

DEPARTAMENTO DE CIENCIAS ECONÓMICAS ADMINISTRATIVAS

ASIGNATURA: **DESARROLLO DE APLICACIONES PARA DISPOSITIVOS
MOVILES II**

NOMBRE DEL ESTUDIANTE: **BLADIMIR DE JESÚS ORTIZ RAMIREZ**

NUMERO DE CONTROL: **18920041**

CARRERA: **INGENIERÍA EN INFORMÁTICA**

Grupo: **I9A**

SEMESTRE 9

GRADO 5

SEMESTRE LECTIVO: **AGOSTO - DICIEMBRE 2022**

DOCENTE: **MTI. AMBROSIO CARDOSO JIMENEZ**

TRABAJO

4. DOCUMENTACION Y EJEMPLO DE APLICACIONES MOVILES HIBRIDAS.

LUGAR Y FECHA: **NAZARENO SANTA CRUZ XOXOCOTLÁN A 04 DE
DICIEMBRE DE 2022.**

ÍNDICE

INTRODUCCIÓN	4
DESARROLLO	5
Definición	5
Características	5
Ventajas.....	5
Desventajas	6
Frameworks para Aplicaciones Móviles Híbridas.....	6
Los mas destacados son:.....	7
Ionic	7
Servicios y características.....	7
Ventajas.....	8
Desventajas	8
Estructura típica y descripción de ficheros	9
React Native	15
Características	15
Ventajas.....	15
Desventajas	17
Estructura del proyecto.....	19
PhoneGap.....	21
Características	21
Ventajas.....	22
Desventajas	22
Arquitectura de PhoneGap	22
JQuery Mobile	24
Características	25
Ventajas.....	25
Desventajas	25
Estructura	26
Flutter	32
Características	32
Ventajas.....	33

Desventajas	33
Aplicación Hello World.....	33
Instalación del Framework.....	33
Crear proyecto.....	37
CONCLUSIÓN	39
REFERENCIAS.....	40



INTRODUCCIÓN

Mediante este documento daré a conocer los temas relacionados con el tema principal de la unidad 4, cuyo tema principal es “Framework para el Desarrollo de Aplicaciones Móviles Híbridas”. Los frameworks son plantillas o estructuras previas que facilitan el desarrollo de software y aplicaciones. De este modo, las aplicaciones híbridas podrán ser utilizadas en cualquier móvil, tablet u ordenador independientemente del modelo y la marca. Con una aplicación híbrida podemos disfrutar de numerosas ventajas como la obtención de un buen rendimiento en cualquier plataforma o un ahorro en su desarrollo en comparación con las apps nativas, por ejemplo. Las aplicaciones nativas son aquellas que solo funcionan en un único sistema operativo, así se diferencian de las aplicaciones híbridas. Por lo tanto, estas apps son dependientes al 100% de la plataforma de la cual es nativa. En este sentido, si se desea utilizar una app nativa en sistemas distintos, entonces se deben crear versiones distintas de esta. Cada una de las versiones tiene que estar desarrollada bajo los estrictos guidelines de cada sistema operativo móvil. Las aplicaciones híbridas tienen una composición diferente de las aplicaciones nativas, más parecida a la construcción de las páginas web. La parte que corresponde al código común es la información que comparten entre los diferentes sistemas operativos. Existen diferentes códigos para esta programación y depende de cada desarrollador. El lenguaje nativo es la parte final de estas aplicaciones, y dependen del sistema operativo donde se subirá la app. Usar el lenguaje del sistema operativo brinda una mejor experiencia de usuario y la integración correcta a las funcionalidades del hardware. Una vez que todo el sistema ha sido unificado, se puede lanzar al público desde cualquier tienda de aplicaciones, y puede ser descargada en cualquier equipo sin importar el sistema operativo.

Empecemos

DESARROLLO

Definición

Las aplicaciones híbridas, a diferencia de las nativas, son aquellas capaces de funcionar en distintos sistemas operativos móviles. Entre ellos: Android, iOS y Windows Phone. De esta manera, una misma app puede utilizarse en cualquier smartphone o tablet, indistintamente de su marca o fabricante.

Características

- Las apps híbridas tienen un costo inferior precisamente porque requieren menos inversión de horas en el proyecto.
- Las aplicaciones híbridas comparten un mismo código para todo tipo de dispositivos y sistemas y pueden ser publicadas en las tiendas online de apps (como Google Play, de Android, o App Store, de iOS).
- Las apps nativas aprovechan mejor las capacidades del hardware, ya que son desarrolladas prácticamente ad hoc.
- La carga y la ejecución de las aplicaciones nativas es más rápida que la de las aplicaciones híbridas, por lo que la experiencia de usuario suele ser mucho mejor.

Por lo tanto y depende de lo que busques o de lo que más te convenga, preferirás antes un tipo de app u otra. Sobre todo, desde el punto de vista del desarrollo de la misma. Si buscas algo sencillo, a bajo coste y más ágil y adaptable, el desarrollo de aplicaciones híbridas te conviene más. Si, por el contrario, cuentas con mayor presupuesto y tiempo y buscas un desarrollo más profundo y complejo, quizás te compense más una aplicación nativa.

Ventajas

- Costo: Ya que se mantiene una base de código única para varios sistemas operativos, es menor el costo para desarrollar.
- Mantenimiento: muchos de los proyectos híbridos están escritos en lenguaje HTML simple, lo que hace más fácil sustentarlo.
- Sin versiones: mientras una aplicación nativa saca versiones nuevas cada determinado tiempo, las apps híbridas producen una app solamente.
- Escalabilidad: son mucho más escalables en las plataformas, por lo que no se tiene que escribir la aplicación de nuevo, sino introducir los cambios.
- Tiempo de desarrollo: ya que solo se escribe el código una vez, el tiempo de desarrollo es más rápido en comparación a las aplicaciones nativas. Este tipo de proyectos tarda semanas en lugar de meses. Además, es más breve enviarlo al control de calidad.
- Actualizaciones: Se lanzan actualizaciones constantes para corregir bugs y otros errores que surjan. Estas no necesitan de la aprobación de la store y los cambios se muestra de manera inmediata.

- Requiere de internet: las apps híbridas pueden trabajar sin necesidad de wi-fi. Algunas almacenan los datos en servidores locales, por lo que, en caso de no tener red, es posible acceder a la información sin ningún problema.
- Soporte de plataformas: el mantenimiento se le da a todos los sistemas operativos con una base única de código.
- Llegar a más público: los desarrollos híbridos permiten captar mayor audiencia y ser instaladas por usuarios de cualquier plataforma.
- Acceso inmediato: estas aplicaciones pueden descargarse a través de enlaces, páginas web y la store, lo cual brinda un mayor número de portales para bajarlas.

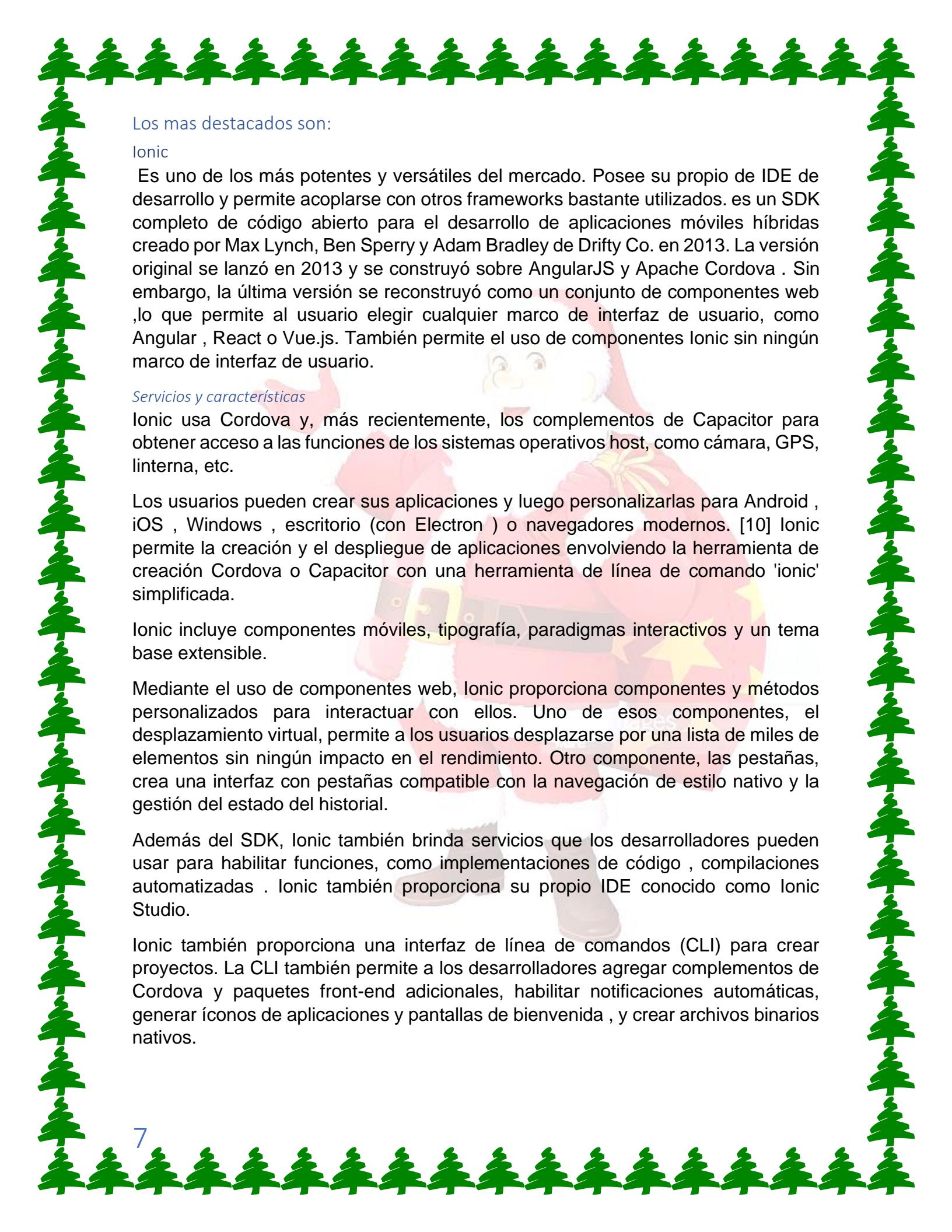
Desventajas

- Rendimiento: es más lento debido a la capa entre el sistema operativo y el código fuente. Esto depende del tamaño de la app, pues cuando es más pequeño, no se nota.
- Funcionalidad: algunas funciones que estarían presentes en los desarrollos nativos no son aprovechadas en los softwares híbridos, por lo que no se tiene el mismo nivel de calidad y estabilidad.
- Experiencia de usuario: la interfaz ofrece una experiencia de usuario deficiente, ya que no puede adaptarse a todas las plataformas en su totalidad.
- Gráficos: las imágenes no tienen una calidad tan buena comparada con las nativas. Y solo mejoran cuando se usan herramientas externas.
- Depuración: Encontrar y limpiar un código resulta ser más difícil por la diferencia entre plataformas. En este sentido, se debe cuidar que no haya errores y hacer pruebas de forma más seguida.
- Fallas: aunque el contenido se revise, pueden existir problemas que solo se detectan cuando el software está trabajando.
- Funcionalidades que demoran en aparecer: cada que los sistemas operativos integran una nueva función, las apps híbridas tardan en añadirlas.

Frameworks para Aplicaciones Móviles Híbridas

Los frameworks son capas de abstracción que adaptan la vista web a la vista de dispositivos móviles. Así, una app web puede verse como una app móvil cuando es usada en un smartphone o tablet. De esta manera, los frameworks permiten que las aplicaciones híbridas (que son apps webs) puedan visualizarse como apps móviles.





Los mas destacados son:

Ionic

Es uno de los más potentes y versátiles del mercado. Posee su propio IDE de desarrollo y permite acoplarse con otros frameworks bastante utilizados. es un SDK completo de código abierto para el desarrollo de aplicaciones móviles híbridas creado por Max Lynch, Ben Sperry y Adam Bradley de Drifty Co. en 2013. La versión original se lanzó en 2013 y se construyó sobre AngularJS y Apache Cordova . Sin embargo, la última versión se reconstruyó como un conjunto de componentes web ,lo que permite al usuario elegir cualquier marco de interfaz de usuario, como Angular , React o Vue.js. También permite el uso de componentes Ionic sin ningún marco de interfaz de usuario.

Servicios y características

Ionic usa Cordova y, más recientemente, los complementos de Capacitor para obtener acceso a las funciones de los sistemas operativos host, como cámara, GPS, interna, etc.

Los usuarios pueden crear sus aplicaciones y luego personalizarlas para Android , iOS , Windows , escritorio (con Electron) o navegadores modernos. [10] Ionic permite la creación y el despliegue de aplicaciones envolviendo la herramienta de creación Cordova o Capacitor con una herramienta de línea de comando 'ionic' simplificada.

Ionic incluye componentes móviles, tipografía, paradigmas interactivos y un tema base extensible.

Mediante el uso de componentes web, Ionic proporciona componentes y métodos personalizados para interactuar con ellos. Uno de esos componentes, el desplazamiento virtual, permite a los usuarios desplazarse por una lista de miles de elementos sin ningún impacto en el rendimiento. Otro componente, las pestañas, crea una interfaz con pestañas compatible con la navegación de estilo nativo y la gestión del estado del historial.

Además del SDK, Ionic también brinda servicios que los desarrolladores pueden usar para habilitar funciones, como implementaciones de código , compilaciones automatizadas . Ionic también proporciona su propio IDE conocido como Ionic Studio.

Ionic también proporciona una interfaz de línea de comandos (CLI) para crear proyectos. La CLI también permite a los desarrolladores agregar complementos de Cordova y paquetes front-end adicionales, habilitar notificaciones automáticas, generar íconos de aplicaciones y pantallas de bienvenida , y crear archivos binarios nativos.

Ventajas

Es fácil de aprender y utilizar

Al basarse en tecnologías web (HTML, CSS y JavaScript), los desarrolladores no tienen que aprender una nueva tecnología para utilizar Ionic.

Numerosas integraciones y plugins

Ionic se integra con los frameworks con los que habitualmente se trabaja, Angular, React y Vue. Además, se integra también con numerosas herramientas y dispone de numerosos plugins.

