Trabajo de final de Máster: *TestIt Oposiciones*

Nombre Estudiante: Manuel Pereira González
Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles

Nombre Consultor/a: Eduard Martín Lineros
Profesor/a responsable de la asignatura: Carles Garrigues Olivella

Versión 1.2 – Enero de 2017

FICHA DEL TRABAJO FINAL

Título del trabajo:	*TestIt Oposiciones*
Nombre del autor:	*Manuel Pereira González*
Nombre del consultor/a:	*Eduard Martín Lineros*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	01/2016
Titulación:	Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles
Idioma del trabajo:	*Castellano*
Palabras clave	*app, oposiciones, test, iOS, Android, ionic*

Resumen del Trabajo (máximo 250 palabras):

El presente trabajo de final de máster consiste en el desarrollo de una aplicación para dispositivos móviles denominada "TestIt Oposiciones". La aplicación permite realizar cuestionarios de tipo test para que los estudiantes de oposiciones puedan practicar con su smartphone o tableta, con especial énfasis en la usabilidad en todas las plataformas para las que está disponible: Web, iOS y Android.

Se trata de una aplicación genérica que permite seleccionar la oposición deseada, no centrada en una única oposición. Sin embargo, a modo de ejemplo se han introducido datos de la oposición del “Cuerpo Administrativo de las Cortes Generales” (más de 2.000 preguntas introducidas).

Como tecnología principal se utiliza ionic framework, que permite el desarrollo de aplicaciones híbridas para diversas plataformas. Este framework se basa en la programación en la capa cliente con HTML5 y Javascript, haciendo gran uso de la librería angularjs. En la parte servidora se utiliza la plataforma en la nube Firebase, tanto para la autenticación de usuarios como para el almacenamiento de datos asociados a las distintas oposiciones.

Se han realizado pruebas de la aplicación con cuatro usuarios reales durante un periodo de un mes, tanto en dispositivos Android como iOS. Como resultado de estas pruebas se han detectado diversas incidencias que se han procedido a solucionar.

El trabajo incluye un breve estudio de mercado y un plan de negocio con diversas ideas para una posible monetización futura, ya que se ha detectado un nicho de mercado con un importante número de usuarios potenciales.

Abstract (in English, 250 words or less):

This final master's work consists of the development of an application for mobile devices called "TestIt Oposiciones". The application allows oppositions students to practice test questionnaires with their smartphone or tablet, with special emphasis on usability on all platforms for which it is available: Web, iOS and Android.

It is a generic application that allows selecting the desired opposition, not focused on a single opposition. However, by way of example, data have been provided for the opposition of the “Cuerpo Administrativo de las Cortes Generales” (more than 2,000 questions introduced).

The main technology used is the ionic framework, which allows the development of hybrid applications for various platforms. This framework is based on programming in the client layer with HTML5 and Javascript, making a big use of the angularjs framework. In the server side, the Firebase platform in the cloud is used, both for the authentication of users and for the storage of data associated with different oppositions.

App testing has been performed with four real users over a one-month period, on both Android and iOS devices. As a result of these tests, several incidents have been detected that have been solved.

A brief market study and a business plan with various ideas for possible future monetization are included, as it has been detected a niche market with a significant number of potential users.

Índice

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Durante mi trayectoria profesional he pasado por dos procesos de oposición bastante complejos, en los que he echado en falta cierto **apoyo tecnológico que sirva de ayuda al estudio y repaso de las pruebas de tipo Test**. A pesar de existir algunas Apps en el mercado que facilitan la realización de exámenes tipo Test para practicar oposiciones, todas tienen una **apariencia bastante rudimentaria, y una funcionalidad limitada que me gustaría ampliar**.

Tampoco existe ninguna aplicación específica que incluya tests para la **práctica de la oposición de Administrativos de las Cortes Generales**, cuya convocatoria se encuentra actualmente abierta y a la que se presentan miles de personas (mencionar que yo ya soy funcionario de las Cortes Generales, del cuerpo de Asesores Facultativos).

En este contexto, **este trabajo de final de máster se centra en el desarrollo de una aplicación para dispositivos móviles que permite realizar cuestionarios de tipo test para practicar oposiciones**. Una de las características fundamentales de esta App es que sea muy sencilla de utilizar, **haciendo énfasis en la usabilidad**. Se trata de una aplicación genérica que permite seleccionar la oposición deseada, no centrada en una única oposición. Sin embargo, a modo de ejemplo se introducirán datos de la oposición del Cuerpo Administrativo de las Cortes Generales.

Respecto a las plataformas en las que funciona la App, de todas las tecnologías abordadas durante al máster la que más me ha atraído es la realización de **Apps híbridas con angularjs e ionic framework**. La realización de esta App, para **plataformas iOS y Android**, me permite realizar algo que sé de primera mano que **es necesario y no existe en el mercado**. Además, aunque la versión inicial es gratuita, tengo distintas **ideas para una posible monetización futura** (acompañan a esta memoria un breve estudio de mercado y un plan de negocio en esta línea).

1.2 Objetivos del Trabajo

A continuación, se muestra una tabla con los principales objetivos del trabajo de final de máster, así como una breve descripción de cada uno de ellos.

Objetivo	Descripción
Poner en práctica lo aprendido en el máster	*A lo largo del máster se han cursado diversas asignaturas de desarrollo de Apps para plataformas iOS y Android. También se han cursado varias asignaturas centradas en el diseño visual, y otras dirigidas a la programación cliente.* *De todas las tecnologías aprendidas, se ha decidido abordar este trabajo de final de máster con el **framework ionic**, que permite el desarrollo de aplicaciones híbridas que se pueden desplegar, entre otras, en las plataformas iOS y Android. Este framework se basa en la programación en la capa cliente con **HTML y Javascript**, haciendo gran uso de la librería **angularjs**.*
Cubrir una necesidad existente en el mercado	*Para elegir la temática de este TFM me ha ayudado mi experiencia personal como opositor y el conocimiento concreto de la **oposición del Cuerpo Administrativo de las Cortes Generales**. Actualmente en el mercado no existe ninguna aplicación que facilite la realización de cuestionarios y exámenes tipo test para ella. Por este motivo, aunque se trata de una aplicación genérica que permite seleccionar la oposición deseada, a modo de ejemplo **se introducirán datos para esta oposición en concreto**.*
Crear una App intuitiva y fácil de usar	*Después de un estudio de mercado sobre aplicaciones con funcionalidad similar, se ha detectado **bastante falta de profesionalidad en las Apps existentes**. Son muy poco intuitivas, **están desarrolladas con tecnologías obsoletas** y carecen de ciertas funcionalidades que considero muy relevantes.* *Por este motivo, la App a desarrollar pone **especial énfasis en el diseño y la usabilidad**, así como en la implementación de estas funcionalidades de que carecen las otras. Se trata de un hándicap importante, sobre todo teniendo en cuenta que se trata de una App híbrida, por lo que **la velocidad de***

	***respuesta y la integración con funcionalidades nativas pueden convertirse en un problema**.*
Facilitar una posible monetización futura	*Se pretende **cubrir un amplio espectro del mercado de dispositivos móviles**, no sólo por dar cobertura a la necesidad existente en el mercado, sino también porque se ha pensado en una posible monetización futura de la App.* *Este es uno de los factores que ha influido notablemente en la selección de una tecnología de desarrollo híbrida, que permita desplegar la App como mínimo las plataformas iOS y Android.* *Con esto se cubriría aproximadamente el 92% del mercado de dispositivos móviles, según datos de Netmarketshare de junio de 2016 (ver figura debajo).*

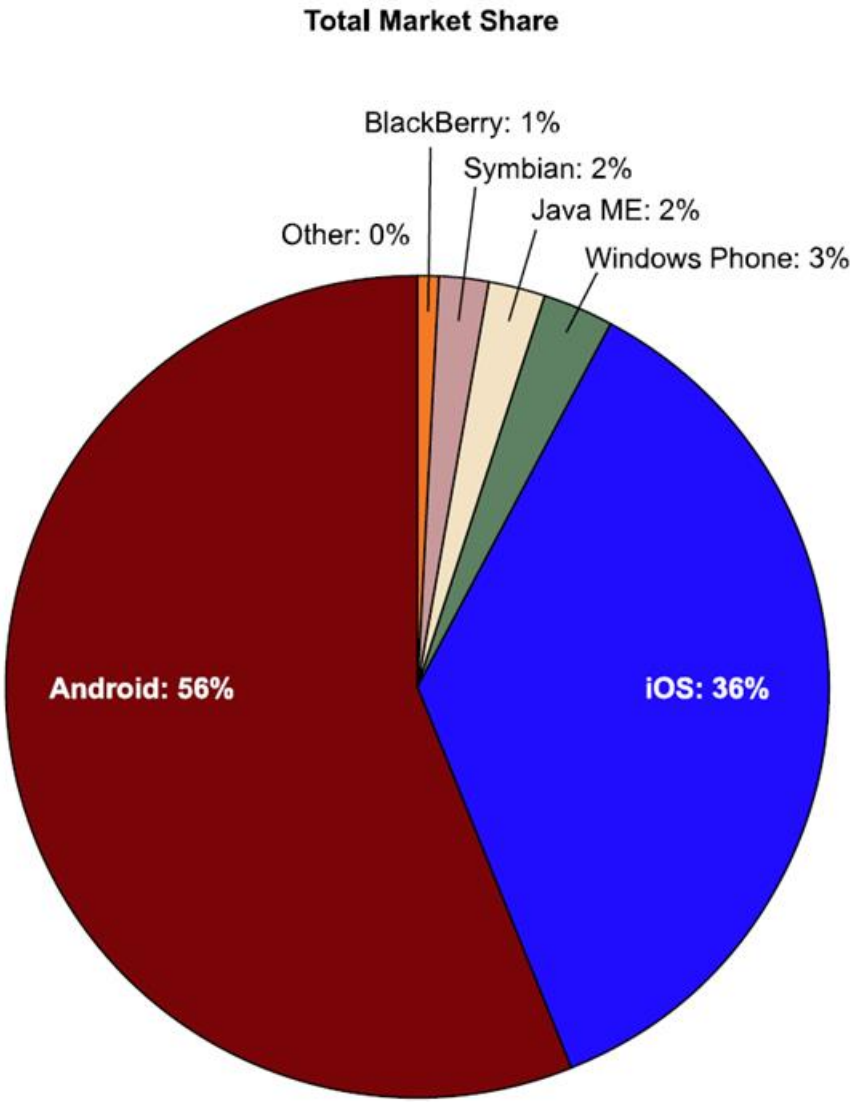


Figura 1: Mercado de dispositivos móviles 2016 (Fuente: Netmarketshare)

1.3 Enfoque y método seguido

Una vez detectada la necesidad, en primer lugar, se ha realizado un **breve estudio del mercado** de las Apps existentes, para ver si alguna la cubría. Descartada la existencia de una aplicación intuitiva que permitiese realizar exámenes tipo Test para las oposiciones de Administrativo de las Cortes Generales, **se decidió abordar esta idea como trabajo de final de máster**.

