**UAH**

16

**Desarrollo de aplicaciones multiplataforma**

**TG2**

Contenido

[1. Autores del trabajo, planificación y entrega 4](#_Toc448145158)

[1.1 Autores 4](#_Toc448145159)

[1.2 Planificación 4](#_Toc448145160)

[1.3 Entrega 4](#_Toc448145161)

[2. Definiciones y siglas 4](#_Toc448145162)

[3. Descripción de las tecnologías 5](#_Toc448145163)

[3.1 Ionic 5](#_Toc448145164)

[3.1.1 Principales características 6](#_Toc448145165)

[3.2 Apache Cordova 7](#_Toc448145166)

[3.2.1 Arquitectura 8](#_Toc448145167)

[4. Criterios de comparación 10](#_Toc448145168)

[4.1 Categoría A: Interfaz 10](#_Toc448145169)

[4.1.1 Criterio A.1: La experiencia y capacidad para el usuario 10](#_Toc448145170)

[4.1.2 Criterio A.2: Interfaz 10](#_Toc448145171)

[4.2 Categoría B: Rendimiento 10](#_Toc448145172)

[4.2.1 Criterio B.1: Recursos 10](#_Toc448145173)

[4.3 Categoría C: Modelo de Negocio 10](#_Toc448145174)

[4.3.1 Criterio C.1: Costes 10](#_Toc448145175)

[4.4 Categoría D: Utilidad 10](#_Toc448145176)

[4.4.1 Criterio D.1: Actualizaciones 10](#_Toc448145177)

[4.4.2 Criterio D.2: Control de distribución 10](#_Toc448145178)

[4.4.3 Criterio D.3: Fragmentación 10](#_Toc448145179)

[4.4.4 Criterio D.4: Disponibilidad de conocimiento de programación 10](#_Toc448145180)

[4.4.5 Criterio D.5: Seguridad 10](#_Toc448145181)

[4.4.6 Criterio D.6: Modo Offline 10](#_Toc448145182)

[4.5 Categoría B 11](#_Toc448145183)

[4.6 Categoría C 11](#_Toc448145184)

[5. Evaluación de los criterios por tecnología 11](#_Toc448145185)

[5.1 Evaluación de los criterios para Ionic 11](#_Toc448145186)

[5.2 Evaluación de los criterios para Apache Cordova 11](#_Toc448145187)

[6. Comparación de las tecnologías 12](#_Toc448145188)

[7. Recomendaciones 13](#_Toc448145189)

[7.1 Situación 1 13](#_Toc448145190)

[7.1.1 Descripción de la situación 13](#_Toc448145191)

[7.1.2 Recomendación de tecnología a utilizar 13](#_Toc448145192)

[7.2 Situación 2 13](#_Toc448145193)

[7.2.1 Descripción de la situación 13](#_Toc448145194)

[7.2.2 Recomendación de tecnología a utilizar 13](#_Toc448145195)

[8. Conclusiones 13](#_Toc448145196)

# Autores del trabajo, planificación y entrega

## Autores

El grupo que desarrolla el trabajo que se presenta a continuación el es grupo 7 de laboratorio del turno de tarde; este grupo está formado por:

* Miguel Ángel Rodríguez Blanco (Coordinador del grupo).
* Luis Nueda García.
* Álvaro Laza Martín.
* Eduardo Dorado Pérez.

## Planificación

La planificación de proyecto ha sido realizada por el Coordinador del grupo utilizando la herramienta de diagramación GanttPro, tras una reunión con el grupo de trabajo en la que se han expuesto las tareas necesarias para la consecución del objetivo final y se ha realizado un reparto/asignación de tareas equitativo.