Más productividad y menos costes

Ionic favorece una mayor productividad de los desarrolladores y reduce los costes de desarrollo de la aplicación. Desarrollar aplicaciones híbridas en un único código propicia un menor tiempo de desarrollo y hace que su mantenimiento y escalado sea más sencillo. El desarrollo de una sola aplicación con un único código para distintas plataformas resulta menos costoso que el desarrollo de una aplicación nativa.

Diseño de interfaces sencillo

Ionic hace más sencillo y rápido el diseño de interfaces de usuario para los desarrolladores. Pueden ir eligiendo elementos UI predeterminados de su librería de componentes en vez de tener que ir codificando uno a uno.

Buena documentación y respaldo de la comunidad

ionic Framework es un proyecto de código abierto, muy bien documentado y con una comunidad muy activa.

Desventajas

Peor rendimiento que las aplicaciones nativas

Si el objetivo principal es el rendimiento de nuestra aplicación, debemos contemplar otras opciones como Xamarin o React, que permiten desarrollar aplicaciones nativas, con un mejor rendimiento que las aplicaciones híbridas.

Dependencia con los plugins

Cada vez que necesitemos acceder a funcionalidad nativa deberemos recurrir a un plugin. Normalmente encontraremos un plugin para implementar la funcionalidad que necesitemos, pero en alguna ocasión muy concreta podemos tener que crearlo nosotros mismos.

Aplicaciones más pesadas que las nativas

Crear nuestra aplicación usando HTML, CSS y JavaScript implica escribir mucho código y agregar librerías, complementos y dependencias que harán que nuestra aplicación sea más pesada que una aplicación nativa.

Estructura típica y descripción de ficheros

La estructura del proyecto sigue los estándares recomendados por Angular.

Las capturas que se muestran en este apartado corresponden a un proyecto creado con el comando “ionic init ‘prueba’ –type=angular”. Básicamente crea un proyecto con estructura Angular llamado “prueba”.

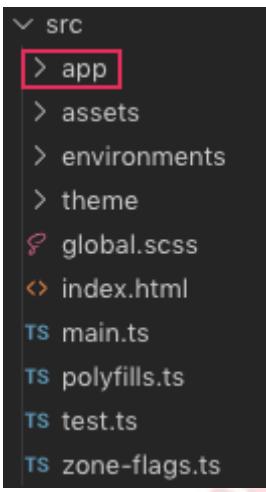
Hay tres tipos de estructura de páginas por defecto y que se pueden inicializar antes de comenzar: tabs (pestañas), side-menu (menú lateral) y blank (en blanco, sin páginas). Por defecto, al utilizar el comando anterior, crea tres páginas de prueba mediante tabs (pestañas).

A continuación se detallan las carpetas y ficheros que se consideran más importantes:

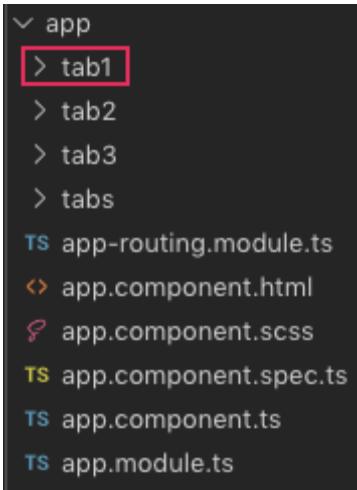
src: Carpeta principal donde se almacena todo el núcleo del código de la aplicación.

```
> e2e  
> node_modules  
  > src  
    .gitignore  
    angular.json  
    browserslist  
    ionic.config.json  
    karma.conf.js  
    package-lock.json  
    package.json  
    tsconfig.app.json  
    tsconfig.json  
    tsconfig.spec.json  
    tslint.json
```

app: Carpeta que incluye e incluirá todos los ficheros de código relativo a las páginas, componentes, servicios, modelos, estilos, etc. Cualquier cambio relativo a funcionamiento o aspecto visual, tendrá lugar en esta carpeta.

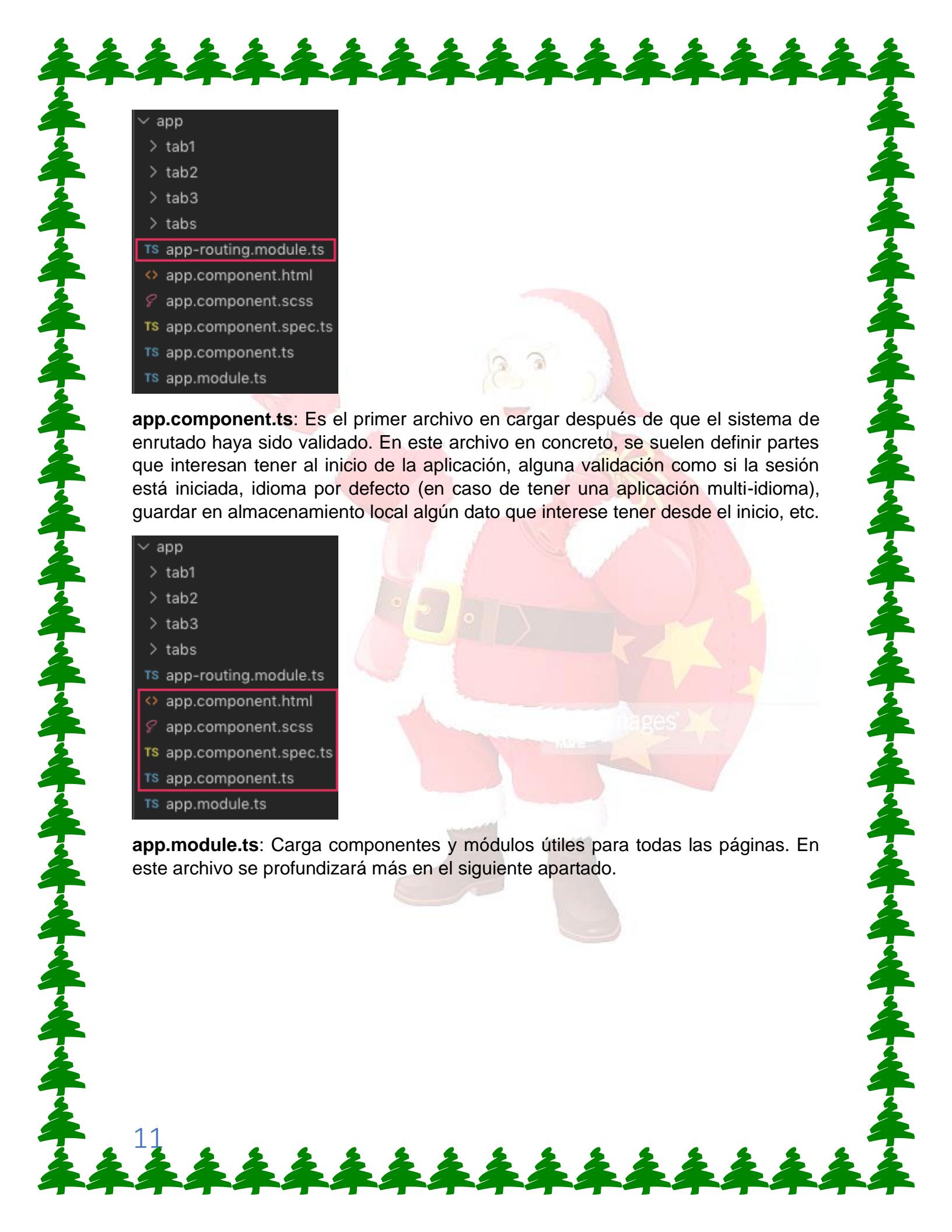


Directarios tabX y tabs: Carpetas correspondientes a las páginas de la aplicación. Cada página creada por defecto se crea en esta sección siguiendo el mismo nivel. Estas páginas han sido creadas por defecto al utilizar el tipo “tabs” a la hora de crear el proyecto. Cada página (o componente en Angular) tiene 4 partes por defecto siguiendo el estándar de Angular: archivo visual (.html), archivo de estilos (.scss), archivo para lógica (.ts) y archivo para testing (.spec.ts). Además, Ionic agrega un archivo módulo (.module.ts) para que cada página pueda ser referenciada en el sistema de rutas.



app-routing.module.ts: Archivo principal de rutas. Desde la versión 4, Ionic sigue el sistema de enrutado de Angular. Para incorporar el “[lazy loading](#)”, lo ideal es referenciar a cada página mediante su invocación al módulo concreto de la página (.module.ts).

Si tuviéramos una estructura de páginas más compleja, con varios niveles, habría que acceder a cada módulo concreto para ver a su vez la organización de rutas de dicho módulo. En el siguiente apartado de “Estructura personalizada” se detalla mejor este punto.



```
< app
  > tab1
  > tab2
  > tab3
  > tabs
  TS app-routing.module.ts
  < app.component.html
  < app.component.scss
  TS app.component.spec.ts
  TS app.component.ts
  TS app.module.ts
```

app.component.ts: Es el primer archivo en cargar después de que el sistema de enrutado haya sido validado. En este archivo en concreto, se suelen definir partes que interesan tener al inicio de la aplicación, alguna validación como si la sesión está iniciada, idioma por defecto (en caso de tener una aplicación multi-idioma), guardar en almacenamiento local algún dato que interese tener desde el inicio, etc.

```
< app
  > tab1
  > tab2
  > tab3
  > tabs
  TS app-routing.module.ts
  < app.component.html
  < app.component.scss
  TS app.component.spec.ts
  TS app.component.ts
  TS app.module.ts
```

app.module.ts: Carga componentes y módulos útiles para todas las páginas. En este archivo se profundizará más en el siguiente apartado.

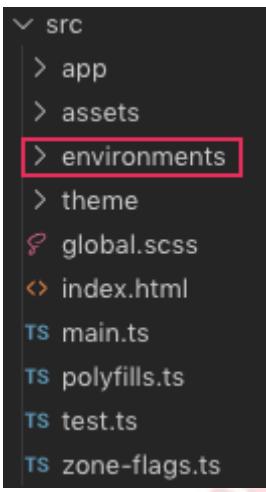
```
< app
  > tab1
  > tab2
  > tab3
  > tabs
  TS app-routing.module.ts
  < app.component.html
  F app.component.scss
  TS app.component.spec.ts
  TS app.component.ts
  TS app.module.ts
```

assets: Carpeta donde se encuentran todos los recursos visuales como los iconos e imágenes estáticas. También se suelen almacenar aquí los ficheros de traducciones con sus literales (en caso de tener una aplicación multi-idioma).

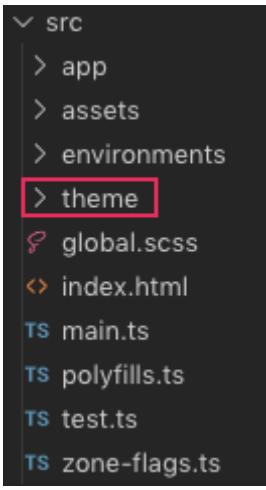
```
< src
  > app
  > assets
  > environments
  > theme
  F global.scss
  < index.html
  TS main.ts
  TS polyfills.ts
  TS test.ts
  TS zone-flags.ts
```

environments: Esta carpeta incluye dos ficheros de variables de configuración global de la aplicación, con el objetivo de especificar que cargue uno u otro según el entorno (desarrollo o producción). Si no hay distinción de entornos, lo ideal es duplicar el contenido de los archivos, por si en un futuro sí es necesario tener los dos entornos disponibles.



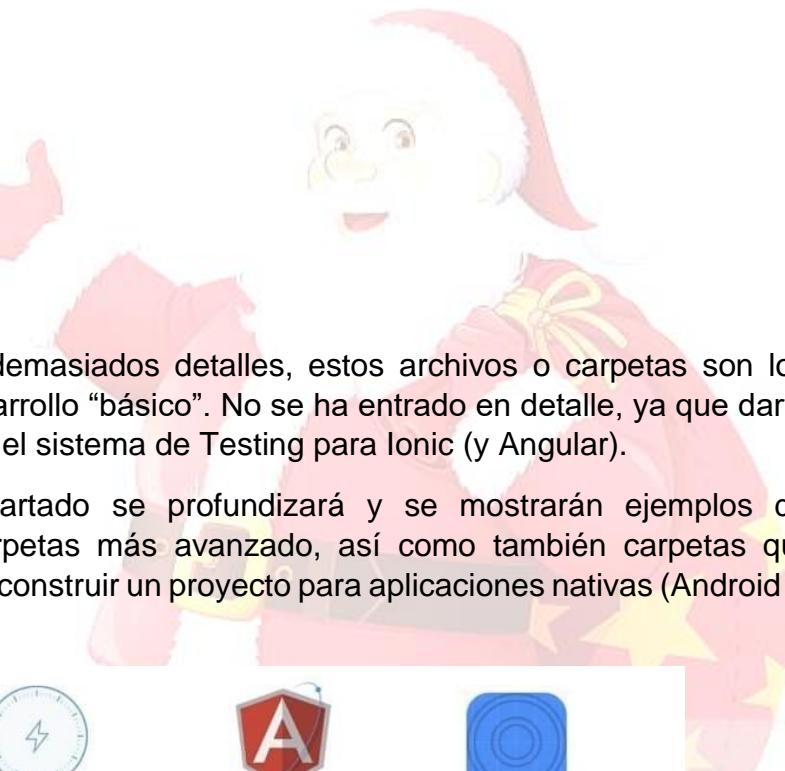
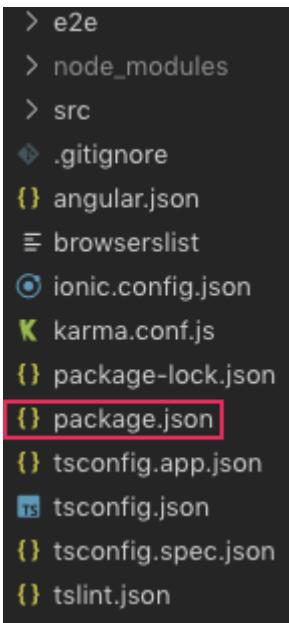


theme: Ficheros de estilos globales. Por defecto, solo incorpora uno.



variables.scss: Aquí se almacenan todas las variables CSS4, tales como colores, diferentes tamaños de letra, altura de las tabs, etc. En el siguiente apartado se mostrarán algunas carpetas que añaden configuración extra a esta carpeta de "theme".

package.json: Este archivo .json está enfocado al gestor de paquetes NPM, que indica todos los paquetes instalados para la aplicación. Se le puede indicar un id, una versión, un nombre, una descripción de aplicación pero esto solo se tiene en cuenta en caso de que la aplicación también fuera una web (PWA), por lo que esta información, aunque recomendable que esté sincronizada con la información del archivo config.xml (se verá en el siguiente apartado), no es requisito imprescindible ya que no se tiene en cuenta para aplicaciones nativas.



