte de los usuarios desconocen de su origen, todo programa informático conlleva un precio, además de horas de estudio y dedicación que los programadores aportan para el desarrollo de una nueva aplicación. Por este motivo a lo largo de los años, se han creado o perfeccionado herramientas que facilite de alguna manera su trabajo, adaptándose a cada necesidad y haciendo posible un desarrollo más eficiente.

Actualmente existen varios frameworks que cuentan con características específicas, cada uno tiene su propio lenguaje, comportamiento y manera de ejecutar que responden a diferentes necesidades. Elegir al marco de trabajo correcto entre todo un grupo de herramientas similares, es de vital importancia para que se pueda obtener el mayor beneficio de los recursos invertidos en un proyecto de desarrollo.

Los resultados obtenidos en estudios de este tipo con el tiempo y la evolución constante que tiene las herramientas que se analizan se van quedando obsoletos, en otras palabras, conforme pasa el tiempo cualquier estudio parecido carece de importancia y su aporte es cada vez menor. Por lo tanto, una actualización a los resultados obtenidos previamente es de gran importancia pues puede ofrecer nuevos puntos de vista, los cuales no han sido considerados debido a la capacidad que tenían los dispositivos o el surgimiento de nuevas tecnologías.

Finalmente, es importante señalar que para que este trabajo tenga relevancia y aporte en conocimiento, se tomará en cuenta solamente bibliografía actual y los más recientes aportes que haya realizado la comunidad. En otras palabras, para la recopilación de información de cada herramienta, solo se utilizará material que se haya publicado a partir del 2018 y los aportes y actualizaciones más recientes que haya tenido cada una de estas.

1.2. Objetivo General

Realizar un estudio comparativo entre frameworks de desarrollo que permitan crear aplicaciones móviles multiplataforma que se ejecuten de manera nativa en un dispositivo.

1.3. Objetivos Específicos

Diferenciar mediante un análisis los tipos de aplicaciones móviles que existen y reconocer cuál se adapta más a una necesidad específica citando las mismas.

Realizar una investigación acerca de los frameworks que permitan la creación de aplicaciones nativas multiplataforma, de tal manera que sirva de base para el análisis comparativo.

Comparar, mediante parámetros establecidos, la utilidad y el beneficio que se puede obtener de los frameworks existentes, facilitando así la elección de una herramienta de desarrollo.

2. Aplicaciones Móviles

2.1. Definición

Una “app” o aplicación es un programa informático generalmente diseñado para funcionar sobre dispositivos móviles, este permite a los usuarios llevar a cabo una o varias tareas en equipos electrónicos fáciles de transportar. Una aplicación móvil puede ser de cualquier tipo, sencilla o compleja, y en cualquier caso está diseñada para que la persona u organización que la haya construido tenga pleno control sobre esta. Las “apps” pueden ser entendidas como un atajo que lleva al usuario a donde quiere ir, sin tener que pasar por una línea intermedia de búsqueda (Gardner & Davis, 2014).

Con el tiempo, las aplicaciones móviles se han convertido en la esencia de cualquier dispositivo, tanto así que en la actualidad es imposible concebir la idea de un teléfono móvil sin aplicaciones. Las aplicaciones tienen un número casi ilimitado de funcionalidades que pueden cumplir, estas permiten acceder a contenido multimedia, información de cualquier tipo, permiten jugar y divertirse y dan la posibilidad de establecer una comunicación en cualquier parte del mundo.

La mayoría de las aplicaciones que se encuentra actualmente en el mercado son destinadas a un gran número de usuarios finales, por lo cual son liberadas en versiones rápidas que permitan satisfacer la demanda del mercado. Aunque existen varios tipos y formas de desarrollar una aplicación móvil, es necesario realizar un análisis de costo/beneficio, el cual evalúe las herramientas a utilizar previo a la construcción de una nueva aplicación (Thomas, y otros, 2018).

El desarrollo de aplicaciones móviles es un fenómeno relativamente joven que crece a un ritmo realmente acelerado. Esto se debe a la popularidad que tienen los teléfonos inteligentes, así como también el aumento progresivo en sus capacidades de almacenamiento y procesamiento, sin dejar de mencionar que cada vez son una poco más accesibles (Erfani, Mesbah, & Kruchten, 2013).

El análisis, diseño y construcción de una aplicación móvil no es una tarea fácil y al contrario de lo que muchos piensan, su desarrollo no es parecido al web. Si bien parecen ser un fenómeno

reciente, las aplicaciones surgieron en conjunto con los dispositivos móviles, de hecho, ya estaban incluidas en sistemas operativos anteriores a los que se usa actualmente. El auge que ha sufrido esta industria es en parte consecuencia de mejoras en el rendimiento de dispositivos y la creación de nuevas herramientas que facilitan una producción de aplicaciones más rápida y eficiente (Cuello & Vittone, 2013).

En la actualidad el mercado de aplicaciones móviles es realmente competitivo y para mantener o mejorar la calidad de sus trabajos, los desarrolladores necesitan datos sobre su rendimiento, la frecuencia que está siendo utilizada, el número de usuarios entre otras métricas. Por esta razón, no existe ni un solo tipo de aplicaciones, ni una sola forma de desarrollarlas, pues la demanda aumenta mucho más rápido que la oferta (Ravindranath, y otros, 2012).

2.2. Aplicaciones Web

Son un tipo de aplicaciones basadas en la web las cuales no hacen uso de los recursos del dispositivo, pero necesitan de un navegador para operar, en otras palabras, necesitan obligatoriamente de una conexión a internet para su funcionamiento. Utilizan tecnologías web (HTML, JavaScript y CSS) por lo que su implementación es sencilla, aunque se ven notablemente desfavorecidas en su acceso limitado al hardware y a la información del dispositivo (Xanthopoulos & Xinogalos, 2013).

Tiene varias limitantes como por ejemplo la interacción cliente-servidor (requiere un tiempo de respuesta considerable) y la falta de acceso a las capacidades del dispositivo, esto hace que la experiencia del usuario se vea claramente afectada, haciendo que esta opción para el desarrollo de “apps” sea en la actualidad la menos preferida (Delia, Galdamez, Thomas, Corbalan, & Pesado, 2014).

Este tipo de aplicaciones no necesitan ser instaladas directamente en el dispositivo para poder ejecutarse pues corren dentro de un navegador. Su objetivo principal es el de mostrar información y se las puede sacar mejor partido aprovechando las opciones que brinda la web, por ejemplo, haciendo uso de la posición geográfica, la información compartida, entre otros (Ramírez, 2012).

Si bien las aplicaciones web no representan un mayor grado de utilidad, su implementación rápida y relativamente sencilla, permite publicar versiones de manera rápida y eficiente, siempre y cuando el objetivo de la aplicación sea simplemente mostrar información y tener una interacción simple con el usuario.

2.3. Aplicaciones Híbridas

Las aplicaciones híbridas son un tipo especial de desarrollo multiplataforma basado en tecnología web, es decir, utiliza como principales componentes a HTML, JavaScript y CSS, pero no son no son ejecutadas por un navegador. Este tipo de aplicaciones funcionan dentro de un contenedor web especial que permite el uso de los recursos propios del dispositivo. Aunque el desarrollo de aplicaciones nativas permita la reutilización de código en diferentes plataformas y el acceso al hardware del equipo, el uso de componentes no nativos en la interfaz vuelve más lento el software y perjudica la experiencia del usuario (Thomas, Galdamez, Delia, Corbalán, & Pesado, 2015).

Se habla de híbridas porque estas aplicaciones son una combinación de aspectos nativos con tecnología web y para funcionamiento necesitan ser instaladas en el dispositivo. Para operar en los sistemas necesitan de contenedores nativos (UIWebView en iOS y WebView en Android). Para acceder a los recursos requieren de API´s especializadas que pueden ser de origen nativo o externo (Xanthopoulos & Xinogalos, 2013).

Representa un mayor grado de utilidad con respecto a las aplicaciones web, pero tienen ciertas limitantes, sobre todo con respecto al acceso de las capacidades del dispositivo. Aunque su implementación demanda cierta complejidad, no deja de ser sencilla pues aprovecha la tecnología web para una construcción rápida, la cual no demanda el uso de herramientas específicas, ni tampoco de lenguajes especiales.

2.4. Aplicaciones Nativas

Las aplicaciones nativas son desarrollos específicos que se hacen para cada uno de los dispositivos móviles que existen. Se los realiza utilizando el lenguaje propio de plataforma, así

como herramientas que lo hacen posible. La ventaja que provee este tipo de aplicaciones es que son desarrolladas a modo de que puedan ser ejecutadas sin necesidad de una conexión a internet. Al ser un software específico, una vez instalado en el dispositivo puede hacer uso pleno de sus capacidades sin tener que conectarse a servicios o recursos en línea (Rodríguez, 2017).

Las aplicaciones nativas tienen ciertas ventajas que sobresalen sobre los demás tipos de aplicaciones. Estas tienen pleno acceso a las capacidades del dispositivo, es decir, pueden hacer uso de la cámara, la información, el giroscopio y de los demás recursos propios del equipo. Así mismo, tienen un alto rendimiento y pueden ser ejecutadas en segundo plano, notificando al usuario sólo en caso de requerir su atención (Thomas, y otros, 2017).

Sin embargo, el uso de tecnologías específicas para representa un desafío considerable para el desarrollo de software, ya que el código escrito para una plataforma móvil (Java para Android y Objective-C para iOS) no se puede ser reutilizado. La fragmentación hace que el desarrollo y mantenimiento de aplicaciones nativas sea uno de los principales desafíos técnicos que afectan a la comunidad de desarrolladores. Como resultado de las consecuencias que produce el desarrollo nativo está que se representa un tiempo de desarrollo potencialmente alto, altos costos de prueba y mantenimiento, y baja portabilidad (Malavolta, 2016).

Aunque, por otro lado, las aplicaciones nativas tienen un nivel de funcionalidad elevado pues están en capacidad de responder a cualquier necesidad que se presente, e incluso pueden ser diseñadas para ser utilizadas sin una conexión a internet. Estas aplicaciones, además, pueden ser publicadas y distribuidas sin problema por las tiendas de aplicaciones correspondientes, llegando a una mayor cantidad de usuarios específicos.

2.5. Aplicaciones Nativas Multiplataforma

Las aplicaciones nativas multiplataforma son desarrollos que procuran optimizar la relación costo/beneficio, esto se lo puede realizar a través de compartir una misma base de código para la implementación de las diferentes plataformas.

Este tipo de aplicaciones tiene la ventaja de que su costo de producción es menor al de las aplicaciones nativas, además el tiempo que se requiere para su construcción, se ve considerablemente acortado con respecto a desarrollar aplicaciones específicas para cada tipo de dispositivos (Delia, Galdamez, Thomas, Corbalan, & Pesado, 2014).

Estas aplicaciones funcionan con un tipo especial de compilación, específicamente con compilación cruzada. Esta forma especial permite ejecutar software de manera nativa, esto es posible gracias a la creación de una versión específica de alto rendimiento para cada plataforma destino. Las herramientas que permiten este tipo de desarrollo generalmente utilizan un lenguaje propio de entrada, y generan código fuente que el dispositivo acepte de forma nativa. (Thomas, y otros, 2017).

Si bien las aplicaciones móviles multiplataforma representan un cierto grado de beneficio con respecto a las completamente nativas, en ocasiones se ven condicionadas debido a que no pueden acceder a los recursos del dispositivo de manera directa. Para hacer uso de las características propias de los teléfonos, se requiere de herramientas externas que, aunque tengan el mismo comportamiento, su rendimiento en ocasiones se ve limitado.

3. Frameworks para el Desarrollo de Aplicaciones Móviles Multiplataforma

3.1. Flutter

Flutter fue lanzado como versión beta en 2015 y presentado oficialmente por Google en su evento I/O Developer Conference del 2017. Este es un framework multiplataforma que tiene como objetivo principal desarrollar aplicaciones móviles de alto rendimiento. Estas aplicaciones pueden ser ejecutadas no solamente en iOS y Android, sino también en Fuschia, el sistema operativo de Google de próxima generación (Wu, 2018).

Según los creadores de Flutter, esta herramienta es un SDK (Software Development Kit) que sirve para crear hermosas aplicaciones, compiladas de forma nativa, para móvil, web y escritorio desde una única base de código. Esta herramienta tiene como objetivo permitir a los desarrolladores ofrecer aplicaciones de alto rendimiento que se sientan naturales en diferentes plataformas (Flutter, s.f.).

Flutter utiliza el lenguaje de programación Dart, este fue introducido por Google en 2011 y no ha sido muy utilizado por la comunidad de desarrolladores. Dart es un lenguaje fácil de usar, sobre todo para personas familiarizadas con los lenguajes de tecnología Java. Es compatible con la mayoría de los conceptos orientados a objetos y posee de una basta información que está disponible su sitio web oficial (Sharma & Gupta, 2020).

Este Framework tiene la característica que en lugar de utilizar widgets propios de la plataforma para la cual se está desarrollando, mediante su propio motor de renderizado, construye cada componente de vista. Este comportamiento y el beneficio que ofrece Dart con la compilación AOT (Ahead of Time), brinda la posibilidad de crear aplicaciones que tengan un rendimiento igual o muy cercano al nativo. (Wu, 2018).

En cuanto a su arquitectura, esta herramienta está organizada en una serie de capas relacionadas entre sí. El objetivo de este diseño es el de ayudar al programador hacer más con menos. Por ejemplo, como se observa en la Ilustración 1, la capa Material se construye en base a la capa de Widgets y esta a su vez se ensambla a partir de niveles inferiores de renderización en la capa de Rendering (Flutter, s.f.).

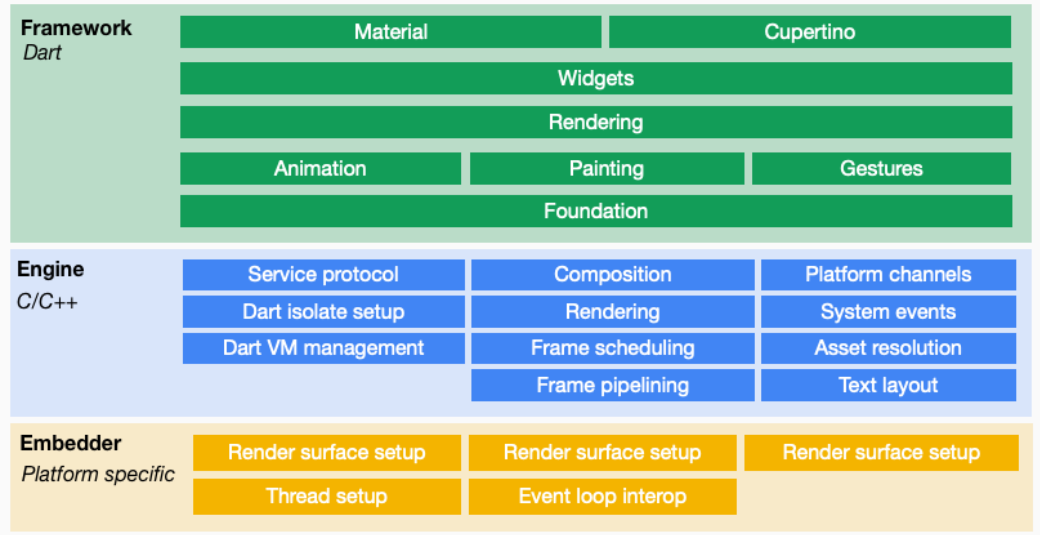


Ilustración 1: Arquitectura de Flutter (Flutter, s.f.)

Una de las características más importantes de este framework, es que se puede notar al tiempo de la codificación, los cambios en la aplicación que se está desarrollando, esto mediante la implementación de Hot-Reload. Este se considera un factor importante durante el ciclo de desarrollo de software, pues no hay la necesidad de compilar frecuentemente una aplicación evitando así perder tiempo durante este proceso (Wu, 2018).

En resumen, Flutter es un framework gratuito para el desarrollo de aplicaciones móviles multiplataforma, es además open source con soporte, documentación y mantenimiento de Google y de la comunidad en general. Actualmente esta herramienta es utilizada por organizaciones en todo el mundo para aplicaciones de producción, debido a su rapidez y a su compilación nativa que permite ejecutar aplicaciones sin tener que usar vistas web o conectores de JavaScript (Payne, 2019).

3.2. NativeScript

Lanzado por la empresa estadunidense de software Progress en 2015, NativeScript es un framework de código abierto para el desarrollo de aplicaciones móviles. Con el uso de esta herramienta, se puede tener acceso a cualquier plataforma (Android o iOS) de manera nativa

utilizando especialmente JavaScript, TypeScript y Angular. Además, este marco de trabajo está diseñado para lograr una reutilización de código no solamente entre plataformas móviles, sino también a nivel web (Progress, s.f.).

NativeScript proporciona al usuario una buena experiencia con el uso de sus aplicaciones, esto lo hace gracias a que no utiliza WebViews como es el caso de algunas otras herramientas. Los módulos multiplataforma que utilizan, le brindan la posibilidad de escribir aplicaciones para iOS y Android desde una única base de código JavaScript y luego montarlas hacia cualquier plataforma (GitHub, s.f.).

La arquitectura de este framework está construida sobre algunas partes fundamentales que son: Runtimes, Módulos Principales, CLI (Command-Line Interface) y Plugins.

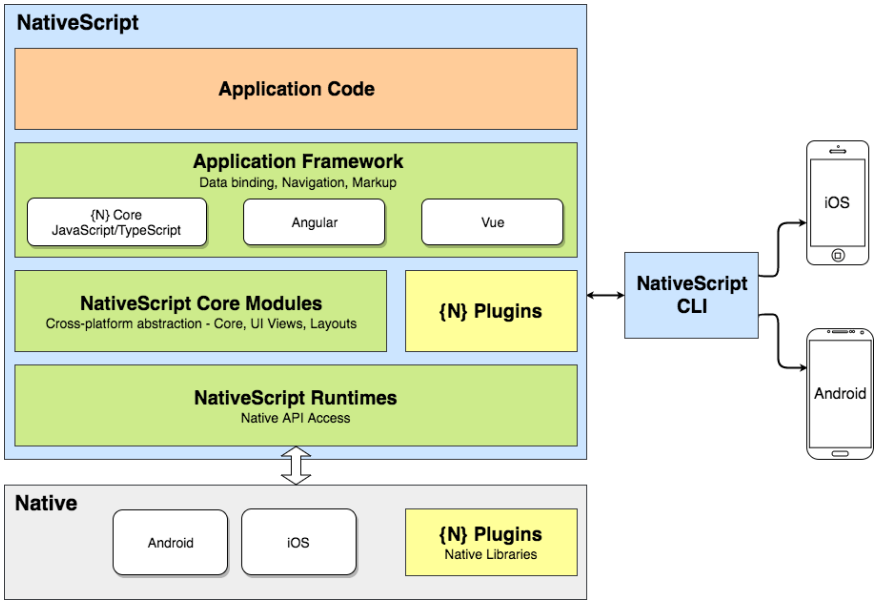


Ilustración 2: Arquitectura de NativeScript (NativeScript, s.f.)