Este trabajo de planificación está disponible en la URL:

[***https://app.ganttpro.com/shared/token/efc3eaab91ed6391cc072e2f6774710be06e949e7d796a915e91287d2e1aefce#!/app/home***](https://app.ganttpro.com/shared/token/efc3eaab91ed6391cc072e2f6774710be06e949e7d796a915e91287d2e1aefce#%21/app/home)

## Entrega

El repositorio creado para la puesta en común de los avances del grupo y en el que se ha guardado el resultado de este trabajo es:

***https://github.com/MiguelRodriguezBlanco/TG2***

En este repositorio se encuentra, además de los archivos subidos a los largo del desarrollo del trabajo los archivos requeridos en la rama máster:

* Trabajo terminado: del trabajo terminado con el nombre TG2\_final.docx
* Presentación del trabajo: TG2\_final.pptx

## Definiciones y siglas

**AngularJS** - [framework](https://es.wikipedia.org/wiki/Framework) de [JavaScript](https://es.wikipedia.org/wiki/JavaScript) de [código abierto](https://es.wikipedia.org/wiki/C%C3%B3digo_abierto), mantenido por [Google](https://es.wikipedia.org/wiki/Google), que se utiliza para crear y mantener [aplicaciones web](https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web) [de una sola página](https://es.wikipedia.org/wiki/Single-page_application). Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de [Modelo Vista Controlador](https://es.wikipedia.org/wiki/Modelo_Vista_Controlador) (MVC), en un esfuerzo para hacer que el desarrollo y las [pruebas](https://es.wikipedia.org/wiki/Pruebas_de_software) sean más fáciles.

**CSS** - Cascading Style Sheets, es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en [HTML](https://es.wikipedia.org/wiki/HTML) o [XML](https://es.wikipedia.org/wiki/Extensible_Markup_Language).

**DOM** - Document Object Model es una interfaz de plataforma que proporciona un conjunto estándar de objetos para representar documentos [HTML](https://es.wikipedia.org/wiki/HTML), [XHTML](https://es.wikipedia.org/wiki/XHTML) y [XML](https://es.wikipedia.org/wiki/Extensible_Markup_Language)[1](https://es.wikipedia.org/wiki/Document_Object_Model#cite_note-1) , un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos. A través del DOM, los programas pueden acceder y modificar el contenido, estructura y estilo de los documentos HTML y XML, que es para lo que se diseñó principalmente.

**Licencia MIT**  - es una de tantas [licencias de software](https://es.wikipedia.org/wiki/Licencia_de_software) que se origina en el [Instituto Tecnológico de Massachusetts](https://es.wikipedia.org/wiki/Instituto_Tecnol%C3%B3gico_de_Massachusetts) (MIT, Massachusetts Institute of Technology). Quizás debería llamarse más correctamente **licencia X11**, ya que es la licencia que llevaba este [software](https://es.wikipedia.org/wiki/Software) de muestra de la información de manera gráfica [X Window System](https://es.wikipedia.org/wiki/X_Window_System) originario del MIT en los años 1980. Pero ya sea como MIT o X11, su texto es idéntico.

**Mixin** - es una [clase](https://es.wikipedia.org/wiki/Clase_(inform%C3%A1tica)), que en los [lenguajes de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) [orientada a objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos), ofrece cierta funcionalidad para ser [heredada](https://es.wikipedia.org/wiki/Herencia_(programaci%C3%B3n_orientada_a_objetos)) por una subclase, pero no está ideada para ser autónoma, sino que es más bien un medio de obtener funcionalidad.

**Sass** - Syntactically Awesome Stylesheets, es un metalenguaje de hoja de estilos (CSS) o [lenguaje de script](https://es.wikipedia.org/wiki/Lenguaje_de_script) traducido a CSS.

**SDK** - Software Development Kit, es un conjunto de herramientas de [desarrollo de software](https://es.wikipedia.org/wiki/Desarrollo_de_software) que le permite al [programador](https://es.wikipedia.org/wiki/Programador) o [desarrollador de software](https://es.wikipedia.org/wiki/Desarrollador_de_software) crear [aplicaciones](https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_(inform%C3%A1tica)) para un sistema concreto.

# Descripción de las tecnologías

## Ionic

|  |  |
| --- | --- |
| logo-ionic-framework-phonegap-spain | Ionic es un potente SDK HTML5, que facilita el desarrollo de aplicaciones móvil nativas e híbridas multiplataforma utilizando tecnologías web como HTML, CSS (generado por SASS), y Javascript. |

Al tratarse de una herramienta de código abierto publicado bajo una licencia MIT, la misma que utilizan “jQuery” y “Ruby on Rails”, permite ser utilizado en proyectos tanto personales como comerciales, de forma gratuita.