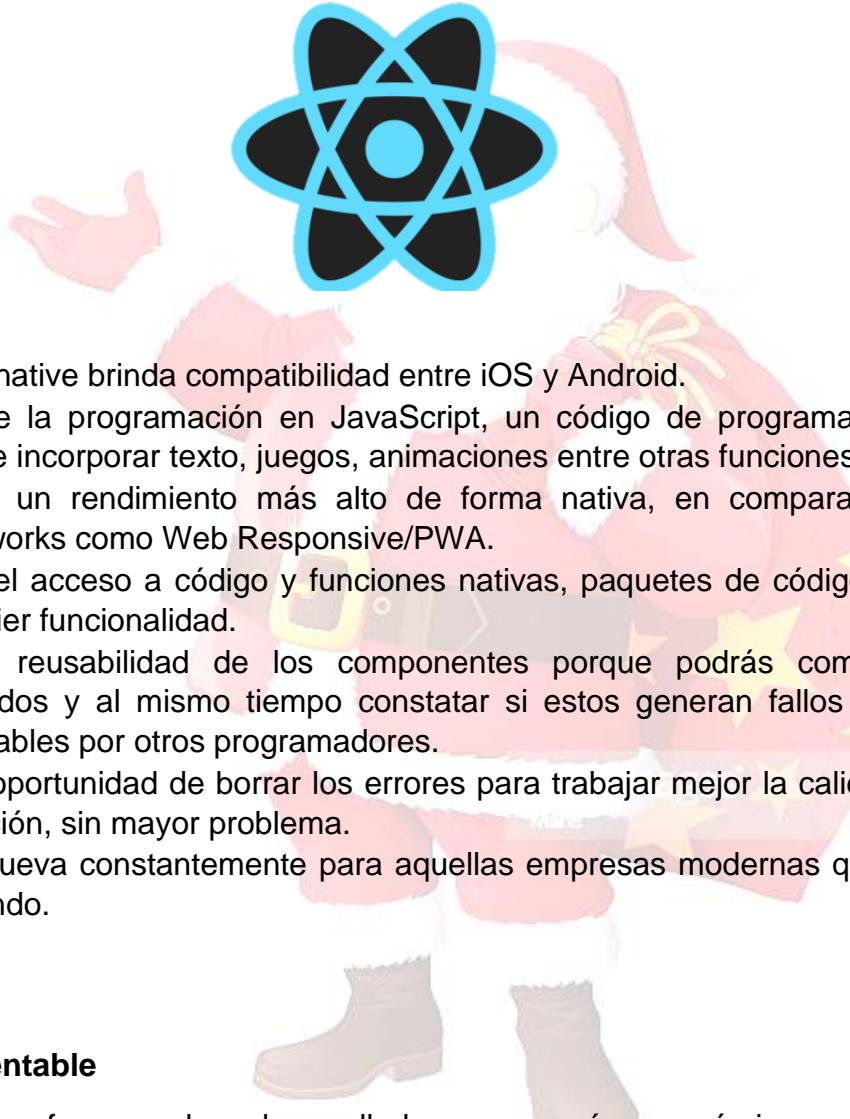
Para no entrar en demasiados detalles, estos archivos o carpetas son los más utilizados en un desarrollo “básico”. No se ha entrado en detalle, ya que daría para otro artículo a parte, el sistema de Testing para Ionic (y Angular).

En el siguiente apartado se profundizará y se mostrarán ejemplos de una organización de carpetas más avanzado, así como también carpetas que son creadas a la hora de construir un proyecto para aplicaciones nativas (Android e iOS), etc.



React Native

Es un framework de código abierto creado por Meta Platforms, Inc.¹ Se utiliza para desarrollar aplicaciones para Android, Android TV,³ iOS, macOS, tvOS, Web, Windows y UWP⁶ al permitir que los desarrolladores usen React con las características nativas de estas plataformas. También se utiliza para desarrollar aplicaciones de realidad virtual con Oculus.



Características

- React native brinda compatibilidad entre iOS y Android.
- Permite la programación en JavaScript, un código de programación que permite incorporar texto, juegos, animaciones entre otras funciones.
- Ofrece un rendimiento más alto de forma nativa, en comparación con frameworks como Web Responsive/PWA.
- Tiene el acceso a código y funciones nativas, paquetes de códigos a casi cualquier funcionalidad.
- Brinda reusabilidad de los componentes porque podrás compilar los resultados y al mismo tiempo constatar si estos generan fallos o si son reutilizables por otros programadores.
- Da la oportunidad de borrar los errores para trabajar mejor la calidad de la aplicación, sin mayor problema.
- Se renueva constantemente para aquellas empresas modernas que vayan surgiendo.

Ventajas

Desarrollo rentable

React Native ofrece a los desarrolladores una vía económica para crear aplicaciones multiplataforma con React Native. En lugar de crear dos aplicaciones diferentes para Android y iOS, el desarrollador puede implementar el mismo código para ambas plataformas. Esto significa reducir los costos de desarrollo en aproximadamente un 50%. Esta característica también hace que las aplicaciones React Native sean más baratas de mantener.

Entrega más rápida de proyectos de aplicaciones

La capacidad de React Native para acelerar el desarrollo de aplicaciones es una de las características más atractivas de la plataforma. Los desarrolladores aprovechan varios componentes listos para usar para crear funciones de aplicaciones más rápido que nunca. React Native requiere menos código en comparación con otras plataformas de desarrollo. Por lo tanto, se necesitan menos esfuerzos para poner en funcionamiento las aplicaciones React Native.

Aprovecha JavaScript

Según el sitio web de Statista, JavaScript es el lenguaje de programación más utilizado en el mundo. Según un estudio de Statista, el 68% de los desarrolladores escriben JavaScript. Esto facilita a los desarrolladores de JavaScript el uso de React Native, ya que el marco está escrito en JavaScript.

Requiere equipos más pequeños

Si bien el desarrollo de React Native aún puede requerir la participación de los desarrolladores principales de iOS y Android, la mayor parte de la tarea de desarrollo está en JavaScript.

Un proyecto de desarrollo de aplicaciones típico necesita equipos de desarrollo de Android y iOS. Tal arreglo puede provocar errores de comunicación e inconsistencias. En algunos casos, el aspecto y las funciones de la aplicación no serían los mismos en ambas plataformas. Sin embargo, el uso de React Native ofrece la ventaja de crear aplicaciones de Android y iOS con un solo equipo. Sería beneficioso contar con un desarrollador nativo experimentado entre los miembros del equipo. Ya no se necesitan dos equipos de desarrollo diferentes.

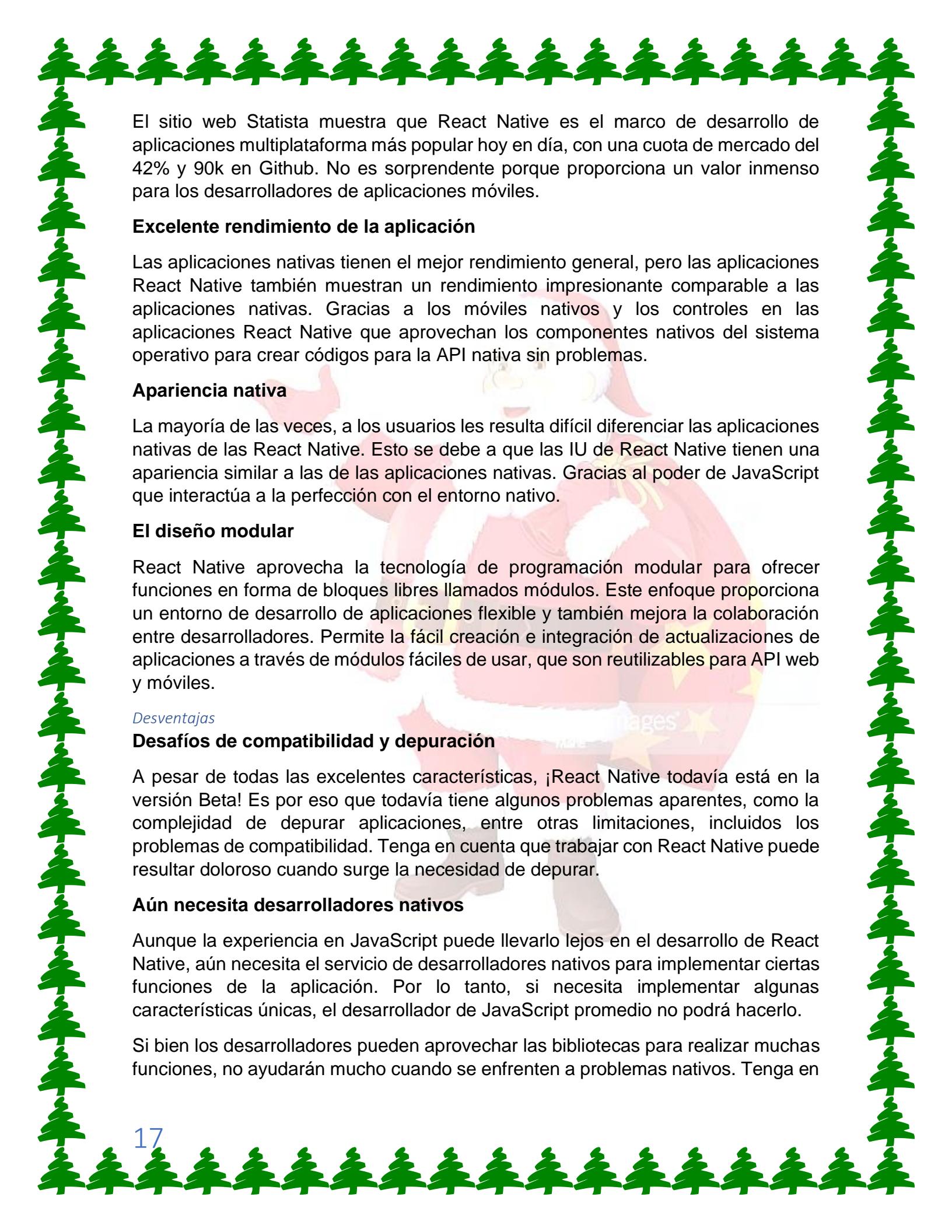
Ventaja del código abierto

Dado el hecho de que React Native es una plataforma de código abierto con licencia del MIT, brinda a los desarrolladores acceso para usar bibliotecas y marcos de forma gratuita. Impone algunas restricciones sobre la reutilización del software, pero también proporciona protección legal para los desarrolladores.

Función de recarga activa (Hot Reloading)

La función de recarga activa es una de las más útiles y atractivas para los desarrolladores. Permite al desarrollador implementar cambios en un código y ver el efecto en la aplicación en tiempo real. Por lo tanto, permite actualizar una aplicación que ya está activa. Todo lo que se requiere es editar el código fuente y la actualización se activa después de guardar el archivo. Por lo tanto, los desarrolladores pueden crear actualizaciones sin un solo segundo de inactividad.

Comunidad de desarrolladores activa



El sitio web Statista muestra que React Native es el marco de desarrollo de aplicaciones multiplataforma más popular hoy en día, con una cuota de mercado del 42% y 90k en Github. No es sorprendente porque proporciona un valor inmenso para los desarrolladores de aplicaciones móviles.

Excelente rendimiento de la aplicación

Las aplicaciones nativas tienen el mejor rendimiento general, pero las aplicaciones React Native también muestran un rendimiento impresionante comparable a las aplicaciones nativas. Gracias a los móviles nativos y los controles en las aplicaciones React Native que aprovechan los componentes nativos del sistema operativo para crear códigos para la API nativa sin problemas.

Apariencia nativa

La mayoría de las veces, a los usuarios les resulta difícil diferenciar las aplicaciones nativas de las React Native. Esto se debe a que las IU de React Native tienen una apariencia similar a las de las aplicaciones nativas. Gracias al poder de JavaScript que interactúa a la perfección con el entorno nativo.

El diseño modular

React Native aprovecha la tecnología de programación modular para ofrecer funciones en forma de bloques libres llamados módulos. Este enfoque proporciona un entorno de desarrollo de aplicaciones flexible y también mejora la colaboración entre desarrolladores. Permite la fácil creación e integración de actualizaciones de aplicaciones a través de módulos fáciles de usar, que son reutilizables para API web y móviles.