La App se va a realizar utilizando **ionic framework** por los motivos ya expuestos en los apartados previos, que permite compilar, entre otras, para las plataformas iOS y Android. Para el desarrollo de la App, disponemos de diversas herramientas que se van a utilizar en las distintas fases:

- **Ionic Market**: Una tienda de componentes y plantillas de partida (templates) que puede ahorrar muchas horas de trabajo. Tiene bastantes componentes de buena calidad, antes de desarrollar algo desde cero es importante primero investigar en el market por si alguien ya ha hecho algo similar. En una búsqueda preliminar se han detectado tres templates que podrían servir de punto de partida:
    - Quizionic: https://market.ionic.io/starters/quizionic
    - Android iOS Quiz App: https://market.ionic.io/starters/android-ios-quiz-app-using-ionic-framework
    - Quizzer: https://market.ionic.io/starters/quizzer
    - Quiz Module: https://market.ionic.io/plugins/quiz-module

  Sin embargo, y tras analizarlos con detenimiento, ninguno cubría los requisitos básicos, por lo que se ha decido partir de un proyecto desde cero utilizando el "Named template starter" denominado "sidemenu" (ver http://ionicframework.com/docs/cli/start.html).
- **Ionic Creator**: Se trata de una herramienta online de diseño visual de pantallas. Se va a utilizar esta herramienta para el desarrollo de los componentes principales del interfaz gráfico.
- **Ionic View App**: Esta herramienta es muy potente, se trata de una App disponible en los markets de los distintos sistemas operativos, que te permite visualizar tus proyectos sin necesidad de haberlos publicado. Esto permitirá probar la aplicación sin necesidad de tener que publicarla en los markets ni desplegarla en nativo (un proceso bastante lento).

1.4 Planificación del Trabajo

A continuación se muestra una planificación temporal a alto nivel del trabajo a realizar:

Figura 2: Planificación Temporal de alto nivel

Algunas notas sobre esta planificación:

- La planificación se ha realizado teniendo en cuenta las fechas de entrega de **cada una de las PECs así como de la entrega final**.
- Como todo el desarrollo de la App va a ser abordado por la misma persona, ciertas tareas que en un proyecto real se podrían haber realizado en paralelo aquí **se muestran secuencialmente**.
- La duración de las tareas se ha calculado en horas, teniendo en cuenta que se prevé una dedicación de **dos horas diarias de lunes a viernes y tres horas diarias los fines de semana**.
- La semana del 26 de diciembre al 1 de enero no se dedicará tiempo, se considera festiva en la planificación.

La siguiente tabla recoge el detalle de cada una de las tareas del diagrama:

Tarea	Horas	Descripción
Estudio de mercado	*8*	*Para cada una de las ideas para el TFM, se realizará un breve estudio de mercado para evaluar aplicaciones existentes que cubran parte o toda la funcionalidad requerida. Se buscarán Apps en los markets de Android e iOS.* *Para TestIT Oposiciones: Dentro de las que permiten hacer Tests para oposiciones se identificará si alguna contiene Tests para la oposición de Administrativo de las Cortes Generales.*
Alternativas y decisión de temática TFM	*4*	*Realizado el estudio de mercado para cada una de las alternativas de TFM, se seleccionará la App que finalmente se llevará a cabo.*
Definición de requisitos	*7*	*Definición formal de los requisitos funcionales de la App a implementar. Se incluye conversación con usuarios potenciales de la App (opositores que estén preparándose para Administrativos de Cortes Generales).*
Desarrollo PEC 1 (planificación temporal)	*10*	*Inicio de escritura de la memoria del TFM, que incluye la introducción y la parte de planificación de recursos.*
Entrega PEC 1	*1*	*Entrega de la PEC 1*
Arquitectura tecnológica	*8*	*Definición de la Arquitectura Tecnológica de la App a implementar. Selección de tecnologías a utilizar en cada una de las capas de la aplicación. Selección de herramientas de ayuda al desarrollo.*

Tarea	Horas	Descripción
Estructuración de ficheros	*4*	*Dentro de la capa cliente/javascript, definición de los distintos controladores y estructura de ficheros para el proyecto (se tendrá en cuenta https://angular.io/docs/ts/latest/guide/style-guide.html).*
Modelo de datos (diseño estructura JSON)	*8*	*Definición de la estructura de datos en formato JSON que contenga preguntas, opciones, exámenes, oposiciones, etc.*
Diseño visual	*16*	*Diseño de la interfaz de usuario, esquemas de las distintas pantallas de la App.*
Funcionalidades nativas	*4*	*Análisis y diseño de uso de funcionalidades nativas de los dispositivos, y tecnología para acceso a ellas (vibración, bloqueo pantalla, etc. usando apache cordova).*
Entrega PEC 2	*1*	*Entrega de la PEC 2*
Preparación del entorno de desarrollo	*12*	*Configuración y preparación del entorno de desarrollo, tanto para el uso de ionic framework como para las pruebas en. dispositivos nativos Android e iOS*
Template y login	*8*	*Inicio del proyecto, punto de partida de un template e implementación de la funcionalidad de login/logout.*
Menú y pantalla principal	*16*	*Implementación del menú de la aplicación y la pantalla principal.*
Integración Firebase	*15*	*Integración con la plataforma Firebase para login de usuarios y almacenamiento de datos de tests.*
Cuestionario y Exámenes	*42*	*Implementación de la funcionalidad principal de la App: Cuestionarios tipo test y exámenes anteriores.*
Pruebas y solución de errores	*16*	*Pruebas de la App en entorno Android e iOS, identificación y corrección de errores.*
Entrega PEC 3	*1*	*Entrega de la PEC 3*
Finalización de la memoria	*13*	*Redacción final de la memoria del TFM*
Preparación de PPT	*17*	*Preparación de powerpoint de presentación del TFM, así como del vídeo de demostración.*

Tarea	Horas	Descripción
Entrega final	*1*	*Entrega final del TFM*
Tribunal virtual	*1*	*Tribunal Virtual del TFM*

1.4 Breve sumario de productos obtenidos

Los entregables finales del Trabajo de Final de Máster serán los siguientes:

1) **Memoria**: Memoria del trabajo, basada en la plantilla proporcionada por los profesores de la asignatura.
2) **Presentación virtual**: Diapositivas y vídeo con la descripción del proyecto y una demostración del funcionamiento de la App.
3) **Código fuente**: Se dará acceso al repositorio Git que contiene el código fuente de la App desarrollada.

   > **Nota Importante**: Junto con esta memoria no se hace público el código fuente de la aplicación, puesto que, como se ha comentado, se pretende realizar una monetización futura del
4) **Ejecutable de la App**, para las siguientes plataformas:
   a. Ionic View: Código de la App en Ionic View para poder descargarla y probarla sin estar publicada en el market.
   b. iOS: Proyecto XCode para poder desplegar la aplicación en un dispositivo iPhone/iPad.
   c. Android: APK y proyecto de Android Studio para poder desplegar la aplicación en un dispositivo Android.

1.5 Breve descripción de los otros capítulos de la memoria

En el **apartado 2** de este documento se introduce el estudio de mercado realizado para validar la funcionalidad de la App a implementar. En el **apartado 3** se muestran los requisitos funcionales de la aplicación, en primer lugar a nivel general y luego abordando en detalle cada uno de ellos. El **apartado 4** contiene el diseño de la aplicación, tanto a nivel de arquitectura como a nivel de diseño técnico detallado. El **apartado 5** describe la implementación de la App, incluyendo su compatibilidad para distintas plataformas así como los principales problemas encontrados durante su desarrollo. En el **apartado 6** se describen las conclusiones de la realización del proyecto. Por último, **los apartados 7, 8** contienen respectivamente el Glosario de términos y la Bibliografía y enlaces relevantes utilizados.