Uno de los elementos más importantes de este framework son sus Plugins, estos son bloques de construcción que encapsula funcionalidades y que proporciona ayuda a los desarrolladores a crear aplicaciones de manera más rápida. La mayoría son publicadas por la comunidad y codificadas en JavaScript o TypeScript (NativeScript, s.f.).

3.3. React Native

React Native es un framework de código abierto para el desarrollo de aplicaciones móviles multiplataforma. Fue desarrollado por Facebook en 2015 e inicialmente presentado como una opción para el desarrollo de aplicaciones en iOS. Posteriormente y en gran parte gracias a los aportes que realizó la comunidad, esta herramienta agregó soporte para Android permitiendo el desarrollo de aplicaciones para esta plataforma (Kuitunen, 2019).

Este framework se basó originalmente en una librería JavaScript de Facebook llamada React, la cual es utilizada para la construcción de interfaces web. Posteriormente la empresa creó un nuevo marco de trabajo llamado React Native, el cual permite el desarrollo aplicaciones para la mayoría de los dispositivos móviles. Ambas tecnologías, es decir, tanto React como React Native para sus desarrollos utilizan una combinación de JavaScript y XML conocida como JSX (Avdic, 2019).

Esta herramienta permite crear aplicaciones nativas sin comprometer la experiencia de los usuarios. Al asignar de forma directa a los bloques de UI nativos, un conjunto básico de componentes como vistas, textos e imágenes, se puede lograr una mayor fluidez y una mejor experiencia en cuanto a la interfaz con la que interactúa el usuario y la aplicación en general (Facebook Inc., s.f.).

Una de las mayores ventajas de este framework es que el procesamiento de sus aplicaciones son realizadas bajo la invocación de API´s nativas, en Objective-C para iOS y JAVA para Android. De esta forma la aplicación móvil utiliza componentes de interfaz reales y propios de cada plataforma, en lugar de hacer uso de elementos web que trate de imitar alguna funcionalidad (Avdic, 2019).

La arquitectura de este framework consta de una máquina virtual JavaScript, un puente React Native y los módulos nativos de cada plataforma. El código escrito se ejecuta en la máquina virtual JavaScript junto con las bibliotecas de terceros. Las llamadas del módulo nativo se enrutan a través del puente de React Native a las API nativas y a las librerías utilizadas de terceros. Este tipo de arquitectura permite el uso de JavaScript para desarrollar una aplicación sin dejar de utilizar los componentes y características nativas de la interfaz de cada plataforma (Kuitunen, 2019).

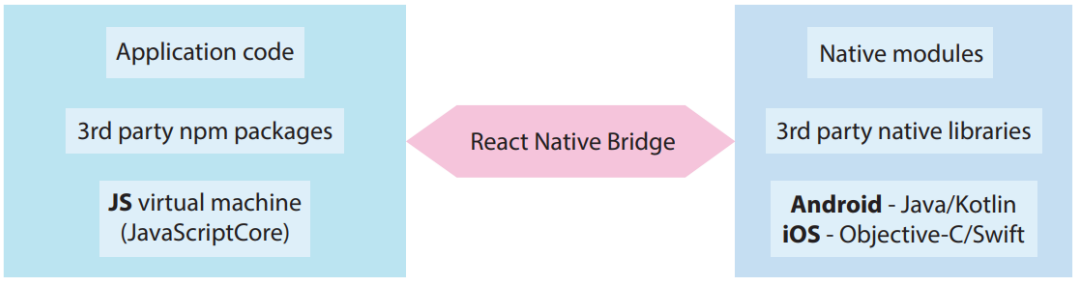


Ilustración 3: Arquitectura de React Native (Kuitunen, 2019)

3.4. Xamarin

Xamarin es un framework de código abierto para el desarrollo de aplicaciones móviles y de alto rendimiento. Inició en 2001 como un proyecto de la compañía que llevaba el mismo nombre que plataforma, conocido entonces como Proyecto Mono. Dicha empresa fue adquirida posteriormente por Microsoft en 2016 y desde entonces es propietaria de la plataforma y es la encargada junto con la comunidad de dar soporte (Microsoft, s.f.).

Esta plataforma permite el desarrollo de aplicaciones móviles mediante el uso del lenguaje diseñado por Microsoft C#. Xamarin hace uso de las ventajas que ofrece .NET para que el desarrollo de una aplicación sea más sencillo y el programador no tenga que preocuparse por detalles. Esta característica permite que la programación sea considerablemente veloz e incluso se puede lanzar versiones de prueba al mismo tiempo en Android, iOS y aunque no tenga mucha aceptación, Windows Phone (Avdic, 2019).

Con Xamarin toda aplicación que se esté construyendo puede ser escrita utilizando C#, desde la base del código, hasta el acceso a datos pasando incluso por la lógica del negocio. Debido a que hace uso también de .NET, se puede utilizar recursos y librerías propias de la plataforma que facilita el desarrollo. Además, mediante la compilación AOT con la que funciona, se reduce el tiempo de inicio, se optimiza el uso de la memoria y se obtiene un alto rendimiento (Microsoft, s.f.).

Si bien Xamarin permite tener una base de código en C#, su arquitectura demanda crear una interfaz de usuario para cada plataforma. Así mismo, la ejecución es diferente para cada sistema operativo, aunque su modelo es basado en Mono, este es un entorno de ejecución

propio de Xamarin que controla de forma automática algunas tareas como asignación de memoria, la recolección de elementos no utilizados y la interoperabilidad con las plataformas (Microsoft, s.f.).

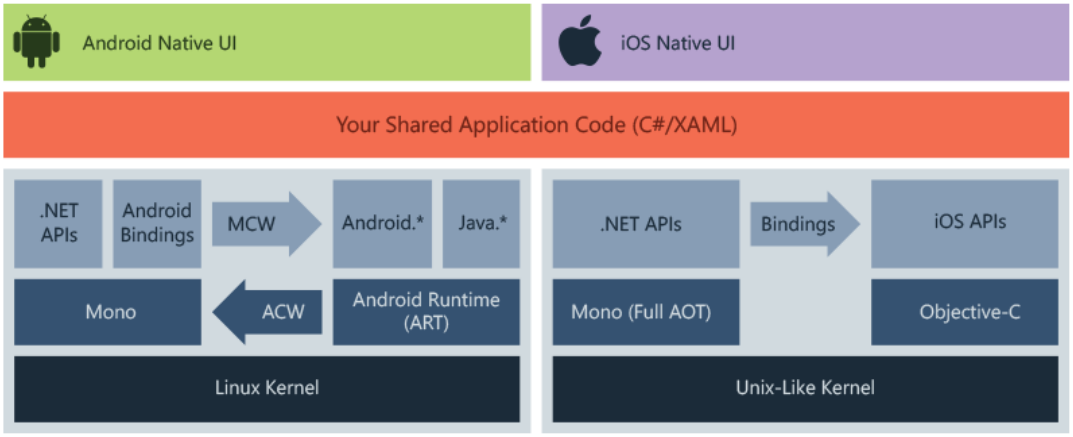


Ilustración 4: Arquitectura de Xamarin (Microsoft, s.f.)

Aunque Xamarin está basado en la idea del desarrollo multiplataforma en la que se comparte la mayor parte del código en la lógica del negocio, sus interfaces deben ser programadas de manera independiente. Todo esto concluye en que, al hacer uso de esta herramienta, en verdad se está compartiendo un promedio de 85% de código, cifra que se puede considerar alta tomando en cuenta lo que implica hacer desarrollos completamente nativos (Delia, Galdamez, Thomas, Corbalan, & Pesado, 2014).

4. Comparación de Frameworks

4.1. Instalación

La instalación de los cuatro frameworks mencionados durante el desarrollo del presente trabajo, es importante mencionar que se lo hará en una computadora con las siguientes características:

EQUIPO	
Marca	Razer
Modelo	Blade Stealth
SO	Windows 10 Pro
Almacenamiento	512 GB SSD
RAM	16 GB
GPU	Intel HD Graphics 620

Tabla 1: Características del equipo a utilizar en el presente trabajo (Zurita, 2020)

PROCESADOR	
Nombre	Intel Core i7-7500
Arquitectura	64 bits
Núcleos	2 núcleos
Hilos	4 hilos
Frecuencia base	2.70 GHz
Frecuencia turbo	2.90 GHz

Tabla 2: Características del procesador del equipo a utilizar en el presente trabajo (Zurita, 2020)

Flutter

Para la instalación y ejecución del este framework, el equipo de trabajo debe cumplir con los siguientes requerimientos (Flutter, s.f.):

- **Sistema Operativo**: Windows 7 (64 bits) en adelante.
- **Espacio en Disco:** 400 MB (sin incluir el espacio para el IDE).
- **Herramientas:** Git y Windows PowerShell 5.0 en adelante.

Procedimiento

1. Descargarse la última versión estable del SDK disponible en la página oficial de Flutter. Para la fecha de realización, la última versión estable es la 1.17.0.

2. Una vez obtenido el archivo zip, se recomienda ubicar el contenido en una dirección diferente a la carpeta que contiene los archivos de programas del disco C. Esto es por motivos de permisos que se requiere como administrador. Aunque la guía oficial menciona que, desde aquí ya se puede hacer uso de la herramienta, hay que hacer algunas configuraciones previas en el equipo de manera que todo marche de forma correcta.

3. Actualizar el path:
    - Abrir el editor de variables de entorno del sistema.

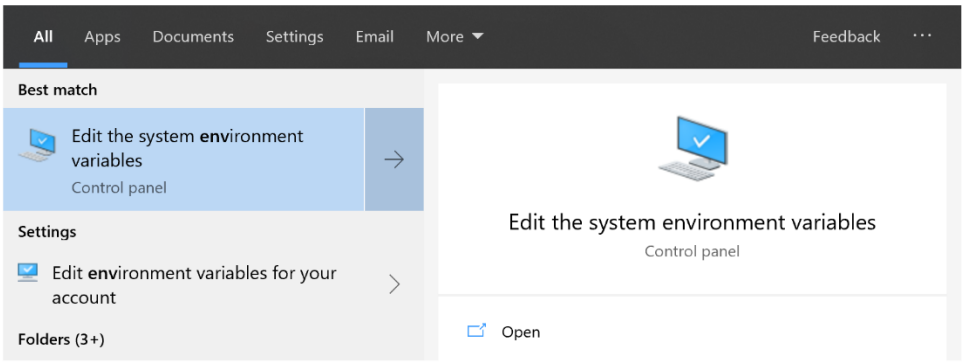


Ilustración 5: Instalación - Flutter (Zurita, 2020)

    - Agregar Flutter al path.

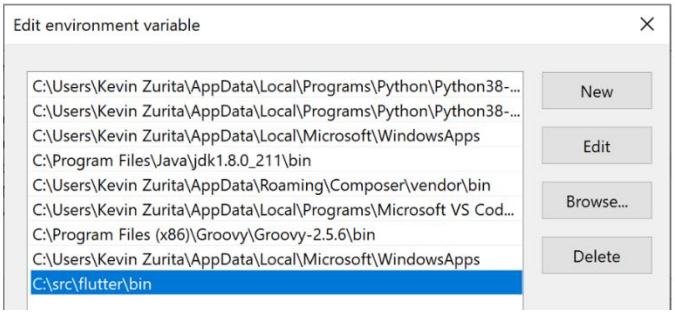


Ilustración 6: Instalación - Flutter (Zurita, 2020)

- Una vez realizado el paso anterior, se puede comprobar que se instaló correctamente mediante el comando: `flutter --version`

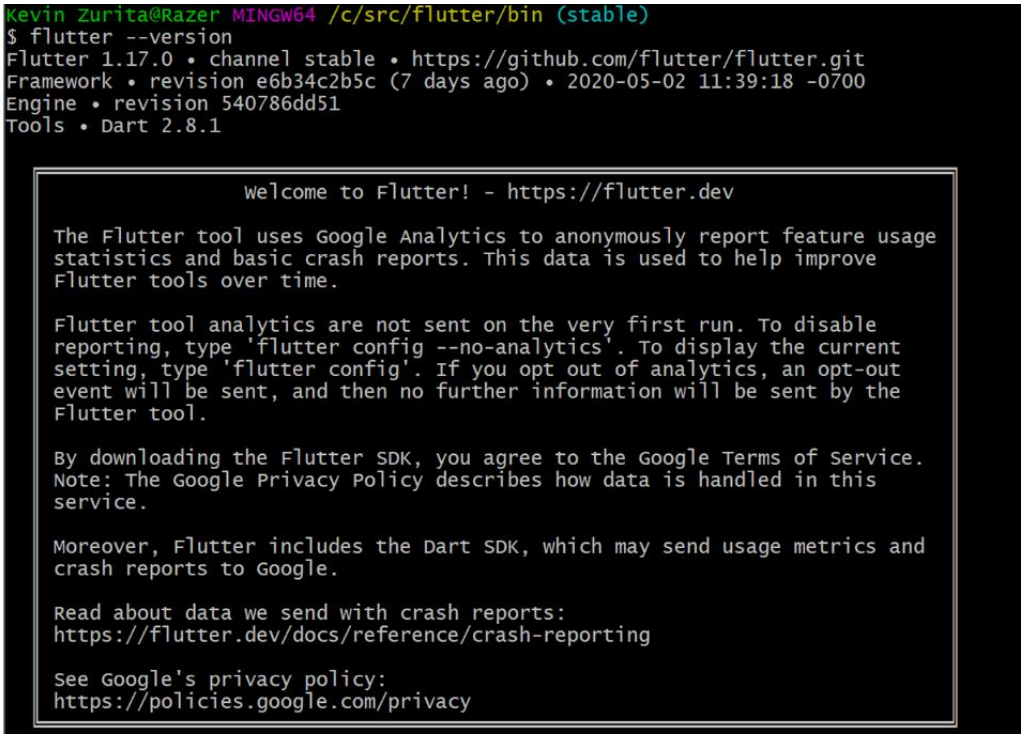


Ilustración 7: Instalación - Flutter (Zurita, 2020)

4. Finalmente, para poder hacer uso de la herramienta, es necesario contar con un IDE que soporte el desarrollo con este framework. La guía oficial recomienda VS Code o Android Studio.

El IDE escogido para el desarrollo del trabajo fue Android Studio, el mismo que una vez instalado requiere incluir un plugin propio de Flutter disponible en el Market Place de la herramienta.

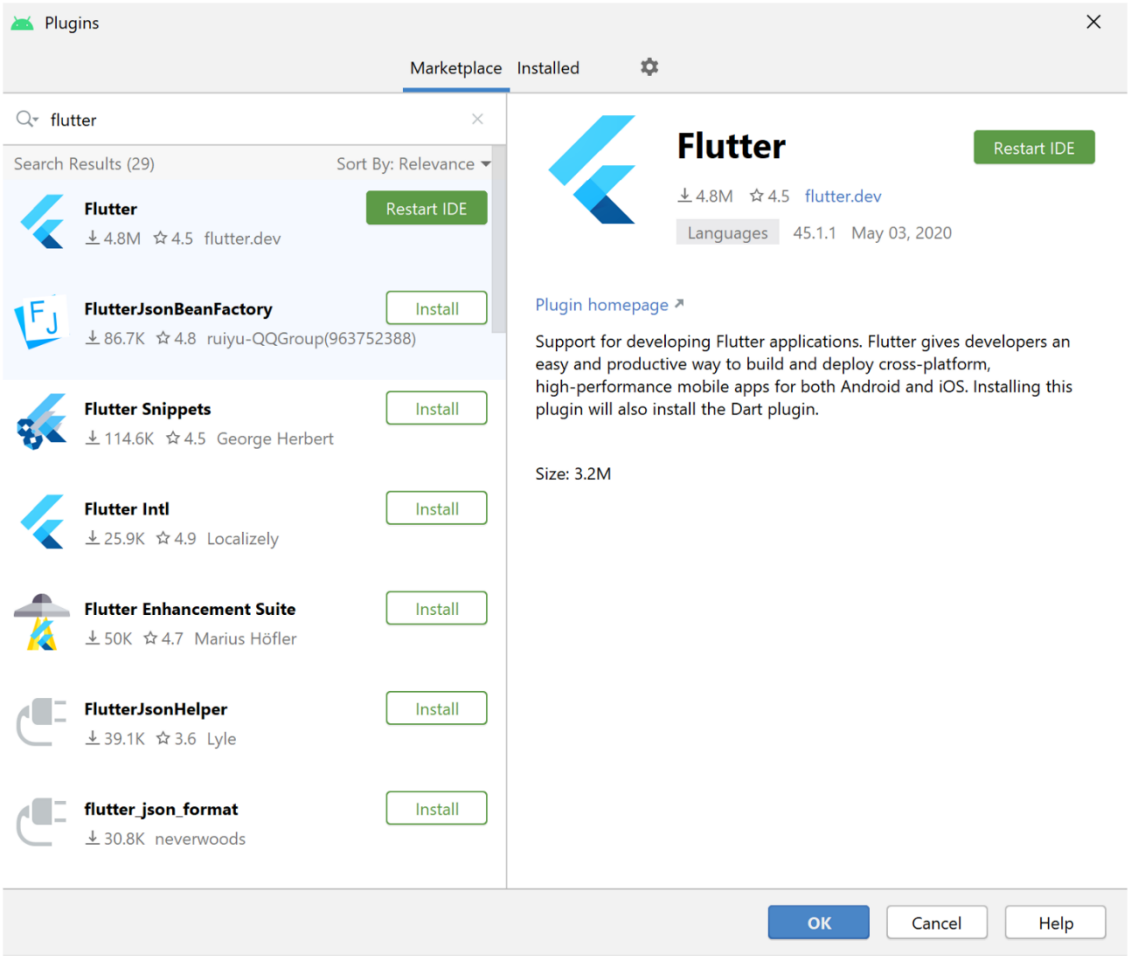


Ilustración 8: Instalación - Flutter (Zurita, 2020)

Una vez instalado este recurso, ya se puede hacer uso del framework creando nuevas aplicaciones mediante Android Studio.

NativeScript

Para la instalación del este framework, la guía oficial de NativeScript menciona que el equipo donde se vaya a trabajar debe cumplir con algunos requerimientos mínimos que son (NativeScript, s.f.):

- **Sistema Operativo:** Windows 7 en adelante (preferible Windows 10).
- **Herramientas:** SDK de Android 6.0 en adelante con un API 23 o superior (si se desea trabajar con un emulador).

Procedimiento

1. Instalar Node.js, este recurso se lo encuentra disponible en sitio web oficial *nodejs.org*. Para la fecha de realización del presente trabajo, la última versión disponible para los usuarios es la 12.16.3. LTS.

