

**Curso: 2020/21**

**Título de la tarea**

**José Mari Tomé Mejías**

**Desarrollo de aplicaciones Multiplataforma**

**Programación multimedia y dispositivos móviles**

**ÍNDICE**

1. INTRODUCCIÓN 1

2. DIFERENTES APARTADOS DE LA TAREA 1- 4

3. AUTOEVALUACIÓN 5

4. BIBLIOGRAFÍA 5

# INTRODUCCIÓN

**Realizar un trabajo para conocer Ionic.**

**Realizar un proyecto nuevo en Ionic con el template blank y cambiar el título por vuestro nombre.**

# DIFERENTES APARTADOS DE LA TAREA

* 1. **¿Qué es un desarrollo híbrido?**

El desarrollo de aplicaciones móviles híbridas es un enfoque de programación para dispositivos móviles que combina los puntos fuertes de la programación nativa y el desarrollo de apps móviles HTML5, usando tecnologías como HTML, CSS y JavaScript, normalmente se ejecutan en el navegador nativo del sistema por lo que (aunque dependan del framework utilizado) no pueden acceder a tantas funcionalidades del hardware del dispositivo ni a las librerías del sistema.

Es una manera de programar para dispositivos móviles que combina las fortalezas de la programación nativa con otras tecnologías (Como tecnologías web o algún lenguaje de programación diferente al de la plataforma) para desarrollar aplicaciones multiplataforma que se ejecuten de forma nativa en Android o IOS.

Para crear una aplicación móvil híbrida el desarrollo se hace en HTML5 y se coloca un empaquetador (wraper) de dispositivo nativo alrededor. El empaquetador nativo actúa como intermediario y traduce las instrucciones a una forma que el dispositivo móvil entiende. El núcleo HTML5 de la aplicación móvil híbrida funciona dentro del contenedor nativo y utiliza el motor del navegador del dispositivo móvil (no el propio navegador) para cargar el HTML5 y procesar el código JavaScript a nivel local. El núcleo HTML5 de la aplicación puede ser reutilizado para otras plataformas móviles. Solo el empaquetador nativo debe ser escrito para cada dispositivo móvil y sistema operativo. [Se puede reducir el tiempo de desarrollo](https://searchdatacenter.techtarget.com/es/consejo/Todo-lo-que-necesita-saber-para-desarrollar-aplicaciones-moviles) generando empaquetadores con software de proveedores como PhoneGap o Ionic.

* 1. **Ventajas de un desarrollo híbrido**

La principal ventaja de un desarrollo híbrido es que se puede crear aplicaciones multiplataforma tanto para Android como para IOS, sin necesidad de tener un programador que programe para Android y otro para IOS, reduciendo el coste de la aplicación.

* 1. **Ventajas de un desarrollo nativo**

El principal motivo que es más factible utilizar aplicaciones nativas es usan el lenguaje nativo de la plataforma en el caso Android será Java, Objetive-C o Swift para IOS y Visual Basic en Windows, entre otros.

Por lo que el rendimiento y el tamaño del programa será inferior al de un desarrollo híbrido al no necesitar librerías ni Frameworks que se usen como intermediarios para que el dispositivo pueda ejecutar el código.

Entre todas las ventajas que tiene una aplicación nativa tenemos:

* Mejor rendimiento
* Menor consumo de memoria
* Mayor velocidad
* Aprovechamiento total del hardware del dispositivo (Cámara, GPS, Sensores, entre otros)
  1. **Escenarios en los que cada tipo de desarrollo es mejor**

Los usos de aplicaciones híbridas son aplicables a aplicaciones que se quieran instalar en distintas plataformas sin necesidad de más personal.

Los usos de aplicaciones nativas son aplicables cuando se requieren muchos recursos de hardware del teléfono y muchas peticiones al servidor con grandes números de usuarios.