Desventajas

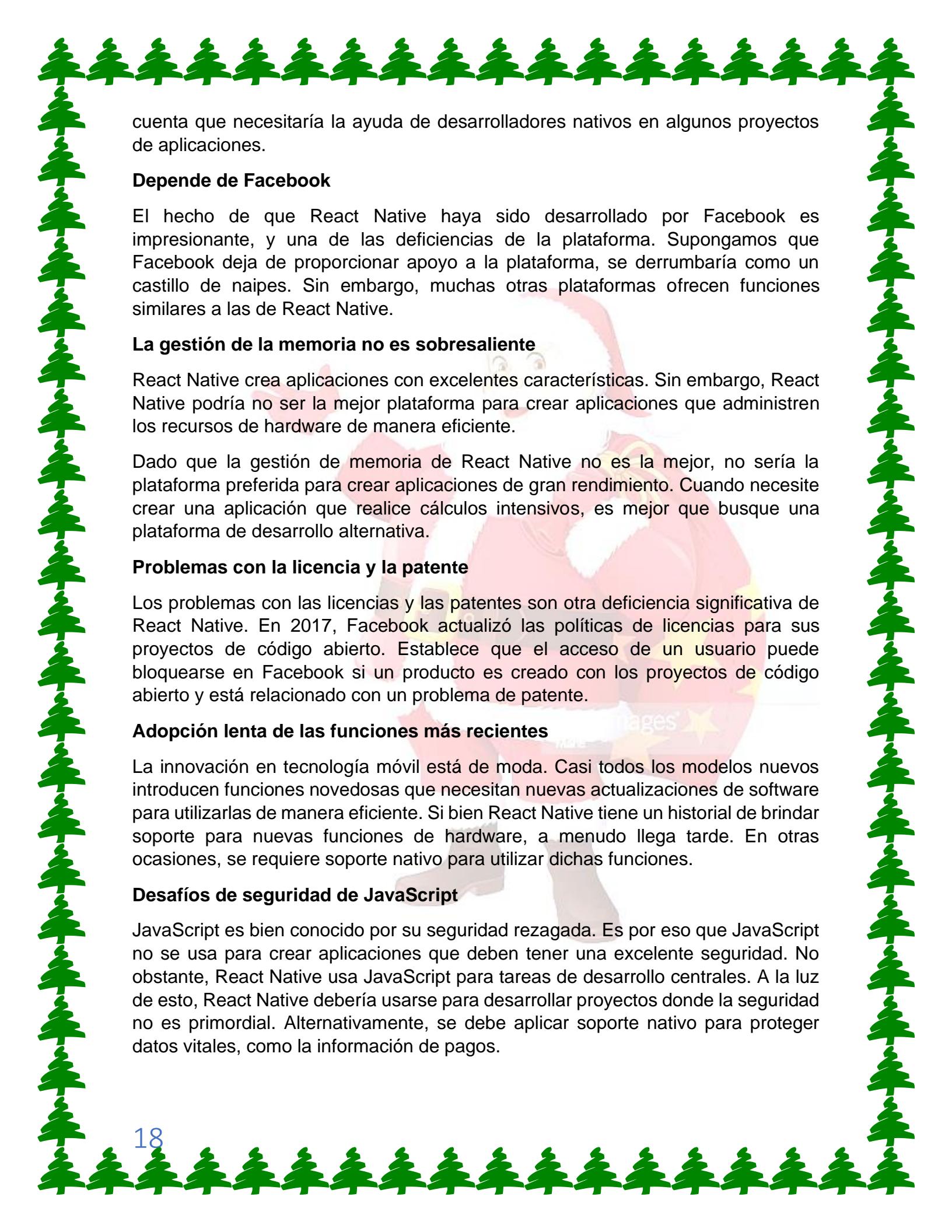
Desafíos de compatibilidad y depuración

A pesar de todas las excelentes características, ¡React Native todavía está en la versión Beta! Es por eso que todavía tiene algunos problemas aparentes, como la complejidad de depurar aplicaciones, entre otras limitaciones, incluidos los problemas de compatibilidad. Tenga en cuenta que trabajar con React Native puede resultar doloroso cuando surge la necesidad de depurar.

Aún necesita desarrolladores nativos

Aunque la experiencia en JavaScript puede llevarlo lejos en el desarrollo de React Native, aún necesita el servicio de desarrolladores nativos para implementar ciertas funciones de la aplicación. Por lo tanto, si necesita implementar algunas características únicas, el desarrollador de JavaScript promedio no podrá hacerlo.

Si bien los desarrolladores pueden aprovechar las bibliotecas para realizar muchas funciones, no ayudarán mucho cuando se enfrenten a problemas nativos. Tenga en



cuenta que necesitaría la ayuda de desarrolladores nativos en algunos proyectos de aplicaciones.

Depende de Facebook

El hecho de que React Native haya sido desarrollado por Facebook es impresionante, y una de las deficiencias de la plataforma. Supongamos que Facebook deja de proporcionar apoyo a la plataforma, se derrumbaría como un castillo de naipes. Sin embargo, muchas otras plataformas ofrecen funciones similares a las de React Native.

La gestión de la memoria no es sobresaliente

React Native crea aplicaciones con excelentes características. Sin embargo, React Native podría no ser la mejor plataforma para crear aplicaciones que administren los recursos de hardware de manera eficiente.

Dado que la gestión de memoria de React Native no es la mejor, no sería la plataforma preferida para crear aplicaciones de gran rendimiento. Cuando necesite crear una aplicación que realice cálculos intensivos, es mejor que busque una plataforma de desarrollo alternativa.

Problemas con la licencia y la patente

Los problemas con las licencias y las patentes son otra deficiencia significativa de React Native. En 2017, Facebook actualizó las políticas de licencias para sus proyectos de código abierto. Establece que el acceso de un usuario puede bloquearse en Facebook si un producto es creado con los proyectos de código abierto y está relacionado con un problema de patente.

Adopción lenta de las funciones más recientes

La innovación en tecnología móvil está de moda. Casi todos los modelos nuevos introducen funciones novedosas que necesitan nuevas actualizaciones de software para utilizarlas de manera eficiente. Si bien React Native tiene un historial de brindar soporte para nuevas funciones de hardware, a menudo llega tarde. En otras ocasiones, se requiere soporte nativo para utilizar dichas funciones.

Desafíos de seguridad de JavaScript

JavaScript es bien conocido por su seguridad rezagada. Es por eso que JavaScript no se usa para crear aplicaciones que deben tener una excelente seguridad. No obstante, React Native usa JavaScript para tareas de desarrollo centrales. A la luz de esto, React Native debería usarse para desarrollar proyectos donde la seguridad no es primordial. Alternativamente, se debe aplicar soporte nativo para proteger datos vitales, como la información de pagos.

Complicaciones de rendimiento vs. de características

React Native es excelente para crear aplicaciones simples con imágenes atractivas. Sin embargo, lograr un rendimiento excelente se convierte en un desafío cuando necesita agregar funciones complejas. Por lo tanto, puede que tenga que conformarse con uno de los dos.

Problemas complejos relacionados con dispositivos

Dado que los códigos en React Native no se generan de forma nativa, puede resultar complicado resolver los desafíos específicos del dispositivo. Tales problemas conducen a desafíos complicados sin una forma viable de resolverlos.

Componentes de desarrollo de terceros

Los componentes necesarios para crear algunos tipos de aplicaciones faltan en React Native. Por lo tanto, debe utilizar recursos de terceros para agregar estos componentes a su aplicación. Si bien el uso de componentes de terceros es aceptable, puede quedarse atascado. En esos momentos, tendrá que encontrar una manera de resolver el problema usted mismo. Por eso es complicado utilizar un componente de terceros en proyectos de desarrollo de aplicaciones.

Estructura del proyecto

Si hemos creado nuestro proyecto utilizando Expo CLI, la estructura de carpetas de nuestro proyecto debería ser similar a la siguiente:

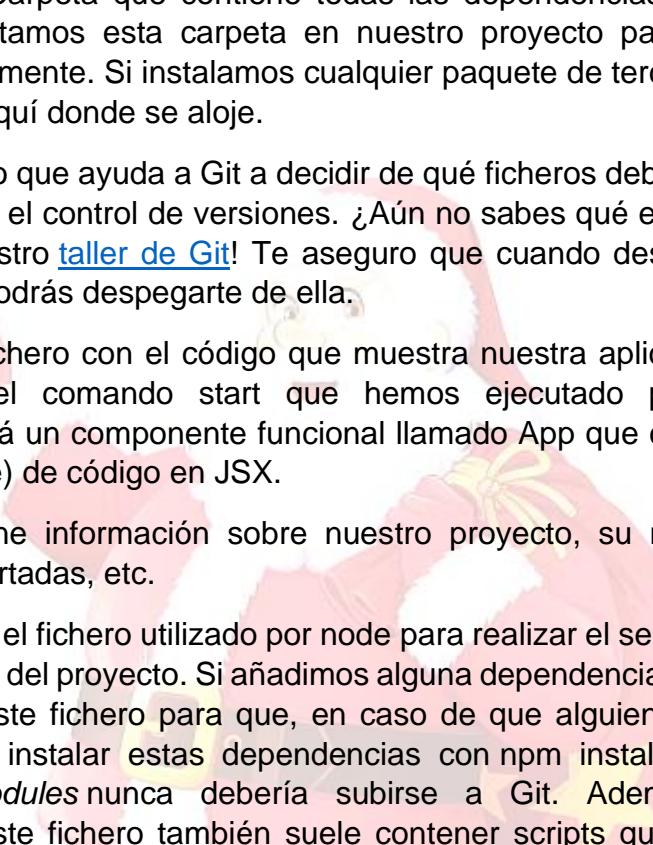


```
✓ FIRS...  📂  📄  ⏪  🗃
  > .expo-shared
  > assets
  > node_modules
  ♦ .gitignore
  JS App.js
  { app.json
  B babel.config.js
  { package.json
  🎁 yarn.lock
```

Tendremos los siguientes ficheros y carpetas:

- .expo-shared: Carpeta con ficheros de configuración de expo, que no nos interesa tocar de momento.

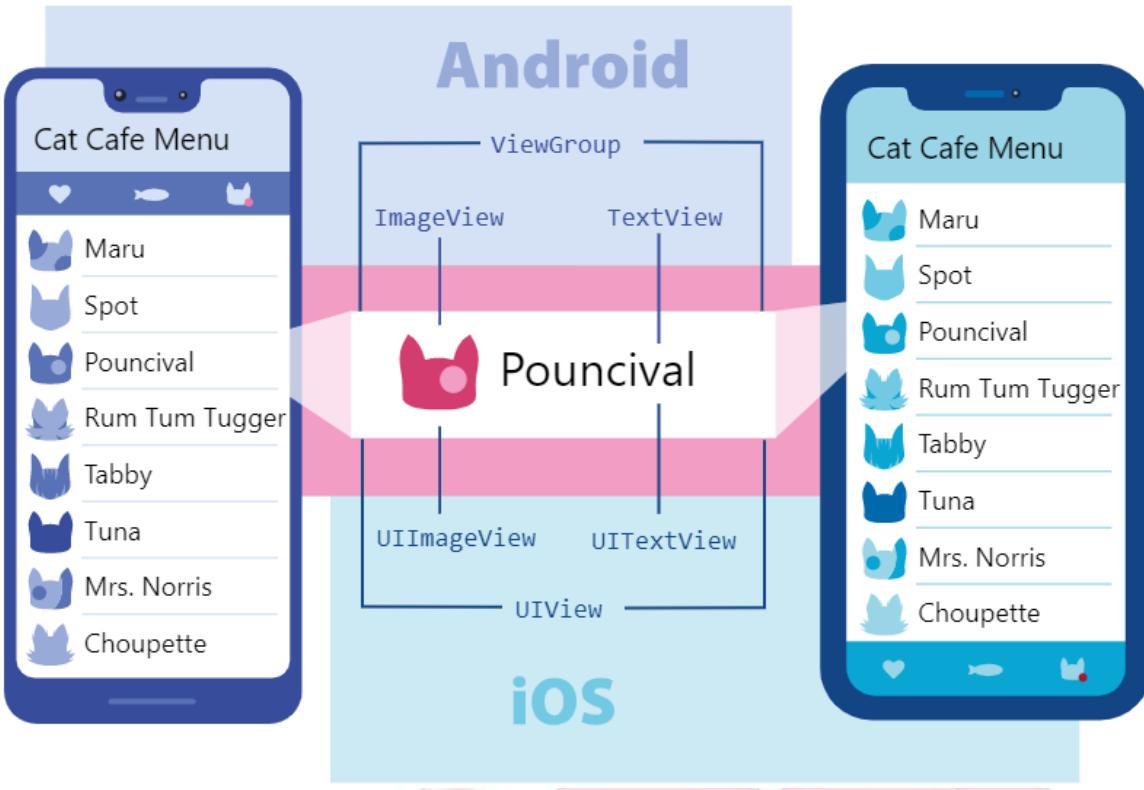
- assets: Carpeta que contendrá cualquier tipo de recurso (imágenes, audios, fuentes, ...) que necesitemos utilizar en nuestra aplicación.
- node_modules: Carpeta que contiene todas las dependencias de nuestro proyecto. Necesitamos esta carpeta en nuestro proyecto para que todo funcione correctamente. Si instalamos cualquier paquete de terceros (que lo haremos), será aquí donde se aloje.
- .gitignore: Fichero que ayuda a Git a decidir de qué ficheros debe realizar un seguimiento para el control de versiones. ¿Aún no sabes qué es Git? ¡Echa un vistazo a nuestro [taller de Git](#)! Te aseguro que cuando descubras esta herramienta no podrás despegarte de ella.
- App.js: Este es el fichero con el código que muestra nuestra aplicación al ser levantada con el comando start que hemos ejecutado previamente. Inicialmente habrá un componente funcional llamado App que contiene una plantilla (template) de código en JSX.
- app.json: Contiene información sobre nuestro proyecto, su nombre, sus plataformas soportadas, etc.
- package.json: Es el fichero utilizado por node para realizar el seguimiento de las dependencias del proyecto. Si añadimos alguna dependencia al proyecto, se reflejará en este fichero para que, en caso de que alguien se clone el proyecto, pueda instalar estas dependencias con npm install, ya que la carpeta node_modules nunca debería subirse a Git. Además de las dependencias, este fichero también suele contener scripts que facilitan la ejecución de algunos comandos que suelen ejecutarse frecuentemente.