2. Estudio de Mercado y Retorno de la Inversión

Se ha realizado un estudio de mercado previo al desarrollo de la aplicación, para analizar la oferta de aplicaciones similares y funcionalidades ofrecidas por estas, así como los distintos modos de monetización que se utilizan en cada una de ellas. La siguiente tabla muestra las características principales de cada una de ellas:

Nombre	**Funcionalidades Principales e Interfaz de Usuario**	**Monetización**
Correos Oposiciones	*- Interfaz muy precaria, no se adapta bien a distintos tamaños de pantalla, scrolls incontrolados, muy poco profesional* *- Versión gratuita con publicidad y limitaciones* *- Posibilidad de test de examen, test por temas, exámenes reales* *- Posibilidad de desordenar las respuestas de las preguntas desde la pantalla de configuración, así como limitar el tiempo y el número de preguntas*	Versión completa: sin publicidad, más preguntas y simulación de examen real: **1,99€**
Tests Adams	*Tests de la academia Adams.* *- No se puede utilizar la aplicación sin darse de alta, sin pagar no se puede realizar ningún test* *- Simulacros de examen con límite de tiempo.* *- Personalización de tests*	Se compran independientemente los bloques de las asignaturas desde la propia aplicación
Ortografía Oposiciones, TestOpos policía nacional, etc. (todas muy similares).	*- Interfaz precaria, no se adapta bien, muy poco profesional* *- Versión gratuita con publicidad* *- Posibilidad de dictados, ejercicios y tests por temas. Se pueden desordenar las respuestas desde la pantalla de configuración*	Versión completa con más preguntas y sin publicidad: **1,99€**

Nombre	Funcionalidades Principales e Interfaz de Usuario	Monetización
TodoTest	*- Tests de permisos de conducir* *- Tests oficiales de la DGT, tests de examen, registro y evolución de fallos* *- Permite corregir una pregunta con la explicación de la respuesta correcta* *- Permite consultar la nota del examen real, se integra con la DGT* *- Actualización automática de las preguntas al conectarse (versiones de la base de datos).*	Gratuita
Test de armas y Test general de oposiciones (similares, mismo desarrollador)	*- Versión lite gratuita* *- Interfaz de usuario muy poco intuitiva y profesional* *- Permite series libres, exámenes y por un tema concreto* *- No se puede ir a las preguntas anteriores* *- Interesante: Permite notificación de errores desde la propia App*	Versión completa: 5,99€ / 10,99€

Como resultado del estudio puede concluirse que, **a pesar de existir** algunas Apps en el mercado que facilitan la realización de exámenes tipo Test para practicar oposiciones, todas tienen una **apariencia bastante rudimentaria**, son muy poco intuitivas, muchas están desarrolladas con tecnologías obsoletas y **carecen de ciertas funcionalidades** que considero muy relevantes, como la posibilidad de elegir los temas concretos sobre los que se quiere realizar el test para una determinada oposición, a aleatorización del orden de las respuestas, la inclusión de exámenes reales de otros años, etc.

2.1 Retorno de la Inversión

Pensando en una posible monetización futura de la aplicación, se han evaluado distintas alternativas:

- **Banners**: Inclusión de Banners con publicidad en una única versión gratuita. Monetización a través de estos.
- **Pago por descargar exámenes**: Versión gratuita de la aplicación que incluye la zona de cuestionarios y algunos exámenes de ejemplo. Importe fijado para la descarga de exámenes de otros años.

- **Límite de tiempo**: Sólo permitir hacer cuestionarios con tiempo limitado de 10 minutos, a no ser que pagues por una versión que no limita dicho tiempo.
- **Límite de preguntas**: Sólo permitir hacer cuestionarios con un número máximo de 20 preguntas, a no ser que pagues por una versión que no limita dicho número.
- **Límite de preguntas al día**: Sólo permitir realizar cuestionarios hasta un máximo número de preguntas al día. Cada día se reinicia el contador.
- **Incluir tus propias preguntas**: Tener una aplicación básica gratuita, y una profesional que habilita la posibilidad de añadir tus propias preguntas.
- **Una App por cada oposición**: En lugar de una aplicación genérica, poder tener una aplicación en el market para cada oposición, y cobrar un importe fijo por cada una.
- **Explorar el nuevo modelo de suscripciones y prueba gratuita de iOS**: Ver https://hipertextual.com/2016/09/app-store-iphone-7
- **InApp purchase para ionic**: Pensar en la monetización utilizando el plugin InApp purchase para ionic: https://alexdisler.com/2016/02/29/in-app-purchases-ionic-cordova/

De todas las posibilidades de monetización expuestas, inicialmente se orientará la aplicación a la **creación de una versión básica** cuya descarga y uso sea **gratuita, y una versión profesional** con un único **importe fijo de descarga inicial** que incluya algunas funcionalidades adicionales (no se incluirá publicidad en ninguna de las versiones). Se establecen las siguientes estimaciones para la **monetización del proyecto**, en función del estudio de mercado y la planificación expuestos en los distintos apartados de este documento:

- **Estimación inicial del coste de desarrollo** de la aplicación: **6.100€** (se trata de un coste muy bajo, gracias a que una sola persona realiza todas las fases del desarrollo, y que gran parte de los trabajos se realizan aprovechando la aplicación como proyecto de final de máster).
- **Oposiciones disponibles** desde el inicio del proyecto, para abarcar un amplio espectro de usuarios potenciales (se podrán ampliar en función de los avances en ventas):
  - Oposición a *Administrativo de las Cortes Generales*
  - Oposición a *Auxiliar Administrativo de la Administración General del Estado*
  - Oposición *TIC A1 de la Administración General del Estado*
- **Número de usuarios** nuevos estimado para la **versión gratuita**:
  - Primer año: **2.000 usuarios**
  - Segundo año: **3.000 usuarios**
  - Siguientes años: **8.000 usuarios**
- **Factor de conversión** de usuarios de la versión gratuita a la versión profesional: **10%**
- **Número de usuarios** nuevos estimado para la versión **profesional**:
  - Primer año: **200 usuarios**
  - Segundo año: **300 usuarios**
  - Tercer año y siguientes: **800 usuarios**

- **Precio** de la versión profesional: **3,99€**
- **Gastos anuales** estimados: Existen muy pocos gastos asociados adicionales a la mano de obra del desarrollo inicial de la Aplicación:
    - Licencias como desarrollador de Apple (para Android es gratuito) y licencia del entorno de desarrollo Webstorm: **200€**
    - Porcentaje de los ingresos que se queda la tienda de aplicaciones: **30%**
- **Retorno de la inversión (ROI)** en función de los años trascurridos desde el arranque de la aplicación profesional:

Año	Usuarios	Ingresos	Gastos	Diferencia	Acumulado
1	200	798,00 €	439,40 €	358,60 €	358,60 €
2	300	1.197,00 €	559,10 €	637,90 €	996,50 €
3	800	3.192,00 €	1.157,60 €	2.034,40 €	3.030,90 €
4	800	3.192,00 €	1.157,60 €	2.034,40 €	5.065,30 €
5	800	3.192,00 €	1.157,60 €	2.034,40 €	**7.099,70 €**
6	800	3.192,00 €	1.157,60 €	2.034,40 €	9.134,10 €
7	800	3.192,00 €	1.157,60 €	2.034,40 €	11.168,50 €
8	800	3.192,00 €	1.157,60 €	2.034,40 €	13.202,90 €
9	800	3.192,00 €	1.157,60 €	2.034,40 €	15.237,30 €
10	800	3.192,00 €	1.157,60 €	2.034,40 €	17.271,70 €

Figura 3: Retorno de la Inversión

Con estas estimaciones, a partir de **mediados del quinto año se amortizaría la inversión inicial**, y la aplicación comenzaría a producir unos **beneficios brutos antes de impuestos de aproximadamente 2.000€ anuales**.

3. Análisis Funcional

3.1 Definición de Requisitos

A continuación se muestra un cuadro con los requisitos principales de la aplicación. Se ha asignado un identificador único a cada requisito, para posteriormente identificarlos cuando sea necesario:

ID	Tipo	Nombre	Descripción
RFA1	Funcional	Acceso autenticado	Deberá permitirse el acceso autenticado a la aplicación. Un usuario sin autenticar no podrá acceder, deberá registrarse previamente.
RFA2	Funcional	Registro	Desde la propia aplicación deberá ofrecerse la posibilidad de registrarse en el sistema a los usuarios que todavía no lo hayan hecho.
RFA3	Funcional	Recordar usuario	Una vez autenticado, desde la aplicación se recordará el usuario actual para no volver a solicitar sus datos las siguientes veces que se acceda a la aplicación.
RFA4	Funcional	Cerrar sesión	Desde la aplicación deberá poder permitirse cerrar la sesión para olvidar al usuario actualmente autenticado.
RFA5	Funcional	Cambiar Oposición	Desde la pantalla principal de la aplicación deberá poder elegirse cuál es la oposición para la que se desea utilizar la aplicación. En cualquier momento se podrá modificar dicha oposición, lo que modificará los temas y exámenes correspondientes.
RFB1	Funcional	Cuestionario	Se deben poder realizar cuestionarios en los que elegir las siguientes opciones: - Número de preguntas - Temas de las preguntas - Límite de tiempo - Puntuación de aciertos y fallos

ID	Tipo	Nombre	Descripción
RFB2	Funcional	Orden aleatorio	Dentro de un cuestionario, el orden de las preguntas así como de las respuestas de cada una debe ser aleatorio (a excepción de aquellas preguntas que no permitan modificar el orden de sus respuestas por el contenido semántico de estas).
RFB3	Funcional	Proporción preguntas	Si se seleccionan varios temas para las preguntas de un determinado cuestionario, aunque se elijan las preguntas de forma aleatoria debe existir similar proporción de preguntas de cada tema.
RFB4	Funcional	Respuesta acertada	En el mismo momento de responder a una pregunta se deberá saber si se ha acertado, y si no es así se deberá indicar cuál era la respuesta correcta
RFC1	Funcional	Examen real	Se deben poder realizar exámenes reales de convocatorias anteriores. El orden de las preguntas y respuestas y la limitación de tiempo deberán ser iguales que el examen original.
RFC2	Funcional	Tests predefinidos	Se deben poder realizar una serie de tests predefinidos, en los que se respetará el orden de las preguntas y respuestas así como la limitación de tiempo definidos para dicho test.
RFC3	Funcional	Comenzar examen o test	Al comenzar un examen o test preestablecido, se mostrarán los datos de dicho examen: nombre, fecha de realización, tiempo disponible, número de preguntas, etc.
RFD1	Funcional	Ver resultados	Finalizado un cuestionario/examen/test deberán mostrarse los resultados obtenidos, que incluirán: - Número total de preguntas - Número de respuestas acertadas - Número de respuestas falladas - Número de preguntas no respondidas - Puntuación total

ID	Tipo	Nombre	Descripción
RFD2	Funcional	Repetir	Finalizado un cuestionario/examen/test deberá poder elegirse entre cualquiera de las siguientes opciones: - Repetir de nuevo el cuestionario/examen/test - Repetir sólo las preguntas no acertadas (falladas o no respondidas). - Realizar un nuevo cuestionario/examen/test
RT1	Técnico	Sistemas Operativos	La aplicación deberá funcionar en múltiples dispositivos, independientemente de su resolución os sistema operativo. Como mínimo se soportarán los smartphones o tabletas con sistema operativo *iOS* a partir de su versión 7 o *Android* a partir de su versión 4.1 *Jelly Bean*.
RT2	Técnico	Usabilidad	La usabilidad de la aplicación estará orientada a su uso desde un smartphone, el acceso a todas las opciones deberá ser intuitivo y poder realizarse con una sola mano. Se evitará en la medida de lo posible la introducción de textos con el teclado por parte del usuario.
RT3	Técnico	Backend	La aplicación deberá almacenar toda la información relativa a oposiciones, preguntas y exámenes en un servidor de backend. La modificación de cualquiera de estos datos podrá realizarse sin necesidad de que el usuario tenga que actualizar su aplicación en el market.

