Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

PoliticApp: Conoce la vida política del Congreso

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Imanol Torres Rodríguez

Tutor: David De Andrés Martínez

2016-2017

Resumen

El presente trabajo abarca el desarrollo de una aplicación móvil que permita a los usuarios conocer de forma rápida y sencilla información de carácter abierto a cerca del Congreso de los Diputados y sus representantes políticos, así como proporcionar un canal en el que poder expresar su opinión respecto a las iniciativas que se debaten a diario. La aplicación pretende dotar de mayor accesibilidad a dicha información que, a pesar de ser datos abiertos y estar a disposición de los ciudadanos, en ocasiones resulta difícil de consultar.

El trabajo se ha enfocado desde la perspectiva de las tecnologías web multiplataforma como alternativa válida al desarrollo nativo tradicional para dispositivos móviles, presentando sus ventajas e inconvenientes. Para abordar el desarrollo se han utilizado frameworks como Ionic y Angular siguiendo el nuevo estándar ECMAscript6 junto con su superset Typescript como lenguaje de programación debido al auge que están teniendo actualmente entre la comunidad web como tecnologías punteras para el desarrollo empresarial. También se ha hecho uso de Firebase para la parte backend, un completo sistema Backend-as-a-Service (BaaS) con características como realtime database y respaldado por Google y toda su infraestructura cloud escalable.

Palabras clave: Desarrollo móvil, web, multiplataforma, datos abiertos, cloud, política, diputados, accesibilidad

Abstract

This project covers the development of a mobile application that allows users to quickly and easily find open information about the Congress of Deputies and their political representatives, as well as providing a channel to express their opinion about the initiatives debated on a daily basis. The application aims to provide greater accessibility to such information which, despite being open data and available to citizens, is sometimes difficult to consult.

The project has focused on the perspective of multiplatform web technologies as a valid alternative to traditional native development for mobile devices, exposing its advantages and disadvantages. In addressing development, frameworks such as Ionic and Angular have been used following the new ECMAScript6 standard along with its superset Typescript as a programming language due to the current boom in the web community as leading technologies for business development. Firebase has also been used for the backend part, a complete Backend-as-a-Service (BaaS) system with features like realtime database and supported by Google and its entire scalable cloud infrastructure.

Keywords: Mobile development, web, multiplatform, open data, cloud, politics, deputies, accessibility

Tabla de contenidos

Índice de Figuras

Figuras:

Tablas:

1. Introducción

1.1 Motivación

En los últimos años el auge de las nuevas tecnologías nos ha permitido vivir en una sociedad hiperconectada, en la que el acceso a la información está literalmente en la palma de nuestra mano. Estas tecnologías han propiciado un cambio en los hábitos de consumo de información, los medios de comunicación tradicionales como la prensa o la televisión han quedado relegados a un segundo plano mientras que Internet se ha consolidado como la principal fuente de información para los usuarios.

El creciente uso del Smartphone[1] y del Internet móvil en la vida cotidiana de las personas es otra de las realidades de la revolución digital que estamos viviendo. Nos permiten hacer infinidad de cosas, desde conocer la frecuencia de paso de los buses de nuestras ciudades, hacer la compra, leer la prensa nacional, etc. Los Smartphone le ofrecen al usuario un amplio abanico de funcionalidades desde la comodidad de un dispositivo portátil que podemos llevar fácilmente en nuestro bolsillo, de esta forma se presentan como el ecosistema de desarrollo perfecto para aplicaciones de consulta de información de forma rápida y cómoda.

A todo lo anterior podemos añadir el desconocimiento generalizado de los ciudadanos respecto a las actividades diarias de nuestro Congreso de los Diputados. La gran mayoría de los ciudadanos desconocen cómo está organizado el Congreso, qué cargos ocupa cada uno de sus diputados, qué propuestas de ley salen adelante, qué iniciativas proponen, etc. Esto contrasta con lo comentado anteriormente, en plena era de la información este desconocimiento es mayormente debido a que dicha información no está presentada al ciudadano de una forma clara y accesible. Normalmente esta información está alojada en sitios web difíciles de consultar desde un dispositivo móvil o que, por la forma en la que están diseñados, la información queda clasificada de forma confusa para el usuario con lo que la navegación se vuelve tediosa hasta el punto de desistir en su búsqueda y abandonar dichas webs.

Otra de las grandes motivaciones que me han llevado a escoger este trabajo de fin de grado ha sido la experiencia laboral adquirida en el último año en el ámbito de las tecnologías web, concretamente en el desarrollo frontend utilizando tecnologías como Javascript o Angular. Dichas tecnologías web están viviendo un momento de gran expansión abarcando incluso el desarrollo en dispositivos móviles, por lo que este trabajo me permitirá aplicar dichas tecnologías al desarrollo móvil y comprobar de primera mano sus principales ventajas e inconvenientes respecto al desarrollo nativo así como seguir adquiriendo conocimientos en los nuevos estándares de la web y las actuales soluciones cloud backend como Firebase.

Por estas razones he considerado que desarrollar una aplicación móvil basada en datos abiertos que unifique todos estos aspectos de la actualidad es una buena solución al problema de accesibilidad de la información pública referente al Congreso de los Diputados.

1.2 Movimiento Open Data

A demás de los motivos de los que hemos hablado, existe otro de gran importancia que es el movimiento Open Data que se ha producido en España desde el año 2010 aproximadamente.

El movimiento Open Data[2] consiste en poner a disposición de los ciudadanos y de la sociedad en general una gran cantidad de información de interés común, que puede proceder de diversas fuentes como organizaciones de la administración pública o empresas privadas que hayan participado en proyectos de ámbito público, de forma libre y sin ánimo de lucro.

Esta información, por lo general, no está presentada en un formato específico siendo los formatos más comunes el XML (en el caso de estar estructurada), los archivos PDF y los documentos en formato Word. Pueden ser de diversas temáticas como, médicos, geográficos, meteorológicos, sobre biodiversidad, de servicios públicos, etc.

Pero este movimiento va más allá de la libre disposición de los datos, se pretende que la sociedad pueda aprovechar estos datos abiertos, que en la mayoría de los casos están destinados a ser archivados ya que estas organizaciones no tienen la capacidad para analizarlos o simplemente no quieren, para construir ideas nuevas que puedan resultar en nuevos datos, conocimiento o incluso nuevos servicios para el conjunto de la sociedad. Se trata de no poner barreras al conocimiento y que la sociedad pueda encontrar nuevas oportunidades de negocio

Los datos abiertos son especialmente útiles en el ámbito docente (aún más si cabe en el ámbito de la Informática). Tanto alumnos como profesores pueden disponer de una enorme cantidad de datos interesantes con los que trabajar que, de otra forma, solo serían accesibles mediante un largo y tedioso proceso de solicitud o simplemente no habría posibilidad de usarlos.

Existen varios ejemplos de este movimiento como puede ser el Proyecto Aporta que persigue la reutilización de los datos públicos, Open Data Euskadi, y el portal Dades Obertes de la Generalitat de Catalunya.

Por nuestra parte, con este proyecto pretendemos sumarnos a este movimiento y aportar una herramienta más para la transparencia de las organizaciones gubernamentales de nuestro país, en este caso el Congreso de los Diputados.

1.3 Objetivos

El objetivo principal del presente trabajo es el desarrollo de una aplicación móvil multiplataforma basada en datos abiertos sobre el Congreso de los Diputados que permita acceder a información referente a dicha cámara de forma cómoda y clara para el ciudadano. Así mismo, se persiguen otros objetivos relacionados en mayor o menor medida con el objetivo principal. Estos objetivos son:

- **Fomentar la Transparencia:** La aplicación pretende ser un ejemplo de transparencia en las instituciones presentando información como declaración de bienes y de ingresos únicamente de los Diputados que hayan decidido hacerla pública.

- **Fomentar la curiosidad política:** Presentando los datos de una forma accesible y amigable se persigue que el ciudadano se interese por el día a día del Congreso de los Diputados.

- **Valor añadido:** La aplicación no pretende ser una simple visualización de datos, busca darle un valor añadido a la información mostrándola de forma visual mediante gráficas y estadísticas.

- **Diseño usable y responsive:** Se pretende desarrollar la interfaz de usuario siguiendo los principios del diseño web responsive (adaptativo) y de la usabilidad (facilidad de uso).

- **Diseño Multiplataforma:** Se persigue la portabilidad de la aplicación a distintas plataformas como iOS, Android, Windows Phone o incluso la Web utilizando tecnologías web estándar.

- **Enfoque Web:** Haciendo uso de las principales tecnologías web del momento se pretende dar una visión novedosa del desarrollo móvil presentando sus puntos fuertes y débiles frente al desarrollo nativo.

- **Enfoque personal:** Desde un punto de vista personal quiero aprovechar este trabajo para adquirir y ampliar conocimientos sobre nuevos frameworks como Ionic, el nuevo estándar ECMAScript6 y el lenguaje de programación TypeScript, que me posicionen como un profesional de referencia en las tecnologías web.

1.4 Estructura

En este apartado explicaré brevemente la estructura seguida para la redacción de este trabajo.

En el **capítulo 1** se desarrolla la introducción a este Trabajo de Fin de Grado, explicando las motivaciones que me han llevado a proponerlo, lo objetivos principales y su estructura por capítulos.

En el **capítulo 2** presentamos un análisis del estado actual de las aplicaciones móviles sobre política en el ámbito nacional. Expondrá las soluciones ya existentes identificando funcionalidades comunes y extrayendo las funcionalidades que nos aporten un valor añadido.

En el **capítulo 3** se muestra la especificación inicial de la aplicación mediante diagramas de casos de usos y sus mockups relacionados. También veremos el modelo de datos utilizado por la aplicación.

En el **capítulo 4** explicaremos a fondo las diversas tecnologías web elegidas para llevar a cabo la implementación de la aplicación y el porqué de esa elección.

En el **capítulo 5** se presentará la arquitectura global de la aplicación y se expondrán los aspectos más relevantes del proceso de implementación.

En el **capítulo 6** veremos los resultados finales del proceso de desarrollo de la aplicación junto con su rendimiento.

En el **capítulo 7** se presentarán las conclusiones del proyecto haciendo una valoración personal sobre las tecnologías utilizadas, el grado de satisfacción con los objetivos planteados en la introducción y las posibles mejoras o funcionalidades como trabajo futuro.

En el **capítulo 8** se recopilan todas las referencias utilizadas en la redacción de este Trabajo de Fin de Grado.

2. Estado del Arte

Curiosamente no existen demasiadas aplicaciones sobre política en las principales tiendas como Google Play o App Store. Navegando por estas Stores podemos encontrar aplicaciones que ofrecen información sobre este tema, pero, en su mayoría, es información de otros países o la presentan de forma confusa. Un ejemplo puede ser "Easy Politica", una aplicación centrada en el sistema político italiano, que permite a los ciudadanos indagar sobre el funcionamiento de éste y conocer el texto de su constitución. Centrándonos en el ámbito nacional tenemos aplicaciones como "Pablo Iglesias Casta Wars" o "PolitiKO" dos aplicaciones de corte satírico que no aportan información sobre la actualidad política pero que pretenden caricaturizar a los políticos que en ellas aparecen, síntoma del descontento general de la sociedad. Entre todas ellas destaca "Congreso 2.0" una aplicación disponible tanto para Android como para iOS que nos ofrece información sobre la actividad legislativa del Congreso de los Diputados.

2.1 "Easy Politica"

Web Site: https://www.facebook.com/pages/Easy-Politica/162002350633994

Easy Politica consiste en una aplicación que permite a los usuarios aprender sobre el sistema político italiano y su constitución de forma fácil e intuitiva.

Figura 1. Easy Politica, Interfaz de usuario

Estas son las principales funcionalidades de Easy Politica:

- **Constitución**: Permite a los ciudadanos leer artículos completos de la constitución de su país para conocerla más a fondo.

- **Gobierno actual**: Permite conocer la composición actual del gobierno.

- **Miembros**: Permite conocer los principales cargos del gobierno actual.

- **Información de contacto**: La aplicación ofrece información para ponerte en contacto con los órganos de gobierno de Italia.

- **División de Poderes**: Ofrece información al usuario sobre la forma en que se dividen los poderes Ejecutivo, Legislativo y Judicial dentro del estado italiano y su distribución.

- **Instituciones**: Permite visualizar imágenes de las instituciones del estado italiano, así como conocer su historia

- **Noticias**: Muestra en tiempo real las últimas noticias relacionadas con la vida política italiana.

La aplicación pretender ser una fuente de conocimiento para los usuarios más que un canal de transparencia y en el que poder expresar su opinión. Tiene varias funcionalidades interesantes como la que te permite conocer la historia de las instituciones, así como visualizar imágenes de éstas, sin embargo, es una aplicación un tanto limitada ya que no te muestra un listado completo de los integrantes de la cámara principal de gobierno, sino un listado de los integrantes del gobierno actual. Tampoco presenta una interfaz de búsqueda de integrantes, ni de instituciones por lo que puede llegar a ser complicado navegar por sus listas.

2.2 "Congreso 2.0"

Web Site: www.congresodoscero.org

Como se indica en la propia descripción de la aplicación en las tiendas de aplicaciones, Congreso 2.0 te permite acceder a información legislativa sobre el Congreso de los Diputados de España. La aplicación es capaz de proporcionar información acerca de las propuestas de ley que están en tramitación, su fecha de presentación e incluso su texto íntegro.

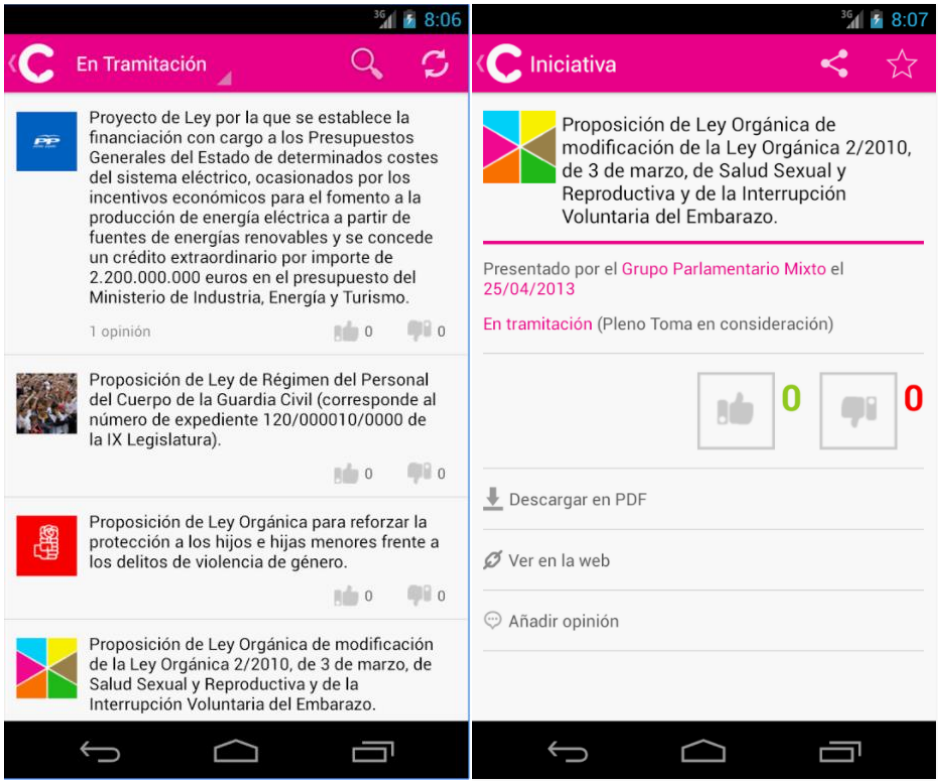


Figura 2. Congreso 2.0, Interfaz de usuario

A continuación, veremos más a fondo sus principales características:

- **Iniciativas en Tramitación**: La aplicación te permite ver las iniciativas que actualmente se encuentran en tramitación presentándolas en forma de lista.

- **Iniciativas Aprobadas**: Consiste en un listado de iniciativas que han sido aprobadas por el Congreso de los Diputados.

- **Iniciativas Rechazadas**: Al igual que los dos listados anteriores, este nos presenta las iniciativas

- **Iniciativas Favoritas**: La aplicación te permite marcar las iniciativas como favoritas, con lo que podemos acceder al listado de iniciativas favoritas.

- **Actualidad**: En este listado podremos ver todas las iniciativas disponibles tanto si están en tramitación como si han sido aprobadas y rechazadas ordenadas por su fecha de presentación, mostrando en primer lugar las más actuales.