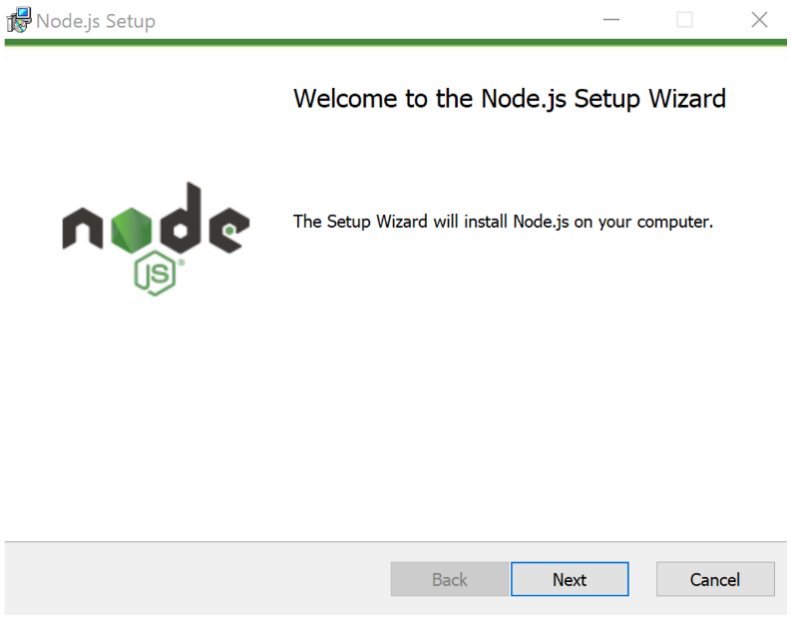


Ilustración 9: Instalación - NativeScript (Zurita, 2020)

2. Una vez instalado el recurso anterior, en un intérprete de comandos se puede continuar con la instalación del CLI de NativeScript mediante: `npm install -g nativescript.`

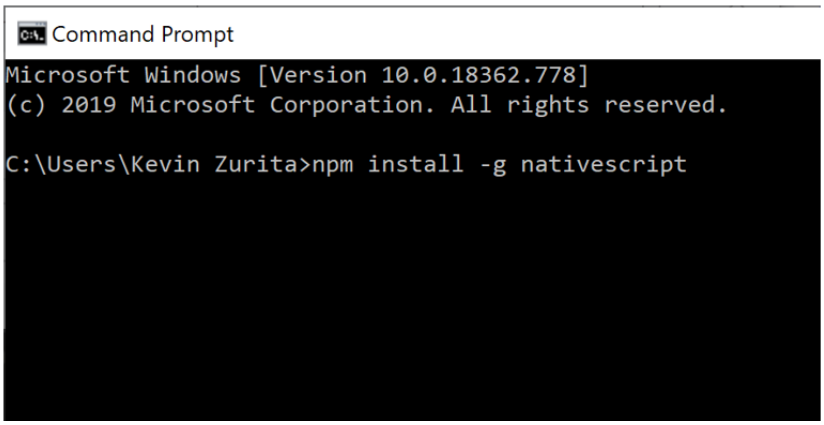


Ilustración 10: Instalación - NativeScript (Zurita, 2020)

Esto instalará una serie de recursos y dependencias que son necesarias para el funcionamiento del framework.

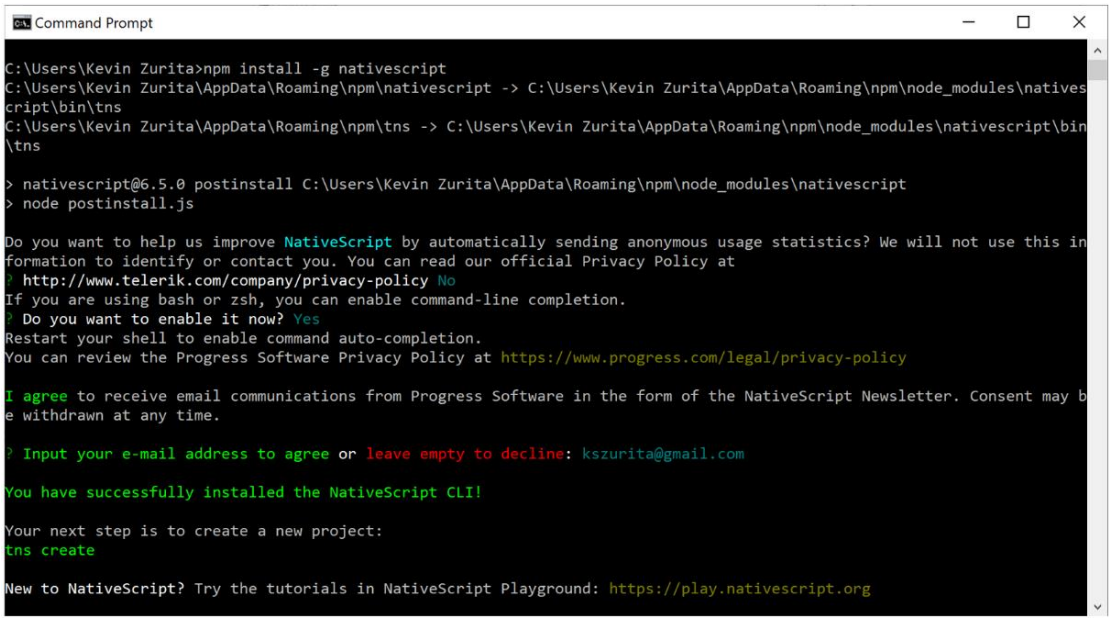


Ilustración 11: Instalación - NativeScript (Zurita, 2020)

3. Una vez completado el paso anterior, se puede iniciar con la creación de nuevos proyectos, para esto se necesita un IDE que cuente son soporte JavaScript.

Finalmente, es importante mencionar que para poder observar los cambios que se realicen en las aplicaciones que se están desarrollando, se debe hacer uso de los emuladores que soporte el sistema operativo que se está utilizado. Aunque también se lo puede hacer mediante las aplicaciones propias de NativeScript.

Estas aplicaciones son diseñadas para ejecutar proyectos de forma directa desde cualquier dispositivo móvil, sin importar el sistema operativo que se utilice. Están son “Preview” y “Playground” ambas están disponibles tanto de la App Store como en Google Play y funcionan de manera complementaria.

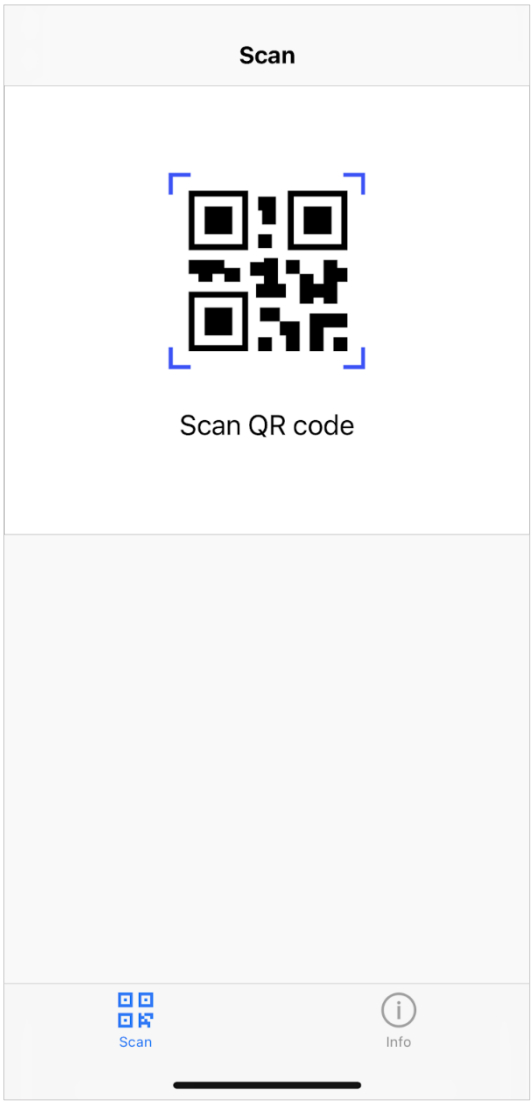


Ilustración 12: Aplicación Playground - NativeScript (Zurita, 2020)

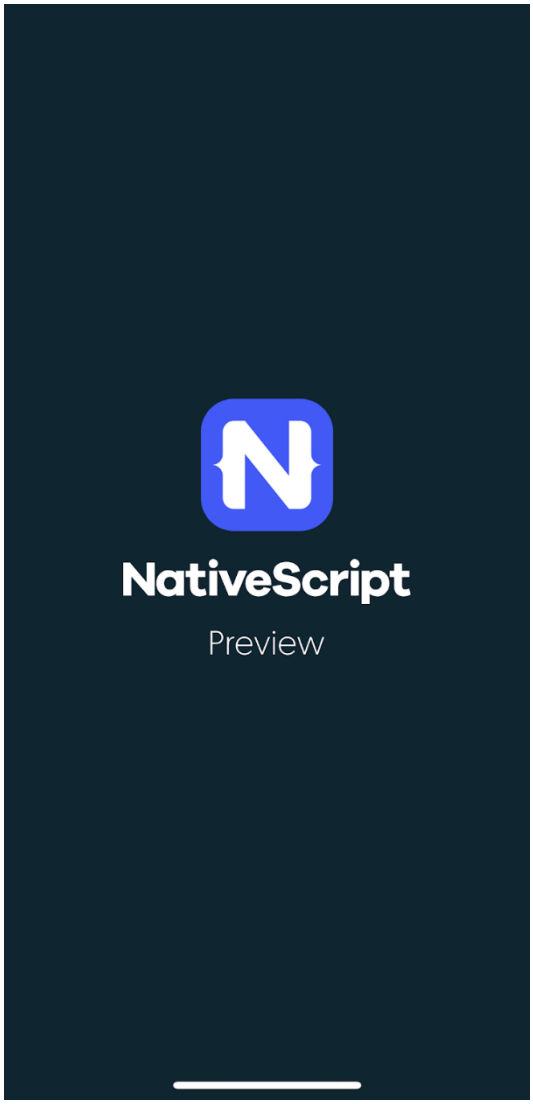


Ilustración 13: Aplicación Preview - NativeScript (Zurita, 2020)

React Native

Al igual que NativeScript, este framework no requiere de mayores requerimientos en el equipo en el que se va a trabajar, antes de iniciar con la instalación, la guía recomienda instalar previamente Node.js.

Procedimiento

1. Previo a la instalación, hay que verificar si en el equipo que se va a trabajar se cuenta instalado Node.js, para esto hacemos uso de un intérprete de comandos con la siguiente instrucción: `node -v`.

   De la misma forma, si quiere verificar la versión del sistema gestor de paquetes de Node.js se utiliza el comando `npm -v`.

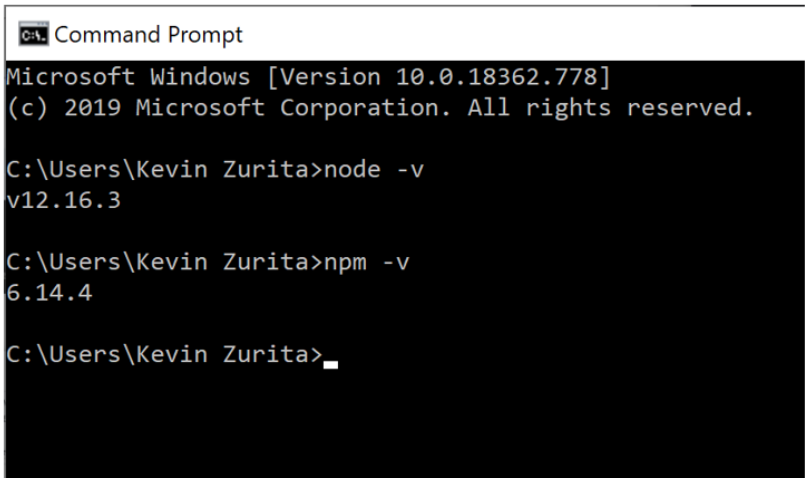


Ilustración 14: Instalación - React Native (Zurita, 2020)

2. Una vez se haya cumplido con el paso anterior, se procede con la instalación del CLI de Expo mediante el comando `npm install expo-cli –global`. Esta sentencia se la ejecuta una vez que se haya ubicado la carpeta en donde vamos a trabajar.

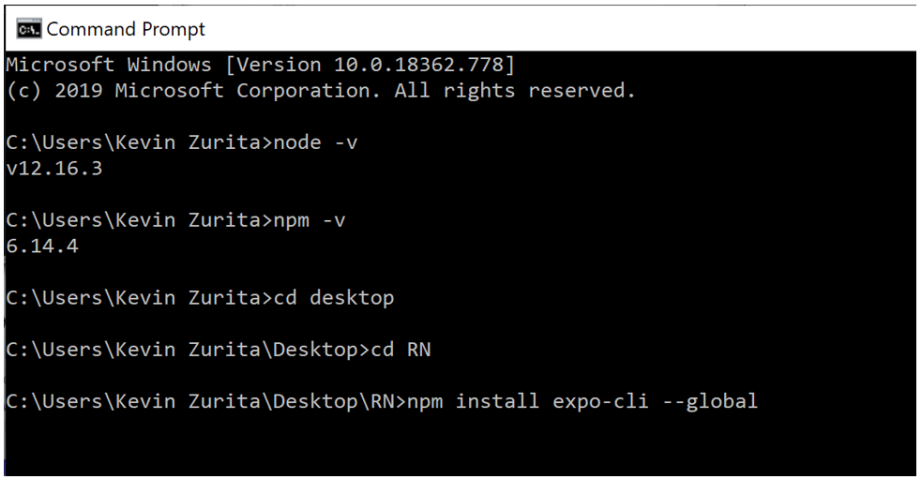


Ilustración 15: Instalación - React Native (Zurita, 2020)

Se instalará varios paquetes necesarios para construir aplicaciones en React Native de manera correcta.

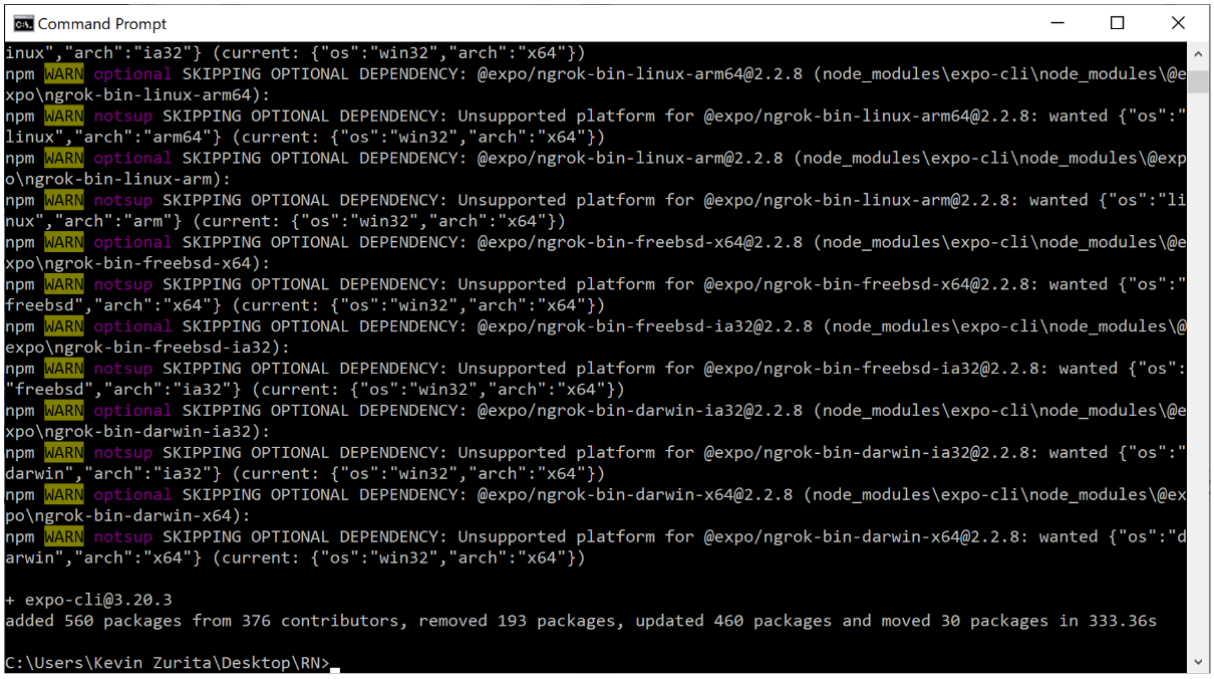


Ilustración 16: Instalación - React Native (Zurita, 2020)

Para poder ejecutar aplicaciones React Native en cualquier dispositivo, es necesario contar con la aplicación “Expo”, la misma está disponible tanto en la App Store como en Google Play. A

diferencia de las aplicaciones de NativeScript, “Expo” es una única aplicación que requiere de un lector de códigos QR para Android y la aplicación de cámara en iOS.

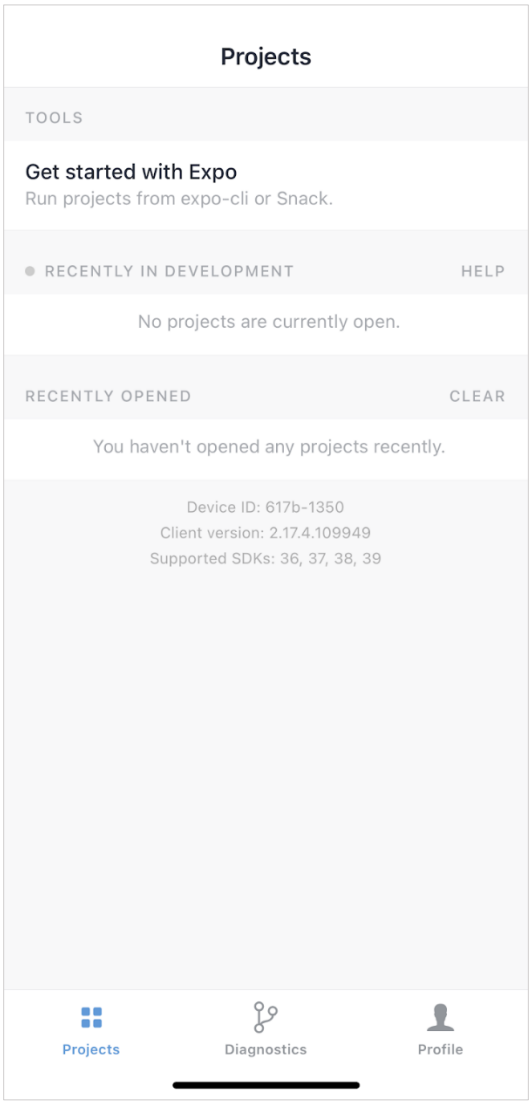


Ilustración 17: Aplicación Expo - React Native (Zurita, 2020)

Para iniciar un proyecto con React Native y poder visualizarlo con Expo es necesario abrir un intérprete de comandos, ubicarse en la dirección en donde se encuentra la carpeta del proyecto y ejecutar el comando: `expo start NombredelProeycto`

Ilustración 18: Ejemplo de código QR - React Native (Zurita, 2020)

Xamarin

Para la instalación de este framework, la guía oficial menciona que no existen prerrequisitos para hacer uso de esta herramienta (Microsoft, s.f.).