Ionic es un MVC (Model-View-Controller o Modelo-Vista-Controlador), un patrón de diseño que separa los datos, la lógica y las interfaces de usuario, está separado en tres componentes: Modelo, Vista y Controlador.

Está basado en la ideología de separación de conceptos y cumple perfectamente con los objetivos de los patrones de diseño.

* **Modelo:** Es la capa encargada de los datos, es decir, la que se encarga de hacer peticiones a las bases de datos para enviar o recibir información. Estas bases de datos pueden estar alojadas de forma local en nuestra app o de forma remota en un servidor externo.
* **Vista:** Se trata del código que nos permitirá presentar los datos que el modelo nos proporciona, como ejemplo podríamos decir que en una aplicación es el código HTML que nos permite mostrar la salida de los datos procesados.
* **Controlador:** Es la capa que sirve de enlace entre la vista y el modelo. Envía comandos al modelo para actualizar su estado, y a la vista correspondiente para cambiar su presentación.

En el caso MVVM (Modelo Vista VistaModelo) la iteracción entre la vista y el controlador será en los dos sentidos, el controlador muestra los datos en la vista y si en la vista hay un cambio de datos, se actualiza el modelo automáticamente.

Ionic está compuesto por dos partes:

* **Sass**, preprocesador CSS, que permite trabajar con elementos CSS y encargado de darle estilo a nuestras aplicaciones.
* **AngularJS**, permite optimizar el anterior y crear todos los componentes que se generan a través de [directivas](http://uno-de-piera.com/directivas-en-angularjs/), [factorías](http://uno-de-piera.com/consumir-datos-con-factorias-y-servicios-en-angularjs/) o [servicios](http://uno-de-piera.com/consumir-json-con-angularjs-con-resources-y-http/), que permite trabajar con el máximo potencial, lo que asegura aplicaciones rápidas y escalables

El CSS puede actuar de forma autónoma, aunque permite ser mejorado por el desarrollador. Permite añadir y utilizar propiedades por defecto al CSS; además con el fin de proporcionar más potencia y flexibilidad, el núcleo está escrito con Sass e incluye variables y mixins fácilmente personalizados. Si bien el diseño por defecto es similar a iOS, el CSS permite ser ampliado para otorgar a la app su propia apariencia.

Ionic se centra principalmente en el aspecto y las sensaciones del usuario al interactuar con la app. No se trata de un sustituto de PhoneGap u otro framework Javascrip, sin embargo Ionic que adapta a estos proyectos para simplificar el desarrollo de una parte muy importante de la app, la “Interfaz”.

### Principales características

(<http://www.phonegapspain.com/que-es-y-como-empezar-con-ionic-framework/>)

|  |  |
| --- | --- |
| **Alto rendimiento** | |
| ionic-1 | Ionic se ha construido para desarrollar y ofrecer el mejor rendimiento para los últimos dispositivos móviles. La mínima manipulación del DOM, con cero jQuery y transiciones aceleradas por hardware, proporcionan a Ionic la capacidad de ser rápido y trasladar esta rapidez a las apps que se desarrollan con él. |

|  |  |
| --- | --- |
| **AngularJS & Ionic** | |
| ionic-2 | Ionic trabaja con AngularJS para crear un potente SDK (Sw Development Kit) apropiado para desarrollar aplicaciones ricas y robustas; así, Ionic presenta una arquitectura central robusta y profesional para el desarrollo de aplicaciones. |

|  |  |
| --- | --- |
| **Centro nativo** | |
| ionic-3 | Ionic es una herramienta a partir de la que se desarrolla código una vez y compila para varios, se modela a partir de los SDK nativos de desarrollo móviles más populares, lo que lo convierte en una herramienta amigable para todos aquellos desarrolladores que ya han trabajado en aplicaciones nativas para iOS o Android. |

|  |  |
| --- | --- |
| **Diseño elegante** | |
| ionic-4 | Ionic ha sido diseñado para poder trabajar y aprovechar todas las capacidades del terminal móvil, integrando la mayor parte de los componentes usados en móviles, como son tipografía, elementos interactivos... mediante la inclusión de infinidad de potentes plugins desarrollados por la comunidad y los propios de Cordova de un modo limpio, sencillo y funcional. |

|  |  |
| --- | --- |
| **Potente CLI** | |
| ionic-5 | Desde el terminal permite crear nuevos proyectos, empezando a partir de un proyecto en blanco o con alguna pequeña estructura determinada acorde a las necesidades del desarrollador o del cliente/usuario final. También es posible compilar el proyecto, emular la app mediante un emulador o hacer que esta corra en el teléfono mediante un solo comando de forma muy sencilla partir de un único comando permite crear, construir, probar y compilar aplicaciones en cualquier plataforma. |