A screenshot of a file explorer window showing the directory structure of a React Native project named "octuweb". The left pane shows files and folders like __tests__, android, ios, node_modules, .babelrc, .buckconfig, .flowconfig, .gitignore, .watchmanconfig, index.android.js, index.ios.js, and package.json. The right pane shows the content of index.ios.js:

```
index.ios.js
1 /**
2  * Sample React Native App
3  * https://github.com/facebook/react-native
4  * @flow
5 */
6
7 import React, { Component } from 'react';
8 import {
9   AppRegistry,
10  StyleSheet,
11  Text,
12  View
13 } from 'react-native';
14
15 export default class octuweb extends Component {
16   render() {
17     return (

```

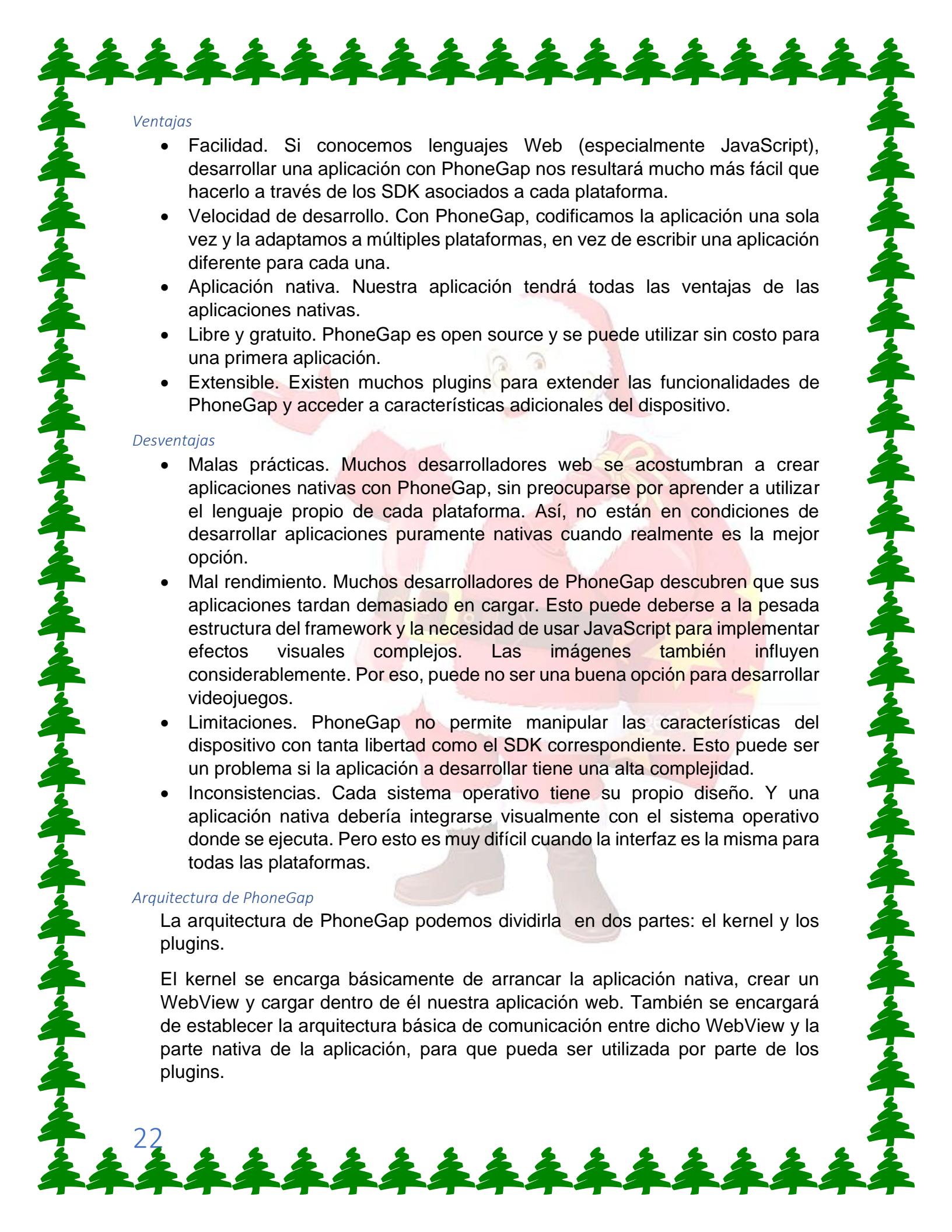


PhoneGap

Es un framework para el desarrollo de aplicaciones móviles producido por Nitobi, y comprado posteriormente por Adobe Systems. Principalmente, PhoneGap permite a los programadores desarrollar aplicaciones para dispositivos móviles utilizando herramientas genéricas tales como JavaScript, HTML5 y CSS3. Las aplicaciones resultantes son híbridas, es decir que no son realmente aplicaciones nativas al dispositivo (ya que el renderizado se realiza mediante vistas web y no con interfaces gráficas específicas de cada sistema), pero no se tratan tampoco de aplicaciones web (teniendo en cuenta que son aplicaciones que son empaquetadas para poder ser desplegadas en el dispositivo incluso trabajando con el API del sistema nativo).

Características

- Permite crear actualmente aplicaciones móviles para: iPhone, Android, Windows Phone, Blackberry, Blackberry 10, webOS, Symbian y Bada.
- Las aplicaciones creadas con PhoneGap sólo pueden nutrirse de HTML, CSS y Javascript. Si requieren lógica generada por otros lenguajes de programación, deberán conseguirla de un backend a través de APIs o webservices
- Ofrece un servicio en la nube llamado PhoneGap Build que permite construir rápidamente apps móviles y compilarlas con facilidad sin necesidad de SDKs, compiladores o hardware específico.
- Tiene una licencia Apache 2.0



Ventajas

- Facilidad. Si conocemos lenguajes Web (especialmente JavaScript), desarrollar una aplicación con PhoneGap nos resultará mucho más fácil que hacerlo a través de los SDK asociados a cada plataforma.
- Velocidad de desarrollo. Con PhoneGap, codificamos la aplicación una sola vez y la adaptamos a múltiples plataformas, en vez de escribir una aplicación diferente para cada una.
- Aplicación nativa. Nuestra aplicación tendrá todas las ventajas de las aplicaciones nativas.
- Libre y gratuito. PhoneGap es open source y se puede utilizar sin costo para una primera aplicación.
- Extensible. Existen muchos plugins para extender las funcionalidades de PhoneGap y acceder a características adicionales del dispositivo.

Desventajas

- Malas prácticas. Muchos desarrolladores web se acostumbran a crear aplicaciones nativas con PhoneGap, sin preocuparse por aprender a utilizar el lenguaje propio de cada plataforma. Así, no están en condiciones de desarrollar aplicaciones puramente nativas cuando realmente es la mejor opción.
- Mal rendimiento. Muchos desarrolladores de PhoneGap descubren que sus aplicaciones tardan demasiado en cargar. Esto puede deberse a la pesada estructura del framework y la necesidad de usar JavaScript para implementar efectos visuales complejos. Las imágenes también influyen considerablemente. Por eso, puede no ser una buena opción para desarrollar videojuegos.
- Limitaciones. PhoneGap no permite manipular las características del dispositivo con tanta libertad como el SDK correspondiente. Esto puede ser un problema si la aplicación a desarrollar tiene una alta complejidad.
- Inconsistencias. Cada sistema operativo tiene su propio diseño. Y una aplicación nativa debería integrarse visualmente con el sistema operativo donde se ejecuta. Pero esto es muy difícil cuando la interfaz es la misma para todas las plataformas.

Arquitectura de PhoneGap

La arquitectura de PhoneGap podemos dividirla en dos partes: el kernel y los plugins.

El kernel se encarga básicamente de arrancar la aplicación nativa, crear un WebView y cargar dentro de él nuestra aplicación web. También se encargará de establecer la arquitectura básica de comunicación entre dicho WebView y la parte nativa de la aplicación, para que pueda ser utilizada por parte de los plugins.



Por otra parte, los plugins son componentes que podremos añadir para hacer uso de funcionalidades, servicios y APIs nativas que no son accesibles de forma estándar por un navegador web, expuestas ante nuestra aplicación como funciones JavaScript que podremos usar desde nuestra aplicación web.

Entre los plugins oficiales (mantenidos por los desarrolladores de PhoneGap) podemos encontrar los siguientes:

- Battery Status: Monitorización y consulta de estado de la batería
- Camera: acceso a la cámara y álbum de fotos del dispositivo
- Contacts: acceso, creación y modificación de contactos existentes en el dispositivo
- Device: Información referente al dispositivo (plataforma, versión de S.O., etc.)
- Device Motion: acceso al acelerómetro del dispositivo
- Device Orientation: acceso a la brújula del dispositivo
- Dialogs: mostrar notificaciones en forma de diálogos
- FileSystem: acceso al sistema de ficheros del dispositivo modelado a través del API File de HTML5
- File Transfer: soporte para descargas y subidas de ficheros
- Geolocation: consulta de posición GPS
- Globalization: soporte para representar distintos objetos según su localización
- InAppBrowser: abrir URLs dentro de la propia aplicación sin salir al navegador principal del dispositivo.
- Media: Grabación y reproducción de archivos de audio
- Media Capture: captura de ficheros multimedia mediante las aplicaciones de captura del dispositivo
- Network Information: información sobre el estado de la conexión del dispositivo, si está online u offline o si está mediante Wifi, 3G, etc.
- Splashscreen: control fino sobre el comportamiento de la pantalla de inicio SplashScreen.
- Vibration: hacer que el dispositivo vibre.

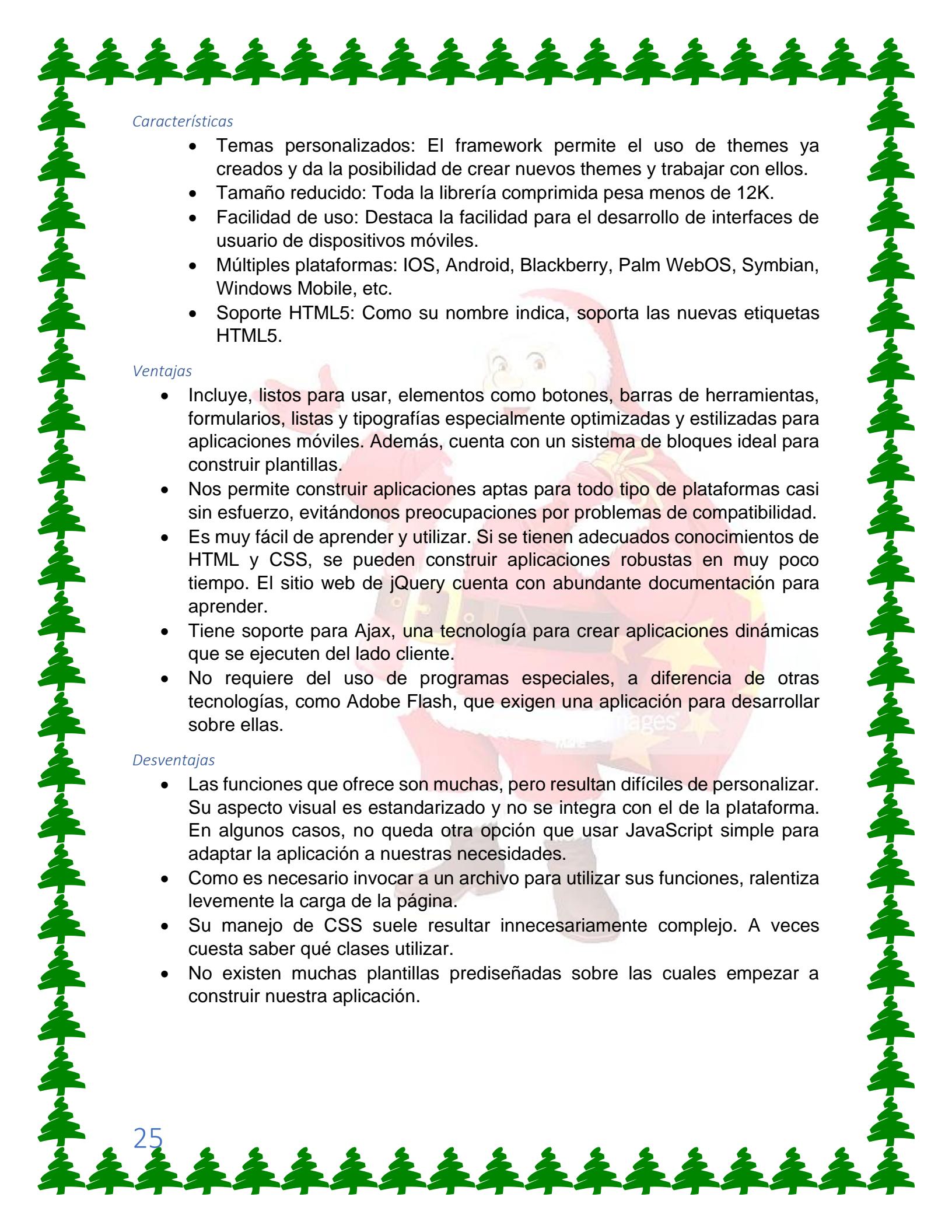
También existen numerosos plugins creados por la comunidad que podremos buscar desde <https://plugins.cordova.io>, y en última instancia siempre tendremos

la posibilidad de crearnos los nuestros propios, aunque en ese caso sí necesitaremos disponer de conocimientos sobre desarrollo nativo en cada plataforma.



JQuery Mobile

Es un Framework optimizado para dispositivos táctiles (también conocido como Framework móvil) que está siendo desarrollado actualmente por el equipo de proyectos de jQuery. El desarrollo se centra en la creación de un Framework compatible con la gran variedad de teléfonos inteligentes y tabletas, algo necesario en el creciente y heterogéneo mercado de tabletas y teléfonos inteligentes. El Framework de jQuery Mobile es compatible con otros frameworks móviles y plataformas como PhoneGap y Worklight entre otros. Todos los proyectos que utilizan jQuery Mobile utilizan más o menos el mismo código. Es importante enlazar las librerías JavaScript de jQuery y jQuery Mobile, así como sus hojas de estilo (estos archivos pueden descargarse y utilizarse localmente, pero se recomienda enlazarlos desde el CDN de jQuery).



Características

- Temas personalizados: El framework permite el uso de themes ya creados y da la posibilidad de crear nuevos themes y trabajar con ellos.
- Tamaño reducido: Toda la librería comprimida pesa menos de 12K.
- Facilidad de uso: Destaca la facilidad para el desarrollo de interfaces de usuario de dispositivos móviles.
- Múltiples plataformas: IOS, Android, Blackberry, Palm WebOS, Symbian, Windows Mobile, etc.
- Soporte HTML5: Como su nombre indica, soporta las nuevas etiquetas HTML5.

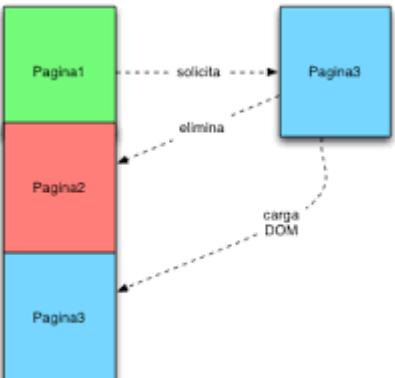
Ventajas

- Incluye, listos para usar, elementos como botones, barras de herramientas, formularios, listas y tipografías especialmente optimizadas y estilizadas para aplicaciones móviles. Además, cuenta con un sistema de bloques ideal para construir plantillas.
- Nos permite construir aplicaciones aptas para todo tipo de plataformas casi sin esfuerzo, evitándonos preocupaciones por problemas de compatibilidad.
- Es muy fácil de aprender y utilizar. Si se tienen adecuados conocimientos de HTML y CSS, se pueden construir aplicaciones robustas en muy poco tiempo. El sitio web de jQuery cuenta con abundante documentación para aprender.
- Tiene soporte para Ajax, una tecnología para crear aplicaciones dinámicas que se ejecuten del lado cliente.
- No requiere del uso de programas especiales, a diferencia de otras tecnologías, como Adobe Flash, que exigen una aplicación para desarrollar sobre ellas.