Procedimiento

1. Descargar e instalar Visual Studio, asegurándose que en las opciones de descarga este seleccionado la opción de *Mobile developmente with .NET*.

   Si ya se tiene instalado una versión de Visual Studio, se puede verificar y si es caso instalar la opción que permite el desarrollo de aplicaciones con Xamarin.

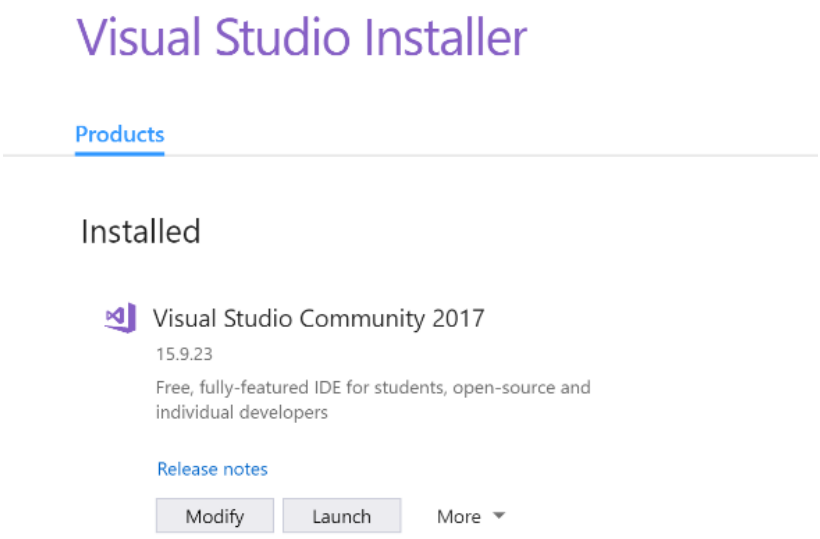


Ilustración 19: Instalación - Xamarin (Zurita, 2020)

   En la opción de *Modify*, dentro de la sección de *Mobile & Gaming*, se puede comprobar que se tenga instalado la opción de Xamarin.

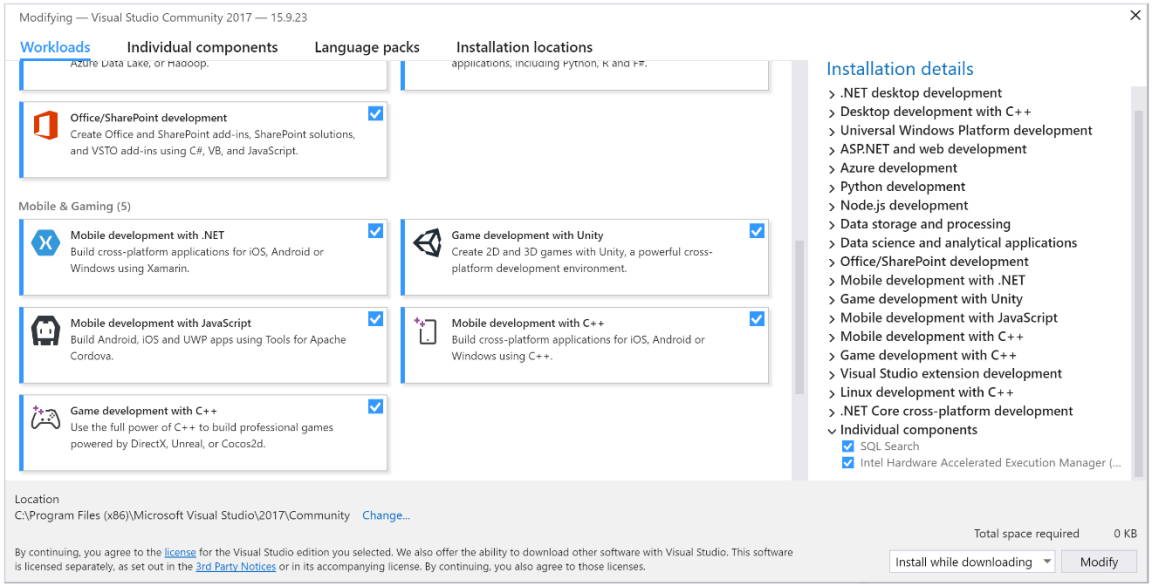


Ilustración 20: Instalación - Xamarin (Zurita, 2020)

2. Para finalizar, en la opción de *Individual components*, específicamente dentro de la sección de *Development activities* y en la sección de *Emulators*, hay que verificar que se cuente instalado todas las opciones de Xamarin y algún emulador de Android que permita la ejecución de aplicaciones.

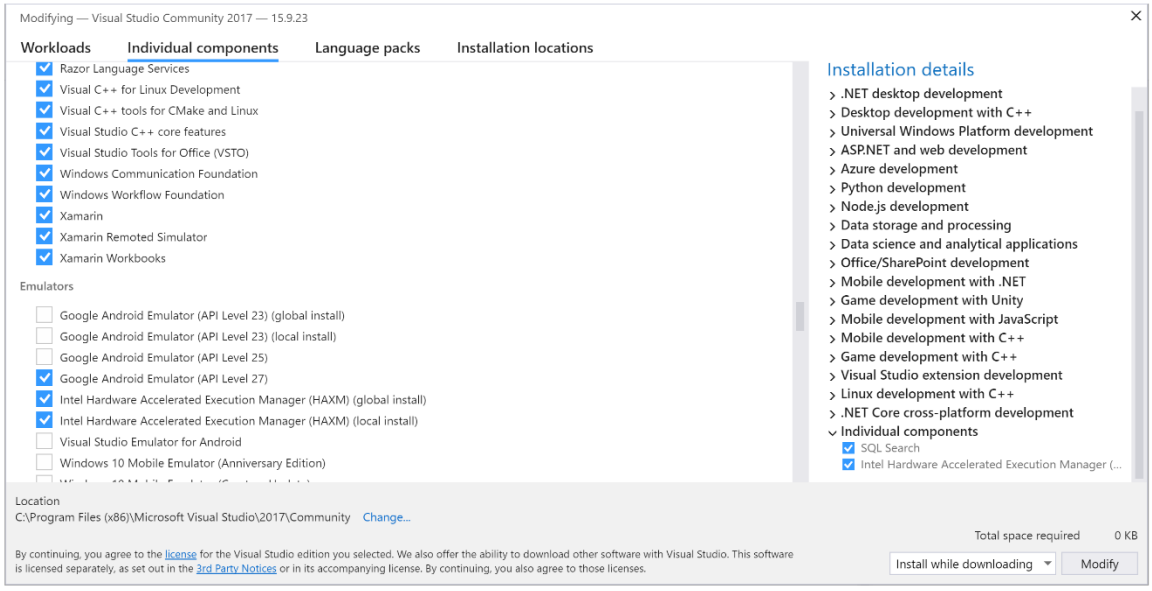


Ilustración 21: Instalación - Xamarin (Zurita, 2020)

4.2. Lenguaje

Flutter - Dart	NativeScript - TypeScript
Presentado de manera oficial en Octubre de 2011, Dart en un lenguaje de programación de código abierto desarrollado por Google y aprobado de manera formal por ECMA (European Computer Manufacturers Association). Según sus creadores, este lenguaje no está pensado para sustituir a JavaScript en los desarrollos web, al contrario, puede ser considerado como una alternativa fresca y moderna. Aunque fue concebido como un lenguaje orientado para la web, actualmente con este se pueden construir aplicaciones de escritorio, de servidor e incluso para dispositivos móviles mediante Flutter (Google, s.f.).	Desarrollado por Microsoft y publicado en Octubre del 2012, TypeScript es un lenguaje de código abierto basado en JavaScript. Fue concebido con la idea satisfacer las necesidades que JavaScript no suplía en proyectos de gran escala (Microsoft, s.f.). TypeScript es una extensión de JavaScript que permite escribir código mejor estructurado de una manera sencilla, dentro de un sistema de módulos, clases e interfaces. Todo programa de TypeScript antes de ser ejecutado, es traducido a JavaScript pues es el lenguaje que entiende la mayoría de los navegadores (Bierman, Abadi, & Torgersen, 2014).
React Native - JavaScript	**Xamarin - C#**
JavaScript es un lenguaje de programación que apareció como tal en 1995, cuando el lenguaje anterior a este (LiveScipt) agregó compatibilidad con la tecnología Java de la época. Netscape que fue la empresa desarrolladora fue adquirida por la fundación Mozilla, quienes tomaron los mandos del negocio hasta el momento (Ayoze, 2017). Técnicamente JavaScript es un lenguaje interpretado y orientado a la web, que surgió como respuesta a una combinación entre aplicaciones cada vez más complejas y velocidades lentas de navegación. El resultado de crear e implementar este lenguaje fue la reducción considerable de tiempo de respuesta de aplicaciones del momento (Pérez, 2019).	C# es un lenguaje de programación orientado a objetos creado por Microsoft dentro del desarrollo de su plataforma .NET. Tiene raíces en la familia de lenguajes C y puede ser de fácil interpretación para los programadores que tienen conocimiento de C, C++, Java e incluso JavaScript (Microsoft, s.f.). Este es un lenguaje seguro y orientado completamente a la programación orientada a objetos. Generalmente es utilizado para desarrollar aplicaciones de escritorio, aplicaciones web basadas en ASP.NET y en la actualidad se lo utiliza también para la creación de aplicaciones móviles compatibles con Android, iOS y Windows Phone (Dimes, 2016).

Tabla 3: Comparación de lenguajes

4.3. Compatibilidad

<table>
<tr><td colspan="2">Android</td></tr>
<tr><td colspan="2">Desarrollado en un principio por la empresa Android Inc. y adquirido en 2005 por Google, Android es un sistema operativo basado en Linux y de código abierto. Fue desarrollado en un principio para teléfonos inteligentes que cuenten con pantalla táctil, aunque en la actualidad es compatible con tabletas, relojes inteligentes, televisores e incluso automóviles. Es tanta la popularidad que este sistema operativo ha alcanzado, que se estima que existen alrededor de 2500 millones de dispositivos activos alrededor del mundo que lo utilizan (Android, s.f.).</td></tr>
<tr><td>Compatible con:</td><td>Flutter, NativeScript, React Native y Xamarin</td></tr>
<tr><td colspan="2">iOS</td></tr>
<tr><td colspan="2">Presentado junto a la primera generación del iPhone en 2007, iOS es un sistema operativo exclusivo de la empresa Apple para sus dispositivos inteligentes. Este sistema fue el primero en adaptar una tienda de aplicaciones móviles, un hecho controvertido para la época que transformó la industria para siempre. Aunque fue concebido para ser utilizado en sus teléfonos móviles, posteriormente iOS se adaptó al iPod, al iPad, e incluso a su gama de relojes inteligentes. En la actualidad se calcula que el número de dispositivos Apple que utilizan este sistema operativo o alguno de sus derivados, está cerca de los 1500 millones (Rodriguez, 2019)</td></tr>
<tr><td>Compatible con:</td><td>Flutter, NativeScript, React Native y Xamarin</td></tr>
<tr><td colspan="2">Windows Phone</td></tr>
<tr><td colspan="2">Presentado en 2010 como una versión mejorada de su predecesor Windows Mobile, Windows Phone es un sistema operativo desarrollado por Microsoft para teléfonos móviles y que competía directamente con iOS y Android. Aunque desde 2015 este sistema operativo se encuentra descontinuado, tuvo cierto momento de popularidad. Según Microsoft llegaron a existir 385.000 aplicaciones en su tienda Windows Phone Store y ese número aumenta cada mes. Con el pasar del tiempo y las mejores opciones que ofrecía el mercado, los desarrolladores perdieron interés y debido a la poca demanda de la plataforma, la empresa decidió descontinuar sus operaciones (Constantin, Gordan, Mihaela, & Berczes, 2017).</td></tr>
<tr><td>Compatible con:</td><td>Xamarin</td></tr>
</table>

Tabla 4: Comparación de compatibilidad

4.4. Desarrollo y Codificación – Ejemplos Prácticos

Ejemplo 1

El primer ejemplo es un aplicativo simple en cual el usuario ingresa en un formulario su nombre y su fecha de nacimiento, luego al presionar un botón, el aplicativo muestra a manera de notificación un saludo y el cálculo de la edad que tiene actualmente.

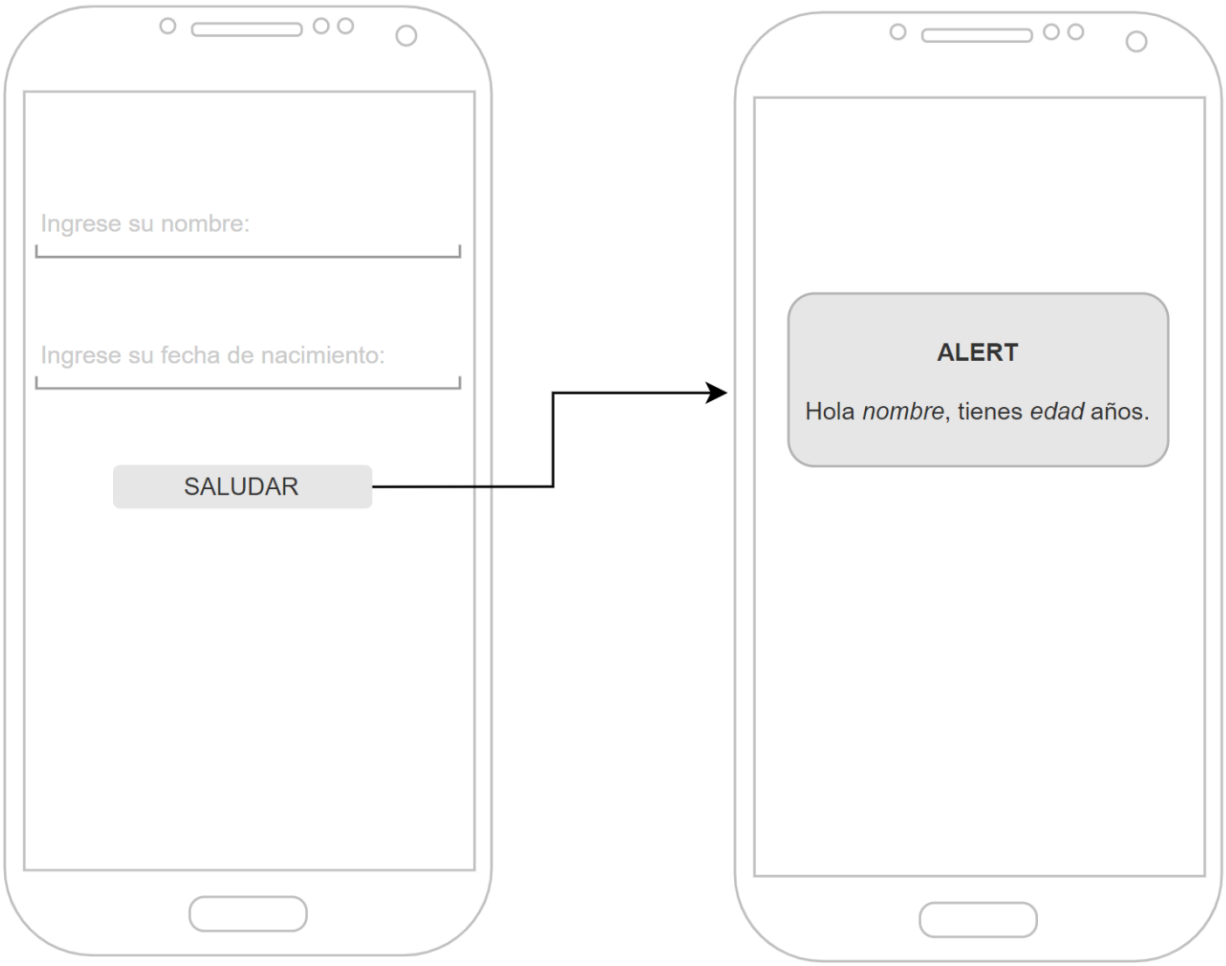


Ilustración 22: Diseño - Ejemplo 1 (Zurita, 2020)

En Flutter

main.dart

```
1. import 'package:flutter/material.dart';
2.
3. void main() {
4.   runApp(MyApp());
5. }
6.
7. class MyApp extends StatelessWidget {
8.   @override
9.   Widget build(BuildContext context) {
10.    return MaterialApp(
11.      theme: ThemeData(
12.        primarySwatch: Colors.blue,
13.        visualDensity: VisualDensity.adaptivePlatformDensity,
14.      ),
15.      home: MyHomePage(),
16.    );
17.  }
18.}
19.
20.class MyHomePage extends StatefulWidget{
21.  @override
22.  MyHomePageState createState() => MyHomePageState();
23.}
24.
25.class MyHomePageState extends State<MyHomePage>{
26.  TextEditingController _controller;
27.
28.  void initState(){
29.    _controller = TextEditingController();
30.    super.initState();
31.  }
32.
33.  @override
```

```
34.  Widget build(BuildContext context){
35.    return Scaffold(
36.      appBar: AppBar(
37.        title: Text('SALUDO'),
38.      ),
39.      body: Padding(
40.        padding: EdgeInsets.all(20),
41.        child: Column(
42.          children: <Widget>[
43.            TextField(
44.              controller: _controller,
45.              decoration: InputDecoration(
46.                filled: true,
47.                labelText: 'Ingrese su nombre',
48.              ),
49.            ),
50.
51.            RaisedButton(
52.              controller: _controller,
53.              child: Text('Ingrese su fecha de nacimiento'),
54.              onPressed:(){
55.                showDatePicker(
56.                    context: context,
57.                    initialDate: DateTime.now(),
58.                    firstDate: DateTime(1970),
59.                    lastDate: DateTime(2100));
60.              }
61.            ),
62.
63.            RaisedButton(
64.              child: Text('Saludar'),
65.              onPressed: () {
66.                _saluda(context, _controller.text, _controller.DateTime);
67.              },
68.            )
69.          ],
70.        )
```

```
71.       ),
72.     );
73.   }
74.
75.   Future _saluda(BuildContext context, String nombre, DateTime fecha){
76.     $edad = DateTime.now().difference($fecha)
77.     return showDialog(
78.         context: context,
79.       builder: (context) => AlertDialog(
80.         title: Text('Alert'),
81.         content: Text('Hola $nombre, tienes $edad años'),
82.       )
83.     );
84.   }
85. }
```