- **Mis votos**: Te permite conocer el listado de iniciativas en las que has votado de forma ficticia.

- **Búsqueda simple**: Mediante un campo de texto, podemos realizar búsquedas simples en las diferentes listas de la aplicación, o realizar una búsqueda general de iniciativas. Esto significa que solo podemos encontrar iniciativas que contengan en su título el texto que introduzcamos.

- **Detalle de Iniciativas**: Pulsando en una de las iniciativas listadas accedemos a una vista de detalle que nos ofrece información más específica sobre dicha iniciativa.

- **Voto Ficticio**: Accediendo al detalle de una iniciativa podemos votar a favor o en contra de forma ficticia y así dar nuestra opinión de forma anónima.

- **Comentar Iniciativa**: Podemos añadir de forma anónima comentarios en cualquiera de las iniciativas a las que accedamos.

- **Descargar en PDF**: Podemos descargar la iniciativa integra en pdf en nuestro dispositivo.

- **Ver en la Web**: Nos redirige a la web oficial del Congreso de los Diputados donde tenemos la información completa sobre la iniciativa.

Como podemos ver, esta aplicación ofrece un conjunto de funcionalidades algo reducido.Como principal aplicación competidora a nivel nacional, no ofrece información sobre el Congreso de los Diputados más allá de las iniciativas, como por ejemplo sobre los grupos parlamentarios que lo componen y sus integrantes. Con el desarrollo de PoliticApp se pretende cubrir la mayor parte de las funcionalidades que ofrece Congreso 2.0 además de ofrecer otras como valor añadido de forma que nuestra aplicación pueda posicionarse como una alternativa sólida en las tiendas de aplicaciones. A continuación, veremos una comparativa funcional entre Congreso 2.0 y nuestra alternativa PoliticApp, principalmente, y en menor medida con la aplicación Easy Politica.

2.3 Comparativa Funcional

Como ya hemos comentado, PoliticApp pretende añadir funcionalidad que aporte valor a la aplicación frente a otras del mismo tipo. Mediante un sistema tipo MoSCoW[3] priorizaremos dichas funcionalidades en función de la utilidad que el usuario medio pueda darles, excluiremos las que consideremos de menor valor y haremos una comparativa con Congreso 2.0 y Easy Politica.

- **Must**:
    - *Listado de Iniciativas*: Al igual que Congreso 2.0, nuestra app debe ser capaz de mostrar el listado de iniciativas del congreso y permitir acceder al detalle al pulsar en ellas. Esto supone un añadido con respecto a la aplicación Easy Politica.
    - *Comentar Iniciativa*: Esta funcionalidad permitirá a los usuarios expresar su opinión acerca de las iniciativas en las que centren su interés. Es un valor añadido frente a Easy Politica.
    - *Votación Ficticia*: Siguiendo en la línea de permitir a los usuarios expresar su opinión personal, la votación ficticia es una funcionalidad clave en nuestra aplicación.
    - *Listado de Diputados*: Mostrará un listado de los diputados actuales del congreso, esto supone una funcionalidad añadida con respecto a Congreso 2.0 ya que nos permitirá conocer más a fondo a nuestros representantes políticos. Respecto a Easy Politica, nuestra funcionalidad ofrece un listado completo de la cámara frente al listado parcial que ésta ofrece.
    - *Búsqueda Avanzada*: La búsqueda avanzada permitirá al usuario buscar, no solo por un campo de texto, sino por parámetros como el grupo parlamentario al que pertenece un diputado, la fecha de presentación de las iniciativas, etc. Esta funcionalidad también supone un añadido respecto al resto de aplicaciones políticas.
- **Should**:
    - *Órganos de Gobierno*: Ofrecerá información sobre los distintos órganos de gobierno que conforman la cámara como la Junta de Portavoces y la Diputación Permanente, y sus principales integrantes. En este apartado Easy Politica va más allá ofreciendo la historia de las instituciones políticas a los usuarios.
    - *Grupos Parlamentarios*: En esta sección encontraremos información sobre los diferentes grupos parlamentarios que constituyen el total de 350 diputados.

- *Estadísticas*: Podremos ver estadísticas como el porcentaje de representación en la cámara de cada grupo parlamentario o iniciativas aprobadas/rechazadas por grupo parlamentario. Esta funcionalidad supone un añadido frente a las dos aplicaciones mostradas anteriormente.

- **Could**:

  - *Autenticación en la App*: Esta funcionalidad la encontramos en el apartado "Could" debido a que no todos los usuarios estarían dispuestos a identificarse en una app política y expresar su opinión abiertamente. Para implementar esta funcionalidad deberíamos realizar una encuesta de opinión acerca de esta nueva característica una vez lanzada en las tiendas de aplicaciones.

- **Won't Have**:

  - *Ver en la Web*: Se ha optado por descartar esta funcionalidad debido al enfoque inicial de PoliticApp. La web oficial del congreso tiene un estilo algo anticuado y no está diseñada de forma responsive, además de esto, la web presenta la información de manera confusa y es tedioso navegar por ella. Por estos motivos, he considerado que mostrar la web oficial dentro de la interfaz de la aplicación va en contra del planteamiento inicial del proyecto.

  - *Descargar en PDF*: Esta funcionalidad también ha sido descartada ya que no cumple con los objetivos del proyecto. PoliticApp pretende presentar la información de forma amigable para los usuarios por lo que, descargar archivos .pdf con textos de leyes completos se ha considerado innecesario.

	Easy Politica	Congreso 2.0	PoliticApp
Listado de Iniciativas	✘	✔	✔
Votación Ficticia	✘	✔	✔
Comentar Iniciativa	✘	✔	✔
Búsqueda Avanzada	✘	✘	✔
Lista de Diputados	◐	✘	✔
Descargar en PDF	✘	✔	✘
Ver en la Web	✘	✔	✘
Órganos de Gobierno	✔	✘	✔
Grupos Parlamentarios	◐	✘	✔
Estadísticas	✘	✘	✔

Tabla 1. Comparativa Funcional entre Congreso 2.0, Easy Politica y PoliticApp

Como podemos apreciar de forma visual en la Tabla 1, PoliticApp cubre las funcionalidades principales de las dos apps más cercanas a la nuestra para partir de una base sólida de cara a la competencia en las Stores, a la vez que añade nuevas características que pueden resultar muy interesantes para los usuarios como la búsqueda avanzada tanto de diputados como de iniciativas y la pantalla de estadísticas.

Por otra parte, se ha prescindido de características tales como la visualización de la web oficial del congreso, o la descarga en pdf del texto completo de las iniciativas presentadas en el congreso puesto que no están claramente alineadas con los objetivos del proyecto.

3. Diseño

El diseño de la aplicación se ha enfocado desde el punto de vista funcional mediante la especificación de casos de uso[4], junto con sus mockups asociados. Al tratarse de una aplicación móvil enfocada en el desarrollo web y con Firebase como parte backend, no presenta un diseño tradicional a 3 capas. En su lugar utilizaremos un modelo Fat-Client donde la mayor carga de cómputo quedará delegada en la capa cliente la cual extraerá los datos de llamadas http a la base de datos de Firebase.

En este mismo apartado presentaremos el modelo de datos de la aplicación con las principales entidades involucradas, sus relaciones y su cardinalidad.

3.1 Casos de Uso

3.1.1 Lista de Diputados (UC-01)

UC-01	**Lista de Diputados**	
Objetivos	Consultar Diputados	
Descripción	La aplicación debe listar todos los Diputados disponibles en BDD	
Precondición	Ninguna	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario despliega el menú lateral de la aplicación
	2	El usuario pulsa en Diputados en el menú lateral
	3	Se muestra una nueva pantalla con un listado de los Diputados disponibles
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
	--	--
Frecuencia esperada	100 veces/día	
Comentarios	Ninguno	

Tabla 2. Caso de Uso: Lista de Diputados (UC-01)

3.1.2 Búsqueda Avanzada (UC-02)

UC-02	**Búsqueda Avanzada**	
Objetivos	Realizar búsquedas con parámetros	
Descripción	La aplicación debe buscar según los parámetros que le indiquemos (en varias listas)	
Precondición	El usuario ha accedido a un listado de la aplicación	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario pulsa en el botón de búsqueda de la lista en la que se encuentra
	2	El usuario elige los parámetros de búsqueda
	3	El usuario pulsa el botón buscar e inicia la búsqueda
	4	Se muestra nuevamente la lista en la que nos encontrábamos, pero esta vez filtrada
Postcondición	La lista debe estar correctamente filtrada. No estará ordenada.	
Excepciones	**Paso**	**Acción**
	2	Si el usuario inicia la búsqueda sin parámetros, la aplicación le informa de que no puede realizar la búsqueda y le permite seguir eligiendo parámetros
Frecuencia esperada	25 veces/día	
Comentarios	Ninguno	

Tabla 3. Caso de Uso: Búsqueda Avanzada (UC-02)

3.1.3 Lista de Iniciativas (UC-03)

UC-03	**Lista de Iniciativas**	
Objetivos	Consultar Iniciativas	
Descripción	La aplicación debe listar las Iniciativas disponibles	
Precondición	Ninguna	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario despliega el menú lateral de la aplicación
	2	El usuario pulsa en Iniciativas en el menú lateral
	3	Se muestra una nueva pantalla con un listado de las Iniciativas disponibles
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
	--	--
Frecuencia esperada	100 veces/día	
Comentarios	Ninguno	

Tabla 4. Caso de Uso: Lista de Iniciativas (UC-03)

3.1.4 Detalle de Entidad (UC-04)

UC-04	**Detalle de Entidad**	
Objetivos	Acceder al detalle de las entidades	
Descripción	La aplicación debe permitir consultar datos detallados de las entidades (diputados, Iniciativas)	
Precondición	El usuario debe estar en una lista	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario pulsa en una entidad de la lista
	2	Se muestra una nueva pantalla con información detallada de la entidad seleccionada
Postcondición	La información debe pertenecer a la entidad seleccionada	
Excepciones	**Paso**	**Acción**
	--	--
Frecuencia esperada	200 veces/día	
Comentarios	Ninguno	

Tabla 5. Caso de Uso: Detalle de Entidad (UC-04)

3.1.5 Comentar Iniciativa (UC-05)

UC-05	**Comentar Iniciativa**	
Objetivos	Añadir un comentario a una Iniciativa	
Descripción	La aplicación debe permitir expresar su opinión al usuario.	
Precondición	El usuario debe acceder al detalle de una Iniciativa	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario pulsa en el botón comentarios.
	2	Se muestra un textarea que permite añadir un comentario.
	3	Al pulsar en el botón comentar, el comentario se añade a la Iniciativa
Postcondición	El comentario se muestra a continuación del último comentario de la Iniciativa	
Excepciones	**Paso**	**Acción**
	3	Si el textarea está vacío, la aplicación informa de que no se puede añadir un comentario vacío
Frecuencia esperada	3 veces/día	
Comentarios	Ninguno	

Tabla 6. Caso de Uso: Comentar Iniciativa (UC-05)

3.1.6 Votar Iniciativa (UC-06)

UC-06	**Votar Iniciativa**	
Objetivos	Añadir un voto a una Iniciativa	
Descripción	Debe permitir al usuario votar a favor o en contra	
Precondición	El usuario debe acceder al detalle de una Iniciativa	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario pulsa en el icono de Like o Dislike
	2	Se añade un voto positivo o negativo a la Iniciativa
Postcondición	El voto se suma al total de votos	
Excepciones	**Paso**	**Acción**
	1	Si el usuario pulsa el botón de like o dislike cuando ya ha sido pulsado, el voto se elimina de la iniciativa
Frecuencia esperada	20 veces/día	
Comentarios	Ninguno	

Tabla 7. Caso de Uso: Votar Iniciativa (UC-06)

3.1.7 Órganos de Gobierno (UC-07)

UC-07	**Órganos de Gobierno**	
Objetivos	Consultar los órganos de gobierno	
Descripción	Mostrará información sobre los distintos órganos de gobierno existentes, sus integrantes, etc.	
Precondición	Ninguna	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario despliega el menú lateral de la aplicación
	2	El usuario pulsa en Órganos de gobierno en el menú lateral
	3	Se muestra una nueva pantalla con las opciones disponibles.
	4	El usuario selecciona una de las opciones
	5	Se muestra la información asociada a ese órgano de gobierno
Postcondición	La información se corresponde con el órgano de gobierno seleccionado	
Excepciones	**Paso**	**Acción**
	--	**--**
Frecuencia esperada	40 veces/día	
Comentarios	Ninguno	

Tabla 8. Caso de Uso: Órganos de Gobierno (UC-07)

3.1.8 Grupos Parlamentarios (UC-08)

UC-08	**Grupos Parlamentarios**	
Objetivos	Consultar los grupos parlamentarios	
Descripción	Mostrará información sobre los distintos grupos parlamentarios que conforman la cámara	
Precondición	Ninguna	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario despliega el menú lateral de la aplicación
	2	El usuario pulsa en Grupos Parlamentarios en el menú lateral
	3	Se muestra una nueva pantalla con las opciones disponibles.
	4	El usuario selecciona una de las opciones
	5	Se muestra la información asociada a ese Grupo Parlamentario
Postcondición	La información se corresponde con Grupo Parlamentario escogido	
Excepciones	**Paso**	**Acción**
	--	--
Frecuencia esperada	60 veces/día	
Comentarios	Ninguno	

Tabla 9. Caso de Uso: Grupos Parlamentarios (UC-08)

3.1.9 Estadísticas (UC-09)

UC-09	**Estadísticas**	
Objetivos	Consultar estadísticas del congreso	
Descripción	Se mostrarán visualmente mediante gráficas estadísticas relacionadas con el congreso	
Precondición	Ninguna	
Secuencia Normal	**Paso**	**Acción**
	1	El usuario despliega el menú lateral de la aplicación
	2	El usuario pulsa en Estadísticas en el menú lateral
	3	Se muestran las gráficas con las estadísticas disponibles
Postcondición	Ninguna	
Excepciones	**Paso**	**Acción**
	--	--
Frecuencia esperada	45 veces/día	
Comentarios	Ninguno	

Tabla 10. Caso de Uso: Estadísticas (UC-09)

3.2 Mockups

Los mockups[5] que veremos a continuación están asociados a cada uno de los casos de uso presentados en la sección anterior. En ellos se muestra el flujo de navegación de forma jerárquica entre las distintas vistas de la aplicación.