Desventajas

- Las funciones que ofrece son muchas, pero resultan difíciles de personalizar. Su aspecto visual es estandarizado y no se integra con el de la plataforma. En algunos casos, no queda otra opción que usar JavaScript simple para adaptar la aplicación a nuestras necesidades.
- Como es necesario invocar a un archivo para utilizar sus funciones, ralentiza ligeramente la carga de la página.
- Su manejo de CSS suele resultar innecesariamente complejo. A veces cuesta saber qué clases utilizar.
- No existen muchas plantillas prediseñadas sobre las cuales empezar a construir nuestra aplicación.

Estructura



ada día desarrollamos mas en entornos móviles y uno de los frameworks de referencia a la hora de abordar proyectos de este tipo **es JQuery Mobile**. Se trata de un framework construido por la gente de JQuery que facilita sobremanera el desarrollo de aplicaciones orientadas a móviles. **El framework se apoya en data-atributos de html5 para generar una presentación orientada al móvil.**

```
1 <html>
2 <head>
3   <link rel="stylesheet"
4     href="http://code.jquery.com/mobile/1.4.2/jquery.mobile-1.4.2.min.css" />
5   <script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
6   <script
7     src="http://code.jquery.com/mobile/1.4.2/jquery.mobile-1.4.2.min.js"></script>
8 </head>
9 <body>
10
11<div data-role="page">
12<div data-role="header">
13<h6>Cabecera de la pagina</h6>
14</div>
15
16<div data-role="content"><a
```

```
17 href="pagina2.html" data-role="button">
```

```
18 Ir a la pagina 2</a>
```

```
19
```

```
20</div>
```

```
21
```

```
22<div data-role="footer">
```

```
23<h6>Pie de la pagina</h6>
```

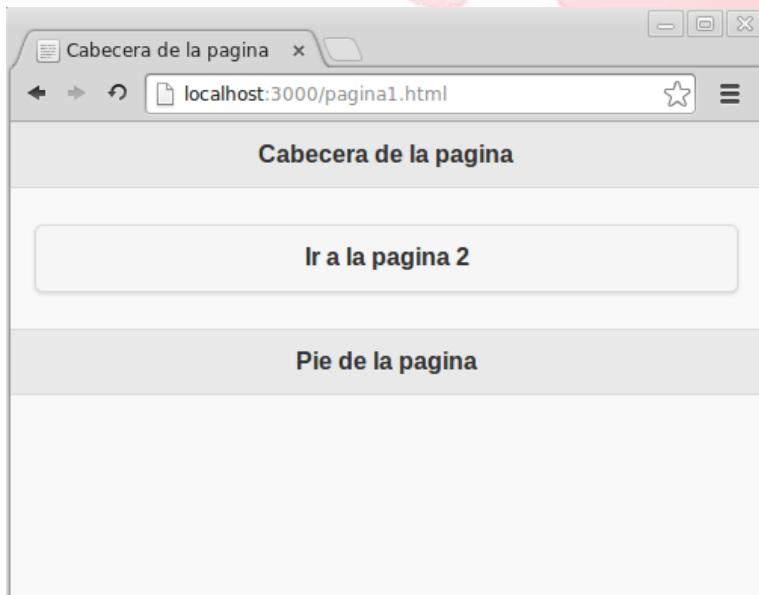
```
24</div>
```

```
25</body>
```

```
26</html>
```

JQuery Mobile Data-Atributos

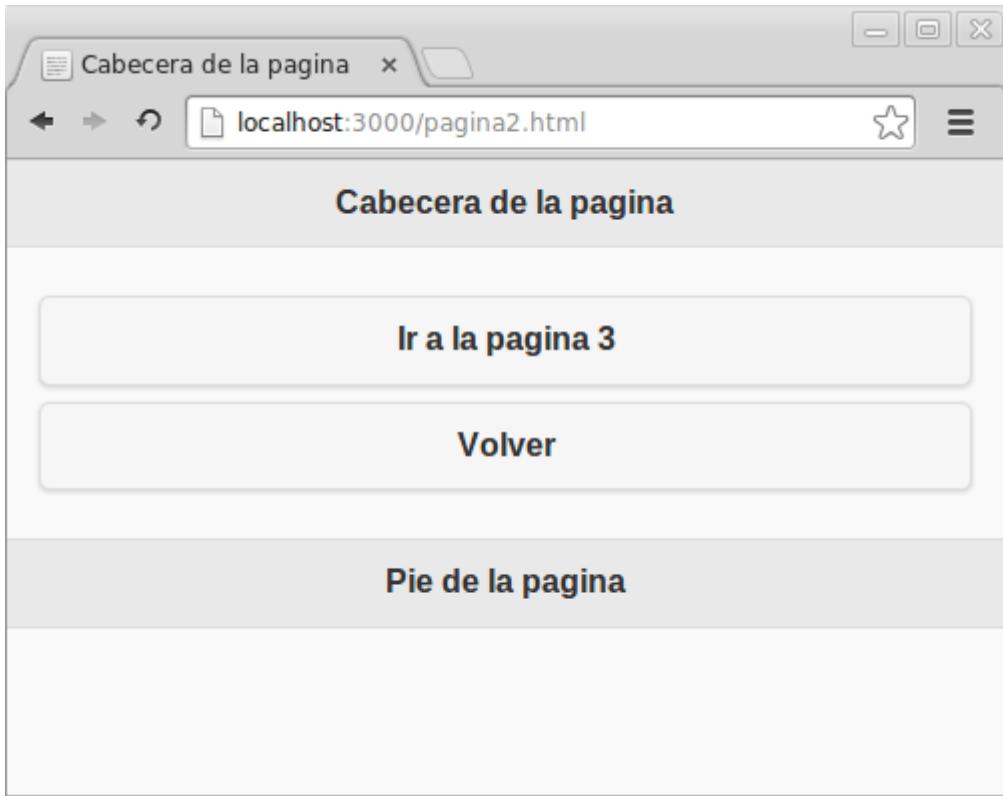
JQuery Mobile utiliza el atributo data-role para asignar diferentes roles a las diferentes partes de la página. En este caso tenemos los roles de página, cabecera, contenido y pie. Si cargamos la página en un navegador veremos lo siguiente .



El código de la página dos es muy similar al de la primera con la única diferencia que tiene un botón de volver que permite cargar la página anterior:

```
1 <html>
2 <head>
3 <link rel="stylesheet"
4 href="http://code.jquery.com/mobile/1.4.2/jquery.mobile-1.4.2.min.css" />
5 <script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
6 <script
7 src="http://code.jquery.com/mobile/1.4.2/jquery.mobile-1.4.2.min.js"></script>
8 </head>
9 <body>
10
11<div data-role="page">
12<div data-role="header">
13<h6>Cabecera de la pagina</h6>
14</div>
15
16<div data-role="content">
17<a href="pagina3.html" data-role="button">Ir a la
18pagina 3</a> <a href="#" data-role="button" data-rel="back">Volver</a></div>
19
20<div data-role="footer">
21<h6>Pie de la pagina</h6>
22</div>
23</body>
24</html>
```

El navegador nos muestra lo siguiente :

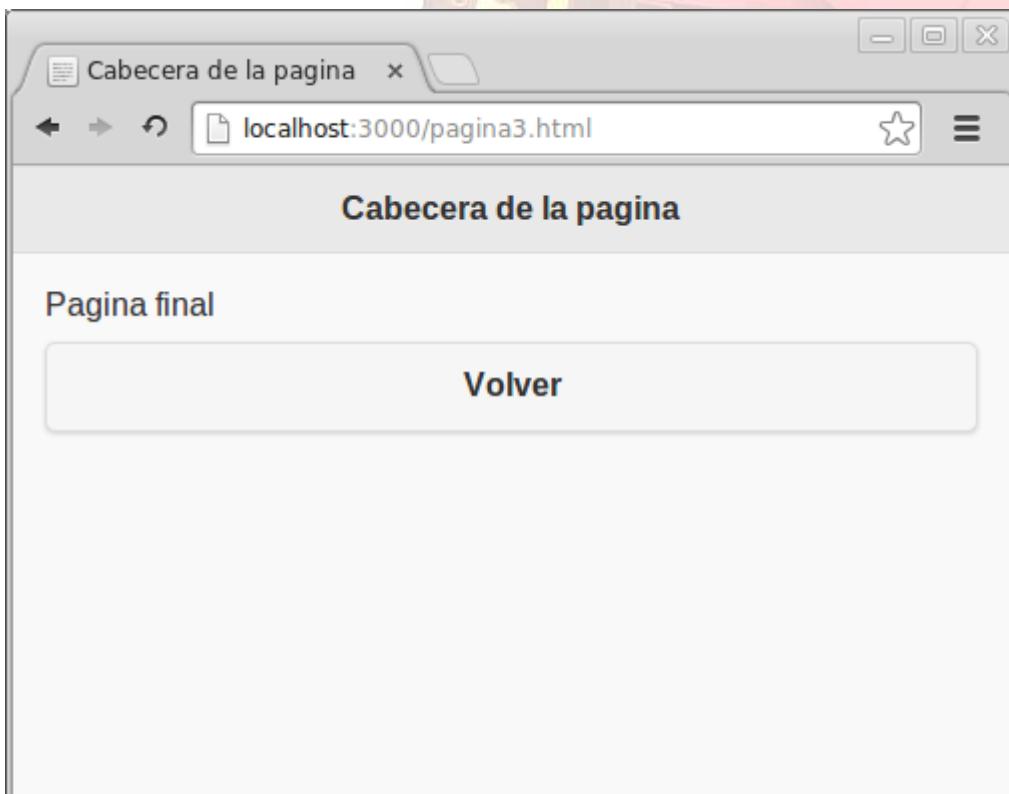


La página 3 es prácticamente idéntica a la página 2 salvo que cambiamos el mensaje.

```
1 <html>
2 <head>
3 <link rel="stylesheet"
4 href="http://code.jquery.com/mobile/1.4.2/jquery.mobile-1.4.2.min.css" />
5 <script src="http://code.jquery.com/jquery-1.9.1.min.js"></script>
6 <script
7 src="http://code.jquery.com/mobile/1.4.2/jquery.mobile-1.4.2.min.js"></script>
8 </head>
9 <body>
10
11<div data-role="page">
12<div data-role="header">
```

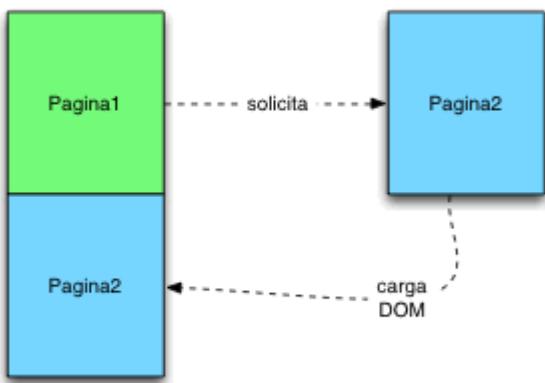
```
13<h6>Cabecera de la pagina</h6>
14</div>
15
16<div data-role="content">
17Pagina final
18<a href="#" data-role="button" data-rel="back">Volver</a></div>
19</div>
20
21<div data-role="footer">
22<h6>Pie de la pagina</h6>
23</div>
24</body>
25</html>
```

El navegador muestra :

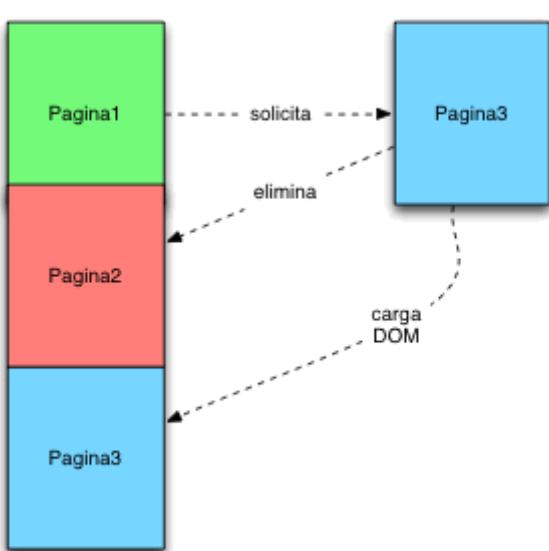


Todo lo que hemos hecho es muy muy sencillo y podemos ver la potencia que tiene el framework

Lamentablemente las cosas no son tan sencillas como parecen de entrada . Si nos fijamos en como **JQuery Mobile ha cargado cada una de las páginas veremos que ha elegido un modelo de SPA** . Esto quiere decir lo siguiente .Cuando solicitamos la segunda página desde la primera ,se muestra en el navegador la segunda página pero no hemos cambiado de página realmente sino que se ha cargado via AJAX el contenido de la segunda página en la primera . Hecho esto JQuery Mobile ha escondido el contenido de la primera página y ha mostrado únicamente el de la segunda.



Algo similar sucederá cuando desde la segunda página solicitemos la tercera .En este caso se descargará la página2 de memoria y se cargará la 3.



Aunque JQuery Mobile parezca un framework muy sencillo de manejar cuando realizamos el ejemplo de hola mundo ,es quizas uno de los frameworks que mas cuesta a los desarrolladores entender su funcionamiento real. Si en algún momento tenemos que trabajar con el ... hay que tomarselo con calma,ya que convertirse en experto no es trivial.

Flutter

Es un SDK de código fuente abierto de desarrollo de aplicaciones móviles creado por Google. Suele usarse para desarrollar interfaces de usuario para aplicaciones en Android, iOS y Web así como método primario para crear aplicaciones para Google Fuchsia. En el último año, ha sufrido un crecimiento muy grande en cuanto a su popularidad. Eso se debe a su velocidad de desarrollo, experiencia nativa y renderización de la interface. El 3 de marzo de 2021, en el evento virtual llamado "Flutter Engage", Google lanzó Flutter 2. Este fue el cambio oficial más grande que tuvo el SDK.