Capturas

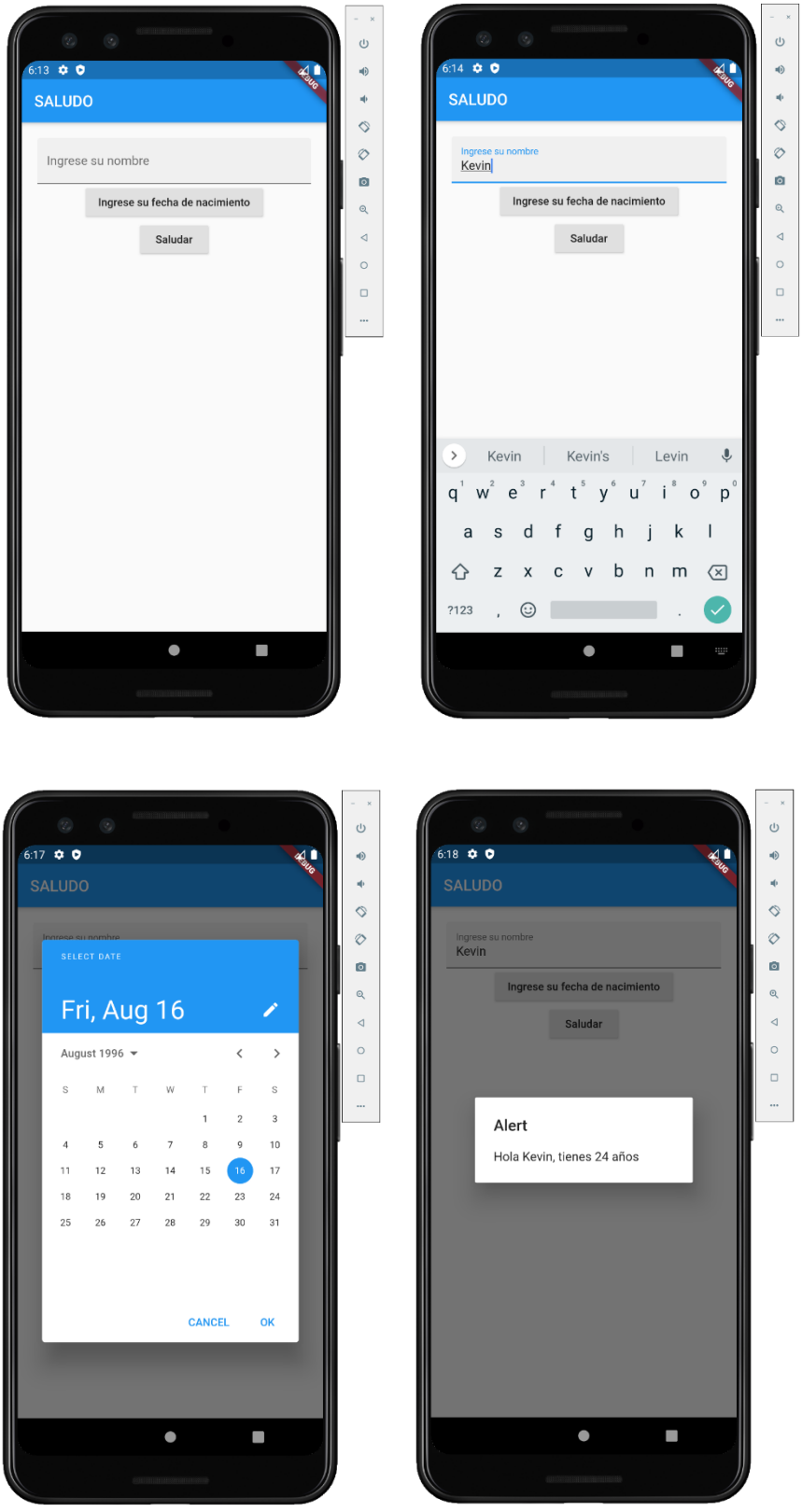


Ilustración 23: Capturas Flutter - Ejemplo 1 (Zurita, 2020)

En NativeScript

home.xml

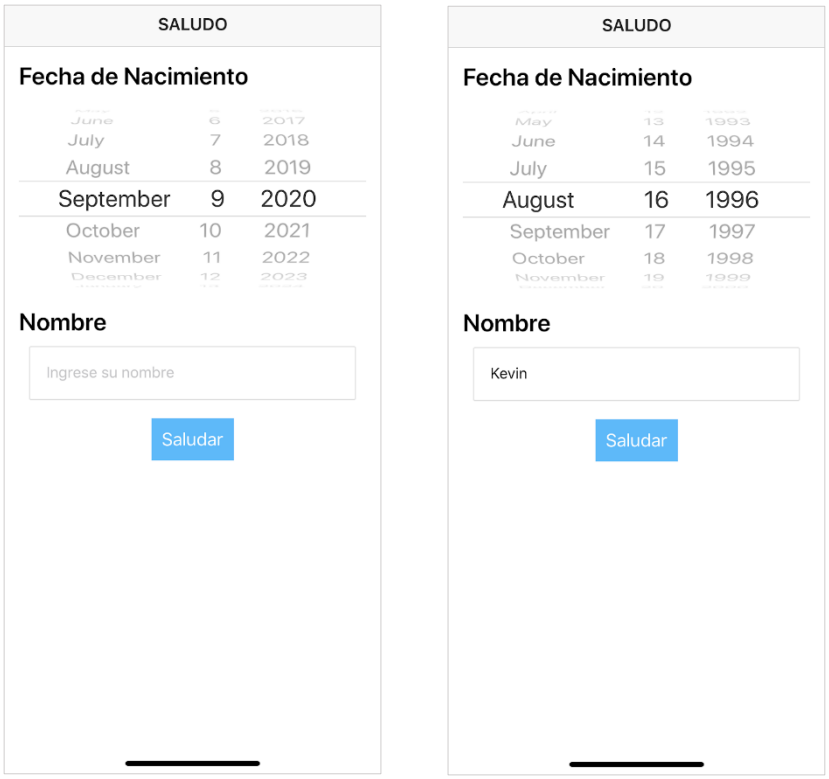
```
<Page class="page">
    <ActionBar title="SALUDO" class="action-bar" />
    <ScrollView>
        <StackLayout class="home-panel form input-field">
            <Label text="Fecha de Nacimiento" class="heading" />
            <DatePicker :year="currentYear" :month="currentMonth"
                :day="currentDay" minDate="1970-01-01"
                maxDate="2100-12-31" />
            <Label text="Nombre" class="heading" />
            <TextField v-model="str" hint="Ingrese su nombre"
                class="input input-border" />
            <StackLayout orientation="horizontal"
                horizontalAlignment="center">
                <Button text="Saludar" @tap="sayhello"
                    class="btn btn-primary" />
            </StackLayout>
        </StackLayout>
    </ScrollView>
</Page>
```

home-view-model.js

```
const appSettings = require("tns-core-modules/application-settings");
const dialogs = require("tns-core-modules/ui/dialogs");

    export default {
        data() {
            return {
                currentDay: new Date().getUTCDate(),
                currentMonth: new Date().getUTCMonth() + 1,
                currentYear: new Date().getUTCFullYear()
            };
        },
```

```
12.
13.         created() {
14.             this.num = appSettings.getNumber("someNumber", null);
15.             this.dte = appSettings.getString("someDate", "");
16.         },
17.
18.         methods: {
19.             calculateAge(dte){
20.                 var dif = Date.now() - dte.getUTCDate();
21.                 var age = new Date(dif);
22.                 return (age.getUTCFullYear());
23.             }
24.             sayhello() {
25.                 appSettings.setString("someString", this.str);
26.                 appSettings.setDate("someDate", this.date);
27.                 dialogs.alert(
28.                     "Hola " +
29.                     appSettings.getString("someString") +
30.                     " tienes" +
31.                     calculateAge("someDate") +
32.                     " años"
33.                 );
34.             }
35.         }
36.     };
```

Capturas



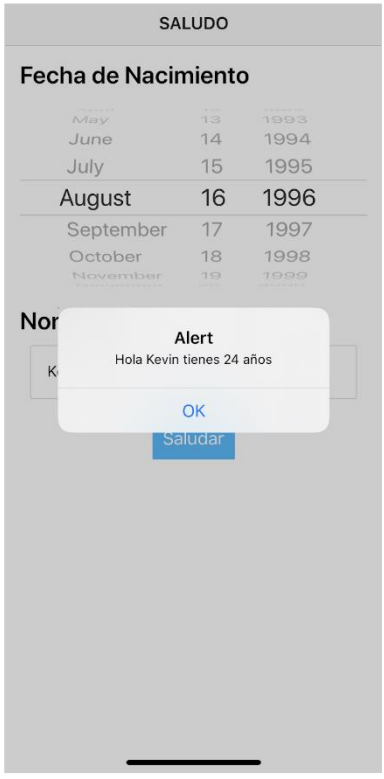


Ilustración 24: Capturas NativeScript - Ejemplo 1 (Zurita, 2020)

En React Native

App.js

```
1. import { StatusBar } from 'expo-status-bar';
2. import React, {useState} from 'react';
3. import {StyleSheet, Text, View, Button, Alert, TextInput, DatePickerIOS } from
   'react-native';
4.
5. export default function App() {
6.
7.   const Separator = () => (
8.     <View style={styles.separator} />
9.   );
10.
11.  const [chosenDate, setChosenDate] = useState(new Date());
12.
13.  return (
14.    <View style={styles.container}>
15.
16.      <Text style={styles.baseText}>
17.      SALUDO
18.      </Text>
19.
20.      <Separator/>
21.
22.      <Text style={styles.labelText}>
23.      Ingrese su nombre:
24.      </Text>
25.
26.      <TextInput
27.      ref={ref => {this.txtinput = ref;}}
28.      style={styles.inputText}
29.      />
30.
31.      <Separator />
32.
```

```
33.       <Text style={styles.labelText}>
34.       Ingrese su fecha de nacimiento:
35.       </Text>
36.
37.       <DatePickerIOS
38.       date={chosenDate}
39.       onDateChange={setChosenDate}
40.       mode = {"date"}
41.       ref={ref => {this.dteinput = ref;}}
42.       />
43.
44.       <Separator/>
45.
46.       <Button
47.       title="SALUDAR"
48.       onPress = {() => Alert.alert('Hola ' + this.txtinput + ' tienes '+
    getAge(this.dteinput) +' años.')}
49.       />
50.       <StatusBar style="auto" />
51.
52.     </View>
53.   );
54.
55.   function getAge(date_str)
56.   {
57.     var today = new Date();
58.     var bdate = new Date(date_str);
59.     var age = today.getFullYear() - bdate.getFullYear();
60.     return age;
61.   }
62.
63. }
64.
65. const styles = StyleSheet.create({
66.
67.   container: {
68.     flex: 1,
```

```
69.     backgroundColor: '#fff',
70.     justifyContent: 'center',
71.   },
72.
73.   baseText: {
74.     fontWeight: 'bold',
75.     fontSize: 25,
76.     paddingBottom: 30,
77.     textAlign:'center',
78.   },
79.
80.   inputText: {
81.     height: 40,
82.     width: 375,
83.     borderColor: 'gray',
84.     borderWidth: 2,
85.
86.   },
87.
88.   labelText: {
89.     fontSize: 15,
90.     paddingBottom: 10,
91.   },
92.
93.   separator: {
94.     marginVertical: 20,
95.     borderBottomColor: '#ffffff',
96.     borderBottomWidth: StyleSheet.hairlineWidth,
97.   },
98.
99. });
```

Capturas

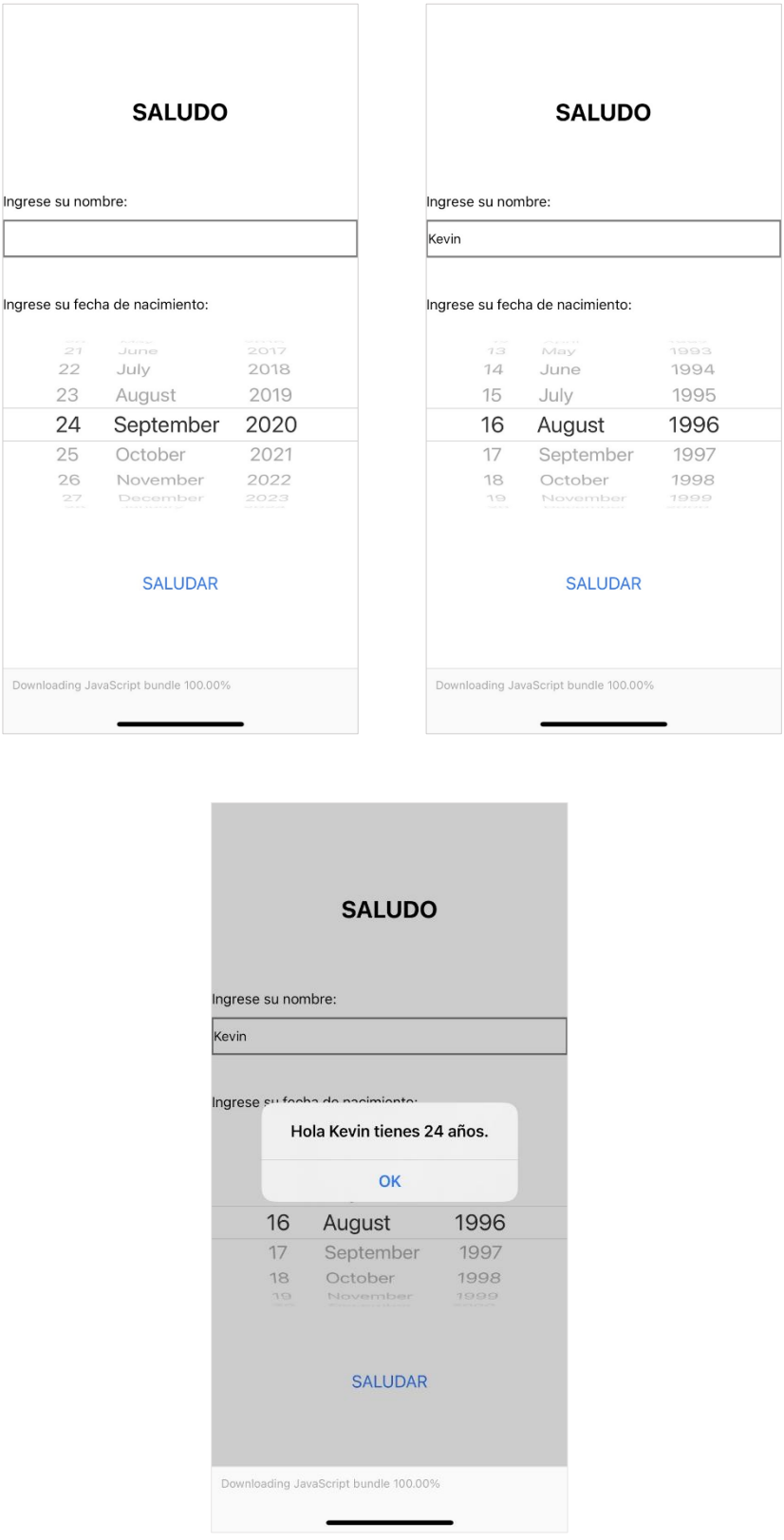


Ilustración 25: Capturas React Native - Ejemplo 1 (Zurita, 2020)

En Xamarin

main.xaml

```
1. <?xml version="1.0" encoding="utf-8" ?>
2.
3. <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
4.              xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
5.              xmlns:local="clr-namespace:Saludo"
6.              x:Class="Saludo.MainPage">
7.
8.     <StackLayout>
9.
10.        <Label Text="SALUDO"
11.               HorizontalOptions="Center"
12.               FontSize="Large"
13.               FontAttributes="Bold"
14.               VerticalOptions="StartAndExpand"/>
15.
16.        <Label Text="Nombre"/>
17.
18.        <Entry x:Name="txt"
19.               Text="Ingrese su nombre"
20.               VerticalOptions="StartAndExpand"/>
21.
22.        <Label Text="Ingrese su fecha de nacimiento"/>
23.
24.        <DatePicker x:Name="dte"
25.                    VerticalOptions="StartAndExpand"/>
26.
27.        <Button x:Name="btn"
28.                Text="Saludar"
29.                Clicked="Btn_clicked"/>
30.    </StackLayout>
31.
32.</ContentPage>
```

main.cs

```
1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. using System.Threading.Tasks;
6. using Xamarin.Forms;
7.
8. namespace Saludo
9. {
10.     public partial class MainPage : ContentPage
11.     {
12.         public MainPage()
13.         {
14.             InitializeComponent();
15.         }
16.
17.         private async void Btn_clicked(object sender, EventArgs e)
18.         {
19.             var today = DateTime.Today;
20.             var age = today.Year - dte.Date.Year;
21.
22.             await DisplayAlert(
23.                 "Alert",
24.                 "Hola " + txt.Text + " tienes " + age.ToString() + " años.",
25.                 "Cancelar"
26.                 );
27.         }
28.     }
29. }
```

Capturas

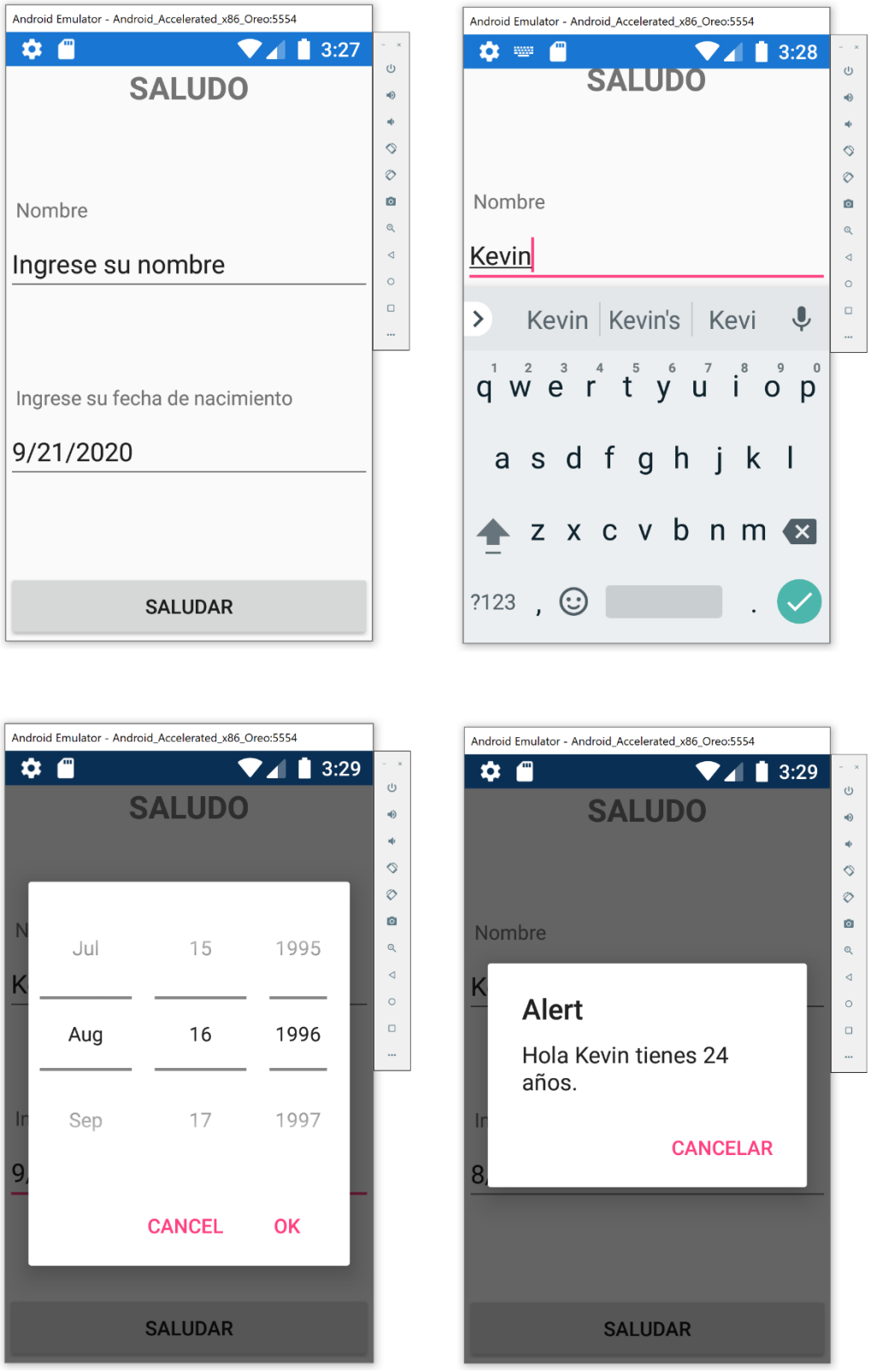


Ilustración 26: Capturas Xamarin - Ejemplo 1 (Zurita, 2020)

Ejemplo 2

El segundo ejemplo es un aplicativo en cual, por medio de un formulario simple en donde se ingrese una dirección de un destinatario, un asunto y un mensaje, el usuario podrá enviar correos electrónicos.