3.2 Usuarios y contexto de Uso

A continuación se muestra la ficha del principal usuario tipo de la aplicación, incluyendo también la descripción de un escenario de uso (se profundizará en este aspecto en los casos de uso del siguiente apartado).

Ficha de Usuario: José, opositor

Nombre: José

Edad: 37 años

Profesión: Técnico Administrativo (actualmente en paro)

Descripción de la persona: José está casado y ha trabajado gran parte de su vida en una empresa de ingeniería. Actualmente está en paro, y ha decidido prepararse la oposición a *Administrativos de las Cortes Generales*. José asiste tres días a la semana a una academia para prepararse la oposición. Además de las horas que dedica al estudio en su casa y en la academia, tiene bastantes "ratos muertos" en los que, si tuviese una aplicación en su dispositivo móvil que se lo permitiese, podría realizar tests de repaso para afianzar conceptos.

Descripción del escenario: Son la 18:00h y como cada lunes José acaba de tomar el tren para dirigirse desde su casa a la academia. El tren tarda aproximadamente 35 minutos en llegar.

José toma su Smartphone y abre la aplicación "*Testit Oposiciones*", para aprovechar este tiempo y realizar algunos tests. Comienza realizando algunos cuestionarios aleatorios, pero cuando lleva un rato decide repasar los tests de los temas que van a estudiar hoy en la academia. Por suerte, el profesor de la academia también cuelga los tests en la aplicación una semana antes, de forma que sus estudiantes pueden preparar también las clases a través de ella.

Otra vez en el tren de vuelta a casa, José aprovecha el tiempo para realizar el examen de Administrativos de la convocatoria anterior. Se encuentra satisfecho, cada vez obtiene mejores resultados, se muestra optimista ante la posibilidad de aprobar la oposición.

4. Diseño de la Solución

4.1 Arquitectura tecnológica

La arquitectura tecnológica de la solución es la habitual de una aplicación realizada con ionic framework, con las siguientes características:

- Ejecución de la lógica en la capa cliente, desarrollo basado en ***ionic framework*** y ***angularjs***.
- Integración con dispositivos basada en ***WebKit*** **(*iOS* y *Android*)**.
- Invocación de funcionalidades nativas del dispositivo utilizando ***apache cordova***.
- Uso de la funcionalidad ***Local Storage*** de Javascript para conservar entre distintas sesiones el estado actual y el usuario autenticado dentro del dispositivo.
- Almacenamiento de datos en la nube en formato JSON utilizando ***Firebase*** con **llamadas asíncronas** que no bloquean al usuario.
- **Compresión de ficheros JSON** para minimizar el tráfico de red entre Firebase y el dispositivo.
- Uso de ***Firebase*** también para la **autenticación** de usuarios basada en contraseñas.

A continuación se muestra un diagrama con la arquitectura tecnológica de la solución completa:

Figura 4: Arquitectura tecnológica

Selección de tecnologías en el lado Cliente

La principal decisión en cuanto a la selección tecnológica en el lado de cliente ha sido la de utilizar **ionic framework**. Ionic permite el desarrollo de **aplicaciones híbridas** para diversas plataformas, lo cual encaja muy bien con la necesidad de implementar la aplicación tanto para sistema operativo Android como iOS.

Ionic framework se basa en la programación en la capa cliente con **HTML5** y **Javascript**, haciendo gran uso de la librería **angularjs**. Las aplicaciones de ionic se empaquetan utilizando **apache cordova**, que se encarga de aglutinar todo el contenido html/css/js en un compilado para poder ser ejecutado en las distintas plataformas. Cordova a su vez ofrece la posibilidad de acceder a las funcionalidades nativas del dispositivo a través de librerías javascript.

La principal razón por la que se han elegido este framework es que de todas las tecnologías abordadas durante el estudio del máster, esta es **la que más me ha atraído y a la que más futuro veo**, por diversos motivos:

- Cuando se planea implementar una aplicación para distintos sistemas operativos móviles y no se cuenta con recursos elevados, el desarrollo de **aplicaciones híbridas** permite implementar una única vez y desplegarlo en diversas plataformas, ahorrando tiempo y esfuerzos.

- **Ionic framework** proporciona una serie de herramientas que agilizan mucho el desarrollo de aplicaciones (ionic view, creator, etc.). La reciente publicación de su **versión 2** ha introducido numerosas mejoras que aportan mucha potencia y versatilidad.

- La **potencia de angularjs** es enorme, y su expansión en el mercado cada vez es mayor, se está consolidando como el framework javascript por excelencia.

Por otro lado, el uso de la funcionalidad ***Local Storage*** de Javascript para conservar entre distintas sesiones el estado actual y el usuario autenticado dentro del dispositivo permite almacenar en el dispositivo toda la información de estado de la aplicación sin necesidad de acceder a funcionalidades nativas, asegurando así su compatibilidad entre plataformas.

Respecto a la arquitectura y funcionamiento interno de **angularjs**, es considerado por la mayoría como un **framework MVVM** (Model-View-View Model). Se trata de un refinamiento del modelo MVC (Model-View-Controller) utilizado para simplificar la separación de la capa de presentación.

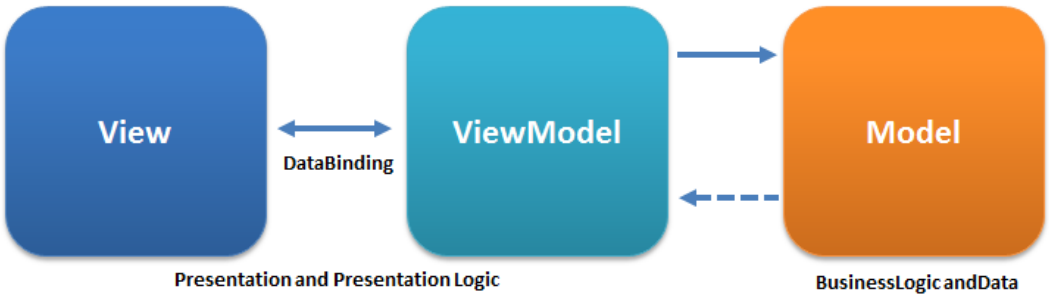


Figura 5: Modelo MVVM

En el modelo MVVM, la lógica de negocio se almacena en el *Presenter*, y la vista está completamente separada del modelo. Mientras que el *Presenter* no es consciente de la vista, la vista sí que es consciente del *Presenter* - el *Presenter* en MVVM se utiliza para representar una vista abstracta de la interfaz de usuario. En el patrón de diseño de MVVM, View está activo y contiene comportamientos, eventos e información de enlazado de datos. La vista en MVVM no es la responsable de la gestión de la información de estado: la vista se sincroniza con el *ViewModel*. El *ViewModel* en MVVM es responsable de la separación de la presentación y expone métodos y comandos para administrar el estado de una vista y manipular el Modelo.

Selección de tecnologías en el lado Servidor

La selección de la plataforma para albergar los datos en el lado servidor no ha sido fácil. En las primeras asignaturas que se realizaron en el máster se utilizó la plataforma ***Parse.com***, que me pareció muy intuitiva y potente, así como fácil de integrar desde las distintas plataformas.

Sin embargo, a finales de Enero de 2016, *Facebook* (que adquirió *Parse*) anunció que **discontinuaba esta plataforma** (así lo refleje en esta entrada de mi blog: http://manuelpereiragonzalez.blogspot.com.es/2016/02/facebook-cierra-parse-el-backend-como.html). En ese momento comencé a evaluar alternativas de "backend como servicio" para poder migrar algunas Apps que tenía implementadas y utilizando Parse, y fue ahí cuando conocí la existencia de *Firebase*, leyendo este post: http://highscalability.com/blog/2016/2/2/the-big-list-of-alternatives-to-parse.html.

Estuve evaluando las distintas alternativas que se ofrecen en este enlace: https://www.maketecheasier.com/facebook-parse-alternatives/ y tras algunas pruebas con Backendless (https://backendless.com/), moBack (https://www.moback.com/) y Firebase (https://firebase.google.com) me decidí por este último, principalmente por los siguientes **motivos**:

- **Funcionalidad**: Para el backend el desarrollo de mi aplicación requería principalmente dos funcionalidades: el almacenamiento de datos en formato json, y la gestión del registro y autenticación de usuarios. Firebase proporcionaba estas dos funcionalidades, además del envío de notificaciones Push para una posible ampliación futura de la funcionalidad de la aplicación cuando se monetice, y la integración de la autenticación con plataformas como Google o Facebook.