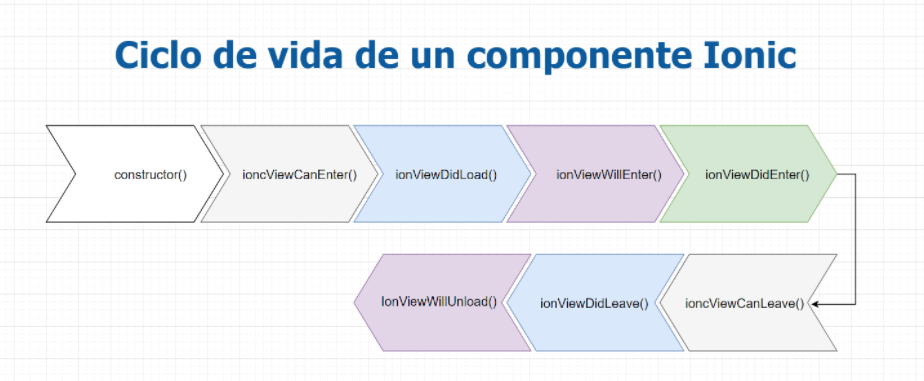
* 1. **¿Qué es Capacitor? ¿Ventajas de Capacitor respeto a Cordoba?**

[Capacitor](https://capacitor.ionicframework.com/) es un nuevo puente nativo para crear aplicaciones multiplataforma (“[Native Progressive Web Apps](https://capacitor.ionicframework.com/docs/)”) sustituye a Apache Cordoba y ha sido creado por la comunidad de Ionic, lo que ya es una ventaja porque no te tienes que mover por distintas comunidades buscando documentación o bugs.

Con el ecosistema Ionic puedes montar las vistas y las interfaces nativas de las mismas, sin dependencia del roadmap de una herramienta de terceros como sería con Cordoba.

Soporte a las PWA. Esto es muy importante, ya hemos hablado otras veces de [PWA](https://alotroladodeltelefonoblog.wordpress.com/2018/07/07/pwa-progressive-web-apps/) y, para sacarles el máximo partido, necesitan mecanismos para poder acceder fácilmente a sistemas del dispositivo. **Capacitor ha nacido con la idea de dar soporte desde el principio a \*PWA** por lo que su integración es más que natural.

**\*PWA** Una aplicación web progresiva es un tipo de software de aplicación que se entrega a través de la web, creado utilizando tecnologías web comunes como HTML, CSS y JavaScript. Está destinado a funcionar en cualquier plataforma que use un navegador compatible con los estándares.

* 1. **Explicar el ciclo de vida de una página. (**<https://ionicframework.com/docs/angular/lifecycle>**)**

**Constructor**

Esto es realmente una característica estándar de las clases en ES6 y no proporcionada directamente por Ionic, pero es el primer evento que se dispara cuando se crea una página.

**ngOnInit**

Este es un evento de Angular, sucede cuando angular ha desplegado los data-bound properties (variables vinculadas a datos) o cuando el componente ha sido inicializado, una vez que ngOnChanges se haya ejecutado. Este evento se puede utilizar para inicializar componentes y realizar llamadas a servicios que solo deben realizarse una vez.

**ngOnDestroy**

También es otro evento de Angular, lanzado justo antes de que Angular destruya la vista. Útil para la limpieza, como cancelar la suscripción a observables. Es el lugar adecuado para realizar las acciones necesarias a ejecutar con la destrucción de un componente.

**ionViewCanEnter**

Este sería nuestro guardia de navegación y es **muy útil si queremos chequear algo antes de entrar a la página**. Es el evento ideal para comprobar si cuentas con los permisos para poder ver el contenido de la página. Debes de asegurarte que la lógica que utilices acá devuelva un booleano.