## Apache Cordova

Apache Cordova es un entorno de desarrollo de aplicaciones móviles de código abierto, que permite la utilización de tecnologías web estándar como son HTML5, CSS3 y Javascript para el desarrollo multiplataforma.

Las aplicaciones se ejecutan dentro de los entornos específicos para cada plataforma, adaptándose a los enlaces de la API compatible con los estándares de acceso a las capacidades de cada dispositivo, tales como sensores, datos, estado de la red, etc.

Usar Apache Cordova si es:

* un desarrollador móvil y quieren extender una aplicación a través de más de una plataforma, sin tener que volver a ponerlo en práctica con el conjunto de herramienta de lenguaje y de cada plataforma.
* un desarrollador web y desean implementar una aplicación web que está acondicionado para la venta en diferentes portales de la tienda de aplicaciones.
* un desarrollador móvil interesados ​​en mezclar los componentes de aplicaciones nativas con un WebView (ventana del navegador especial) que se puede acceder a las API a nivel de dispositivo, o si se quiere desarrollar una interfaz de complementos entre los componentes nativos y WebView.

### Arquitectura

Hay varios componentes en una aplicación Cordova, en el siguiente esquema se muestra una vista de alto nivel de la arquitectura de la aplicación Cordova.



Figura 1.- Aquitectura Cordova (<https://cordova.apache.org/docs/en/latest/guide/overview/>)

#### WebView

La WebView de Cordova le proporciona a la aplicación una completa interfaz de usuario. Para algunas plataformas puede formar parte de una aplicación hibrida mayor mezclando componentes nativos de la aplicación con esta WebView.

#### Web App

Esta es la parte de la aplicación donde reside el código.  La propia aplicación se implementa como una página web, por defecto un archivo local llamado index.html , que hace referencia a CSS, JavaScript, imágenes, archivos multimedia, u otros recursos son necesarios para que se ejecute. La aplicación se ejecuta en el WebView dentro de la envoltura de aplicación nativa, que se distribuye a las “app stores”.

Este recipiente tiene un archivo muy importante - [config.xml](https://cordova.apache.org/docs/en/latest/config_ref/index.html), que proporciona información sobre la aplicación y especifica los parámetros que afectan a la forma en que funciona, por ejemplo, si responde a los cambios de orientación.

#### Plugins

Los plugins son una parte integral del ecosistema Córdoba; estos proporcionan una interfaz para Cordova y los componentes nativos, que les permite comunicarse entre sí y los enlaces a las API del dispositivo estándar. Esto permite invocar código nativo de JavaScript.

Un proyecto Apache Cordova dispone de un conjunto de plugins conocido como “Core Plugins” que proporcionan a la aplicación la capacidad de acceso a las utilidades del dispositivo, tales como la batería, cámara, contactos…

Además de estos, hay otros plugins tercer nivel que proporcionan enlaces adicionales para características no necesariamente disponibles en todas las plataformas. Estos plugins Cordova se pueden buscar a través de [plugin search](https://cordova.apache.org/plugins/) o [NGP](https://www.npmjs.com/search?q=ecosystem%3Acordova) . También es posible desarrolla plugins propios, siguiendo la [Guía de Programas para el Desarrollo](https://cordova.apache.org/docs/en/latest/guide/hybrid/plugins/index.html).

**NOTA**: Cuando se crea un proyecto de Córdoba, no tiene ningún tipo de plugins presentes (este es el nuevo comportamiento predeterminado). Cualquier plugin necesario para la aplicación incluso los “Core Plugins”, se deben añadir de forma explícita.

Córdoba no proporciona ningún widget de interfaz de usuario o framework, si no que ofrece sólo el tiempo de ejecución en los que se pueden ejecutar, por lo que para utilizar los widgets de interfaz de usuario y / o framework, es necesario seleccionarlos e incluirlos en la aplicación.