Características

Motor Propio Renderizado Basado En SKIA

Una de las principales características de Flutter es que no utiliza los widgets que ya vienen en los móviles como lo hacen React Native o NativeScript, Flutter ha ido más allá y ha creado un motor propio renderizado basado en Skia por lo que no utiliza Web Wiew ni los OEM Widgets de los dispositivos.

Gran Librería De Widgets

Este es un apartado en el que Flutter ha trabajado bastante bien, lo primero que se observa cuando se empieza ha trabajar con esta herramienta, es que uno tiene al alcance de su mano todos los controles que necesita.

También te puede interesar: [Beneficios de tener app móviles en negocios pequeños](#)

Por la forma en la que está distribuido Flutter que todo es un Widget, trabajar resulta muy fácil para el usuario.

Hot Reload

Esta es otra de las características destacables de Flutter, esta funcionalidad permite al usuario hacer cambios en el programa, se actualiza y ya se ven los cambios en la aplicaciones móviles, esto adelanta muchísimo el trabajo.

El término hot reload, hace referencia a la rápida recarga en caliente, esta funcionalidad permite hacer un cambio en una app en Flutter mientras se está ejecutando, recargando el código de la aplicación que ha cambiado y dejando que continúe desde donde la dejaste, todo ello en menos de un segundo.

Si tu app encuentra un error, en la mayoría de los casos puedes corregirlo para luego continuar como si el error nunca hubiera pasado, el proceso es rápido incluso cuando tienes que hacer una recarga completa.

Ventajas

- Una única base de código para las principales plataformas de destino.
- Lenguaje de programación Dart fácil de aprender.
- El paradigma todo es un widget ofrece numerosas posibilidades.
- Ejecución potente de las aplicaciones nativas en los smartphones.
- Bibliotecas amplias con elementos de interfaz gráfica prefabricados.
- Implementación sencilla de flujos de datos para proporcionar información actual a todos los usuarios.
- Hot Reload acelera las pruebas durante el desarrollo.

Desventajas

- El código del programa puede volverse confuso al integrar los widgets.
- En caso de actualizar aspectos del diseño en los sistemas operativos, hay que actualizar los módulos Flutter. Como los módulos se integran en el programa de manera fija, también hay que compilar el programa e instalarlo en los dispositivos.
- Todavía es un lenguaje nuevo y poco extendido, cuenta con una comunidad reducida.

Aplicación Hello World

Instalación del Framework

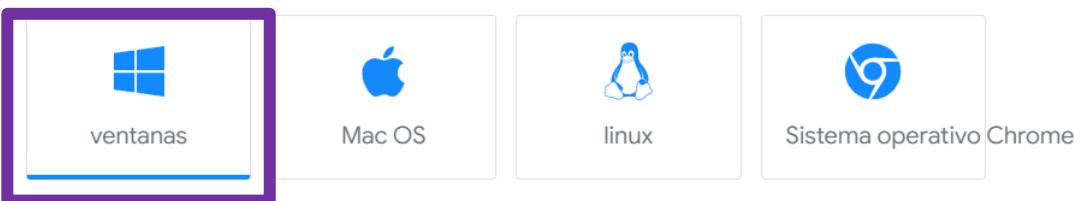
1.- Dirigirse a la documentación oficial de Flutter y dar clic en “Get Start” (Empezar)

https://flutter.dev/?gclid=CjwKCAiAhKycBhAQEiwAgf19eu0xNkAwITvsW4Wvcu1mpB8s0A-FrZ402OhVD8KKtKW51EBeYS0KgxoClv0QAvD_BwE&gclsrc=aw.ds



2.- Selecciona tu sistema operativo

Seleccione el sistema operativo en el que está instalando Flutter:



! Importante: si estás en China, primero lee [Uso de Flutter en China](#).

3.- Descargar el archivo de contenido del SDK y extraemos el archivo

Obtenga el SDK de Flutter

1. Descargue el siguiente paquete de instalación para obtener la versión estable más reciente del SDK de Flutter:

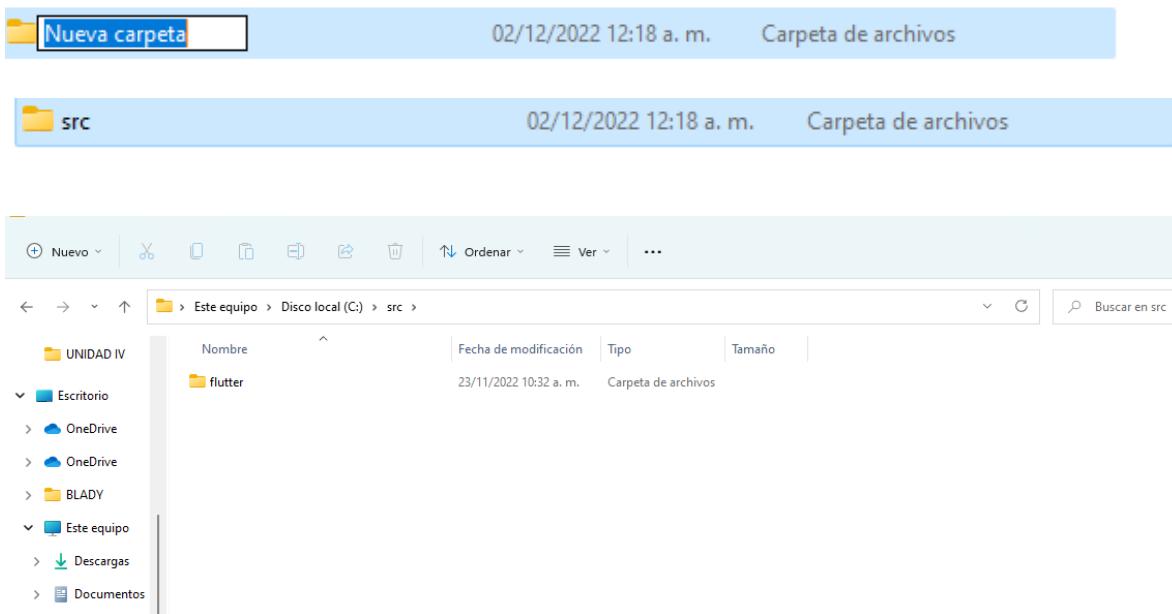
[flutter_windows_3.3.9-estable.zip](#)

Para ver otros canales de lanzamiento y compilaciones anteriores, consulte la página [de lanzamientos de SDK](#).

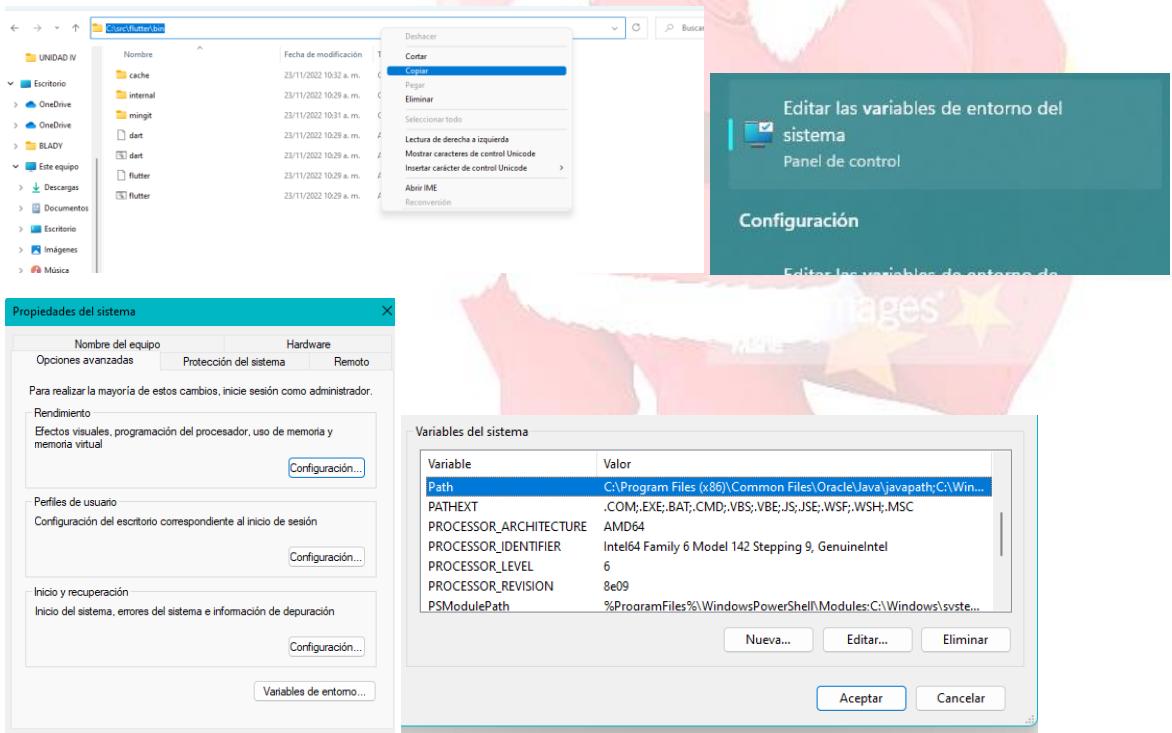
2. Extraiga el archivo zip y coloque el contenido **flutter** en la ubicación de instalación deseada para el SDK de Flutter (por ejemplo, `C:\src\flutter`).

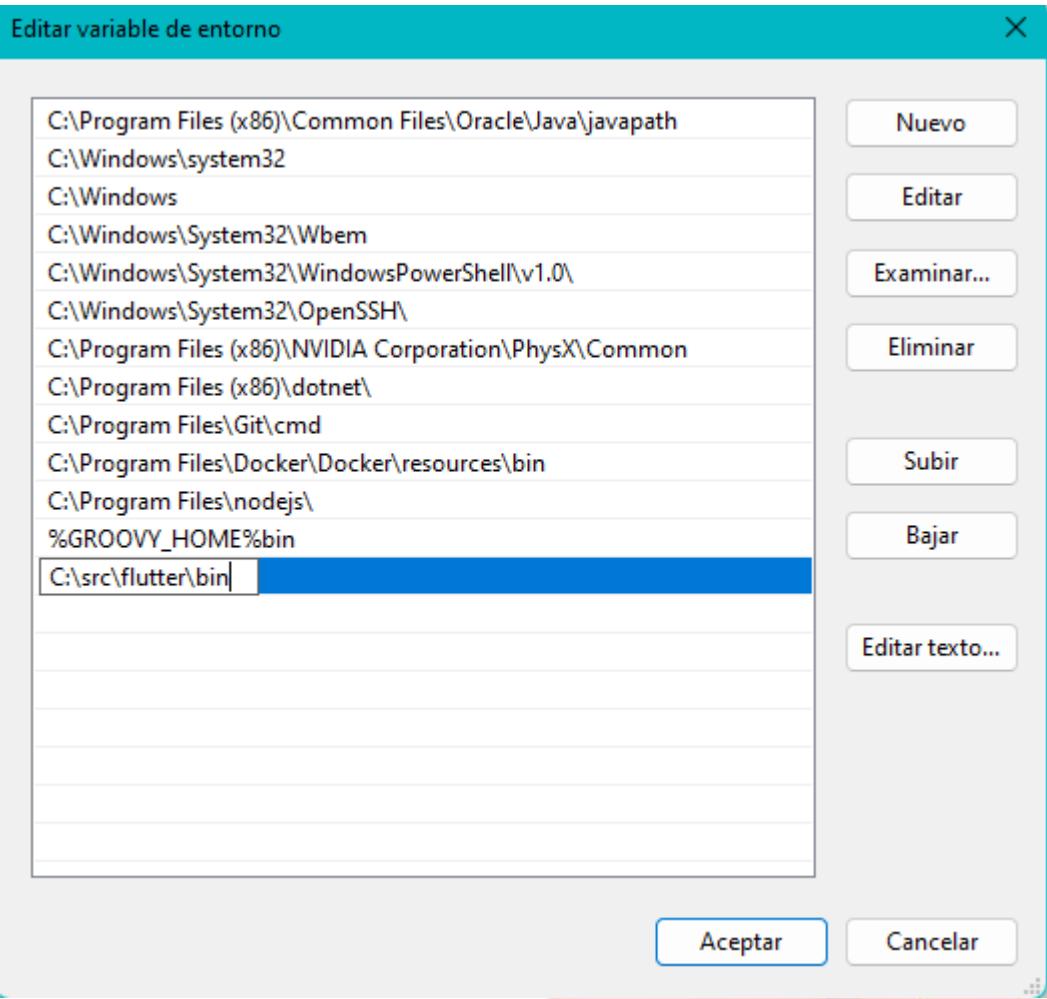


4.- Crear carpeta en disco local C , la renombremos como src y pego la carpeta raiz

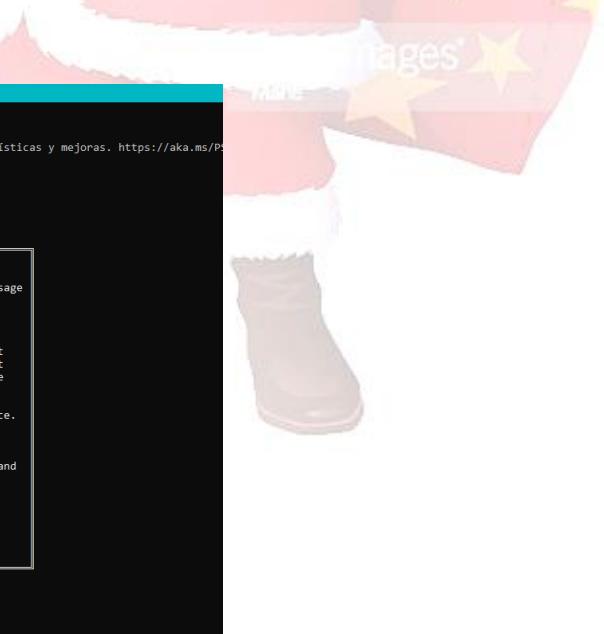


5.- Crear las variables de entorno





Verificamos la versión de flutter



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PowerShell-Upgrade

PS C:\Users\BLADY> flutter --version
Flutter 3.3.9 • channel stable • https://github.com/flutter/flutter.git
Framework • revision b8f71f9f986 (9 days ago) • 2022-11-23 06:43:51 +0900
Engine • revision 8f2221fbef
Tools • Dart 2.18.5 • DevTools 2.15.0

Welcome to Flutter! - https://flutter.dev

The Flutter tool uses Google Analytics to anonymously report feature usage
statistics and basic crash reports. This data is used to help improve
Flutter tools over time.