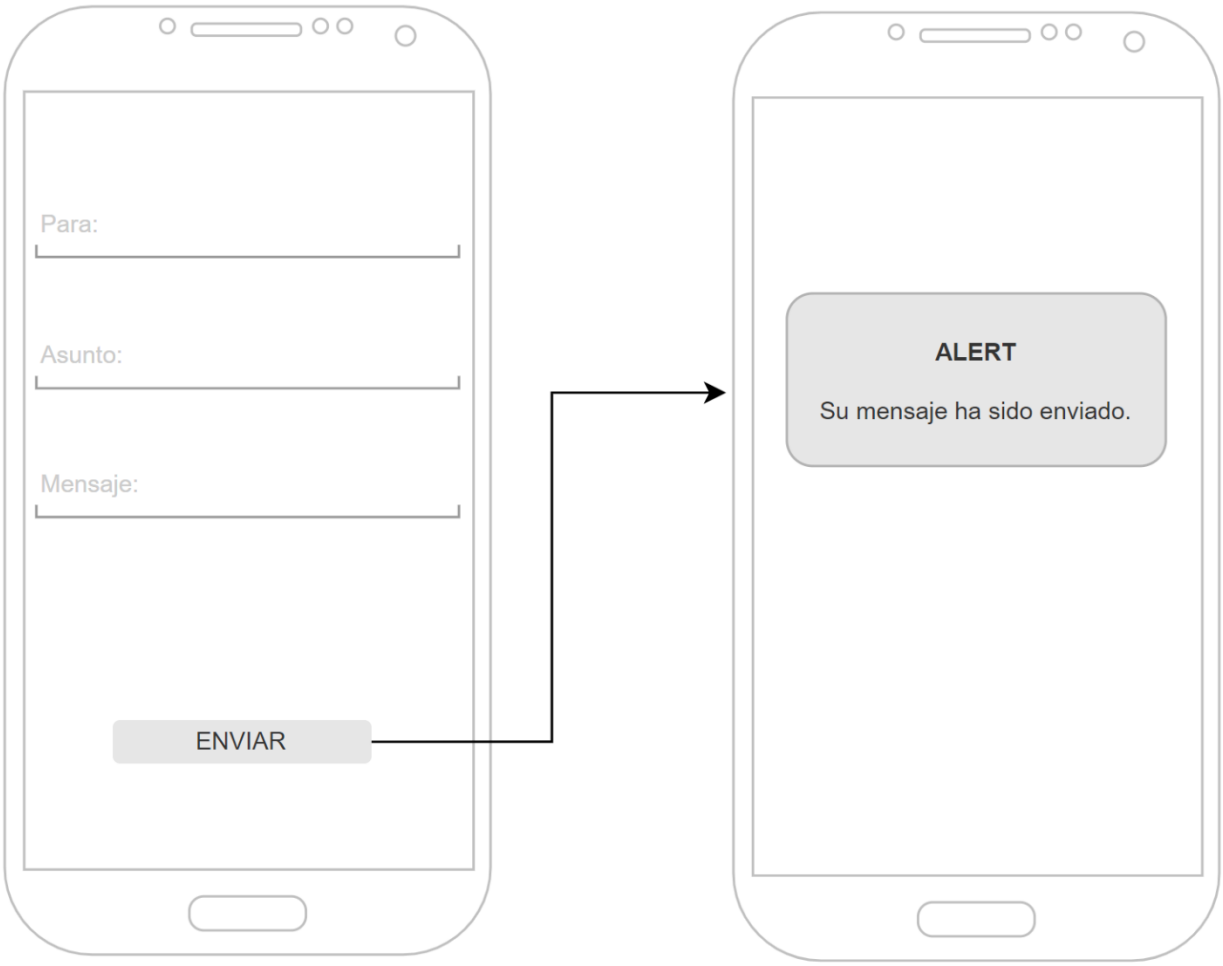


Ilustración 27: Diseño - Ejemplo 2 (Zurita, 2020)

En Flutter

main.dart

```
1. import 'package:flutter/material.dart';
2. import 'package:url_launcher/url_launcher.dart';
3.
4. void main() {
5.   runApp(MyApp());
6. }
7.
8. class MyApp extends StatelessWidget {
9.   @override
10.  Widget build(BuildContext context) {
11.    return MaterialApp(
12.      theme: ThemeData(
13.        primarySwatch: Colors.blue,
14.        visualDensity: VisualDensity.adaptivePlatformDensity,
15.      ),
16.      home: MyHomePage(),
17.    );
18.  }
19.}
20.
21.class MyHomePage extends StatefulWidget{
22.  @override
23.  MyHomePageState createState() => MyHomePageState();
24.}
25. class MyHomePageState extends State<MyHomePage>{
26.  TextEditingController _toMail;
27.  TextEditingController _asunto;
28.  TextEditingController _mensaje;
29.
30.  void initState(){
31.    _toMail = TextEditingController();
32.    _asunto = TextEditingController();
33.    _mensaje = TextEditingController();
```

```
34.     super.initState();
35.   }
36.
37.   @override
38.   Widget build(BuildContext context){
39.     return Scaffold(
40.       appBar: AppBar(
41.         title: Text('ENVIAR CORREO ELECTRÓNICO'),
42.       ),
43.
44.       body: Padding(
45.         padding: EdgeInsets.all(20),
46.         child: Column(
47.
48.           children: <Widget>[
49.             TextField(
50.               controller: _toMail,
51.               decoration: InputDecoration(
52.                 filled: true,
53.                 labelText: 'Para',
54.               ),
55.             ),
56.
57.             SizedBox(height: 15),
58.
59.             TextField(
60.               controller: _asunto,
61.               decoration: InputDecoration(
62.                 filled: true,
63.                 labelText: 'Asunto',
64.               ),
65.             ),
66.              SizedBox(height: 15),
67.              TextField(
68.               controller: _mensaje,
69.               decoration: InputDecoration(
70.                 filled: true,
```

```
71.                 labelText: 'Mensaje',
72.               ),
73.             ),
74.             SizedBox(height: 15),
75.
76.             RaisedButton(
77.               child: Text('ENVIAR'),
78.               onPressed: () {
79.                 _launchURL(_toMail, _asunto, _mensaje);
80.               },
81.             )
82.           ],
83.         )
84.       ),
85.     );
86.   }
87.
88.   Future _launchURL(String toMail, String asunto, String mensaje) async {
89.     var url = 'mailto:$toMail?subject=$asunto&body=$mensaje';
90.     if (await canLaunch(url)) {
91.       builder: (context) => AlertDialog(
92.           title: Text('Alert'),
93.           content: Text('Mensaje Enviado'),
94.       );
95.           await launch(url);
96.     }
97.   } else {
98.       throw 'Error: $url';
99.   }
100. }
```

Capturas

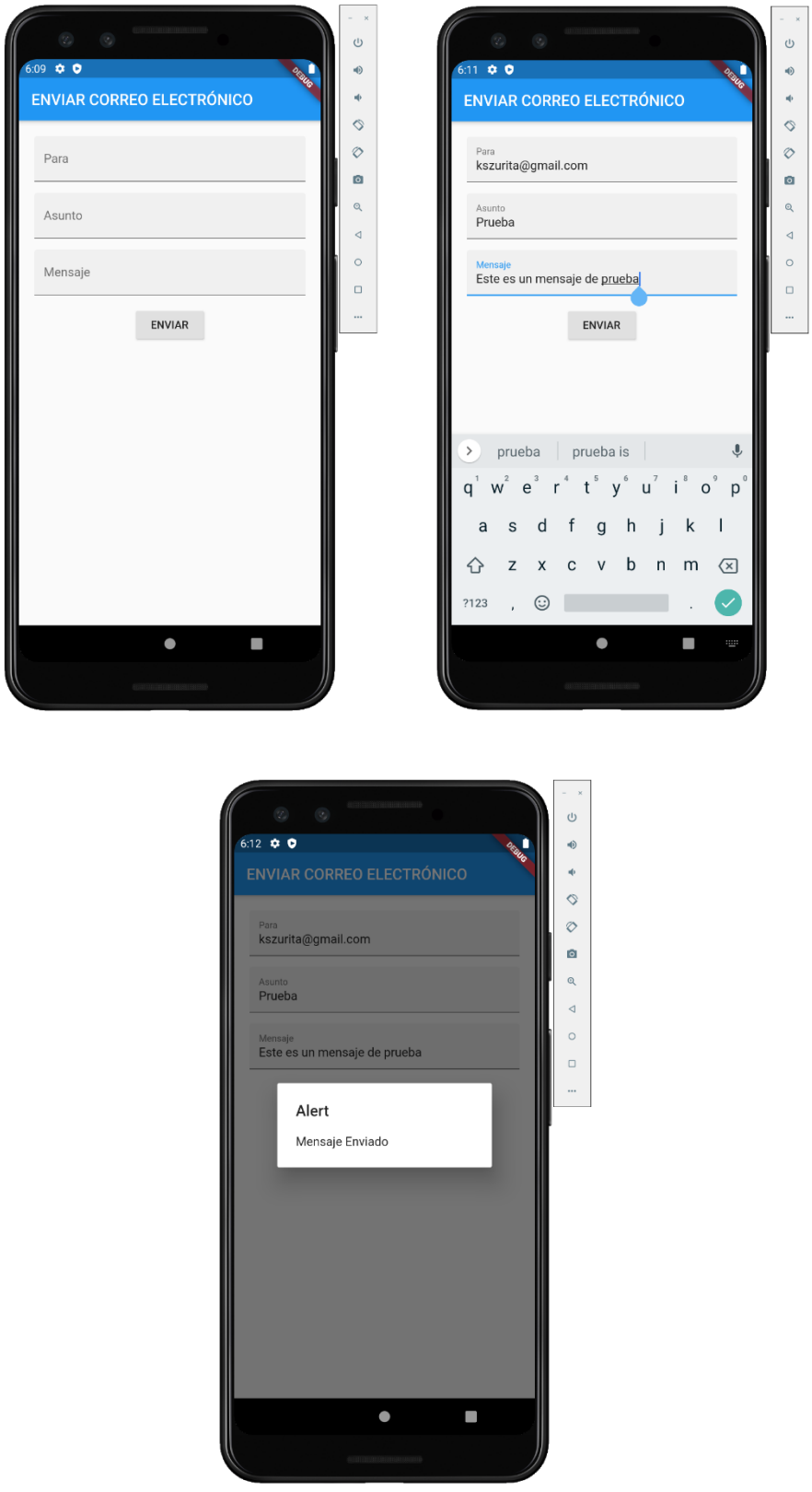


Ilustración 28: Capturas Flutter - Ejemplo 2 (Zurita, 2020)

En NativeScript

home.xml

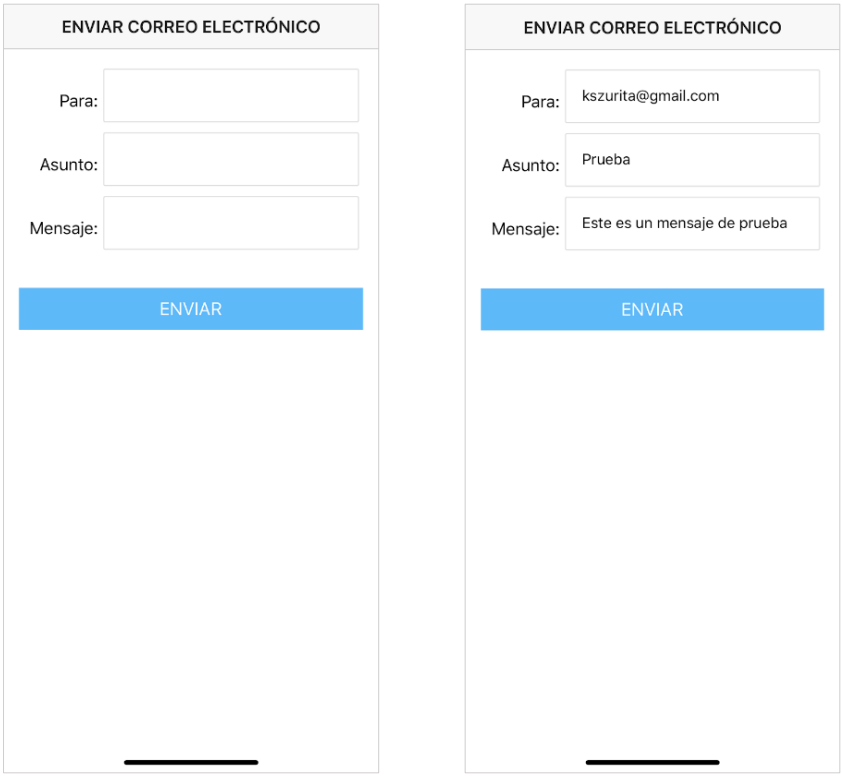
```
1. <Page loaded="pageLoaded" class="page"
   xmlns="http://www.nativescript.org/tns.xsd">
2.     <ActionBar title="ENVIAR CORREO ELECTRÓNICO" class="action-
   bar"></ActionBar>
3.     <ScrollView>
4.         <StackLayout>
5.             <GridLayout rows="*, *, *, *, *" columns="80, *" class="form">
6.                 <Label row="0" col="0" text="Para:"
   horizontalAlignment="right" class="label" />
7.                 <TextField row="0" col="1" text="{{ toEmail }}" class="input
   input-border" keyboardType="email" />
8.                 <Label row="3" col="0" text="Asunto:"
   horizontalAlignment="right" class="label" />
9.                 <TextField row="3" col="1" text="{{ subject }}" class="input
   input-border" />
10.                <Label row="4" col="0" text="Mensaje:"
   horizontalAlignment="right" class="label" />
11.                <TextField row="4" col="1" text="{{ message }}" class="input
   input-border" />
12.            </GridLayout>
13.            <Button text="ENVIAR" tap="{{ onEmailSend }}" class="btn btn-
   primary" />
14.        </StackLayout>
15.    </ScrollView>
16.</Page>
```

home-view-model.js

```
1. var observableModule = require("data/observable");
2. var email = require("../nativescript-email");
3. var dialogs = require("ui/dialogs");
4.
5. function HomeViewModel() {
```

```
6.    var viewModel = observableModule.fromObject({
7.      onEmailSend: function () {
8.
9.        // basic validation
10.       if (!this.get("subject") || !this.get("message") ||
    !this.get("toEmail")) {
11.         return;
12.       }
13.
14.       email.compose({
15.         subject: this.get("subject"),
16.         body: this.get("message"),
17.         to: this.get("toEmail").split(" "),
18.
19.       }).then(
20.         function () {
21.           alert("Mensaje Enviado");
22.         }, function (err) {
23.           alert("Error: " + err);
24.         });
25.     },
26.   });
27.
28.   return viewModel;
29. }
30.
31. module.exports = HomeViewModel;
```

Capturas



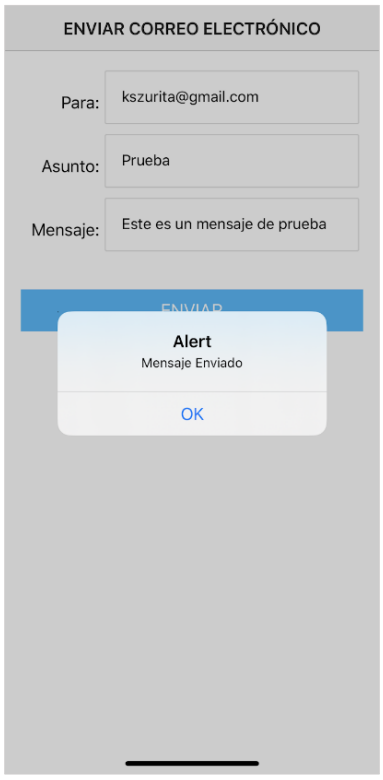


Ilustración 29: Capturas NativeScript - Ejemplo 2 (Zurita, 2020)