- **Lista de Diputados (UC-01)**

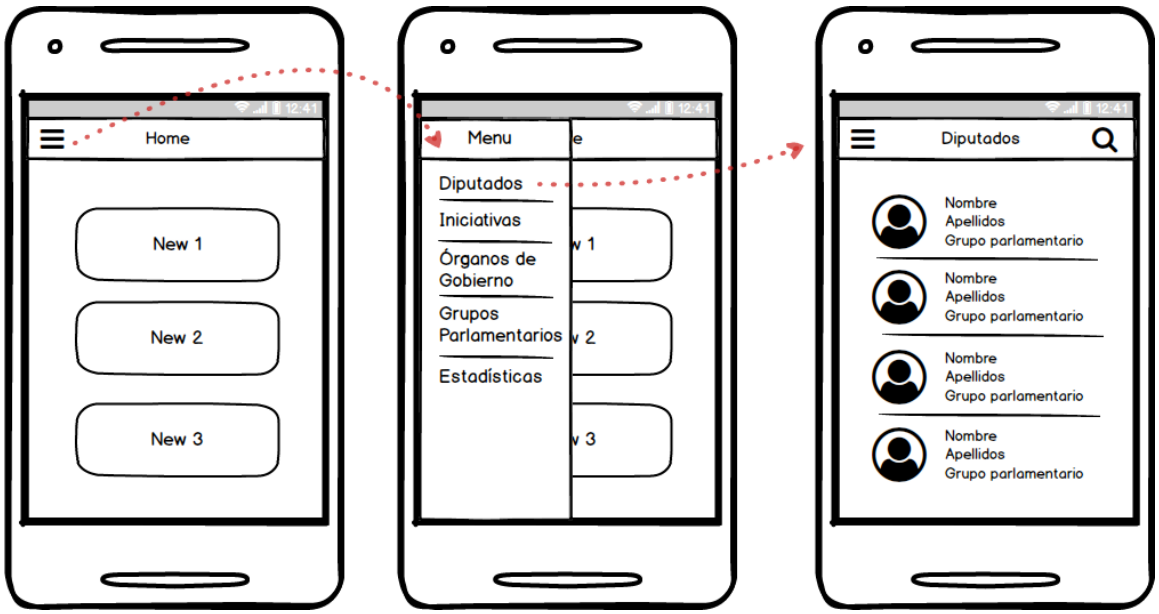


Figura 3. Mockup: Lista de Diputados (UC-01)

- **Búsqueda Avanzada (UC-02)**

En este caso de uso tenemos dos vistas diferenciadas en función de la entidad buscada:

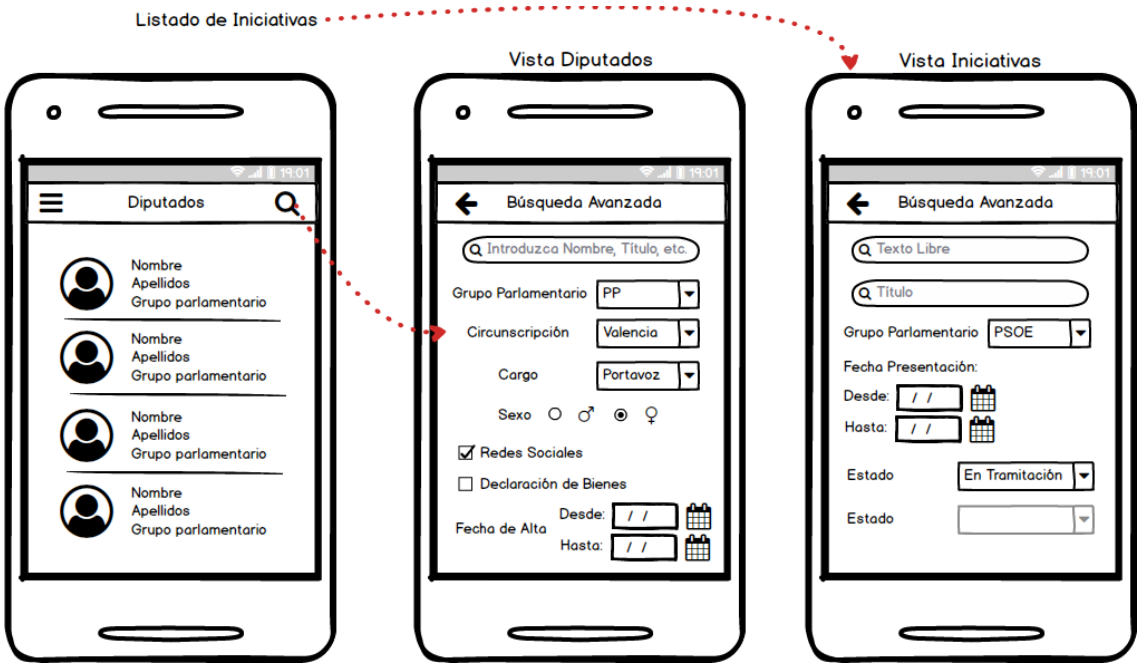


Figura 4. Mockup: Búsqueda Avanzada (UC-02)

- **Lista de Iniciativas (UC-3)**

Partiendo de la vista "Home" (ya presentada en la Figura. 2) desplegando el menú lateral accedemos a la vista del Listado de Iniciativas (Figura. 4).

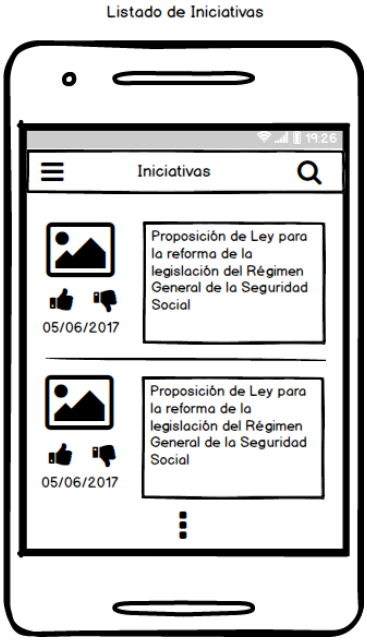


Figura 5. Mockup: Lista de Iniciativas (UC-03)

- **Detalle de Entidad (UC-04)**

En este caso de uso también nos encontramos con dos vistas dependiendo de si accedemos al detalle de una entidad u otra (Figura. 2 y Figura. 4).

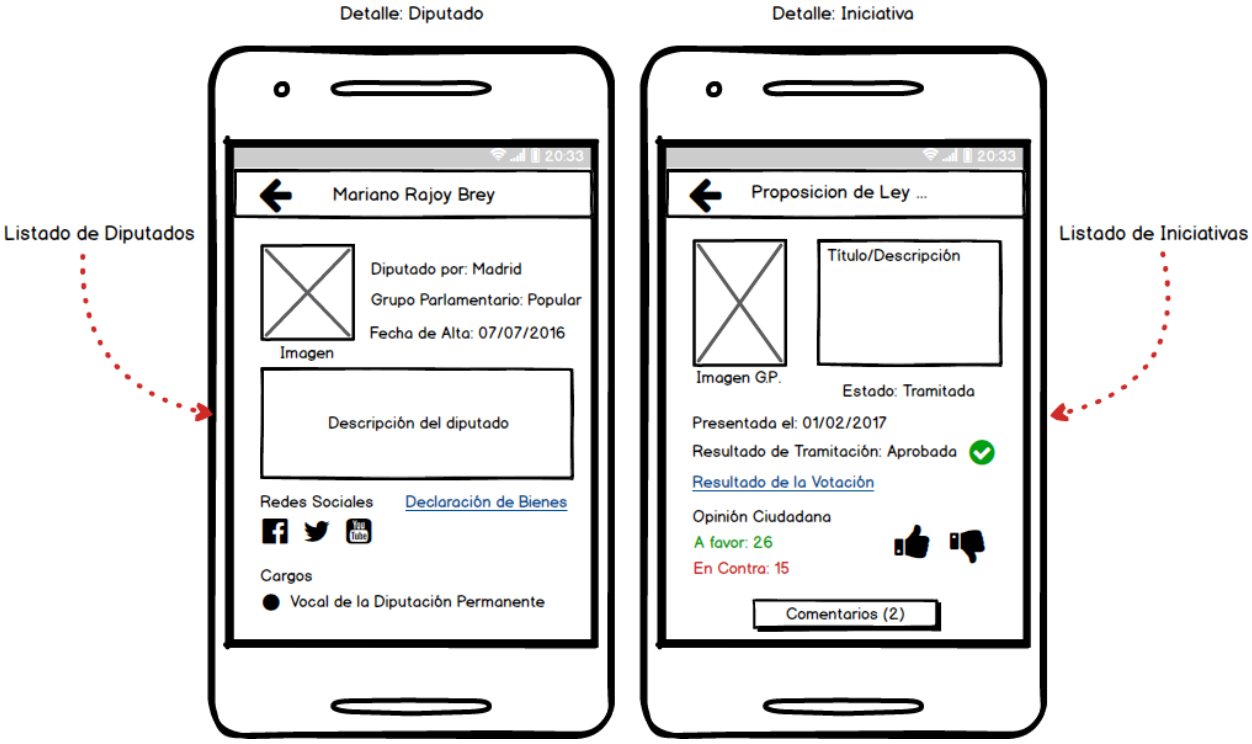


Figura 6. Mockup: Detalle de Entidad (UC-04)

- **Comentar Iniciativa (UC-05)**

Desde la pantalla de detalle de una Iniciativa (Figura 5.) podemos acceder a sus comentarios mediante el botón “Comentarios”

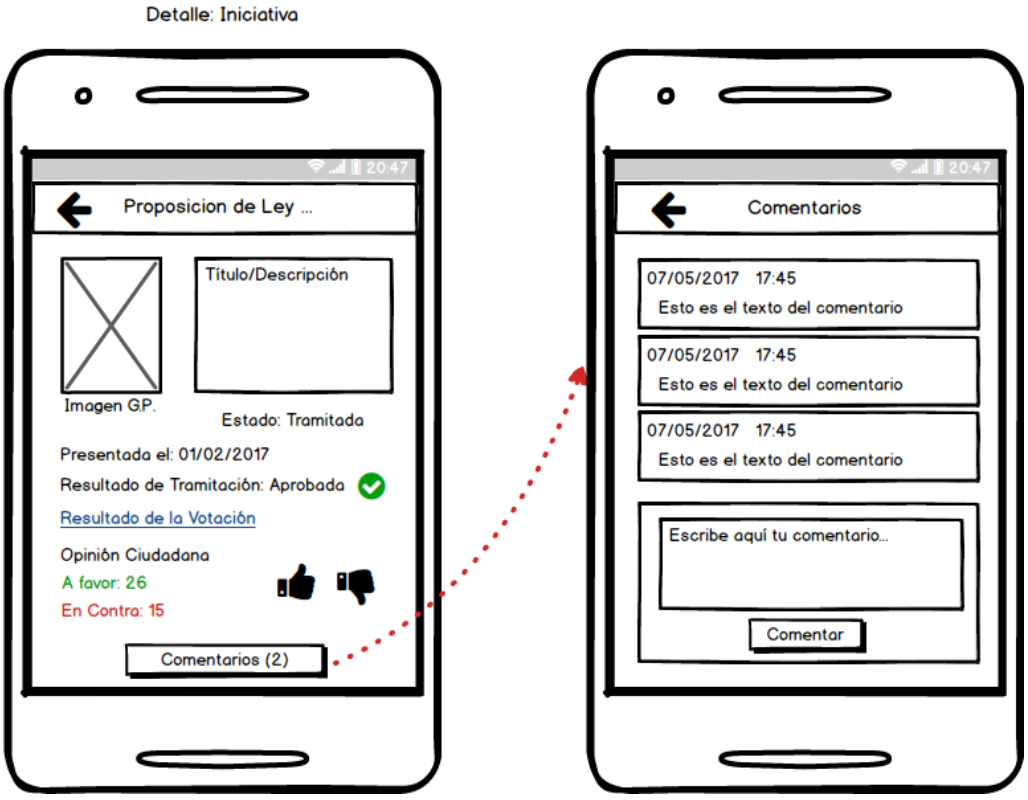


Figura 7. Mockup: Comentar Iniciativa (UC-05)

- **Votar Iniciativa (UC-06)**

El caso de uso Votar Iniciativa está contenido en la pantalla de detalle de una iniciativa (Figura 5.) y no requiere de una nueva pantalla. Al pulsar en los iconos de *like* y *dislike* se actualizan los valores “A favor” y “En contra”.

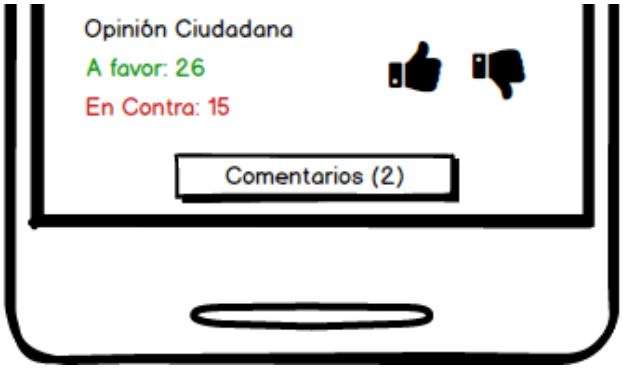


Figura 8. Mockup: Votar Iniciativa (UC-06)

- **Órganos de Gobierno (UC-07)**

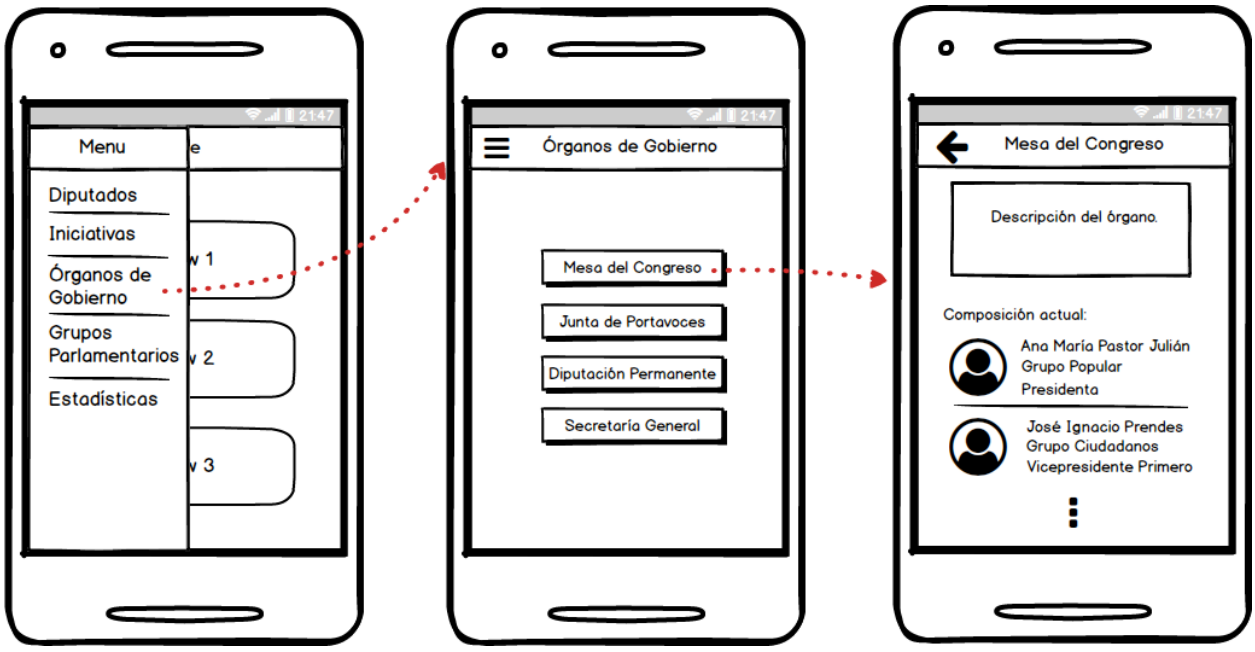


Figura 9. Mockup: Órganos de Gobierno (UC-07)

- **Grupos Parlamentarios (UC-08)**

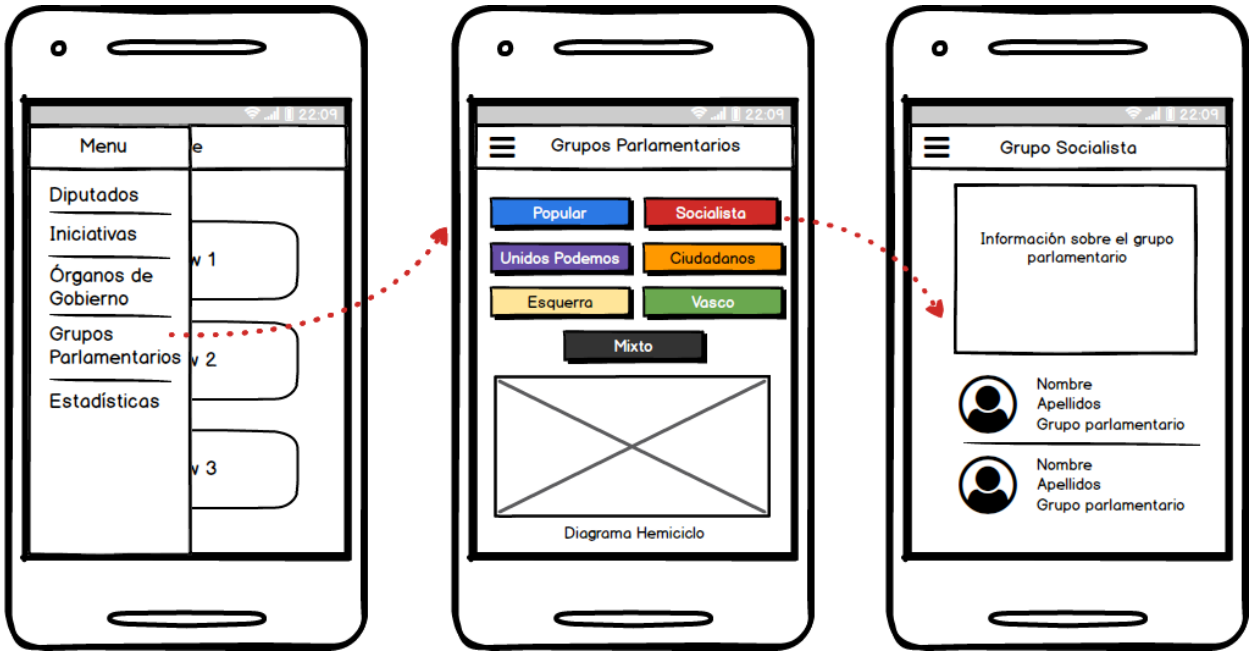


Figura 10. Mockup: Grupos Parlamentarios (UC-08)