Flutter tool analytics are not sent on the very first run. To disable
reporting, type 'flutter config --no-analytics'. To display the current
setting, type 'flutter config'. If you opt out of analytics, an opt-out
event will be sent, and then no further information will be sent by the
Flutter tool.

By downloading the Flutter SDK, you agree to the Google Terms of Service.
Note: The Google Privacy Policy describes how data is handled in this
service.

Moreover, Flutter includes the Dart SDK, which may send usage metrics and
crash reports to Google.

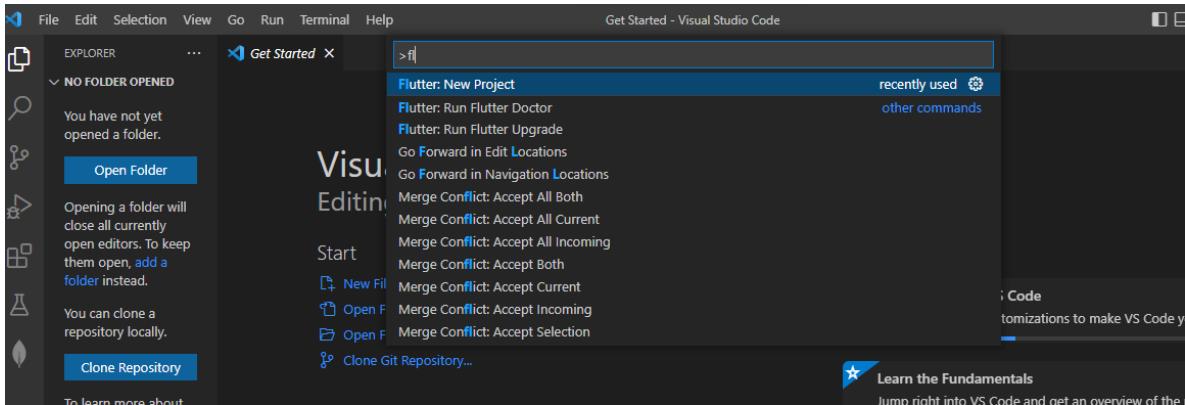
Read about data we send with crash reports:
https://flutter.dev/docs/reference/crash-reporting

See Google's privacy policy:
https://policies.google.com/privacy

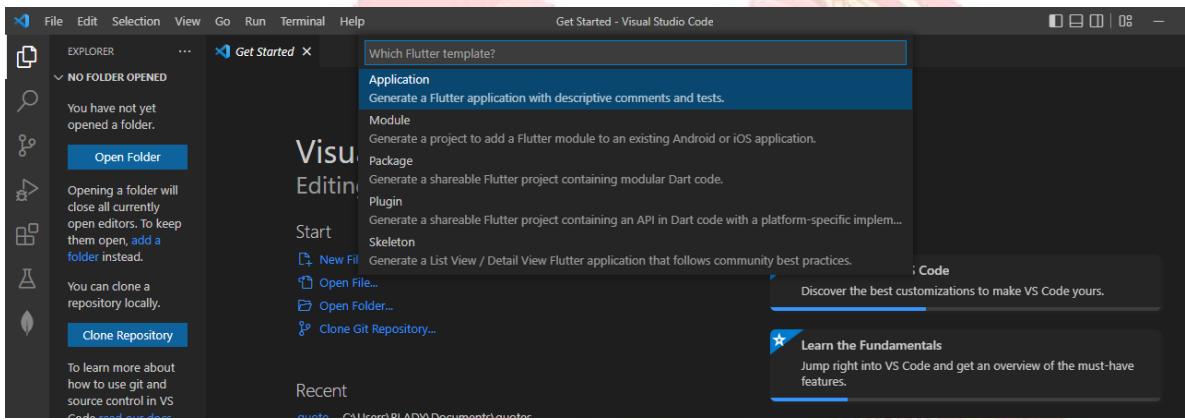
PS C:\Users\BLADY>
```

Crear proyecto

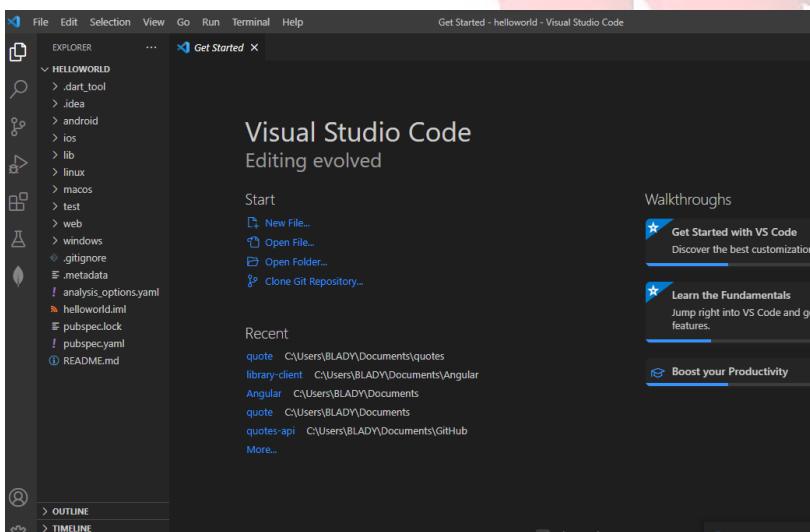
1.- Presionamos la tecla ctrl + shipt + p



2.- Nuevo proyecto de tipo aplicación



3.- Lo nombramos y veremos la siguiente estructura



Creamos nuestro hola mundo con el siguiente comando

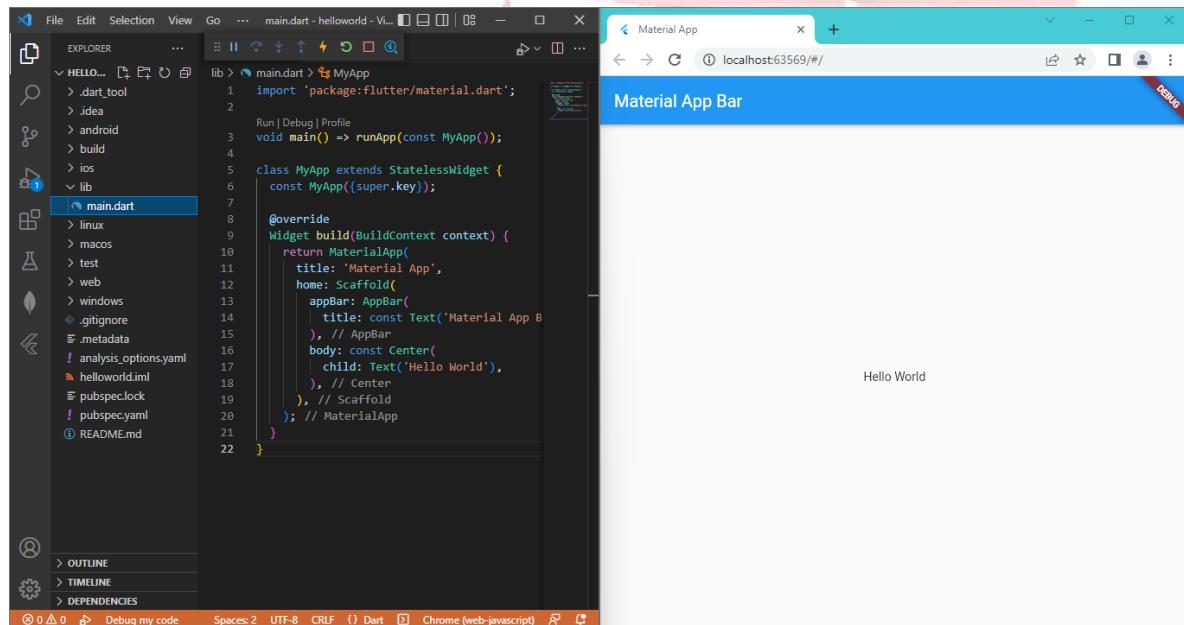
```
import 'package:flutter/material.dart';

void main() => runApp(const MyApp());

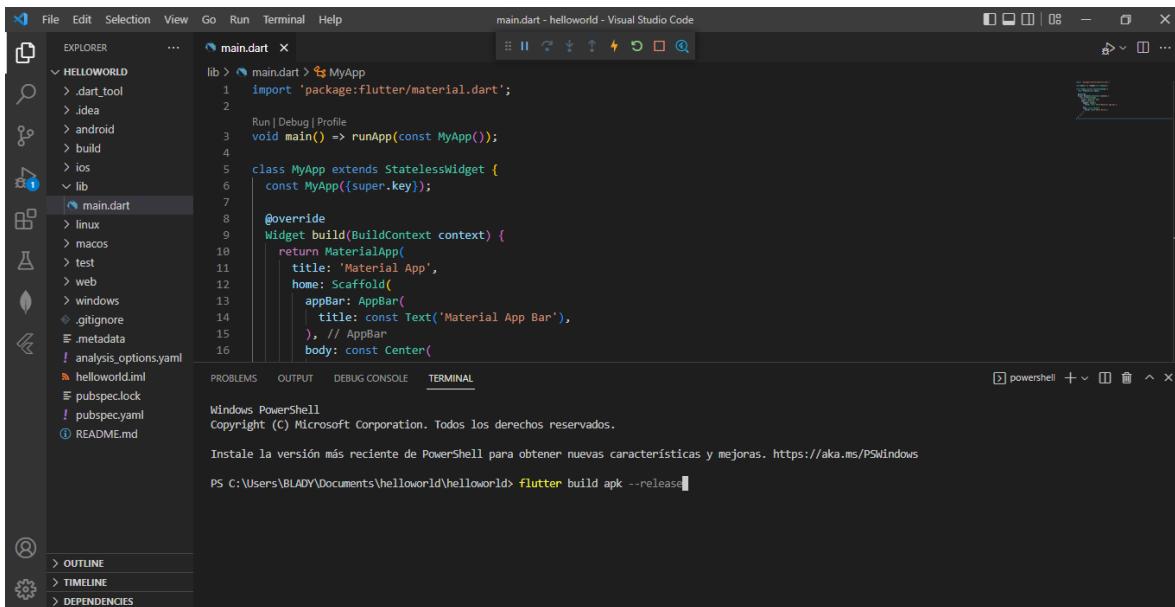
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Material App',
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Material App Bar'),
        ),
        body: const Center(
          child: Text('Hello World'),
        ),
      );
  }
}
```

Test



Generar archivo apk



```
lib/main.dart:1:1: Error: This file has been generated by the build system. It should not be modified directly.
main.dart
^
```

```
File Edit Selection View Go Run Terminal Help
EXPLORER main.dart ×
lib/main.dart > MyApp
1 import 'package:flutter/material.dart';
2
3 Run | Debug | Profile
void main() => runApp(const MyApp());
4
5 class MyApp extends StatelessWidget {
6   const MyApp({super.key});
7
8   @override
9   Widget build(BuildContext context) {
10     return MaterialApp(
11       title: 'Material App',
12       home: Scaffold(
13         appBar: AppBar(
14           title: const Text('Material App Bar'),
15         ), // AppBar
16         body: const Center(
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.
Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows
PS C:\Users\BLADY\Documents\helloworld\helloworld> flutter build apk --release
@powershell + + + + +
```

CONCLUSIÓN

Hoy en día, se habla mucho de las aplicaciones móviles híbridas a la hora de tener una app, y es que resultan ser las más utilizadas a la hora de desarrollar. Cuando hablamos de aplicaciones híbridas, nos estamos refiriendo a un tipo de aplicaciones móviles que constan de una tecnología base de código web, sobre la que se añade código nativo. Este último es el responsable de la mayoría de funcionalidades y ventajas de las apps híbridas. En otras palabras, no son más que una aplicación construida para ser usada o implementada en distintos sistemas operativos móviles, tales como, iOS, Android o Windows Phone, evitando de esta manera, la tarea de crear una aplicación de un especialista para cada sistema operativo. De esta manera, una aplicación híbrida puede ser adaptada a múltiples plataformas móviles sin crear nuevos códigos. Los grandes avances en tecnología y la evolución en el desarrollo app han sido dos factores claves para poder realizar aplicaciones móviles eficaces con un gran número de funcionalidades y adaptadas a los requerimientos de los usuarios. Por ello, es importante conocer cuáles son los mejores frameworks para crear aplicaciones móviles híbridas y ofrecer a nuestro público objetivo el servicio más completo y de mayor calidad posible.

¡MUCHAS GRACIAS POR TODO SU APOYO FELIZ NAVIDAD Y PROSPERO AÑO NUEVO!



REFERENCIAS

- App nativa o híbrida ¿Qué decisión tomar? (2018). OnDesarrollo. <https://ondesarrollo.com/app-nativa-o-hibrida-que-decision-tomar/#:~:text=En%20conclusi%C3%B3n%2C%20su%20rendimiento%20es,funciones%20de%20acorde%20al%20terminal>.
- IONOS Digital Guide. (2020). IONOS. IONOS Digital Guide. <https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/que-es-flutter/>
- de, C. (2018). Flutter (software). Wikipedia.org; Wikimedia Foundation, Inc. [https://es.wikipedia.org/wiki/Flutter_\(software\)](https://es.wikipedia.org/wiki/Flutter_(software))
- Cecilio Álvarez Caules, & Cecilio Álvarez Caules. (2014). JQuery Mobile Arquitectura. Arquitectura Java. <https://www.arquitecturajava.com/jquery-mobile-arquitectura/>
- Flutter - Build apps for any screen. (2022). Flutter.dev. https://flutter.dev/?gclid=CjwKCAiAhKycBhAQEiwAgf19eu0xNkAwITvsW4Wvcu1mpB8s0A-FrZ402OhVD8KKtKW51EBeYS0KgxoClv0QAvD_BwE&gclsrc=aw.ds
- de, C. (2013). PhoneGap. Wikipedia.org; Wikimedia Foundation, Inc. <https://es.wikipedia.org/wiki/PhoneGap>
- React Native: Ventajas y desventajas reveladas. (2021). Back4App Blog. <https://blog.back4app.com/es/react-native-ventajas-y-desventajas-reveladas/>