**ionViewDidLoad**

Se ejecuta cuando la página se ha cargado. **Este evento sólo se activa una vez** y sólo una vez como la función constructor(), pero la diferencia es que en este punto, puede estar seguro de que todas sus variables y dependencias inyectadas están disponibles para su uso. Esta es la función donde usted puede hacer todas sus llamadas HTTP iniciales para obtener sus datos y hacer el levantamiento pesado principal de su aplicación.

**ionViewWillEnter**

Se ejecuta cuando la página está a punto de entrar y convertirse en la página activa.

**ionViewDidEnter**

Se ejecuta cuando la página ha entrado completamente y ahora es la página activa. Este evento se disparará, ya sea la primera carga o al cargar la página en caché. Es la función final que se activa al navegar en una página como parte del ciclo de vida de ella.

**ionViewCanLeave**

Se ejecuta antes de que la vista pueda salir. Esto puede usarse como una especie de “guardia” en las vistas autenticadas donde necesita comprobar los permisos antes de que la vista pueda salir. En esencia es lo mismo a ionViewCanEnter() pero en este caso te deja salir de la página si la lógica que utilizaste dentro de ella retorna true.

**ionViewWillLeave**

Se ejecuta cuando la página está a punto de salir. En el punto en que se desencadena este evento, sigue siendo la página activa, pero **se ha puesto en cola para que se elimine y ya no se puede evitar salir de la página**. Este evento puede ser un buen lugar para preparar los datos que se podría utilizar en la siguiente página o realizar una llamada asíncrona alguna API que se necesitará en la próxima vista.

**ionViewDidLeave**

Se ejecuta cuando la página ha terminado de salir y ya no es la página activa. Esta función es un gran lugar para guardar datos o estados de la página que está dejando, así cómo activar algunas operaciones de fondo que no requieren que la vista sea visible.

**ionViewWillUnload**

Es el último evento que se dispara en el ciclo de vida de una página típica de Ionic. Este es el lugar donde se liberan los recursos que ya no son necesarios y todo tipo de limpieza para evitar posibles pérdidas de memoria. **Aquí se hace el “unsubscribe” a los eventos** y observables que estén suscritos. Esta función sólo se ejecuta una vez y es la última parada antes de que la página se destruya y se desvincula de la vista con todos sus elementos eliminados.

* 1. **Opinión personal de los desarrollos híbridos**

Los desarrollos hibridos tienen una gran ventaja en su desarrollo que es el tiempo de desarrollo con lo que conlleva a aplicaciones más económicas, aunque se puedan usar ya la mayoría del hardware del teléfono su rendimiento de las aplicaciones baja en comparación a los desarrollos nativos.

# AUTOEVALUACIÓN

|  |  |
| --- | --- |
| La app se ejecuta sin problemas | **2,50** |
| El título está cambiado | **1,00** |
| El trabajo incluye los aspectos más relevantes | **6,00** |
| El formato del estudio es correcto | **0,50** |
| Total Autoevaluación | **0,50** |
|  |  |

# BIBLIOGRAFÍA

<https://www.northware.mx/2017/09/12/desarrollo-de-aplicaciones-moviles-hibridas/>

<https://searchdatacenter.techtarget.com/es/definicion/Desarrollo-de-aplicaciones-moviles-hibridas>

<http://blog.sandiamedia.agency/desarrollo-de-aplicaciones-moviles-como-elegir-entre-nativas-o-hibridas/>

<ftp://ftp.software.ibm.com/la/documents/gb/commons/27754_IBM_WP_Native_Web_or_hybrid_2846853.pdf>

<https://es.bccrwp.org/solution/hybrid-app-vs-native-app/>

<https://alotroladodeltelefonoblog.wordpress.com/2020/02/01/capacitor-el-nuevo-apache-cordova/>

<https://medium.com/williambastidasblog/el-sue%C3%B1o-de-crear-una-aplicaci%C3%B3n-ios-android-electron-y-web-pwa-desde-una-sola-base-de-c%C3%B3digo-ab66ad903919>

<https://blog.maestriajs.com/blog/tips/lifecycle-ionic/>