- **Estadísticas (UC-09)**

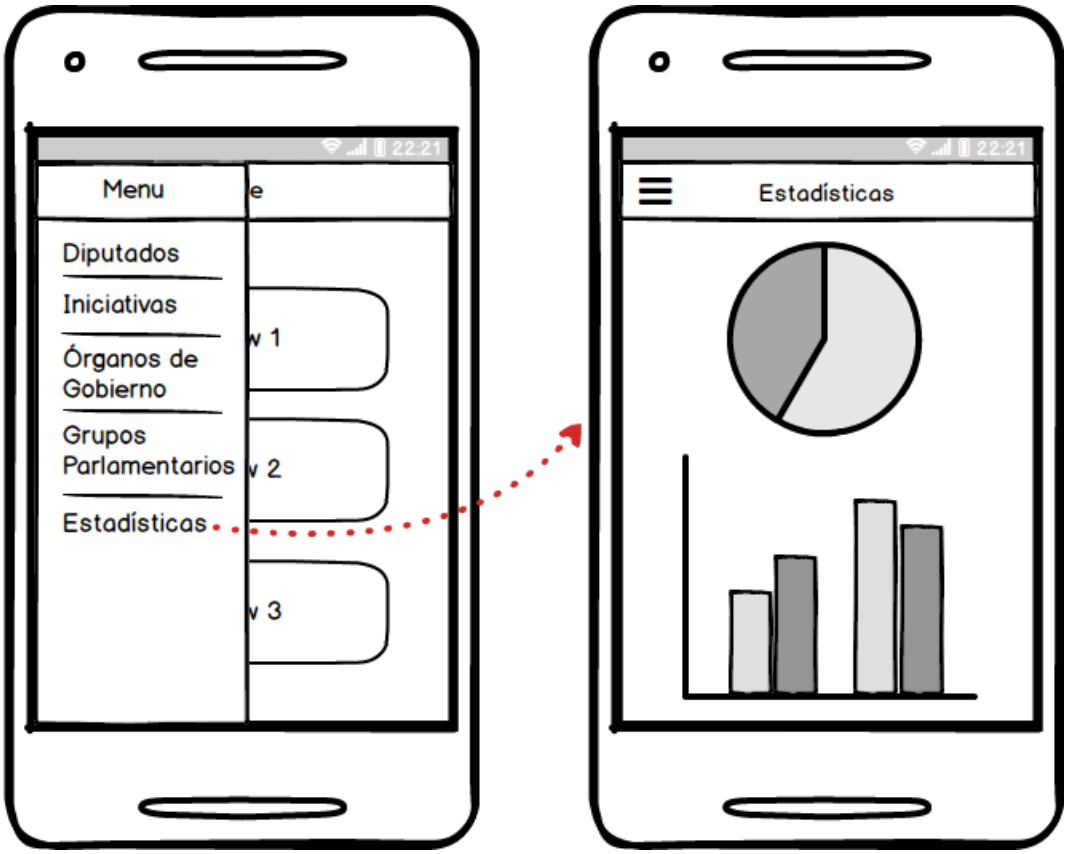


Figura 11. Mockup: Estadísticas (UC-09)

3.3 Modelo de Datos

En el modelo de datos de nuestra aplicación podemos identificar 5 entidades principales que se almacenarán en la base de datos de Firebase y de las cuales harán uso las diferentes pantallas de nuestra aplicación, recuperándolas y modificándolas cuando sea necesario. En base de datos también se almacenarán las listas de valores posibles de los campos desplegables, de esta forma si en un futuro se deben cambiar estos valores bastaría con modificar la base de datos en lugar de cada uno de los archivos HTML de nuestra aplicación, con esto ganamos en mantenibilidad de cara a futuras modificaciones. Estas listas de valores no se han considerado entidades principales como tal por lo que no están incluidas en nuestro modelo de datos.

3.3.1 Entidades

- **Diputado**: Se trata del modelo de un diputado del Congreso.

Figura 12. Modelo de Diputado.

- **Iniciativa**: Modelo de una iniciativa parlamentaria como pueden ser Proyectos de Ley o Reales Decretos. Serán modificadas en base de datos cuando un usuario realice una votación o inserte un comentario.

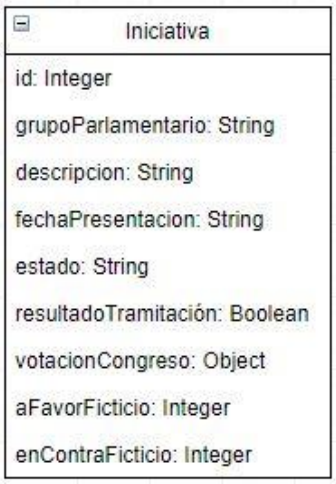


Figura 13. Modelo de Iniciativa

- **Comentario**: Modelo básico de comentario asociado a una iniciativa. Mediante los comentarios, los usuarios expresaran su opinión acerca de las iniciativas en las que estén interesados.

Figura 14. Modelo de Comentario

- **Grupo Parlamentario**: Representa los diferentes grupos parlamentarios del Congreso de los Diputados como el grupo Socialista, o el grupo Mixto.

Figura 15. Modelo de Grupo Parlamentario

- **Órgano de Gobierno**: Representa cada uno de los Órganos de Gobierno internos del Congreso de los Diputados.

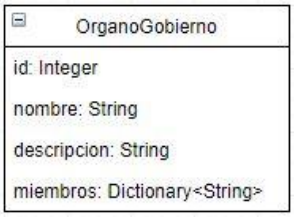


Figura 16. Modelo de Órgano de Gobierno

3.3.2 Diagrama de Entidades

En la figura 17 que veremos a continuación podemos comprobar de forma visual las relaciones entre las entidades presentadas en la sección anterior. Se indica además la cardinalidad de dichas relaciones indicando en cada caso su máximo o mínimo.

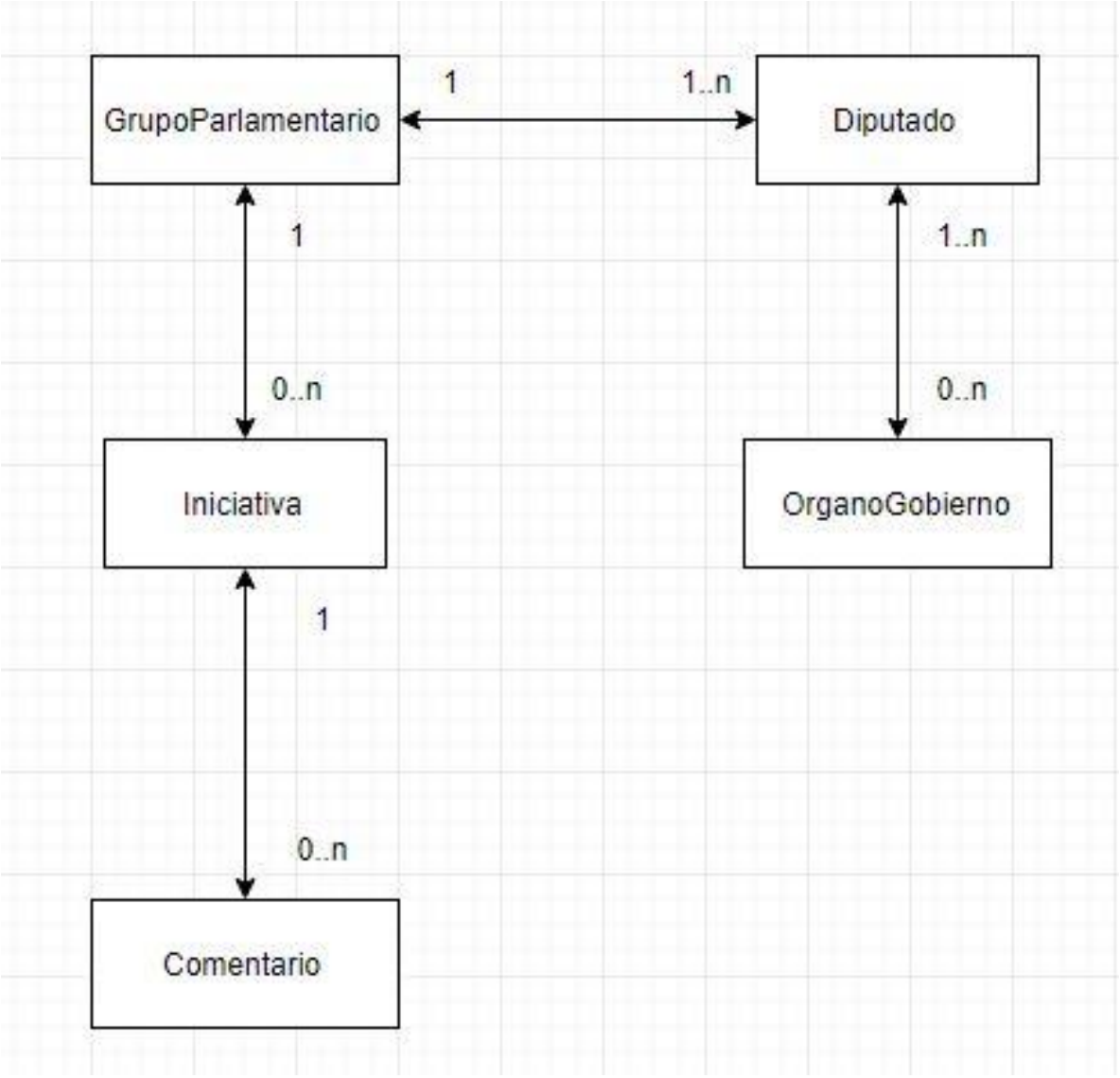


Figura 17. Diagrama de Entidades

4. Tecnologías

Desde hace unos años el desarrollo web ha experimentado una etapa de notable crecimiento frente al desarrollo para escritorio tradicional, las mejoras en lenguajes como HTML con su última versión HTML5 que permiten a los desarrolladores generar interfaces de forma rápida y sencilla, entre otras, han propiciado este crecimiento. Lenguajes como JavaScript se han convertido en la primera opción para muchos desarrolladores que han visto en el mundo de las tecnologías web la oportunidad de crear aplicaciones muy potentes con unos costes de desarrollo y formación menores que en entornos para escritorio. Por otra parte, en el ámbito empresarial la tendencia actual es la migración de muchas de las aplicaciones ya existentes de gestión hacia la web e incluso desarrollar sus nuevas aplicaciones en entornos web en los que solo necesitamos un navegador para tener la aplicación en funcionamiento.

Con un panorama tan prometedor no es de extrañar la aparición de frameworks de desarrollo web basados en JavaScript tales como AngularJS de Google, o React de Facebook. Estos frameworks, que gozan de una tremenda popularidad entre la comunidad de desarrolladores web, nos brindan un entorno de trabajo claro y estructurado que sigue una metodología de desarrollo concreta, incluyendo patrones de diseño, con la que poder desarrollar todo tipo de aplicaciones web.

Los últimos avances en el desarrollo web junto con el nuevo estándar para JavaScript conocido como ECMAScript6 nos permiten incluso la programación orientada a objetos, algo impensable hace apenas un par de años, además se han introducido conceptos como la herencia lo que hace la programación web actual mucho más cercana a la programación tradicional. Pero eso no es todo, con la aparición de TypeScript por parte de Microsoft, el cual cuenta con un gran apoyo de la comunidad web, se introdujo el tipado y la compilación al lenguaje JavaScript por lo que una gran parte de la comunidad está empezando a utilizar este superset de JavaScript del que hablaremos más a fondo en las siguientes secciones. Con todas estas mejoras en un lapso de tiempo relativamente corto, las tecnologías web han conseguido abrirse hueco incluso en entornos de desarrollo móvil con frameworks como Ionic basado en Angular, de esta forma un desarrollador web no necesita aprender una nueva tecnología para introducirse en proyectos de desarrollo para móvil.

Ahora pasaremos a ver de forma más detallada cada una de las tecnologías escogidas para el desarrollo de este proyecto.

4.1 Angular

Angular, comúnmente conocido como Angular2, es uno de los frameworks para desarrollo web más populares y demandados del momento siendo utilizado en el ámbito empresarial para desarrollo de aplicaciones de todo tipo, desde aplicaciones bancarias pasando por aplicaciones de gestión, venta online, etc. Este framework, respaldado por Google, ha sufrido grandes cambios desde que vio la luz su primera versión llamada AngularJS en el año 2010.

Sus principales mejoras con respecto a AngularJS se centran en su adopción del estándar Web Components[6], recomendado por W3C desde el año 2014 y que permite extender el código HTML creando nuestros propios elementos. Además de esto, Angular ha introducido notables mejoras en el rendimiento respecto a su versión anterior mediante la eliminación del two-way binding característico de AngularJS y adoptando un flujo de datos unidireccional.

4.1.1 Arquitectura del Framework

Ahora veremos los principales elementos que nos ofrece este framework para estructurar nuestras aplicaciones y cómo se relacionan entre sí.

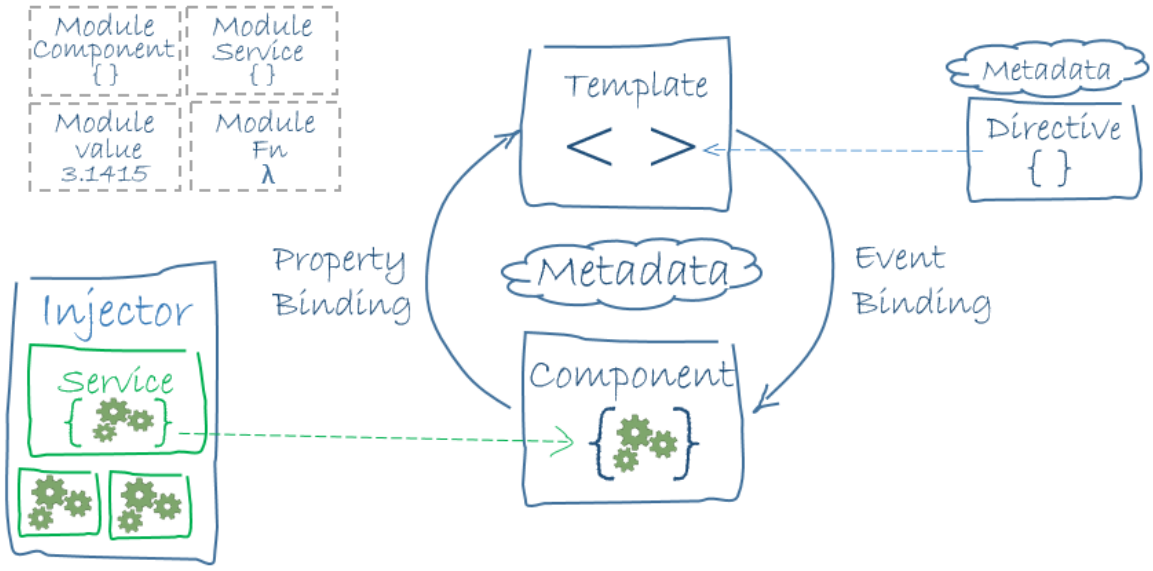


Figura 18. Arquitectura de Angular

Como podemos ver en la figura 18, el framework Angular consta de varios bloques principales:

- **Módulos**

Las aplicaciones Angular son modulares. Esto quiere que están compuestas por módulos de código diferenciados que se encargan de aspectos específicos de la aplicación separando de esta forma las responsabilidades, por ejemplo podríamos tener un módulo encargado de realizar las llamadas a backend y otro encargado de mostrar un calendario y gestionar fechas. Estos módulos pueden contener a su vez componentes angular, servicios, etc.

- **Componentes**

Cada componente en Angular controla una parte de la vista de la aplicación. El componente define propiedades y métodos que estarán disponibles en la vista para dotarla de un comportamiento dinámico, pero, siguiendo los principios de diseño S.O.L.I.D.[7], la lógica de negocio deberá estar contenida en servicios para ser consumidos desde los componentes. Estos componentes podrán ser reutilizados en otras partes de nuestra aplicación.

- **Template**

Los templates nos permiten definir las vistas de nuestros componentes en HTML. Angular permite enriquecer estos templates mediante expresiones, directivas y data bindings para enriquecer el comportamiento de estos.

- **Metadatos**

Los metadatos de Angular nos permiten añadir información de configuración a los bloques de nuestra aplicación como pueden ser los módulos y componentes. Por ejemplo, si tenemos un componente que muestra un calendario, mediante metadatos podemos indicarle el nombre del tag HTML con el que lo utilizaremos en nuestras vistas como podría ser `<custom-calendar></custom-calendar>`.