En React Native

App.js

```
1. import React from 'react';
2. import {StyleSheet, Text, View, Button, Alert, TextInput, Linking} from
   'react-native';
3.
4. export default function App() {
5.
6.   const Separator = () => (3
7.     <View style={styles.separator}
8.     />
9.   );
10.
11.  return (
12.    <View style={styles.container}>
13.
14.      <Text style={styles.baseText}>
15.        ENVIAR CORREO ELECTRÓNICO
16.      </Text>
17.
18.      <Separator />
19.
20.      <Text style={styles.labelText}>Para:</Text>
21.
22.      <TextInput
23.      style={styles.inputText}
24.      ref={ref => {this.para = ref;}}
25.      />
26.
27.      <Separator />
28.
29.      <Text style={styles.labelText}>Asunto:</Text>
30.
31.      <TextInput
32.      style={styles.inputText}
```

```
33.       ref={ref => {this.asunto = ref;}}
34.       />
35.
36.       <Separator />
37.
38.       <Text style={styles.labelText}>Mensaje:</Text>
39.
40.       <TextInput
41.       style={styles.inputText}
42.       ref={ref => {this.mensaje = ref;}}
43.       />
44.
45.       <Separator />
46.
47.       <Button
48.       title="ENVIAR"
49.       onPress = {
50.         () => Linking.openURL(
51.           'mailto:' + this.para + '?' +
52.           'subject=' + this.asunto + '&' +
53.           'body=' + this.mensaje
54.         ),
55.         () => Alert.alert('Mensaje Enviado')
56.       }
57.       />
58.
59.     </View>
60.   );
61. }
62.
63. const styles = StyleSheet.create({
64.
65.   container: {
66.     flex: 1,
67.     backgroundColor: '#fff',
68.     justifyContent: 'center',
69.   },
```

```
70.
71.   baseText: {
72.     fontWeight: 'bold',
73.     fontSize: 25,
74.     paddingBottom: 30,
75.     textAlign:'center',
76.   },
77.
78.   inputText: {
79.     height: 40,
80.     width: 375,
81.     borderColor: 'gray',
82.     borderWidth: 2,
83.
84.   },
85.
86.   labelText: {
87.     fontSize: 15,
88.     paddingBottom: 10,
89.   },
90.
91.   separator: {
92.     marginVertical: 15,
93.     borderBottomColor: '#ffffff',
94.     borderBottomWidth: StyleSheet.hairlineWidth,
95.   },
96.
97. });
```

Capturas

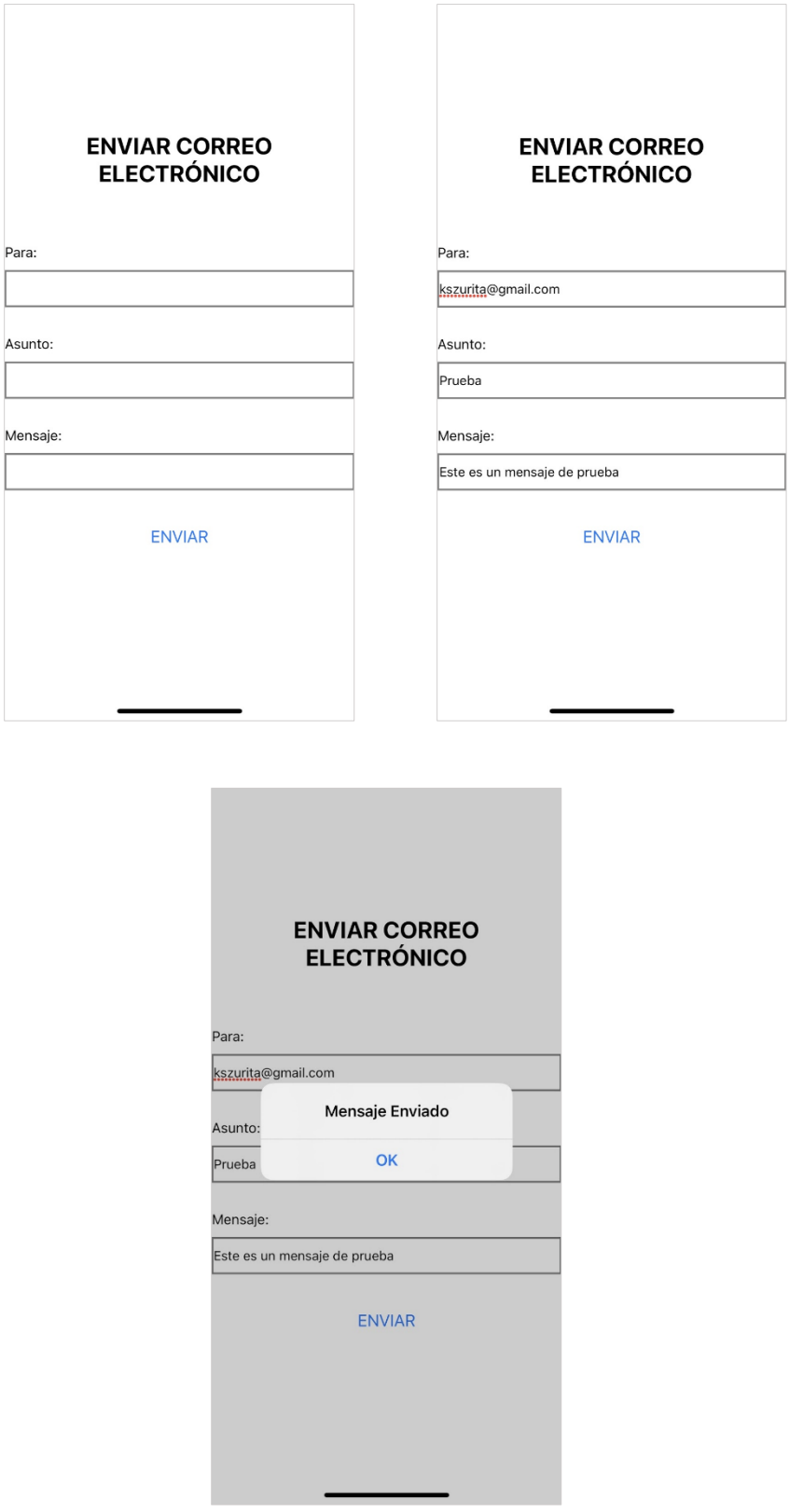


Ilustración 30: Capturas React Native - Ejemplo 2 (Zurita, 2020)

En Xamarin

main.xaml

```
1. <?xml version="1.0" encoding="utf-8" ?>
2.
3. <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
4.              xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
5.              xmlns:local="clr-namespace:Email"
6.              x:Class="Email.MainPage">
7.
8.     <StackLayout>
9.
10.        <Label Text="ENVIAR CORREO ELECTRÓNICO"
11.               FontSize="Small"
12.               FontAttributes="Bold"
13.               HorizontalOptions="Center"
14.               VerticalOptions="StartAndExpand" />
15.
16.        <Label Text="  Para"
17.               FontSize="Micro"/>
18.        <Entry Placeholder="Para"
19.               Keyboard="Email"
20.               FontSize="Micro"
21.               HorizontalOptions="FillAndExpand"
22.               VerticalOptions="StartAndExpand"
23.               x:Name="email"/>
24.
25.        <Label Text="  Asunto"
26.               FontSize="Micro"/>
27.        <Entry Placeholder="Asunto"
28.               FontSize="Micro"
29.               HorizontalOptions="FillAndExpand"
30.               VerticalOptions="StartAndExpand"
31.               x:Name="asunto"/>
32.
33.        <Label Text="  Mensaje"
```

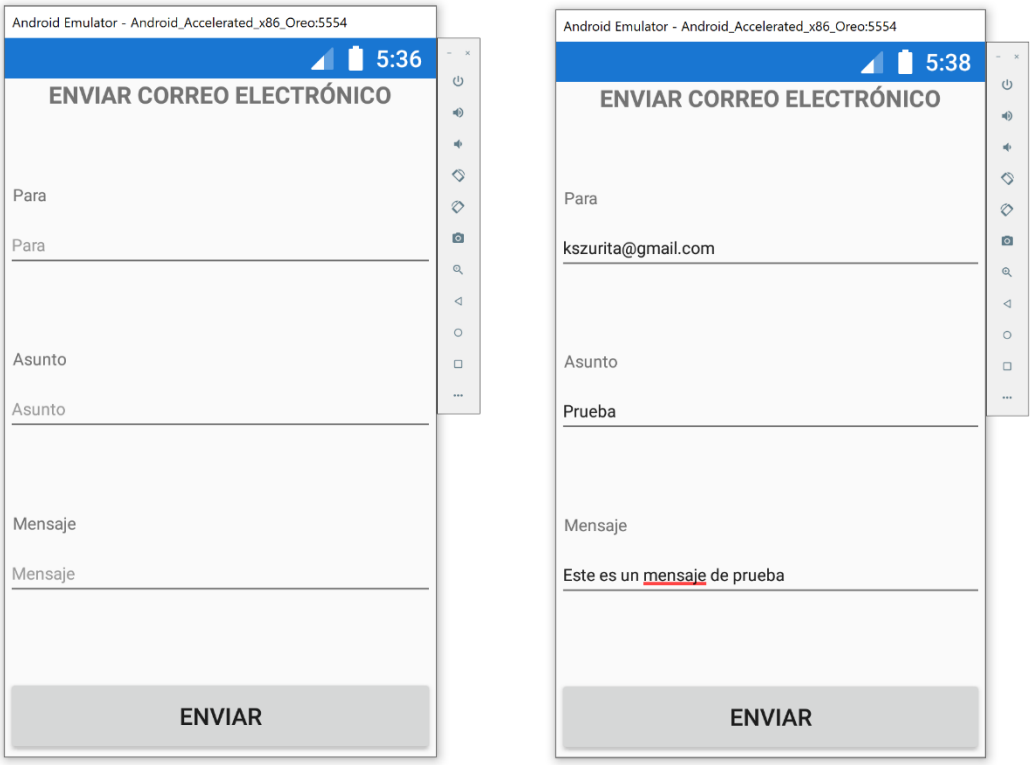
```
                FontSize="Micro"/>
        <Editor Placeholder="Mensaje"
                FontSize="Micro"
                HorizontalOptions="FillAndExpand"
                VerticalOptions="StartAndExpand"
                x:Name="mensaje"/>

        <Button Text="Enviar"
                FontSize="Small"
                Clicked="Button_Clicked"
                x:Name="btn"/>

    </StackLayout>

</ContentPage>
```

main.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Net.Mail;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;


namespace Email
{
    public partial class MainPage : ContentPage
    {

        SmtpClient SmtpServer;