- **Fácil integración y facilidad de uso**: La integración de firebase a través de su API javascript es muy sencilla, y existe una amplia documentación al respecto (ver https://firebase.google.com/docs/web/setup?hl=es y https://firebase.google.com/docs/database/web/start?hl=es).

- **Llamadas asíncronas**: El API de javascript de Firebase incluye llamadas asíncronas, lo que permite no bloquear el interfaz de usuario mientras que se espera la respuesta del servidor.

- **Google**: El hecho de que detrás de Firebase se encuentre una empresa como google me proporciona seguridad respecto a su continuidad.

4.2 Diseño del modelo de datos

Como se ha mencionado previamente, los datos principales de la aplicación serán almacenados en ***Firebase***, en formato ***JSON***. Para ello se utilizará la característica "Storage" de firebase para almacenar ficheros completos. Adicionalmente, se utilizará la autenticación de firebase basada en usuario y contraseña (en el futuro podría permitirse la autenticación OAuth con usuarios de redes sociales).

A continuación se muestran ejemplos de los distintos ficheros en formato JSON que conformarán los datos de la aplicación:

Listado de Oposiciones

Nombre del fichero: *oposiciones.json*
Ejemplo:

```
{
  "oposiciones":
  [
    { "numero" : "0", "nombre": "2016: Administrativos Cortes Generales"},
    { "numero" : "1", "nombre": "2016: TIC A1 AGE"}
  ]
}
```

Listado de exámenes de una oposición

Nombre del fichero: *exámenes_oposicion_[IDOPOSICION].json*
Ejemplo:

```
{
  "examenes":
  [
    { "numero" : "0", "nombre": "Admtvos. Cortes Generales 2014"},
    { "numero" : "2", "nombre": "Aux. Admtvos. AGE 2000"},
    { "numero" : "6", "nombre": "Aux. Admtvos. AGE 2001"},
    { "numero" : "5", "nombre": "Aux. Admtvos. AGE 2002"},
    { "numero" : "7", "nombre": "Aux. Admtvos. AGE 2003"},
    { "numero" : "8", "nombre": "Aux. Admtvos. AGE 2004"},
    { "numero" : "9", "nombre": "Aux. Admtvos. AGE 2005"},
    { "numero" : "10", "nombre": "Aux. Admtvos. AGE 2006 (preguntas 1-30, 42-45, 61-67)"},
    { "numero" : "11", "nombre": "Aux. Admtvos. AGE 2007 (preguntas 1-30, 46-49, 57-65)"},
    { "numero" : "12", "nombre": "Aux. Admtvos. AGE 2008 (preguntas 1-19, 21-30, 36-40, 46
    { "numero" : "1", "nombre": "Ujieres Cortes Generales 2002"},
    { "numero" : "3", "nombre": "Ujieres Cortes Generales 2005"},
    { "numero" : "4", "nombre": "Guías Senado 2016 - Interna"}
  ]
}
```

Preguntas de un tema concreto de una oposición

Nombre de fichero: *preguntas_oposicion_[IDOPOSICON]_tema_[IDTEMA].json*
Ejemplo:


```
{
  "tema": "32",
  "titulo": "La Unión Europea (I). Los Tratados originarios y modificativos. El proces
  "preguntas": [
    {
      "numero": "e12_18",
      "tema": "34",
      "enunciado": "Señale a quién esta atribuido el derecho de iniciativa normativa,
      "alternativas": [
        {"valor": "La Comisión."},
        {"valor": "La Comisión y el Parlamento Europeo."},
        {"valor": "El Consejo y la Comisión."},
        {"valor": "Ninguna de las anteriores"}],
      "correcta": "0", "inmutable": "1"
    },

    {
      "numero": "e12_17",
      "tema": "32",
      "enunciado": "Señale cuándo se celebraron las primeras elecciones al Parlamento
      "alternativas": [
        {"valor": "1974"},
        {"valor": "1977"},
        {"valor": "1979"},
        {"valor": "1981"}],
      "correcta": "2"
    }
  ]
}
```


Como puede comprobarse, además de los datos habituales de cada pregunta como el enunciado o las posibles respuestas, se ha incluído un indicador opcional denominado ***inmutable***. Si aparece este indicador con valor "1", esto significa que no se pueden cambiar el orden de las respuestas de esta pregunta a la hora de aleatorizarlas (porque perdería su sentido).

Examen o Test preestablecido

Nombre del fichero: examen_[IDEXAMEN].json
Ejemplo:


```
{
  "parametros" : {
    "nombre": "Admtvos. Cortes Generales 2014",
    "puntosacierto": "1",
    "puntosfallo": "0.5",
    "limitarTiempo": "true",
    "tiempoLimite": "120"
  },
  "preguntas": [
    {
      "numero" : "e0_1",
      "tema": "",
      "enunciado": "Señale la frase incorrecta:",
      "alternativas": [
        {"valor": "Pedro no habla; está siempre callado."},
        {"valor": "Este coche se ha hecho un boyo al chocar con el otro."},
        {"valor": "María colecciona sellos."},
        {"valor": "Suelo desayunar leche con bollos."}],
      "correcta": "1"
    },
    {
      "numero" : "e0_2",
      "tema": "",
      "enunciado": "Señale la frase correcta:",
      "alternativas": [
        {"valor": "Las sandías que más me gustan son las rayadas."},
        {"valor": "MI padre pescó una ralla en el mar."},
        {"valor": "Con el pan duro haremos pan rayado."},
        {"valor": "Se ha rallado el cristal de la mesa."}],
      "correcta": "0"
    },
    {
      "numero" : "e0_3",
      "tema": "",
      "enunciado": "Marque la frase con un verbo incorrecto:",
      "alternativas": [
        {"valor": "Han salido cuatro voluntarios."},
        {"valor": "Habían muchas niñas invitadas a la fiesta."},
        {"valor": "Ellos han salido cuando nosotros hemos entrado."},
        {"valor": "Cuando hubo suficientes personas, cerraron la puerta."}],
      "correcta": "1"
    },
```


4.3 Estructuración de ficheros de código fuente

> **Nota Importante**: Junto con esta memoria no se hace público el código fuente de la aplicación, puesto que, como se ha comentado, se pretende realizar una monetización futura del producto

Al tratarse de un proyecto realizado con *ionic framework*, parte de la estructuración del código en carpetas viene determinada por los templates de partida de dicho framework (más información en http://ionicframework.com/docs/concepts/structure.html y en http://mcgivery.com/structure-of-an-ionic-app/).

Se han tomado algunas decisiones no predeterminadas que afectan a la estructuración de los ficheros y su ubicación:

- Para multitud de decisiones de diseño de código se han utilizado las directrices definidas en la **guía de estilo de angularjs** de la siguiente URL: https://angular.io/docs/ts/latest/guide/style-guide.html
- Se ha utilizado el siguiente **repositorio git** donde almacenar el código: https://gitlab.com/manuelpereira/cuestionario.git
- Controladores: Se ha separado la lógica de negocio en dos controladores javascript distintos:
    - appController.js: Para la gestión del login, registro y menú principal.
    - quizController.js: Para la gestión de cuestionarios, exámenes y tests predefinidos.
- En cada controlador se ha creado una variable que actúa como **flag para modo debug**, de forma que pueda activarse este modo de depuración para detertar errores (tanto técnicos como en el contenido de las preguntas).

En la siguiente figura se muestra a nivel orientativo la estructura de ficheros del directorio *www* donde ionic framework almacena el contenido de la capa cliente y lógica de negocio:

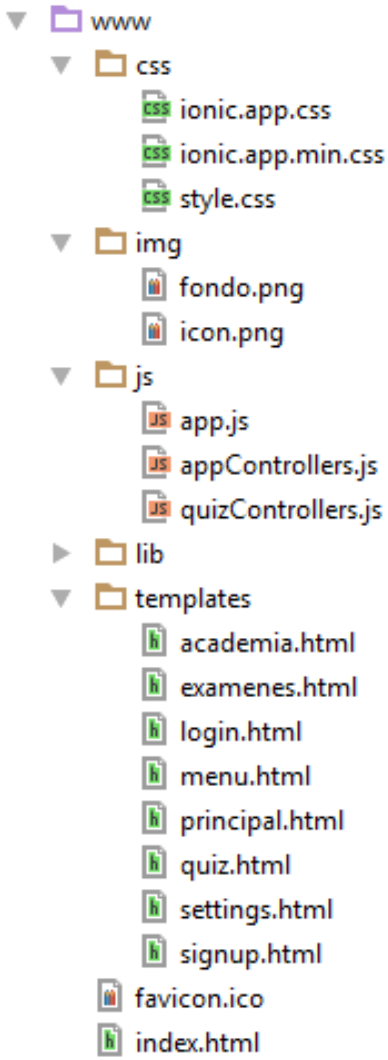


Figura 6: Estructura de ficheros

4.4 Uso de funcionalidades nativas de los dispositivos

En ocasiones es necesario acceder a funcionalidades nativas del dispositivo, no accesibles de forma habitual desde el estándar HTML5/CSS3 de las aplicaciones web no desplegadas en dispositivos móviles.

Para acceder a estas funcionalidades nativas desde ionic framework se ha utilizado el **plugin ng-cordova para angujarjs** (http://ngcordova.com/), para el uso de funcionalidades como:

- Plugin ***insomnia***: Impedir que se bloquee el dispositivo en ciertas ocasiones. En concreto, cuando se está realizando un cuestionario o examen, se utiliza este plugin para que el dispositivo no se bloquee ni entre en modo standby.
- Plugin ***vibration***: Hacer que vibre el dispositivo en determinados momentos, por ejemplo cuando se está realizando un cuestionario o examen y se acaba el tiempo preestablecido para realizarlo.