- **Data Binding**

Angular nos permite enlazar datos, propiedades y funciones a nuestros templates con lo que evitamos tener que manejar la lógica para insertar y actualizar datos en el DOM, lo que anteriormente se realizaba con la líbrería JQuery y llevaba a numerosos errores si no conocías correctamente la estructura del DOM. Angular tiene tres data bindings principales.

- *Interpolación*: Consiste en introducir en el template de un componente entre dobles corchetes una propiedad del componente por ejemplo `{{diputado.nombre}}`. Angular evalúa la interpolación e inserta el valor en el DOM.

- *Property Binding*: Angular enlaza un valor a la propiedad del elemento HTML que le indiquemos. Por ejemplo, `[componenteInput]="diputado"`. De esta forma pasamos información a los componentes de nuestra aplicación.

- *Event Binding*: Angular es capaz de enlazar funciones a los eventos emitidos por los elementos del DOM como por ejemplo un click en un botón: `<button (click) = "seleccionarDiputado(diputado.id)"></button>`. Con esto podemos definir lo que cada elemento del DOM debe realizar cuando se interactúan con ellos.

- **Directivas**

En Angular los templates son dinámicos, cuando el framework los renderiza transforma el DOM basándose en las directivas que encuentra, entonces podemos decir que las directivas son elementos que Angular nos proporciona para generar HTML dinámico en función de nuestras necesidades. Existen dos tipos de directivas:

- *Estructurales*: Este tipo de directivas van precedidas por un asterisco y alteran la propia estructura del DOM eliminando, generando o reemplazando elementos HTML. Por ejemplo, `*ngIf` que inserta un elemento en función de si se cumple la condición indicada o `*ngFor` que nos permite repetir un elemento HTML en base a la iteración de una colección.

- *Directivas Atributo*: Estas directivas actúan como atributos de elementos HTML alterando la apariencia o el comportamiento de dicho elemento. Por ejemplo, `[ngClass]` que nos permite variar sus clases CSS dinámicamente.

- **Servicios**

Los servicios son una parte fundamental de Angular, en ellos encapsulamos desde constantes a funciones que realicen cálculos de temperatura, es decir, toda nuestra lógica de negocio. La responsabilidad de estos servicios debe ser muy concreta como por ejemplo controlar el login en la aplicación, toda esta lógica estaría encapsulada en el servicio de Login. Estos servicios son simples clases javascript (ES6) que contienen propiedades y métodos y pueden ser consumidos por los componentes de nuestra aplicación.

4.2 TypeScript

TypeScript es un nuevo lenguaje de programación, pensado para grandes proyectos, de código abierto y mantenido por Microsoft que apareció en el año 2012 y desde entonces su popularidad ha ido en aumento, tanto que se ha convertido en el lenguaje de programación oficial para Angular. Tal y como hemos comentado anteriormente en este proyecto, se trata de un superconjunto de JavaScript, eso significa que está escrito sobre este lenguaje extendiéndolo y añadiendo nuevos beneficios. Supuso una revolución en el ámbito del desarrollo web debido a que es un lenguaje compilado, su código se compila a JavaScript plano en su versión ES6 proporcionando análisis estático del código, algo impensable con JavaScript hace solo unos años. Al extender la sintaxis de JavaScript, todo el código escrito en este es totalmente compatible con la sintaxis particular de TypeScript, muy cercana al lenguaje C# también respaldado por Microsoft. Estas similitudes con C# han permitido a muchos programadores de escritorio familiarizarse rápidamente con la programación web, siendo esta mucho más cercana a la programación orientada a objetos que hace unos años.

TypeScript añade esencialmente tipado estático. Al contrario que JavaScript que es débilmente tipado y los tipos de sus variables son inferidos, el compilador de TypeScript no permite la asignación de valores de un tipo distinto al indicado en la declaración de sus variables lo que nos evita muchos errores en tiempo de ejecución a la vez que ahorramos en horas de debugging. Otra de sus grandes mejoras es la introducción de objetos basados en clases e interfaces obligando a programar en base a la implementación de estas, generando código mucho más estructurado y mantenible.

El desarrollo en TypeScrip es cada vez más popular debido, en parte, a sus beneficios sobre JavaScript y al soporte que da Microsoft en sus IDEs Visual Studio y Visual Studio Code (del que hablaremos más adelante) mediante plugins y herramientas de ayuda para el proceso de codificación como Intellisense, una herramienta que autocompleta tu código sugiriéndote palabras reservadas del lenguaje, etc. Por estas razones es muy cómodo trabajar con TypeScript y desarrollar código de calidad con él.

4.3 Desarrollo Híbrido frente a Nativo

Uno de los objetivos de este proyecto es mostrar las virtudes del desarrollo web para dispositivos móviles frente al nativo por lo que en esta sección explicaremos en que consiste cada uno.

Antes de la aparición de frameworks híbridos como Ionic o React Native, el desarrollo de aplicaciones móviles estaba muy enfocado al desarrollo nativo. Este tipo de aplicaciones se desarrollan exclusivamente para un sistema operativo utilizando un lenguaje específico para cada plataforma como Swift en iOS o Java en Android. Al estar diseñadas para un sistema operativo en concreto, estas aplicaciones aprovechan mejor las funcionalidades del smartphone lo que se traduce en una mejor experiencia de usuario y un rendimiento mayor que en aplicaciones web o híbridas. Normalmente el coste de desarrollo y de aprendizaje suele ser mayor en las aplicaciones nativas puesto que hay que desarrollar una versión para cada sistema operativo.

Por otro lado, tenemos las aplicaciones híbridas que intentan aunar lo mejor del desarrollo nativo junto con las ventajas de las aplicaciones web. Se desarrollan usando tecnologías web estándar como HTML, CSS y JavaScript por lo que serán ejecutadas por el motor del navegador nativo o WebView (no hay que confundirlo con el propio navegador) esto repercute en un rendimiento menor que las aplicaciones nativas, aunque consiguen mantener unas cotas correctas para una experiencia de usuario satisfactoria. Al igual que las aplicaciones nativas, tienen acceso a las funcionalidades del smartphone como la cámara, el gps, etc aunque de una forma un poco más compleja mediante librerías de plugins nativos como Cordvoa o Phonegap. La gran ventaja de este tipo de aplicaciones reside en su portabilidad, permitiéndonos codificar una única vez la aplicación teniendo la garantía de que nuestro código será compatible con cualquier sistema operativo. Además de esto cualquier persona familiarizada con el desarrollo web podría empezar a desarrollar para móviles casi de inmediato en lugar de aprender varias tecnologías nativas. Aunque las aplicaciones híbridas no presenten una interfaz de usuario tan refinada como las nativas en los últimos años los frameworks existentes han mejorado de forma muy considerable su look&feel adaptándose a cada sistema operativo.

Como hemos visto, no podemos considerar un tipo de desarrollo mejor que el otro, los dos tienen sus virtudes. Sin embargo, el desarrollo híbrido puede ser una muy buena opción cuando los recursos del proyecto son escasos y en aplicaciones que no requieran de un rendimiento crítico, creando en muchos casos aplicaciones de gran calidad con unos costes menores.

Apps Híbridas	Apps Nativas
Desarrollo en HTML, JavaScript y CSS	Desarrollo en tecnología concreta a la plataforma, Java en Android, Swift para iOS, etc.
Un único desarrollo cross platform	Una versión por plataforma
Rendimiento medio	Alto rendimiento
Ciclos de desarrollo más cortos	Alto coste de desarrollo

Tabla 11. Comparativa Apps Híbridas y Nativas

4.4 Ionic Framework

Ionic[8] es uno de los principales frameworks de desarrollo de aplicaciones móviles híbridas de carácter open source. Ionic ofrece un conjunto muy amplio de herramientas para el desarrollo móvil desde la perspectiva de las tecnologías web. Está basado en Angular por lo que podemos construir nuestras aplicaciones siguiendo la estructura de componentes y servicios de Angular y utilizando características como los data bindings en los templates de nuestra app además de obtener todos los beneficios que nos aporta el lenguaje TypeScript respecto a estructuración del código, mantenibilidad y análisis estático. Para la comunicación con el dispositivo móvil utiliza su propio paquete de plugins nativos escritos sobre la librería Cordova permitiéndonos acceder de forma muy sencilla a características como la cámara, el giroscopio, el gps, el bluetooth, etc. Ionic ofrece un look&feel distinto para cada una de las plataforma de desarrollo muy cercano al de aplicaciones nativas, el desarrollador no debe preocuparse por cómo se verá su aplicación, Ionic nos garantiza que dependiendo del sistema operativo la aplicación cambiara su look&feel de forma automática a no ser que el propio desarollador lo indique expresamente lo que resulta en un desarrollo sumamente cómodo y fluido.

Ionic nos propone desarrollos mucho más modularizados que otros frameworks dividiendo nuestras aplicaciones en bloques llamados pages, cada page podemos interpretarla como un template con su component entre los que podemos navegar. Cada vez que navegamos en Ionic se inicializa una nueva page (hacia la que hemos navegado) y se 'apila' en un stack de pages donde tendremos el historial de las pages por las que hemos navegado, para retroceder simplemente tenemos que 'desapilar' la última de estas pages. Con este sistema de navegación Ionic ofrec una forma mucho más sencilla de gestionar el ciclo de vida de las pantallas que en aplicaciones nativas.

4.4.1 Principales Características

- **UI Components**: Ionic pone a nuestra disposición una amplia gama de componentes de interfaz de usuario listos para utilizar como modales, popups, campos de texto, etc. Cada uno de estos componentes podemos visualizarlo en su web oficial pudiendo consultar su look&feel para cada plataforma.

- **API Docs**: La documentación de este framework es una de las más profesionales actualmente, siendo actualizada constantemente y ofreciendo información clara y detallada junto con ejemplos de uso de sus métodos, propiedades, eventos, etc. Además cuenta con foros oficiales donde la comunidad resuelve la mayoría de dudas de los desarrolladores.

- **Ionic Native**: Se trata del paquete de funcionalidades nativas que ofrece Ionic a sus usuarios. Parte de la base de plugins nativos de la librería Cordova permitiéndonos acceder a funcionalidades como 3D Touch en Iphone, a la geolocalización e incluso el giroscopio.

- **Theming**: Ionic aplica estilos a tu aplicación de forma automática, sin embargo, permite una total personalización mediante Sass sobrescribiendo ciertas variables específicas por lo que podemos crear nuestros propios temas para las aplicaciones e incluso publicarlos en una tienda de temas o comprar temas populares entre la comunidad.

- **Ionicons**: Ofrecen un paquete de 900 iconos aproximadamente listos para usar en nuestras aplicaciones. Estos iconos disponen, en la mayoría de caso, de tres versiones similares dependiendo de la plataforma para la que estemos desarrollando.

- **Ionic CLI**: Es la principal herramienta de línea de comandos utilizada durante el proceso de desarrollo de nuestras aplicaciones. Esta potente herramienta nos permite por ejemplo crear la plantilla inicial de un nuevo proyecto, creando por nosotros la estructura de carpetas y archivos. Nos permite realizar multitud de tareas como crear nuevas pages en nuestro proyecto, lanzar un servidor de desarrollo para poder debuggear nuestra aplicación, instalar cualquier plugin de ionic native que necesitemos, generar el icono principal de nuestra app a partir de una imagen png, e incluso lanzar nuestra aplicación en modo depuración en un dispositivo conectado a nuestro ordenador.

4.4.2 Ionic frente a React Native

Como Ionic, existen otros frameworks disponibles para desarrollar aplicaciones híbridas. El más popular de ellos es React Native, un framework basado en ReactJS la líbrería para desarrollo web de Facebook. Este framework se basa en la utilización de JXS para la creación de las vistas de la aplicación, una extensión del lenguaje JavaScript muy semejante a HTML. Este JXS será transpilado más tarde a código JavaScript. Al contrario que Ionic, que utiliza un WebView para renderizar la aplicación, Reac Native renderiza componentes nativos a la plataforma con lo que gana en rendimiento, pero sacrifica parte de la portabilidad del desarrollo. En el caso de React Native debemos utilizar componentes distintos en función de la plataforma de desarrollo, pero podremos reutilizar en cualquier caso la lógica de negocio.

Ahora veremos todas estas características mediante una tabla comparativa entre Ionic y React Native junto con mi valoración personal a favor de Ionic.

Ionic Framework	React Native
Fácil de aprender	Curva de aprendizaje mayor que Ionic
Muchos componentes con etilos predefinidos	Menos componentes pero estos son nativos
Desarrollo en TypeScript	Desarrollo en JavaScript y JSX
Rendimiento medio	Rendimiento cercano a aplicaciones nativas
Portabilidad total del código	Portabilidad parcial del código
Documentación de calidad	Documentación escasa en comparación con Ionic
Se necesita Cordova para acceder a las funcionalidades del dispositivo	Compila a código nativo por lo que tenemos acceso a las características del dispositivo
Respaldado por Google debido a Angular	Respaldado por Facebook

Tabla 12. Comparativa: Ionic Framework y React Native

Como podemos ver en la tabla 12, los dos frameworks tienen sus puntos fuertes, sin embargo, para mi situación creo que Ionic se adapta mucho mejor a mis necesidades. En mi vida laboral trabajo con tecnologías como Angular, JavaScript, TypeScript, HTML, etc, por lo que el aprendizaje de este framework me ha resultado bastante familiar y sencillo. Otro punto importante ha sido la cantidad de componentes predefinidos que tiene, con estos componentes no tenemos que desarrollar nosotros nuestros propios inputs para fechas (por ejemplo), de esta forma podemos centrarnos más en la lógica de negocio. Un aspecto que me ha parecido de gran importancia ha sido la portabilidad, por la oportunidad de llegar al mayor número de usuarios posibles, algo importante en nuestra aplicación.

4.5 Firebase

Por último, en esta sección hablaremos del servicio Firebase[9] de Google y por qué es una buena opción para el desarrollo de aplicaciones móviles.

Figura 19. Firebase: Características Principales

Firebase es una nueva plataforma de desarrollo de aplicaciones en la nube propiedad de Google. Esta plataforma ofrece a los desarrolladores una API para guardar datos en la nube y sincronizarlos en tiempo real, junto con una documentación oficial realmente bien estructurada ofreciendo gran cantidad de información sobre el uso del producto. Ofrece una gran cantidad de funcionalidades como podemos ver en la figura 19, desde bases de datos en tiempo real, autentificación, pasando por datos analíticos como flujo de usuarios, hasta un servicio de anuncios con la intención de monetizar tu aplicación. Gracias a estas características los desarrolladores de aplicaciones pueden ahorrar gran cantidad de tiempo de desarrollo simplemente haciendo uso de ellas, por ejemplo, con las bases de datos en tiempo real no es necesario construir un back-end, simplemente definimos nuestro modelo de datos por lo que podemos centrar el desarrollo en la parte cliente

4.5.1 Funcionalidades Destacadas

- **Realtime Database**: Se trata de una base de datos no relacional en la nube. Permite sincronizar datos entre varios dispositivos en milisegundos.

- **Authentication**: Permite la autentificación de usuarios en tu aplicación mediante el método correo electrónico y contraseña, o mediante proveedores externos como Facebook.

- **Cloud Messaging**: Nos permite enviar mensajes y notificaciones a los usuarios de nuestras aplicaciones pudiendo enviarlos a dispositivos concretos, grupos de dispositivos y usuarios.

- **Storage**: Firebase ofrece almacenamiento de archivos como imágenes, video o audio, contenido generado por los usuarios, integrándolo con su servicio de autentificación para permitir su carga y descarga.

- **Crash Reporting**: Nos proporciona información detallada de los errores que se producen en nuestra app por lo que podremos resolverlos antes de que afecten a la mayor parte de usuarios. En el panel de control de Crash Reporting podremos ver esta información a través de informes detallados y gráficas muy visuales.

- **Remote Config**: Mediante el panel de configuración remota podemos elegir de qué manera se mostrará la aplicación para cada grupo de usuarios, mostrar contenido personalizado, incluso ocultar parte de la aplicación o habilitar funcionalidades.

- **App Indexing**: Mediante esta característica, Google nos permite indexar nuestra aplicación en sus resultados de búsqueda cuando los usuarios busquen contenido que nuestra app podría ofrecerles. De esta forma ganamos visibilidad y, con ello, nuevos usuarios.

- **AdMob**: Es una plataforma de publicidad para apps móviles que permite monetizarlas de una forma sencilla añadiendo anuncios dentro de nuestra app.