#### Development Paths

Cordova proporciona dos flujos de trabajo básicos para crear una aplicación móvil, que pueden ser utilizados para realizar la misma tarea, aunque cada uno de ellos ofrece unas ventajas.

* **Flujo de trabajo multiplataforma (CLI – Command Line Interface/Interfaz de línea de comandos)**: este flujo de trabajo facilita que una aplicación se ejecute en tantos sistemas operativos móviles como sea posible, sin necesidad de un desarrollo específico para cada plataforma. Este flujo de trabajo se centra en torno la CLI Cordova. La CLI es una herramienta de alto nivel que permite construir proyectos para muchas plataformas a la vez, abstrayéndose de gran parte de la funcionalidad de secuencias de comandos shell de nivel inferior. La CLI copia un conjunto común de recursos web en subdirectorios para cada plataforma móvil, hace los cambios de configuración necesarios para cada uno, y ejecuta scripts de creación para generar binarios de la aplicación. La CLI también proporciona una interfaz común para aplicar plugins para la aplicación.
* **Flujo de trabajo centrado en la plataforma**: este flujo de trabajo permite centrarse en la construcción de una aplicación para una sola plataforma que tiene que ser capaz de modificarla en un nivel más bajo. Este enfoque, es útil, para mezclar los componentes nativos personalizados con componentes Cordova basadas en la Web. Como regla general, este flujo de trabajo debe utilizarse siempre que sea necesario modificar el proyecto dentro del SDK.

Este flujo de trabajo se basa en un conjunto de secuencias de comandos shell de nivel inferior que se adaptan para cada plataforma compatible, y una utilidad Plugman independiente que permite aplicar plugins.

Aunque también puede utilizarse para crear aplicaciones multiplataforma, por lo general es más difícil debido a la falta de una herramienta de nivel más alto, además requiere ciclos de compilación separadas y modificaciones de los plugin para cada plataforma.

Cuando se empieza, puede resultar más fácil de utilizar el flujo de trabajo multiplataforma para crear una aplicación, y después, permite cambiar a un flujo de trabajo centrado en la plataforma si es necesario tener mayor control del SDK.

# Criterios de comparación

## Categoría A: Interfaz

### Criterio A.1: La experiencia y capacidad para el usuario

### Criterio A.2: Interfaz

## Categoría B: Rendimiento

### Criterio B.1: Recursos

## Categoría C: Modelo de Negocio

### Criterio C.1: Costes

## Categoría D: Utilidad

### Criterio D.1: Actualizaciones

### Criterio D.2: Control de distribución

### Criterio D.3: Fragmentación

### Criterio D.4: Disponibilidad de conocimiento de programación

### Criterio D.5: Seguridad

### Criterio D.6: Modo Offline

## Categoría B

## Categoría C

# Evaluación de los criterios por tecnología

## Evaluación de los criterios para Ionic

|  |  |
| --- | --- |
| **CRITERIOS** | **EVALUACIÓN** |
| Criterio A.1: Nombre |  |
| Criterio A.2: Nombre |  |
| Criterio A.n: Nombre |  |
| … |  |
| Criterio B.1: Nombre |  |
| Criterio B.2: Nombre |  |
| … |  |

## Evaluación de los criterios para Apache Cordova

|  |  |
| --- | --- |
| **CRITERIOS** | **EVALUACIÓN** |
| Criterio A.1: Nombre |  |
| Criterio A.2: Nombre |  |
| Criterio A.n: Nombre |  |
| … |  |
| Criterio B.1: Nombre |  |
| Criterio B.2: Nombre |  |
| … |  |

# Comparación de las tecnologías

|  |  |  |  |
| --- | --- | --- | --- |
| **CRITERIOS** | **TECNOLOGÍA A** | **TECNOLOGÍA B** | **COMENTARIOS** |
| A.1 |  |  |  |
| A.2 |  |  |  |
| … |  |  |  |
| B.1 |  |  |  |
| B.2 |  |  |  |
| … |  |  |  |

# Recomendaciones

## Situación 1

### Descripción de la situación

### Recomendación de tecnología a utilizar

## Situación 2

### Descripción de la situación

### Recomendación de tecnología a utilizar

# Conclusiones