        public MainPage()
        {
```

```
20.             InitializeComponent();
21.         }
22.
23.         private void Button_Clicked(object sender, EventArgs e)
24.         {
25.             try
26.             {
27.                 SmtpServer = new SmtpClient("smtp.gmail.com");
28.                 SmtpServer.Port = 587;
29.                 SmtpServer.Host = "smtp.gmail.com";
30.                 SmtpServer.EnableSsl = true;
31.                 SmtpServer.UseDefaultCredentials = false;
32.                 SmtpServer.Credentials = new
    System.Net.NetworkCredential("kszurita@outlook.com","*********");
33.                 SmtpServer.SendAsync("kszurita@outlook.com", email.Text,
    asunto.Text, mensaje.Text,"xyz123d");
34.                 SmtpServer.SendCompleted += emailSendCompleted;
35.             }
36.             catch (Exception ex)
37.             {
38.                 DisplayAlert("Error", ex.Message, "Cancelar");
39.             }
40.         }
41.
42.         private void emailSendCompleted(object sender, AsyncCompletedEventArgs
    e)
43.         {
44.             DisplayAlert("Alert", "Mensaje Enviado", "OK");
45.         }
46.     }
47. }
```

Capturas



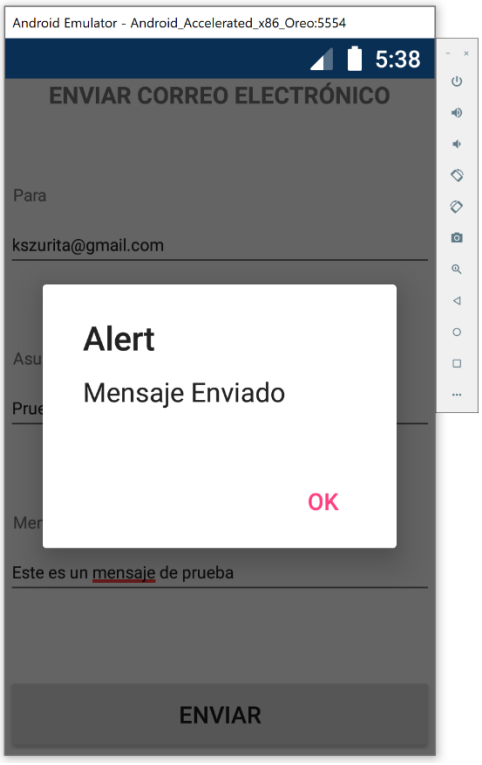


Ilustración 31: Capturas Xamarin - Ejemplo 2 (Zurita, 2020)

4.5. Rendimiento

Flutter	NativeScript
Flutter ofrece quizás el mejor rendimiento entre todos los demás frameworks utilizados en el desarrollo del presente trabajo. Al utilizar lenguaje Dart para sus productos, elimina el puente de JavaScript que tienen sus competidores, esto permite que su rendimiento sea mucho más similar al que ofrecen los desarrollos nativos. El framework, además, permite desarrollar productos de forma más eficiente cuyo resultado no solamente se observa en el rendimiento de la aplicación, sino también, en la reducción de tiempos y costos que implica este tipo de proyectos.	Aunque su arquitectura es similar a la de React Native, su rendimiento es ligeramente inferior. Debido a que utiliza TypeScript para sus desarrollos, este primero debe ser traducido a JavaScript para que posteriormente sea compilado. Es importante mencionar que, aunque esta tecnología ofrece varios medios en los cuales se puede desarrollar sus aplicaciones, carece de aportes tanto del propietario como de la comunidad en general, afectando el desarrollo de productos e incluso incrementando el tiempo y los recursos que se invierten.
React Native	**Xamarin**
El rendimiento del framework React Native es muy similar al que ofrecen las aplicaciones nativas. Además, al poder compartir hasta un 80% de código en el desarrollo de aplicaciones móviles, esta tecnología brinda constituye una gran opción para la creación de cualquier producto. Ahora, debido a que el lenguaje que se utiliza para su codificación es JavaScript, pueden surgir inconvenientes para el manejo de grandes volúmenes de datos, sin embargo, existen componentes adicionales que ayudan a solucionar este tipo de problemas. Por tal motivo, Facebook Inc. que es la empresa creadora del framework, hace uso de este para el desarrollo de sus productos tales como, Facebook, Instagram y WhatsApp.	Las aplicaciones desarrolladas en el framework Xamarin, tienen un rendimiento considerablemente inferior a las aplicaciones nativas. Esto se debe a que su arquitectura no le permite soportar ni funcionalidades ni mucho menos gráficos pesados. Por tal motivo, es recomendable el uso de cualquier otra tecnología si la aplicación es rica en diseño. Aunque el mayor porcentaje de su código es compatible con Android, IOS e incluso con el descontinuado Windows Phone, su rendimiento se ve considerablemente afectado mientras las aplicaciones son robustas, por lo que para este tipo de desarrollos se debería considerar otras opciones.

Tabla 5: Comparación de rendimiento (Zurita, 2020)

4.6. Soporte

Generalmente dentro del campo de la tecnología, se desarrolla productos que no se gastan con el uso que se les da, pero que si necesitan soporte para ampliar sus capacidades. Dar soporte a estos productos no significa restaurarlos hasta su estado original, al contrario, se refiere a la corrección de pequeños defectos que tienen o incluso a la adición de nuevas funcionalidades (Cataldi, Lage, Pessacq, & García, 1999).

El soporte que se brinda en las tecnologías utilizadas en el presente trabajo (entiéndase este como aportes que brindan sus propietarios y la comunidad en general), tienen grandes brechas de diferencia, pues mientras unos con el tiempo han ido incrementado en popularidad, otros no han tenido un mayor crecimiento e incluso en algunos casos sus usuarios han perdido interés. Para muestra obsérvese la Ilustración 32 en esta se puede observar como en el lapso de un año, las personas que realizan búsquedas en Google se han interesado más por Flutter que cualquier otro entorno de desarrollo.

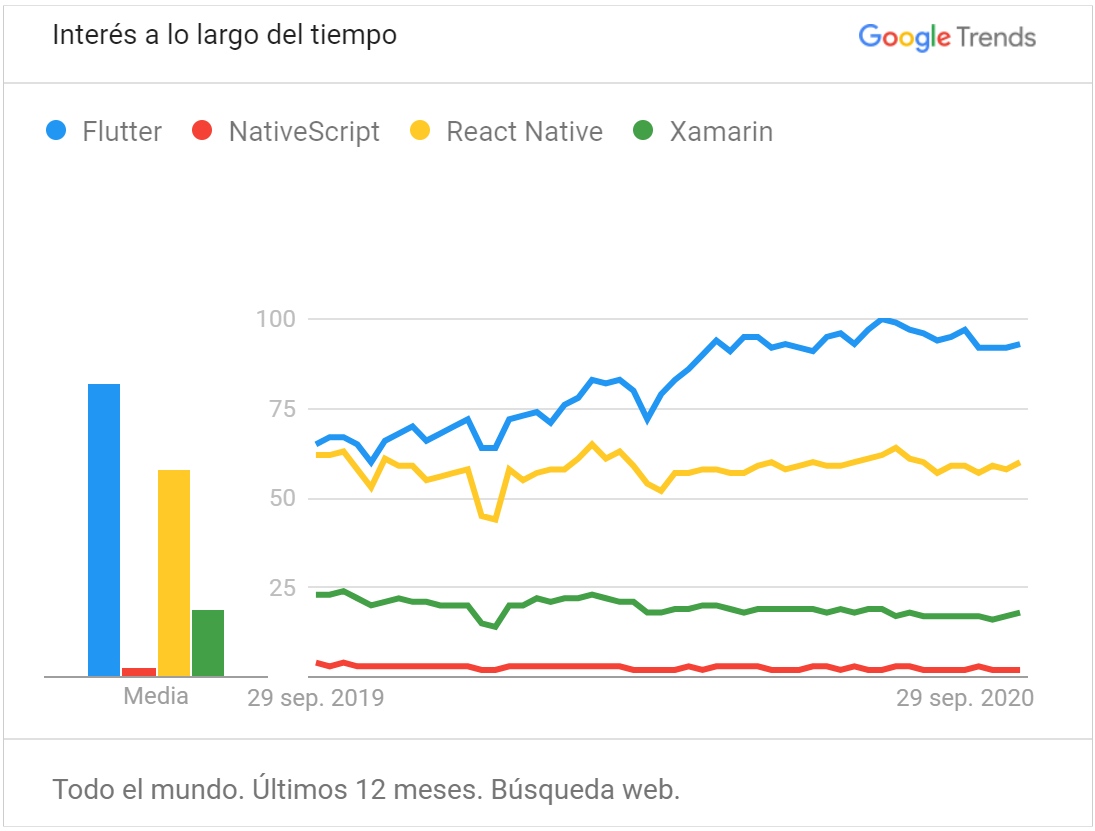


Ilustración 32: Tendencias de búsqueda entre Flutter, Native Script, React Native y Xamarin (Google, s.f.)

En la Ilustración 33, la cual evalúa el nivel de popularidad en los países en que Google está presente, de la misma manera, se puede observar que Flutter es la tecnología con mayor interés alrededor del mundo (incluso en Ecuador), seguido en menor parte de React Native y con un nulo interés por NativeScript y Xamarin.

Ilustración 33: Tendencias de búsqueda en países entre Flutter, Native Script, React Native y Xamarin (Google, s.f.)

Finalmente, y para dar por sentado que Flutter es la tecnología que actualmente cuenta con un mayor número de recursos, o por lo menos que su cantidad crece contantemente, podemos observar la Ilustración 34. Está figura hace referencia al portal especializado en dudas sobre software Stack Overflow y sus rankings anuales. Para este 2020, se determinó que Flutter está entre los frameworks más utilizados, seguido de React Native y Xamarin. Esto no solo demuestra la popularidad de la tecnología creada por Google, sino que también muestra que existen un mayor número de dudas que han sido solucionadas durante los últimos meses.

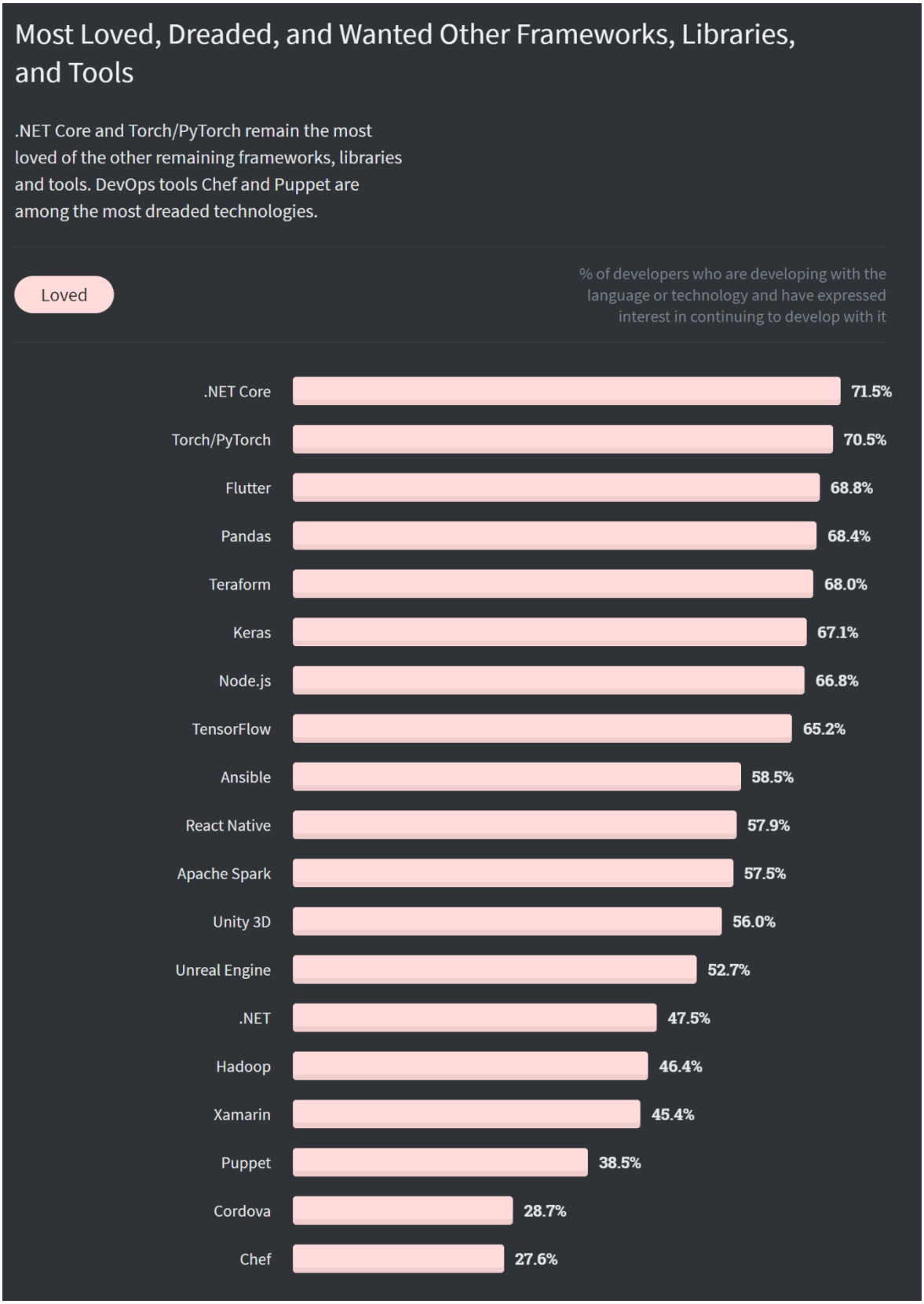


Ilustración 34: Ranking de los frameworks más utilizados (Stack Overflow, 2020)

4.7. Ventajas y Desventajas

	Ventajas	Desventajas
Flutter	Aunque utiliza Dart como lenguaje principal, se compila en C, lo que permite tener un código muy similar al nativo. Los cambios realizados se pueden ver al instante mediante Hot-Reload, si tener la necesidad de compilar el proyecto completo. Se puede desarrollar productos mediante una misma base de código para dispositivos móviles, para escritorio e incluso para web.	Al ser Dart un lenguaje relativamente nuevo y poco popular por el momento, se encuentra en constante cambio, por lo que dificulta su el mantenimiento a largo plazo. Flutter es un framework todavía joven y no ha despegado del todo, por lo sus recursos e información son limitados. El tamaño de sus aplicaciones son más grandes que el promedio, además su construcción requiere de más recursos que el resto de frameworks.

Tabla 6: Ventajas y Desventajas de Flutter (Zurita, 2020)

	Ventajas	Desventajas
NativeScript	Debido a que utiliza TypeScript para la construcción de sus aplicaciones, se puede codificar mediante el paradigma de programación orientada a objetos. Se puede utilizar Playground como entorno de desarrollo. Esta es una herramienta web y de fácil uso para cualquier desarrollador. Sus desarrollos se ejecutan de forma más rápida y precisa en comparación con las otras alternativas.	NativeScript es un framework poco conocido, por lo que se vuelve difícil conseguir cualquier tipo de información. Aunque en la web oficial existan varios recursos que faciliten el desarrollo, estos se encuentran en menor cantidad a los de su competencia. Este framework solo permite desarrollos móviles, por lo que es necesario el uso de alguna otra tecnología para aplicativos tipo web.

Tabla 7: Ventajas y Desventajas de NativeScript (Zurita, 2020)

	Ventajas	Desventajas
React Native	Este framework además del desarrollo móvil multiplataforma, permite construir a la par aplicaciones web. React Native es la tecnología de la que más recursos e información se puede encontrar, esto facilita en gran medida el trabajo de los desarrolladores. Al igual que Flutter, esta herramienta soporta Hot-Reload, por lo que permite un desarrollo más rápido y eficiente.	Para la ejecución de proyectos directamente en los dispositivos, es necesario instalar la aplicación Expo, además de tener una conexión común entre el teléfono y la máquina donde se está desarrollando. React Native utiliza como lenguaje principal JavaScript, por lo que se dificulta la tarea cuando se trabaja con un gran volumen de datos. Al no tener un entorno de desarrollo oficial, es necesario instalar componentes adicionales dentro de la herramienta con la que se está trabajando.

Tabla 8: Ventajas y Desventajas de React Native (Zurita, 2020)

	Ventajas	Desventajas
Xamarin	Al utilizar C# como lenguaje principal, existe la suficiente cantidad de información para solventar cualquier problema. El entorno de desarrollo recomendando para este framework es Visual Studio. Este es un IDE completo que cualquier desarrollador puede utilizar. Al igual que algunas de las tecnologías con la que compite, este framework permite el desarrollo no solo de aplicaciones para teléfonos, sino también para cualquier dispositivo inteligente.	Aunque Xamarin es una herramienta relativamente popular, no existen suficientes recursos, sobre todo librerías para tareas específicas. Xamarin no tiene la capacidad suficiente como para desarrollar aplicaciones de gran tamaño, o que manejen una buena cantidad de datos. Para poder observar los cambios realizados en un desarrollo, es necesario compilar todo el proyecto, consumiendo así más tiempo que sus rivales.

Tabla 9: Ventajas y Desventajas de Xamarin (Zurita, 2020)

5. Conclusiones y Recomendaciones

5.1. Conclusiones

- Mediante el análisis realizado en el presente trabajo, se logró concluir que, se puede notar con claridad el interés que ha surgido por las aplicaciones móviles durante los últimos años, obteniendo como resultado el desarrollo de nuevas tecnologías que permiten su construcción.
- Las aplicaciones móviles multiplataforma representan una opción considerable para las compañías que no cuentan con el tiempo, ni tampoco con los recursos necesarios para desarrollar aplicaciones totalmente nativas.
- Las aplicaciones móviles multiplataforma compiladas de forma nativa, son la mejor opción para los desarrolladores que desean lanzar al mismo tiempo, y de manera ágil nuevas actualizaciones en sus proyectos sin importar la plataforma en la que se encuentran.
- Los frameworks que permiten el desarrollo de aplicaciones multiplataforma son una herramienta cuyo objetivo principal es el de reducir tiempo, recursos e incluso código, sin embargo, estos no reemplazan a las tecnologías que permiten un desarrollo complemente nativo.
- Debido a la rapidez con la que evoluciona los dispositivos móviles, y por ende su software, existen cada vez más opciones que permiten crear nuevas aplicaciones del tipo multiplataforma. Esto no solo por parte de las empresas líderes en tecnología, sino también por parte de la comunidad de software libre.
- Aunque todas las tecnologías analizadas en este estudio utilizan lenguajes de programación diferentes, todos cuentan con un nivel de información suficiente para resolver cualquier duda que se presente en el proceso de codificación.
- Flutter es el framework que más ha incrementado su nivel de popularidad en los últimos años, por esta razón es que existe un gran volumen de información que sirve de ayuda para los desarrolladores.
- NativeScript es la tecnología menos conocida en comparación a los otras que han sido analizadas en el presente trabajo, aunque es un framework relativamente joven, aún no ha tenido un nivel de aceptación aceptable por parte de los desarrolladores.

- React Native es la herramienta más sólida hasta el momento, aunque su nivel de popularidad no ha aumentado, se pueden encontrar en el mercado varias aplicaciones desarrolladas con esta tecnología.
- Xamarin es un framework que, aunque permite el desarrollo de aplicaciones multiplataforma de una manera sencilla, no tiene la capacidad de soportar proyectos complejos y de gran tamaño.
- Por el momento, las aplicaciones desarrolladas en los frameworks anteriormente mencionados no tienen la capacidad de alcanzar el nivel de usabilidad, experiencia y rendimiento de las que son desarrolladas en código nativo, sin embargo, sus capacidades son muy cercanas y dependiendo del proyecto la diferencia puede llegar a ser casi imperceptible.

5.2. Recomendaciones

- Debido al avance considerablemente acelerado de la tecnología en los últimos años, es recomendable dentro de esta área mantenerse al tanto sobre las nuevas herramientas que existen en el mercado, generalmente surgen soluciones innovadoras que facilitan varias tareas en la creación de software.
- Previo al proceso de construcción de aplicaciones móviles, es recomendable evaluar las ventajas y desventajas que conlleva el desarrollo multiplataforma, así como también el desarrollo nativo.
- Es importante considerar los recursos tanto de software como de hardware disponibles previo al desarrollo de una aplicación, de esta manera, la tarea de elegir un framework con el cual trabajar en el desarrollo de apps, se vuelve menos complicado.
- Se recomienda evaluar el tipo de aplicación a desarrollar, muchas veces se invierte más de lo necesario en la construcción de aplicaciones nativas, cuando el proyecto no demandaba más de un desarrollo híbrido.
- Se recomienda mantener actualizados los frameworks a sus últimas versiones, por lo general, estas traen nuevas funcionalidades que suelen ser de ayuda en el proceso de desarrollo de software.

- Es recomendable analizar previamente las habilidades y capacidades de las personas relacionadas al proceso de construcción de software, de esta forma se puede asegurar que tanto los recursos como el tiempo invertido en el desarrollo, no se vean seriamente afectados ante la presencia de un problema.
- Es importante tener métricas previamente establecidas para que, una vez lanzada una aplicación, los usuarios puedan ofrecer una retroalimentación que permita saber si las herramientas utilizadas en el desarrollo fueron correctamente escogidas.
- Se recomienda finalmente considerar que, las capacidades de los frameworks analizados en el presente trabajo son diferentes y su uso depende de los trabajos a realizar, de la compatibilidad que ofrece o incluso de las habilidades del programador.

6. Referencias

Android. (s.f.). What is Android. Recuperado el 26 de Agosto de 2020, de Android: https://www.android.com/what-is-android/

Avdic, D. (2019). React Native vs Xamarin – Mobile for industry. Lund: Lund University.

Ayoze, A. (2017). Curso de Programación Web - JavaScript, Ajax y jQuery. ITcampus Academy.

Bierman, G., Abadi, M., & Torgersen, M. (2014). Understanding TypeScript. Berlín: European Conference on Object-Oriented Programming.

Cataldi, Z., Lage, F., Pessacq, R., & García, R. (1999). INGENIERIA DE SOFTWARE EDUCATIVO. Buenos Aires: Facultad de Ingeniería UBA.

Constantin, O., Gordan, C., Mihaela, N., & Berczes, T. (2017). Comparative Study of Google Android, Apple iOS and Microsoft Windows Phone Mobile Operating Systems. Debrecen: University of Debrecen.

Cuello, J., & Vittone, J. (2013). Diseñando apps para móviles. Barcelona: Catalina Duque Giraldo.

Delia, L., Galdamez, N., Thomas, P., Corbalan, L., & Pesado, P. (2014). Análisis Experimental de desarrollo de Aplicaciones Móviles Multiplataforma. La Plata: Universidad Nacional de La Plata.

Dimes, T. (2016). Programación en C# para Principiantes. Babelcube Inc.

Erfani, M., Mesbah, A., & Kruchten, P. (2013). Real Challenges in Mobile App Development. Vancouver: University of British Columbia.

Facebook Inc. (s.f.). Native Development For Everyone. Recuperado el 12 de Abril de 2020, de React Native: https://reactnative.dev/

Flutter. (s.f.). Technical Overview. Recuperado el 10 de Marzo de 2020, de Flutter: https://flutter.dev/docs/resources/technical-overview

Flutter. (s.f.). Windows install. Recuperado el 12 de Mayo de 2020, de Get started: https://flutter.dev/docs/get-started/install/windows

Gardner, H., & Davis, K. (2014). La generación APP. Barcelona: Espasa Libros.

GitHub. (s.f.). NativeScript. Recuperado el 27 de Marzo de 2020, de GitHub: https://github.com/NativeScript/NativeScript

Google. (s.f.). Dart. Recuperado el 06 de Julio de 2020, de Google: https://opensource.google/projects/dart

Kuitunen, M. (2019). CROSS-PLATFORM MOBILE APPLICATION DEVELOPMENT WITH REACT NATIVE. Tampere: Tampere University of Technology.

Malavolta, I. (2016). Beyond Native Apps: Web Technologies to the Rescue! Amsterdam: Vrije Universiteit Amsterdam.

Microsoft. (s.f.). ¿Qué es Xamarin? Recuperado el 16 de Abril de 2020, de Xamarin: https://docs.microsoft.com/es-es/xamarin/get-started/what-is-xamarin

Microsoft. (s.f.). A tour of the C# language. Recuperado el 18 de Agosto de 2020, de .NET: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

Microsoft. (s.f.). What is TypeScript? Recuperado el 06 de Agosto de 2020, de TypeScript: https://www.typescriptlang.org/

Microsoft. (s.f.). What is Xamarin? Recuperado el 15 de Abril de 2020, de Xamarin: https://dotnet.microsoft.com/learn/xamarin/what-is-xamarin

Microsoft. (s.f.). Xamarin. Recuperado el 13 de Mayo de 2020, de Microsoft: https://dotnet.microsoft.com/learn/xamarin/hello-world-tutorial/intro

NativeScript. (s.f.). System Requirements. Recuperado el 12 de Mayo de 2020, de Installation: https://docs.nativescript.org/start/general-requirements#full-setup-requirements-macos

NativeScript. (s.f.). Technical Overview. Recuperado el 2020 de Marzo de 27, de NativeScript: https://docs.nativescript.org/core-concepts/technical-overview

Payne, R. (2019). Beginning App Development with Flutter. Dallas: Apress.

Pérez, J. (2019). Introducción a JavaScript.

Progress. (s.f.). NativeScript. Recuperado el 24 de Marzo de 2020, de Progress: https://www.progress.com/nativescript

Ramírez, R. (2012). Métodos para el desarrollo de aplicaciones móviles. Barcelona: Universitat Oberta de Catalunya.

Ravindranath, L., Padhye, J., Agarwal, S., Mahajan, R., Obermiller, I., & Shayandeh, S. (2012). AppInsight: mobile app performance monitoring in the wild. Broomfield: Symposium on Operating Systems Design and Implementation.

Rodriguez, C. (2019). "Desarrollo de aplicación para plataformas iOS basada en analítica y minería de textos. Madrid: UC3M.

Rodríguez, J. (2017). PROPUESTA NORMATIVA PARA APLICACIONES MÓVILES EN COLOMBIA: DERECHOS Y DEBERES DE ACTORES INVOLUCRADOS EN LA CREACIÓN Y GESTIÓN DE APLICACIONES NATIVAS. Bogotá: Universidad Santo Tomás.

Sharma, Y., & Gupta, S. (2020). A Study of Flutter and React Native for Mobile App Development. Pune: SINHGAD INSTITUTE OF MANAGEMENT AND COMPUTER APPLICATION (SIMCA).

Thomas, P., Delia, L., Corbalan, L., Cáseres, G., Galdamez, N., Cuitiño, A., . . . Pesado, P. (2017). Análisis de Enfoques de Aplicaciones para Dispositivos Móviles. Buenos Aires: Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC).

Thomas, P., Delia, L., L, C., G, C., J, S., F, T., . . . P, P. (2018). Tendencias para el desarrollo de Aplicaciones para Dispositvos Móviles. Buenos Aires: Centro Asociado a la Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC).

Thomas, P., Galdamez, N., Delia, L., Corbalán, L., & Pesado, P. (2015). Dispositivos Móviles: Desarrollo de Aplicaciones Multiplataforma. Buenos Aires: Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC).

Wu, W. (2018). React Native vs Flutter, cross-platform mobile. Helsinki: Metropolia University of Applied Sciences.

Xanthopoulos, S., & Xinogalos, S. (2013). A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications. Tesalónica: University of Thessaloniki.