- **Google Analytics**: Nos proporciona informes sobre eventos de nuestra aplicación. Estos datos pueden ser estadísticas de uso de la aplicación, como datos demográficos, numero de descargas por país, usos diarios, conexiones a base de datos, etc. Podemos configurar eventos que sean clave para medir factores de gran importancia para nosotros o para nuestra organización y poder tomar mejores decisiones de marketing o de desarrollo de nuevas funcionalidades.

5. Implementación

5.1 Metodología

Como en cualquier proyecto de desarrollo de software, se ha llevado a cabo siguiendo una metodología de trabajo concreta para poder organizar el desarrollo de los diferentes módulos. Al ser una única persona desarrollando la aplicación no ha sido posible aplicar metodologías Agile, tan de moda últimamente, en su lugar se ha seguido una metodología tradicional, empezando por el diseño inicial y seguida de la implementación de los casos de uso.

A pesar de esto, sí que he podido aplicar una metodología de trabajo concreta para la gestión de versiones y la creación de ramas de desarrollo en el proyecto. La metodología seguida ha sido Git Flow, que no se debe confundir con la herramienta informática del mismo nombre que implementa los comandos que esta metodología recomienda para el desarrollo de nuevas funcionalidades.

5.1.1 GitFlow

Git Flow es una metodología de trabajo que define la forma de gestionar las ramas de nuestros repositorios Git (software de control de versiones). Fue definida por Vincent Driessen en 2010 y, desde entonces, se ha convertido en una de las formas de trabajo más populares y extendidas hoy en día. Git Flow define 5 tipos de ramas distintos en su modelo, 2 principales y 3 secundarias que veremos a continuación:

- **Ramas Principales**

  - ***Master***: Será la rama principal de nuestro proyecto. En ella estará todo el código que esté listo para pasar a producción.

  - ***Develop***: Esta será nuestra rama de desarrollo principal. De esta rama partirán las nuevas funcionalidades.

Estas dos ramas principales siempre existirán en nuestro proyecto, siendo las dos únicas ramas que no eliminaremos. Cuando develop se encuentre en un estado estable, su código se fusionará con la rama master generando una nueva versión numerada para nuestra aplicación.

- **Ramas Secundarias**

Además de sus dos ramas principales, Git Flow define 3 tipos adicionales de ramas que facilitarán el trabajo en paralelo del equipo. Estas ramas suelen tener ciclos de vida mucho más reducidos que las principales creándose cuando son necesarias y eliminándolas cuando se termina de trabajar con ellas

- ***Feature***: Este tipo de rama se utilizará para llevar a cabo el desarrollo de nuevas funcionalidades. Solo existirá de forma local mientras se esté llevando a cabo, por lo que en nuestro repositorio remoto nunca tendremos ramas *Feature*. Se crearán desde la rama principal *Develop* y una vez terminado el desarrollo tendremos que fusionarlas con *Develop* nuevamente. Esto se haría fusionando nuestros cambios con un checkout local de *Develop*, resolviendo posibles conflictos si fuera necesario, y creando después una pull request, una petición para incorporar nuestros cambios a la rama *Develop* del repositorio remoto. Mediante estas pull request, cualquier compañero podrá inspeccionar nuestros cambios y sugerirnos mejoras o correcciones y, así, mantener la calidad del código que entrará en *Develop* y posteriormente en *Master*.

- ***Release***: Estas ramas se utilizarán solo para fusionar la rama de *Develop* con la rama *Master* y poder testear y estabilizar el código antes de dicha fusión. Se crearán a partir de la rama de *Develop* y contendrán todas funcionalidades de las ramas *Feature* que se hayan fusionado hasta el momento.

- ***Hotfix***: Estas ramas sirven para corregir errores que se detectan en las versiones de producción de *Master*. Estos errores son errores críticos que necesitan ser solucionados de inmediato por lo que la rama de partida será *Master* y, una vez resueltos, deberán fusionarse tanto con *Master* como con *Develop* para poder desarrollar a partir de la corrección.

5.2 IDE: Visual Studio Code

El IDE escogido para la implementación del proyecto ha sido Visual Estudio Code, propiedad de Microsoft. Este IDE no lleva mucho tiempo en el mercado, su primera versión estable fue lanzada en 2015 hace apenas dos años, pero se ha posicionado como uno de los IDEs preferidos por la comunidad de desarrollo web debido a su filosofía open source multiplataforma y su gran personalización permitiéndonos descargar numerosos plugins y extensiones que adaptan el IDE a nuestras necesidades específicas.

VS Code cuenta con una excelente con herramientas de control de versiones Git. Mucha de las funcionalidades ofrecidas por otros IDEs en forma de extensiones viene ya incluidas por defecto en VS Code por lo que resulta más cómo trabajar con él. Al tratarse

de un producto de Microsoft tiene un soporte excelente para lenguajes como TypeScript, remarcando la sintaxis con colores, autocompletando nuestras sentencias y aportándonos opciones muy interesantes de depuración de código.

Otro de sus puntos a favor, muy apreciado por los desarrolladores, es su ligereza. VS Code apenas ocupa 160MB en disco y tiene un rendimiento realmente bueno permitiéndonos tener múltiples instancias del programa con varios proyectos abiertos a la vez funcionando en la misma máquina sin por ello verse ralentizado.

Debido a que este proyecto ha sido desarrollado en lenguaje TypeScript me ha parecido una muy buena opción utilizar VS Code y aprovechar todos sus beneficios para este lenguaje además de su integración con el sistema de control de versiónes Git que funciona a la perfección.

5.3 Arquitectura de la Aplicación

Ahora veremos cómo está organizado nuestro proyecto y su estructura interna de carpetas y archivos y qué papel juegan cada uno.

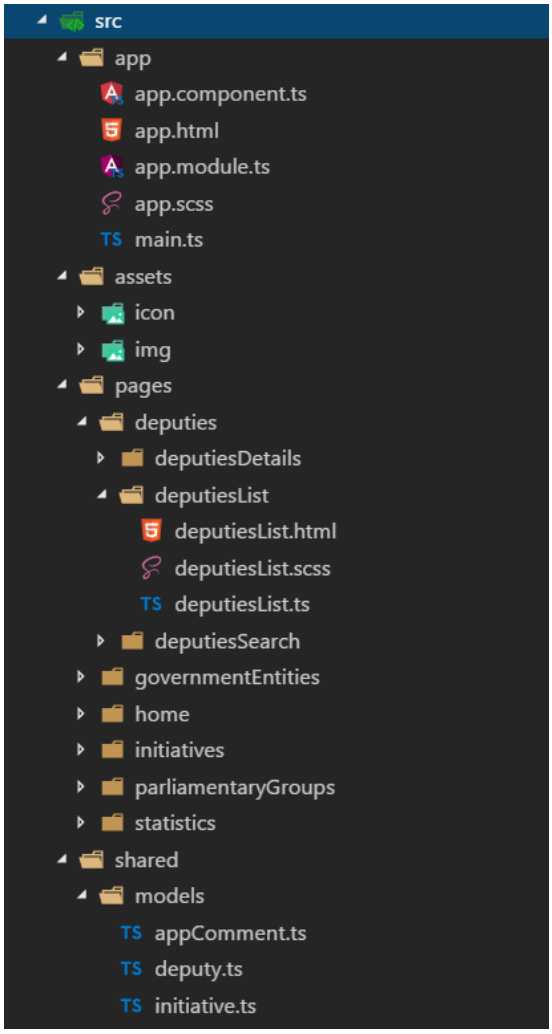


Figura 20. Estructura del Proyecto.

Como se puede apreciar en la figura 20, nuestra aplicación está estructurada de forma jerárquica y organizada en diferentes secciones que están separadas por responsabilidades. Toda esta jerarquía parte de la carpeta **src** que es la que contiene todo nuestro desarrollo, tanto código, como iconos, imágenes, etc. Al mismo nivel que la carpeta SRC tenemos otras carpetas como resources que contiene los iconos principales de la aplicación, platforms que contiene toda la información sobre las builds del proyecto por plataforma y donde se instalan los plug-in, etc. Estas carpetas no se ha creído conveniente mostrarlas ya que no contienen nada relacionado con el desarrollo en sí de nuestra aplicación, de hecho, estas carpetas estarán incluidas en el archivo *.gitignore* para obviarlas en el sistema de control de versiones.

En primer lugar, tenemos la carpeta **app**, en ella encontramos el componente principal de nuestra aplicación que actuará como punto de entrada a la misma. En la vista de este componente principal es donde se renderizarán el resto de **pages** de la aplicación y donde se generará el menú lateral que nos redirigirá al resto de **pages** principales. Contiene un archivo **app.scss**, este archivo contendrá los estilos css generales de la aplicación, que serán usados por varias pages. Además en el archivo **app.module.ts** deberemos importar las nuevas pages que vayamos creando para incluirlas como componentes de entrada en el módulo principal de nuestra aplicación. Si no hacemos esto el sistema de navegación no podrá conocer las pages a las que queremos navegar ya que no estarán declaradas en ningún módulo.

La siguiente carpeta que veremos es la carpeta **assets**, que a su vez contendrá dos carpetas más: **icon** y **img**. En estas carpetas guardaremos todos los recursos que nuestra aplicación necesite para funcional, como ciertos iconos o imágenes de cada uno de los grupos parlamentarios de la aplicación, imágenes de background, de diputados, etc.

En la carpeta **pages** guardaremos todas las pages que declaremos en la aplicación. La decisión de dividirlas en subcarpetas por nombre de entidad ha sido una elección totalmente personal que no influye en la declaración de nuevas pages, aunque se podrían haber guardado todas al mismo nivel. Se ha organizado de esta forma jerárquica porque me ha parecido más claro y visual a la hora de trabajar con pages relacionadas con una única entidad. Cada page tendrá siempre un archivo **.ts** en el cual se definirá toda la lógica de negocio de dicha page, un archivo **.scss** en el que se codificarán los estilos concretos a esa page que solo serán utilizados por ella, y un archivo **.html** para las vistas de las pages.

Por último, tendríamos la carpeta **shared**. En esta carpeta se definirán los modelos de la aplicación para poder trabajar con el tipado de TypeScript. En esta carpeta también se incluirán todos los componentes genéricos que creemos además de directivas customizadas o pipes para transformaciones de datos. En nuestro caso no ha sido necesario desarrollar nuevos elementos de estos tipos puesto que tanto Angular como Ionic ya nos los proporcionaban.

5.4 Filtrado de Listas

El filtrado de las listas de Diputados e Iniciativas ha sido unos de los problemas principales que se ha tenido que solucionar en la aplicación. Se decidió crear un algoritmo de filtrado estricto, esto es un algoritmo que compruebe que todas las condiciones indicadas deban cumplirse. Ahora veremos el flujo de ejecución de esta funcionalidad en nuestra app.

En primer lugar, debemos acceder a una de las dos listas de nuestra aplicación (diputados o iniciativas). Cada vez que accedamos desde el menú lateral el filtro será reiniciado. En esta pantalla podremos acceder a la búsqueda avanzada mediante el botón de la lupa en la esquina superior derecha. Este botón ejecuta una función para abrir el modal de la búsqueda avanzada.

```
openAdvancedSearch() {
  let searchModal = this.modalCtrl.create(DeputiesSearchModal);
  searchModal.onDidDismiss((filters: any) => {
    if (filters) {
      // Convert the modal filters result to iterable array
      this.filtersArray = Object.keys(filters).map((filterName) => {
        return { name: filterName, value: filters[filterName] };
      });
      // Create filter loader
      let filterDataLoading = this.loadingCtrl.create({
        content: 'Filtrando Diputados...'
      });
      filterDataLoading.present();
      // Filter logic
      this.filterDeputies(this.filtersArray);
      // Dismiss filter loader
      filterDataLoading.dismiss();
    }
  });
  searchModal.present();
}
```

Figura 21. Función: openAdvancedSearch.

Como podemos ver en la figura 21, esta función crea un modal a partir del servicio **modalCtrl** de Ionic. Antes de abrir dicho modal se le pasa un callback mediante la función **onDidDismiss** con la lógica que se deberá ejecutar cuando el modal se cierre. Esta función comprobará si recibimos filtros cuando eso ocurra, en caso de haberlos, la lógica de filtrado quedará relegada en la función **filterDeputies** pasándole como parámetro el correspondiente array de filtros.

En el modal aparecerá un formulario en el que todos sus inputs estarán enlazados con una propiedad de un objeto. Cada propiedad de ese objeto representará un filtro de nuestro **advancedSearch**. Este objeto se lo pasaremos a la lista de entidades mediante la función **dismiss** del modal pasándolo como argumento. Al ejecutar el **dismiss** se ejecutará el callback que definimos anteriormente con la función **onDidDismiss**. De esta forma ya tendremos disponibles los filtros necesarios para ejecutar la función **filterDeputies**.

```
private filterDeputies(filters: Array<{ name: string, value: any }>) {
  let passFilter;
  this.filteredDeputies = this.deputies.filter((deputy: Deputy) => {
    passFilter = true;
    filters.forEach((filter) => {
      switch(filter.name) {
        case 'freeText':
          let text = deputy.name.toLowerCase() + ' ' + deputy.lastName.toLowerCase()+ ' ' + deputy.description.toLowerCase();
          passFilter = passFilter && text.includes(filter.value.toLowerCase());
          break;
        case 'parliamentaryGroup':
          passFilter = passFilter && deputy.parliamentaryGroup === filter.value;
          break;
        case 'district':
          passFilter = passFilter && deputy.district === filter.value;
          break;
        case 'position':
          let positions = deputy.positions.filter((position: string) => {
            return position.toLowerCase().includes(filter.value.toLowerCase());
          });
          passFilter = passFilter && positions.length > 0;
          break;
        case 'gender':
          passFilter = passFilter && deputy.gender === filter.value;
          break;
        // rest of cases here
      }
    });
    return passFilter;
  });
}
```

Figura 22. Función: filterDeputies.

Como vemos en la figura 22, el algoritmo está basado en la ejecución de la función **filter**, accesible para cualquier array en JavaScript. Esta función recorre los diputados disponibles y comprueba si el diputado cumple todos los filtros que hemos indicado. Esto se hace mediante un callback que se ejecutará una vez por diputado y que, a su vez, recorrerá los filtros disponibles para ejecutar las comprobaciones de dichos filtros. Como podemos ver en la sentencia **`passFilter = true`**, el algoritmo supone en un primer caso que el diputado pasará el filtro y después trata de encontrar una condición que lo niegue (**`passFilter = passFilter && logic`**). Si el diputado no cumple con todos los filtros indicados, se retornará un valor false y ese diputado no estará contenido en el array resultado.

5.5 Publicación de comentarios

La publicación de comentarios es un buen ejemplo de data binding que nos proporciona Angular, y de cómo nos beneficiamos del uso de una base de datos en tiempo real con el patrón Observable.

```
<button
  ion-button
  clear
  [disabled]="!commentText"
  (click)="sendComment()"
  icon-only
  large>
  <ion-icon name="send"></ion-icon>
</button>
```

Figura 23. Vista: SendComment Button.

En la figura 23 podemos ver una porción de la vista de la pantalla de comentarios. Concretamente vemos el botón que nos permite publicar el comentario y guardarlo en base de datos. Se puede apreciar que el botón tiene dos bindings de Angular, un event binding (click) y un property binding (disabled). Mediante el property binding **`[disabled]="!commentTrxt"`** le decimos a angular que el botón debe estar desactivado cuando la propiedad **commentText** de nuestro componente no tenga valor por lo que, si no hemos escrito nada en el cuadro de texto, el botón nunca lanzará el evento de **click** y no ejecutará la función **sendComment** bindeada a dicho evento.

En el caso en que la propiedad **disabled** se evalúe a false el botón se activará permitiéndonos publicar el comentario. La lógica para la publicación es bastante sencilla y podemos verla en la figura 24. Simplemente enviamos a base de datos un objeto nuevo mediante la función **push**.

```
sendComment() {
  this.database.list('/comments').push({
    initiativeId: this.initiativeId,
    text: this.commentText,
    date: new Date().getTime()
  });
  this.commentText = '';
}
```

Figura 24. Función: sendComment.

Lo interesante de todo esto es que la lista de comentarios se actualizará automáticamente cuando se ejecute el **push** a base de datos. Esto es posible gracias a la inicialización del componente y al patrón Observable.

```
ngOnInit() {
  this.initiativeId = this.navParams.data;
  this.database.list('/comments')
    .subscribe((data: AppComment[]) => {
      this.comments = data.filter((comment) => {
        return comment.initiativeId === this.initiativeId;
      });
    });
}
```

Figura 25. Función: ngOnInit.

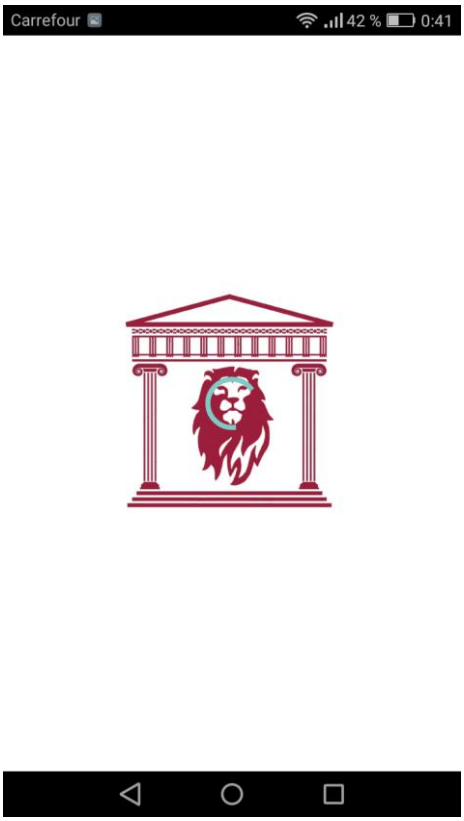
Como podemos ver en la figura 25 al inicializar el componente mediante la función **ngOnInit** de Angular hacemos una llamada **list** a base de datos. Esta llamada nos devuelve un Observable, lo cual nos permite **suscribir** una función anónima (arrow function en TypeScript) a dicha lista, que se ejecutará cada vez que se detecten cambios en la base de datos actualizando los valores de la lista recuperada. De esta forma el usuario percibirá que la vista se actualiza sin necesidad de recargar la ventana o salir y volver a acceder a los comentarios, tanto si él publica un comentario como si lo publica un usuario distinto. A este tipo de programación se la conoce como programación reactiva ya que la lógica reacciona a los cambios que sufre el modelo sin necesidad de generar nuevas llamadas HTTP a nuestro bakend.

6. Resultado

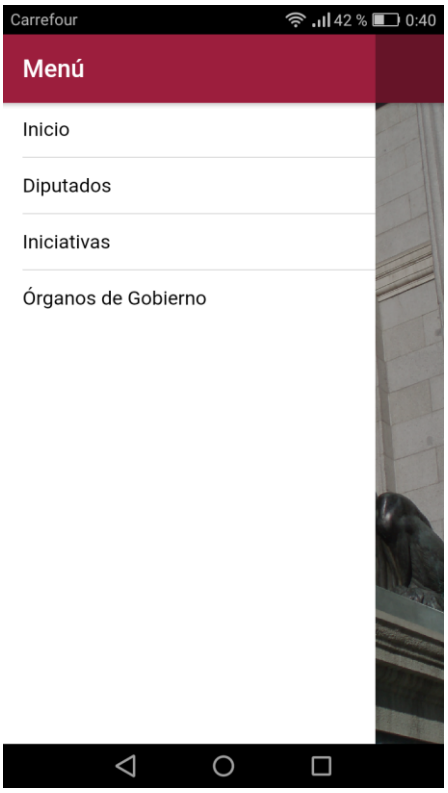
Tras implementar la aplicación y debuggearla tanto en el navegador, como en un emulador y en un dispositivo Android, se ha podido crear una build estable para producción. Todas las figuras que veremos a continuación se han realizado sobre esta versión estable funcionando en un Huawei P7, con Android 6.0 y presentan un rendimiento fluido y con tiempos de carga inferiores a un segundo, a excepción de la pantalla de carga.

- **Pantalla de Carga Inicial**

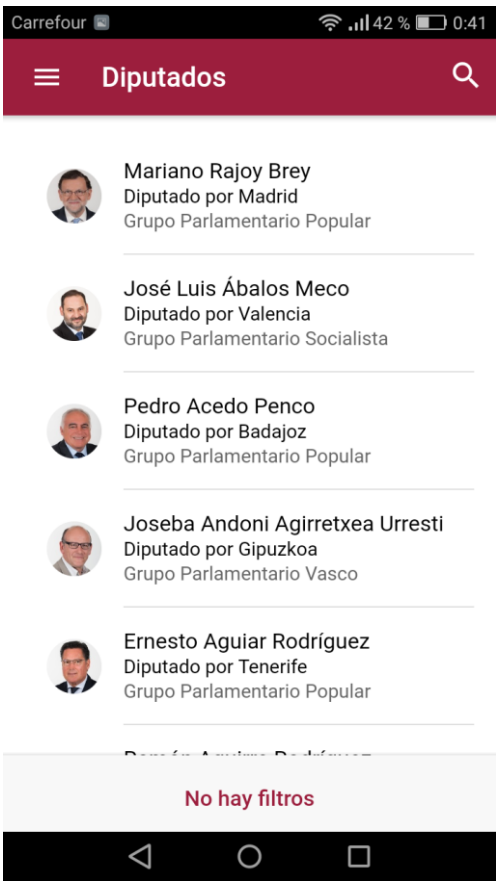
  En esta pantalla se observa que el tiempo de carga es de unos 4,6 segundos hasta que la aplicación está completamente cargada y funcional, esto es debido a la estructura de las aplicaciones Ionic. Todas las pages de la aplicación se declaran en el módulo principal y se deben cargar al iniciarse la aplicación. Este problema se soluciona con la actualización 3.3.0 del framework que incluye la estrategia lazy loading, esto se verá más a fondo en el apartado de Conclusiones y Trabajo Futuro.



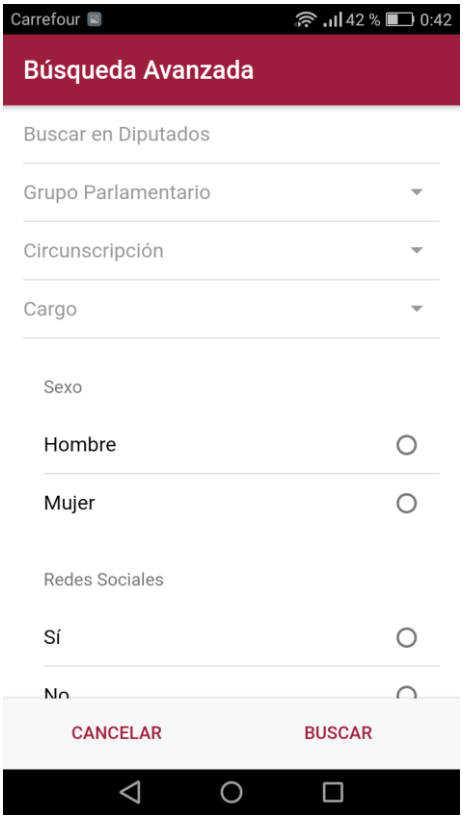
- **Menú Lateral**

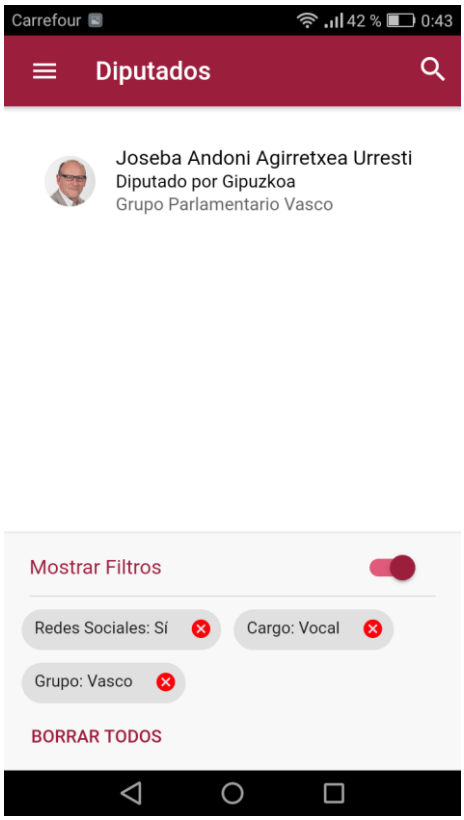


- **Lista de Diputados y Detalle**

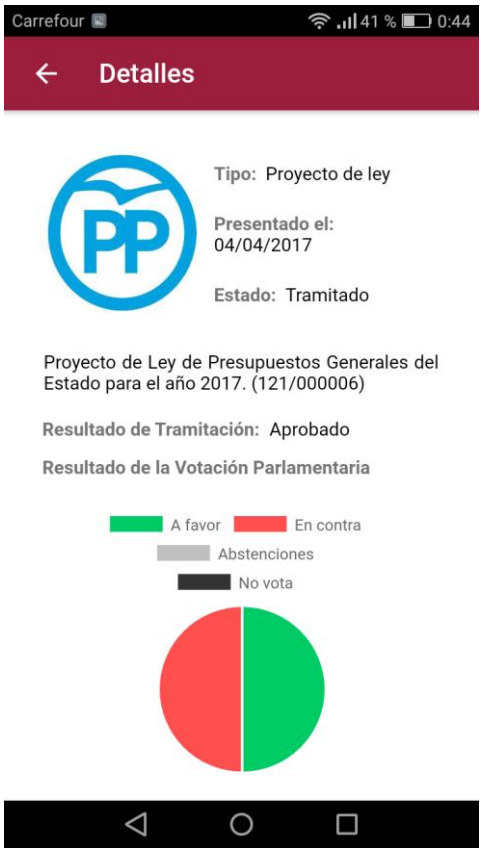


- **Lista de Diputados Filtrada y Búsqueda Avanzada**

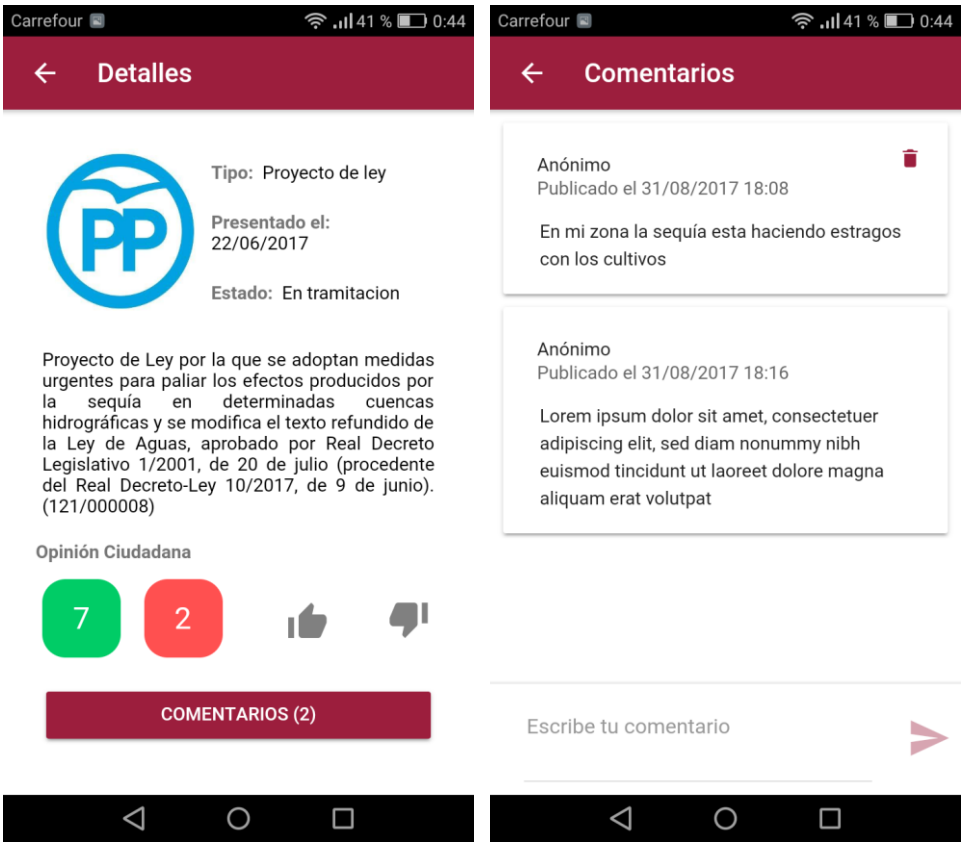




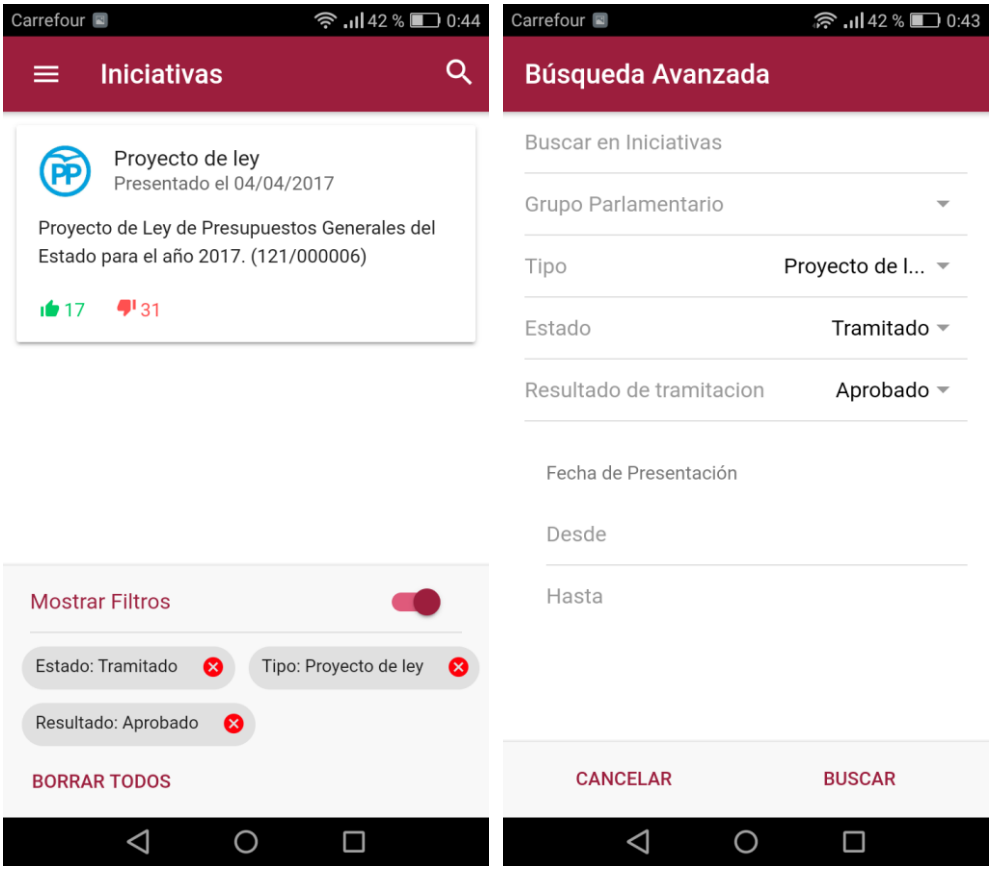
- **Lista de Iniciativas y Detalle**



- **Detalle de Iniciativa con Opinión Ciudadana y Comentarios**

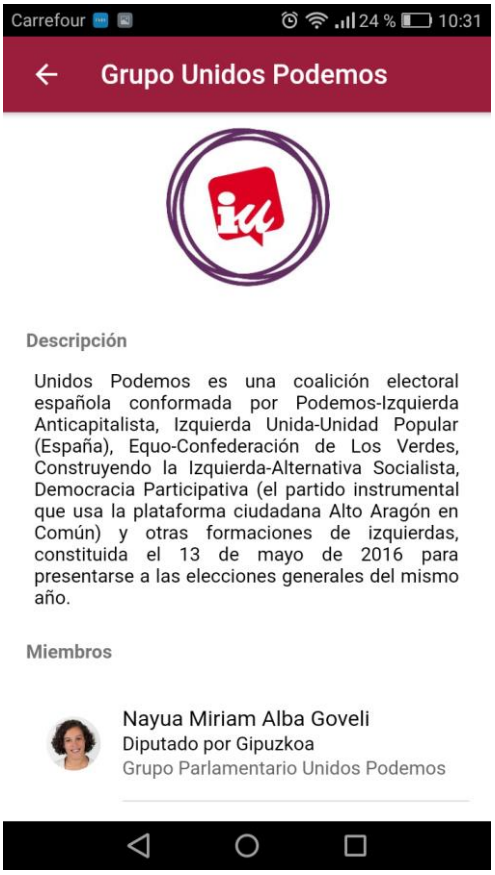


- **Lista de Iniciativas Filtrada y Búsqueda Avanzada**

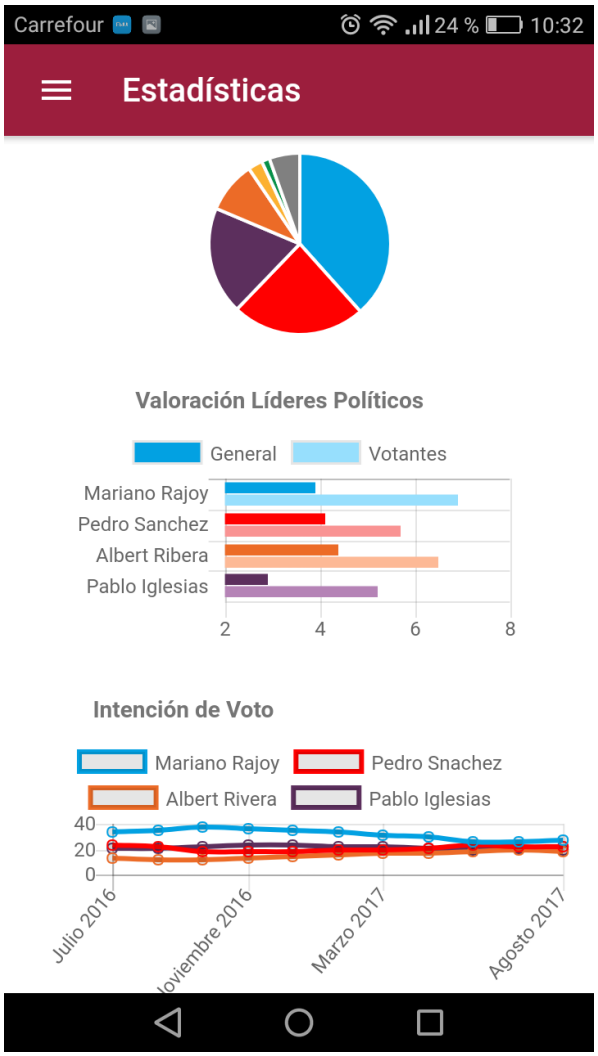


- **Órganos de Gobierno y Detalle**

- **Grupos Parlamentarios y Detalle**



- **Estadísticas**



7. Conclusiones y Trabajo Futuro

Como hemos podido ver a lo largo de todo este proyecto las tecnologías web han experimentado una enorme evolución desde hace unos años. Conceptos como la orientación a objetos o el desarrollo móvil, que parecían tan lejanos a la web, ahora son características sólidas que podemos explotar con garantías a la hora de desarrollar nuestras aplicaciones, convirtiéndose en uno de los entornos de desarrollo más demandados para realizar nuevos proyectos de desarrollo.

Desde mi punto de vista personal las tecnologías web me han permitido el desarrollo de una aplicación móvil multiplataforma con un coste de aprendizaje y desarrollo mucho menor que si hubiera utilizado tecnologías nativas. En concreto Angular y Ionic me han parecido frameworks muy asequibles, con una curva de aprendizaje suave pudiendo desarrollar con ellos casi en el mismo momento que empiezas a aprenderlos. Estos frameworks me han permitido utilizar tecnologías web estándar como HTML que me servirán en mi futuro profesional, tecnologías muy demandadas actualmente. Ionic, además, me ha resultado muy cómodo de utilizar. La librería de componentes predefinidos que tiene es una gran herramienta para lograr desarrollos coherentes, visualmente hablando, además de evitar tener que desarrollar nuestros propios componentes. Utilizando esta librería nos aseguramos de que son componentes testeados y que nos ofrecen un mínimo de calidad a la hora de desarrollar. En cuanto al rendimiento, como pudimos ver en el apartado anterior, Ionic aún tiene que mejorar el aspecto de la carga inicial de la aplicación cuando se ejecuta en el dispositivo final, sin embargo una vez cargada podemos ver que el rendimiento es bastante bueno comparado con las aplicaciones nativas por lo que el resultado ha sido más que satisfactorio.

Firebase, por su parte, me ha sorprendido muy gratamente. Su base de datos no relacional en tiempo real me parece una característica realmente potente para este tipo de aplicaciones móviles, siendo su uso realmente sencillo. La integración con Ionic ha sido total gracias a que Ionic está construido sobre Angular, por lo que solo hemos tenido que añadir la librería *angularfire*[10] a nuestro proyecto para gestionar la comunicación con backend.

Como ya se ha visto, el desarrollo web sigue teniendo sus limitaciones para el desarrollo de aplicaciones móviles frente al desarrollo nativo, pero en estos momentos esas limitaciones se han reducido considerablemente pudiendo optar por la alternativa web como una alternativa sólida para crear aplicaciones de calidad, cosa que difícilmente se podía conseguir hace apenas unos años. No podría asegurar que tecnología es mejor para un desarrollo móvil. Este proyecto me ha ayudado a comprender que esto es muy relativo. En aplicaciones de compraventa como Wallapop o de gestión de la información, con vistas sencillas, sin grandes animaciones que renderizar, el desarrollo web para móvil puede ser una muy buena opción para lograr un desarrollo de calidad en un corto espacio de tiempo que sea reutilizable en todas las plataformas, sin embargo para aplicaciones de emergencias (por ejemplo), que requieran de un alto rendimiento pienso que la opción más viable sería la tecnología nativa.

En conclusión, y después de la experiencia que me ha aportado el proyecto, no dudaría en recomendar estas tecnologías web tanto a desarrolladores principiantes que quieran aprender una tecnología con un futuro brillante, como a empresas de desarrollo móvil pudiendo incluir estos frameworks en su ámbito laboral con garantías de calidad. Creo sinceramente que a todas estas tecnologías tan novedosas tienen un futuro muy prometedor por delante y que, aunque ya son muy populares entre la comunidad web, su uso crecerá en gran medida en el ámbito profesional.

Por otra parte, con respecto a la consecución de objetivos en el proyecto creo que se han conseguido alcanzar la mayoría de ellos. Se ha conseguido desarrollar interfaces responsive, intuitivas y fáciles de usar por los usuarios tal y como se pretendía, además de que el proyecto será totalmente portable al resto de plataformas sin tener que desarrollar nuevas versiones de la app y todo ello desde el punto de vista de las tecnologías web. En lo personal, este proyecto me ha aportado un amplio conocimiento sobre el framework Ionic, conocimiento que podré aplicar en un futuro en mi puesto de trabajo. En lo referente a Open Data y transparencia, se ha conseguido crear la aplicación utilizando únicamente datos públicos ofrecidos por el propio Congreso de los Diputados e incluso ofrecer las declaraciones de la renta de los diputados que han accedido a hacerla pública, algo de vital importancia en el marco político actual debido a la baja confianza de la ciudadanía hacía sus representantes. Por todas estas razones estoy muy satisfecho con el trabajo realizado además de haber podido contribuir a la concienciación política de la sociedad y a la libertad de expresión, sin ningún tipo de posicionamiento ideológico.

Ahora pasaremos a comentar las posibles ampliaciones que se podrían incluir en este proyecto como futuros desarrollos o nuevas características que podrían ser interesantes para el proyecto para potenciar aún más la consecución de sus objetivos.

7.1 Mayor rendimiento: Lazy Loading

La arquitectura habitual de una aplicación Ionic, como vimos en el apartado de Implementación, consiste en un módulo principal en el que declaramos cada uno de los submódulos (pages) y los importamos cuando necesitamos navegar hacia ellos. Esta solución presenta un problema en la carga inicial de la aplicación, al declararse todas las pages en un único módulo principal, la aplicación carga todas las pages antes de estar lista para ser utilizada, con lo que la carga inicial suele resultar bastante pesada. Aunque nuestra aplicación no tiene un tiempo de carga inicial demasiado elevado, esto puede ser un problema muy grave en aplicaciones de mayor envergadura cuyo rendimiento sea crítico.

Es a partir de la versión 3.3.0 de Ionic Framework cuando se incluye un nuevo concepto llamado Lazy Loading. El concepto es simple, en lugar de cargar todas las pages de nuestra aplicación al principio, cargaremos cada módulo según su demanda, solo cuando se necesite disponer de él con lo que la aplicación se iniciara antes.

Ahora cada vez que declaremos nuevas pages (componentes angular) deberemos incluir al inicio el decorador @IonicPage para que el framework sepa que debe cargarlas bajo demanda.

El cambio principal reside en el archivo **app.module.ts**:

```
@NgModule({
  declarations: [
    MyApp,
    HomePage,
    DeputiesListPage,
    DeputiesSearchModal,
    DeputiesDetailsPage,
    InitiativesListPage,
    InitiativesSearchModal,
    InitiativesDetailsPage,
    InitiativesCommentsPage,
    DashboardEntitiesPage,
    EntitiesDetailsPage,
    DashboardParliamentariGroupsPage,
    ParliamentariGroupsDetailsPage,
    StatisticsPage
  ],
  imports: [
    BrowserModule,
```

```
@NgModule({
  declarations: [
    MyApp,
    HomePage
  ],
  imports: [
    BrowserModule,
```

Figura 26. Cambios en app.module.ts

Como podemos ver en la figura 26, ahora no declaramos todas las pages de nuestra aplicación en el mismo módulo principal. Ahora debemos crear un archivo **module.ts** para cada una de estas pages y que de esa forma se puedan cargar de manera independiente como vemos en la figura 27.

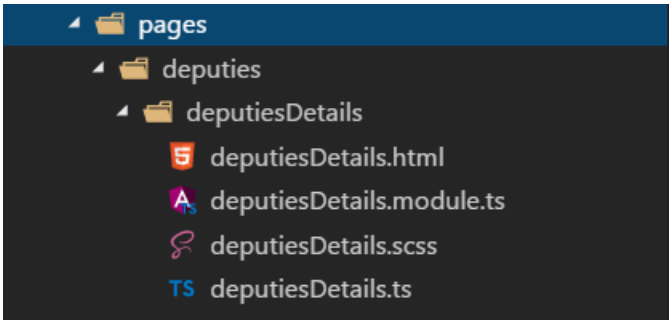


Figura 27. Nuevo archivo module.ts.

Ahora no será necesario importar las pages en los lugares de la aplicación desde los que naveguemos. Ahora simplemente referenciaremos las pages mediante un string sin importarlas al inicio del archivo:

```
  deputyTapped(deputy: Deputy) {
    this.navCtrl.push(DeputiesDetailsPage, deputy);
  }
```

```
deputyTapped(deputy: Deputy) {
  this.navCtrl.push('DeputiesDetailsPage', deputy);
}
```

Figura 28. Nueva navegación.

Con estas mejoras en el framework obtenemos un rendimiento en la carga inicial mucho mayor que en la arquitectura anterior, por lo que sería algo para tener en cuenta en futuros desarrollos.

7.2 Autenticación en la App

Como ya comentamos en la fase diseño, se prescindió de esta característica debido a que no parece una funcionalidad que todo el mundo esté dispuesto a utilizar. En temas de política no todo el mundo se muestra de acuerdo en divulgar su opinión personal por lo que posiblemente algunos usuarios nunca la utilizarían.

En esta primera implementación se trabaja de forma anónima dentro de la aplicación, por lo que para añadir esta nueva funcionalidad deberíamos emitir una encuesta sobre la base de usuarios de nuestra aplicación una vez lanzada en las tiendas de aplicaciones para conocer la opinión general de los usuarios. De esta forma se podrá tomar una decisión de desarrollo correcta, en lugar de ofrecer cosas que quizá nuestros usuarios no utilicen o no necesiten.

7.3 Monetización: Google Analitics

Uno de los aspectos más relevantes del desarrollo de una aplicación móvil es precisamente la monetización de la misma. Algunas apps son de pago por lo que si quieres descargarlas primero tienes que pasar por caja, este modelo no me ha parecido adecuado para nuestra aplicación ya que queremos llegar al mayor número posible de usuarios y no todos están dispuestos a pagar por este tipo de aplicaciones.

La segunda opción sería adoptar un modelo de publicidad en la app permitiendo así generar ingresos por publicidad pudiendo ofrecer la aplicación de forma gratuita. Aunque puede parecer una buena opción, muchas veces la publicidad entorpece la navegación por la app y su look&feel final. Este tipo de publicidad podría hacer que la aplicación tuviera menos descargas de las que podría tener ofreciéndola de forma gratuita y sin publicidad.

Por último, si queremos que sea viable su monetización, pero queremos ofrecerla de modo gratuito y libre de publicidad, la única opción que nos queda sería sacar partido al servicio de Google Analitics integrado en Firebase sobre el que hablamos en el apartado de Tecnologías. Con Google Analitics tendríamos acceso a muchísima información estadística sobre uso de nuestra app, accesos diarios a la misma, diputados consultados más veces, estadísticas de valoración de iniciativas, grupos parlamentarios más consultados, etc. Esta información, generada a partir del uso de datos abiertos en nuestra app, puede ser de gran valor para el gobierno o para los partidos políticos en los periodos de campaña electoral que podrían llegar a pagar por ella. De esta forma podríamos obtener beneficios sin tener que recurrir a la molesta publicidad in-app.

7.4 Web Workers

Ya hemos hablado sobre el rendimiento superior que se consigue en las aplicaciones móviles si elegimos un desarrollo nativo frente a uno web. Esto se debe en gran parte porque en los desarrollos web no se puede paralelizar el renderizado de la vista con la ejecución de la lógica de negocio, por ejemplo. La necesidad de ejecutar toda la aplicación en un mismo hilo hace que este tipo de aplicaciones no puedan aprovechar el multithreading de los últimos procesadores para móviles con lo que su rendimiento siempre será menor. Pero ¿Qué son los Web Workers?

Los Web Workers son herramientas HTML5 que nos permiten ejecutar diferentes procesos JavaScript en paralelo. Dichos procesos se cargan en archivos **.js** separados sin acceso al DOM por lo que la comunicación con ellos será a través del paso de mensajes entre los hilos de ejecución. De esta forma podríamos delegar en un web worker la lógica de negocio más pesada de nuestra aplicación.

Angular ya ofrece soporte para dicha tecnología por lo que las aplicaciones Ionic también son compatibles. Esto es un gran punto a favor, ya que pudiendo paralelizar nuestro código mediante el uso de web workers podemos liberar nuestra interfaz de usuario, mejorando la fluidez y la experiencia final para los usuarios. Si bien una aplicación híbrida nunca será tan rápida o fluida como una nativa, con el uso de web workers podemos alcanzar un rendimiento casi idéntico en el que la diferencia sea despreciable, por lo que eliminaríamos la desventaja del rendimiento en las aplicaciones híbridas.

Web Workers me ha parecido una característica web muy potente, podría ser muy interesante considerarla como una posible área de trabajo futuro para este proyecto. Es cierto que nuestra aplicación no presenta problemas de rendimiento tan serios como para plantearnos la paralelización de nuestra lógica de negocio, por lo que mi interés en añadir esta tecnología en un futuro es más bien académico, para aprender a utilizarlo y aportar valor a mis futuros desarrollos.

8. Referencias

[1] Cadena SER, David Justo. “El uso de 'smartphones' en España se duplica en los últimos cinco años”. 2017
http://cadenaser.com/ser/2017/02/28/ciencia/1488281552_888684.html
[Consulta: 13/09/2017]

[2] Fundación Telefónica. “La sociedad de la información en España”. Ed: Ariel S. A. 2010.

[3] Agile Business Consortium. “DSDM Atern Handbook, MoSCoW Prioritisation”. 2008. https://www.agilebusiness.org/content/moscow-prioritisation-0
[Consulta: 13/09/2017]

[4] Kurt Bittner, Ian Spence. “Use Case Modeling”. Ed: Addison Wesley. 2006.

[5] Blog EstudioKA, Claudia Bravo. “¿Qué es un Mock Up?”. 2015.
http://estudioka.es/que-es-un-mock-up/
[Consulta: 13/09/2017]

[6] BBVA Open4U, Carlos Azaustre. “'Web components': presente y futuro en el desarrollo web”. 2016. https://bbvaopen4u.com/es/actualidad/web-components-presente-y-futuro-en-el-desarrollo-web
[Consulta: 13/09/2017]

[7] Rafael G. Blanes. “S.O.L.I.D.”. 2017.
http://www.ellibronegrodelprogramador.com/blog/s-o-l-i-d
[Consulta: 13/09/2017]

[8] Ionic Framework Documentation. 2017. https://ionicframework.com/docs/
[Consulta: 13/09/2017]

[9] Google Firebase Documentation. 2017. https://firebase.google.com/docs/?hl=es-419
[Consulta: 13/09/2017]

[10] Firebase team. Aangularfire2. 2017. https://github.com/angular/angularfire2
[Consulta: 13/09/2017]