4.5 Diseño visual

Al tratarse de una aplicación pensada en una posible monetización futura, se ha tratado de definir una **imagen lo más profesional posible**, centrando todo el diseño en la **usabilidad** tanto en dispositivos **smartphone** como en **tablets**.

En los siguientes apartados se muestran algunas decisiones que se han tomado respecto a los distintos aspectos del diseño.

Elección de colores

Ionic framework provee una paleta de colores preestablecidos, que abarcan una gama de colores/emociones a trasmitir:

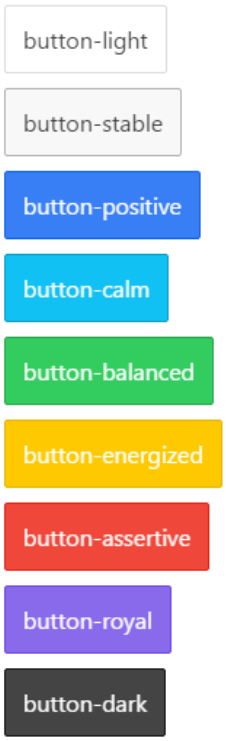


Figura 7: Colores predefinidos Ionic Framework

En general se ha optado por **tonos fríos**, predominando los azules, violetas y grises, y sólo utilizando otros tonos más cálidos para determinados botones o mensajes que requieran la atención del usuario.

Fondo y Logotipo

Como imagen de fondo se ha seleccionado una con los tonos apropiados, que permita su dimensionamiento en distintas resoluciones y que no distraiga la atención de los componentes visuales situados encima de ella.

Se ha tenido que dimensionar la ***splashscreen*** y el ***icono*** de la aplicación para todas las resoluciones y orientaciones posibles (**más de 30 combinaciones** entre dispositivos *iOS* y *Android* en orientación *laptop* y *portrait*).

A continuación se muestra una parte de la imagen seleccionada para el fondo, así como el icono diseñado expresamente para la aplicación (por separado y sobre el fondo):

Figura 8: Fondo e icono de la aplicación

Figura 9: Icono sobre fondo cuadrado

4.6 Árbol de Navegación

El árbol de navegación nos permite establecer de manera jerárquica y estructurada la información y funcionalidades de la aplicación. El mero ejercicio de realizar el árbol de navegación sirve para aclarar las ideas al diseñador, haciendo que se plantee las preguntas oportunas y permitiendo identificar de un vistazo callejones sin salida, funcionalidades inaccesibles, duplicidades, etc. A través del árbol de navegación debe poder accederse a todas las funcionalidades de la aplicación definidas en el apartado de análisis de requisitos de este documento.

A continuación se muestran una serie de decisiones y características que se han aplicado para el diseño del árbol de navegación de la aplicación:

- A cada pantalla del árbol de navegación se le ha dado un **nombre descriptivo**, que se utilizará posteriormente en los siguientes apartados.
- Las **transiciones** entre pantallas que han dibujado con flechas direccionales que incluyen un texto con la acción del usuario que lleva de una pantalla a la otra.
- Se han utilizado recuadros de tamaño menor y con línea discontinua para mostrar popups informativos que aparecen en la transición de una pantalla a otra.
- Para evitar caminos sin salida, cada pantalla o pantallas inaccesibles, cada pantalla contiene **como mínimo una flecha de entrada y una de salida**.
- Se ha definido una pantalla con un menú principal que contiene las principales funcionalidades de la aplicación. Cada opción del menú principal y sus pantallas asociadas se ha dibujado con un color diferente.
- El menú principal **tres opciones** principales:
    - Cuestionario: Realización de cuestionario parametrizable (tiempo, preguntas, temas, etc.).
    - Exámenes reales: Realización de exámenes de otras convocatorias.
    - Tests predefinidos: Realización de tests predefinidos, por ejemplo los de una academia determinada.
- Adicionalmente, desde la pantalla principal también se puede modificar la oposición activa, así como acceder a la pantalla de configuración de la aplicación.
- Durante el desarrollo del árbol de navegación se planteó la duda de cómo facilitar la posibilidad de “volver atrás” en las distintas pantallas. Como decisión general de diseño, se ha optado por **siempre permitir la opción de volver atrás en cualquier pantalla** (en algunos casos esto significa cancelar la opción actual).

El siguiente diagrama muestra el árbol de navegación:

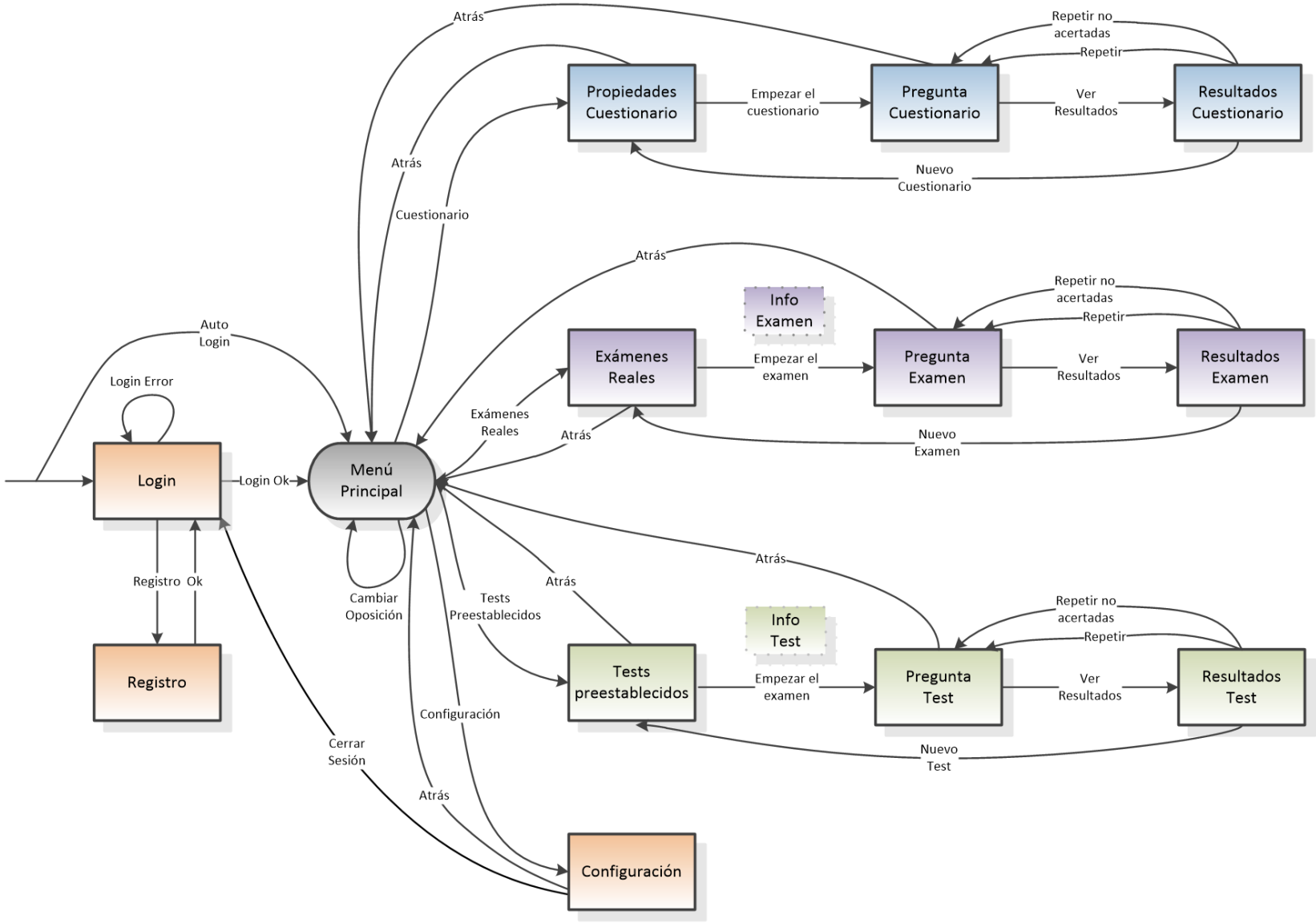


Figura 10: Árbol de navegación

4.7 Prototipo de pantallas

En este apartado se muestra el diseño prototipado de algunas de las pantallas de la aplicación. Para cada pantalla se muestra una **vista previa en un smartphone de alta gama y alta resolución**, tanto en orientación *portrait* como *landscape*:

Pantalla de login: Permite el acceso del usuario al sistema introduciendo su usuario y contraseña:

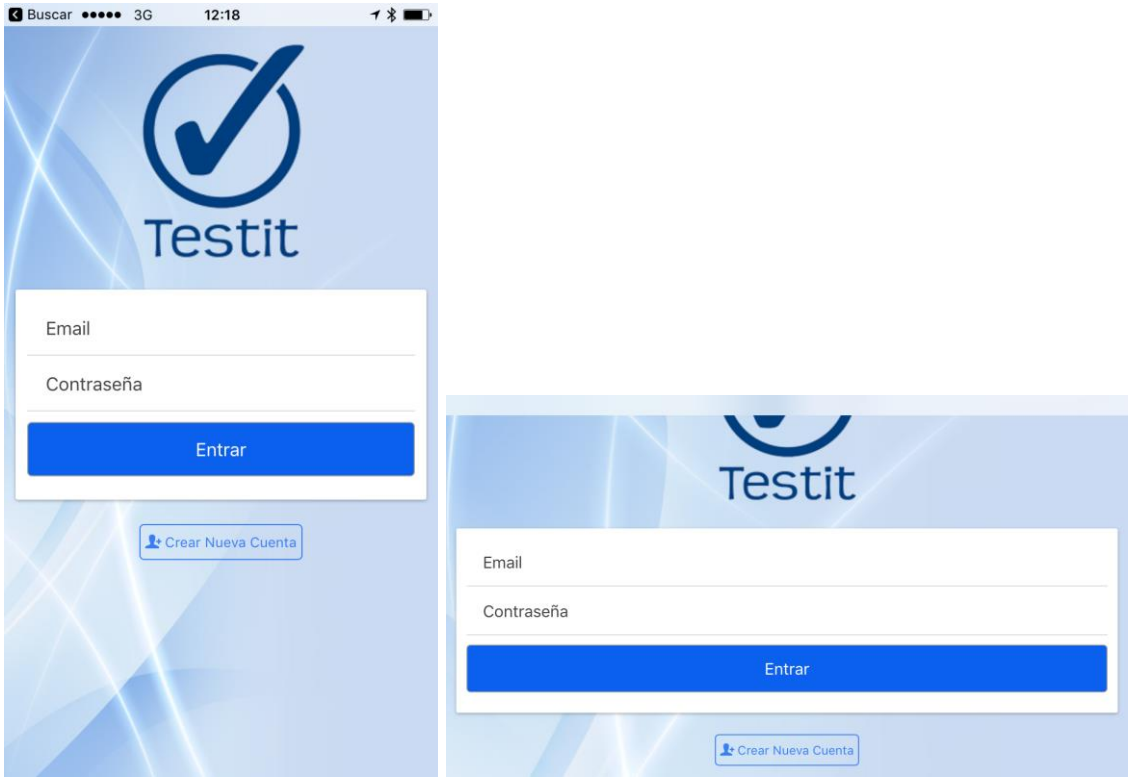


Figura 11: Pantalla Login

Menú principal: Muestra las tres opciones principales de la aplicación, así como la posibilidad de cambiar de oposición o acceder a las opciones de configuración:

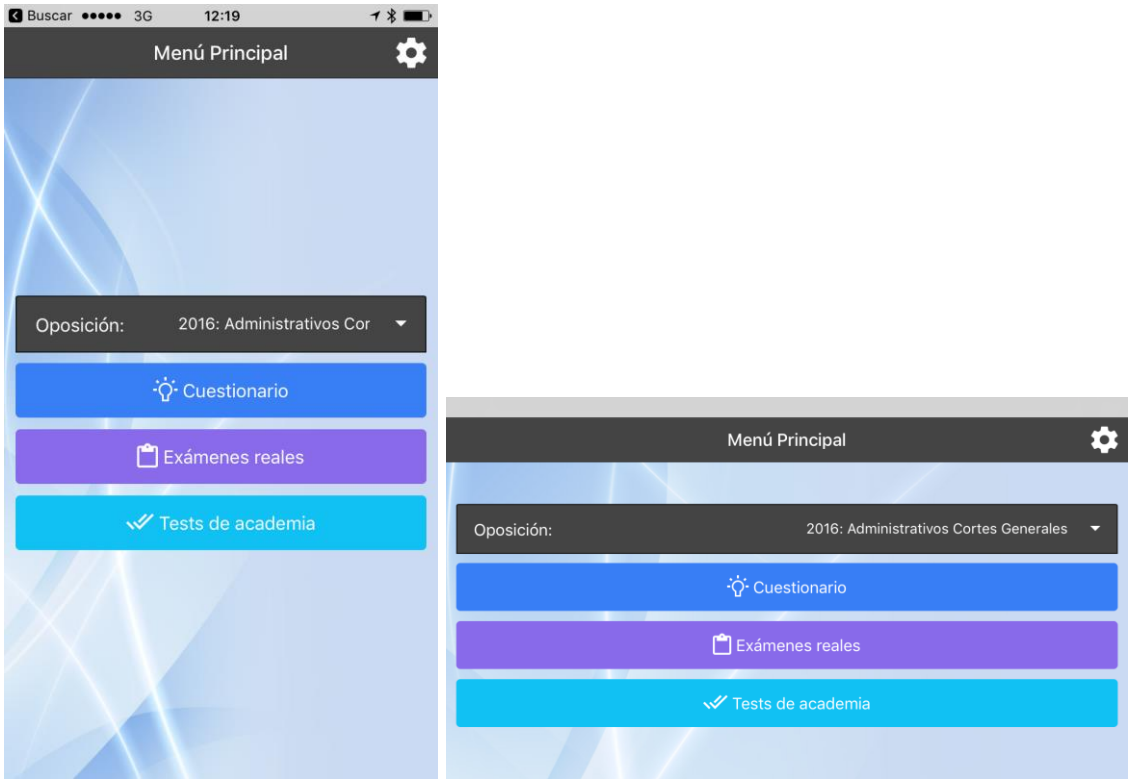


Figura 12: Pantalla Menú principal

Cuestionario: Permite realizar un cuestionario, seleccionando distintas opciones:

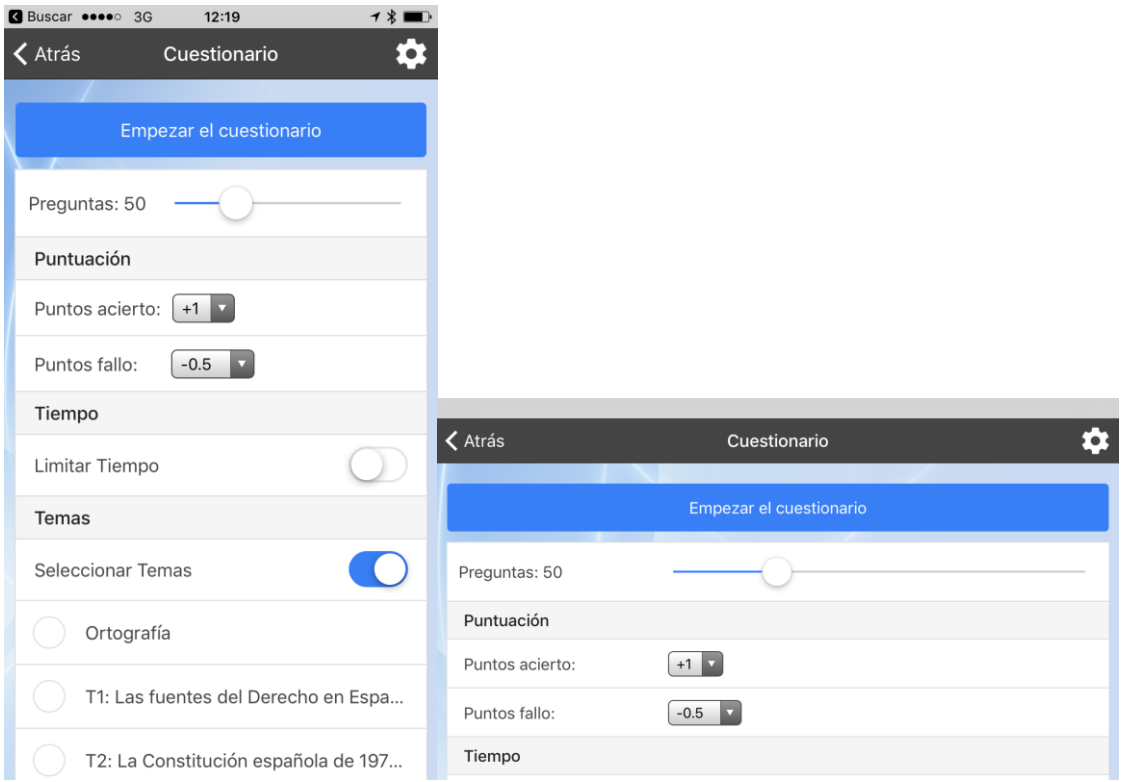


Figura 13: Pantalla Propiedades de cuestionario

Pregunta: Muestra una pregunta concreta del cuestionario, permite responder pulsando sobre una de las opciones mostradas:

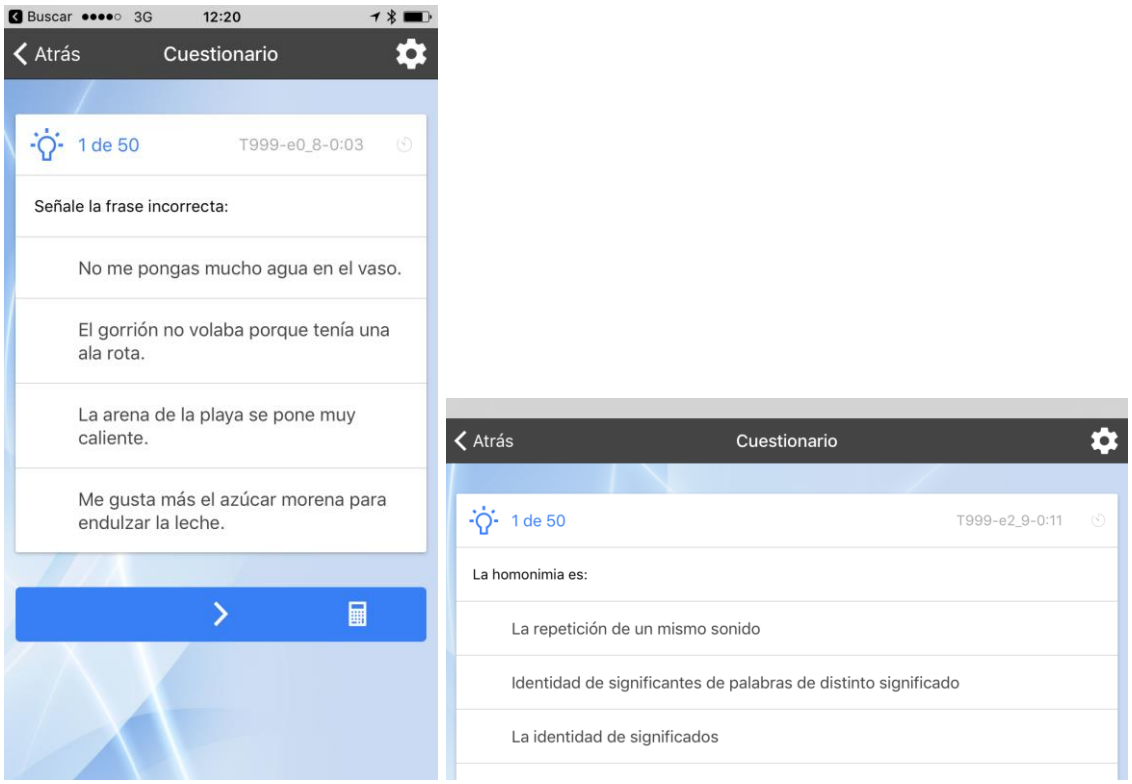


Figura 14: Pantalla Pregunta cuestionario

Resultados: Muestra los resultados del cuestionario realizado:

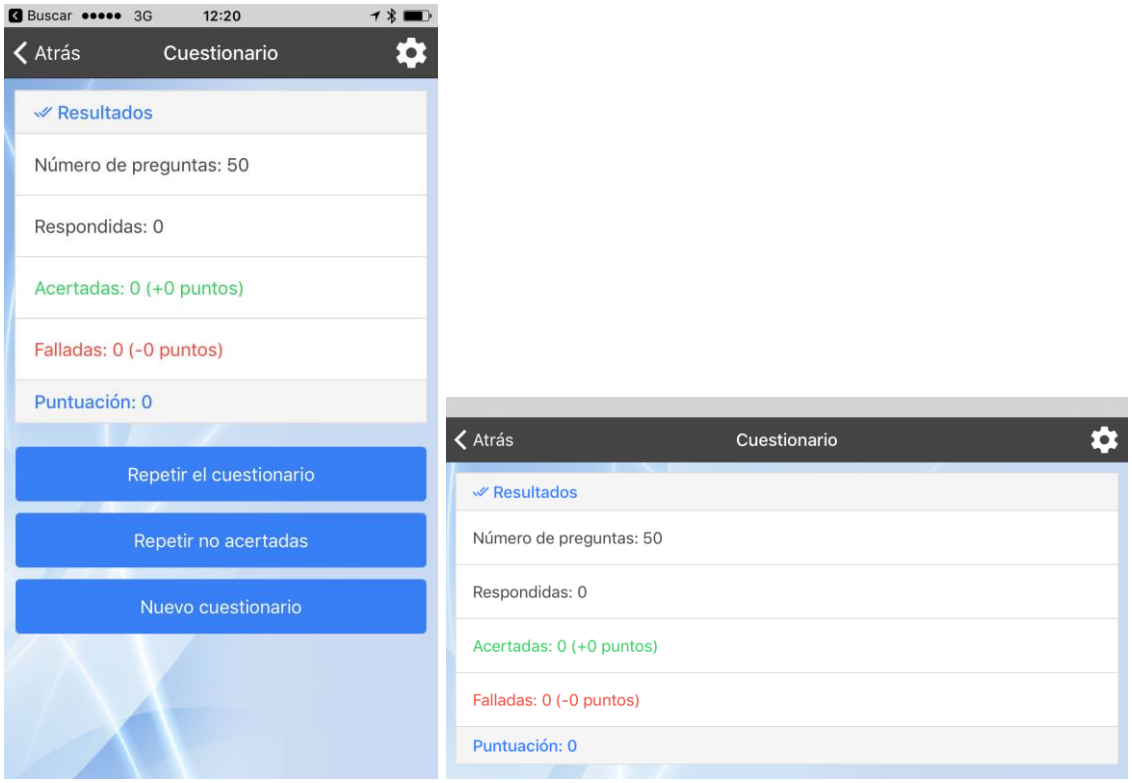


Figura 15: Pantalla Resultados cuestionario

Exámenes reales: Permite realizar exámenes reales de anteriores convocatorias:

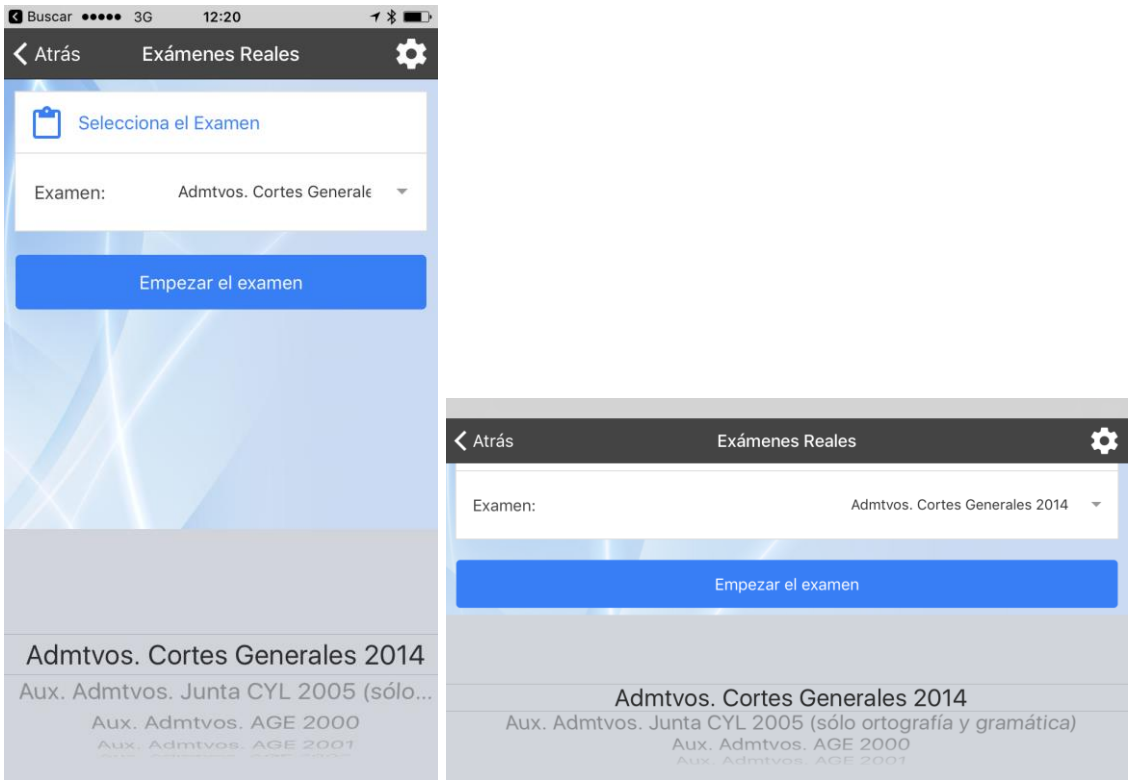


Figura 16: Pantalla Exámenes reales

Inicio de examen: Muestra los datos de un examen antes de comenzarlo:

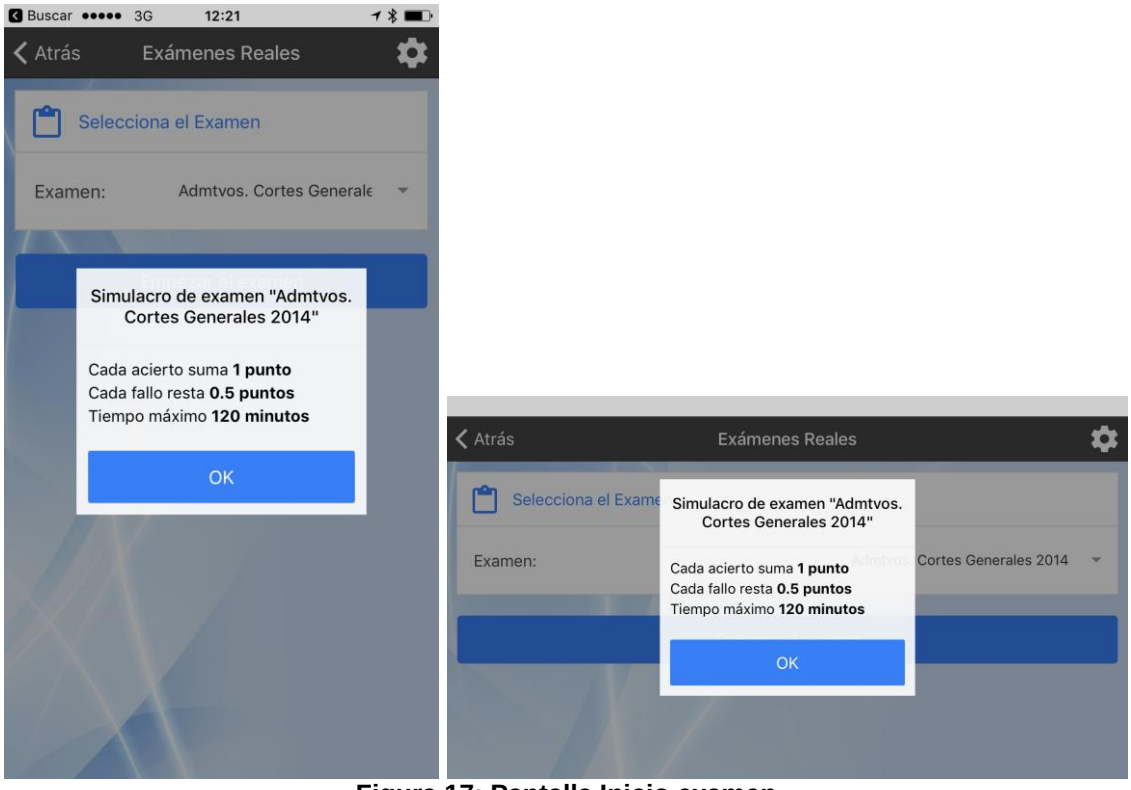


Figura 17: Pantalla Inicio examen

5. Implementación

5.1 Entorno de desarrollo

Como entorno de desarrollo para la implementación de la aplicación se ha utilizado ***WebStorm***. Este IDE se integra perfectamente con *ionic framework*, y permite el desarrollo ágil de código en *HTML* y *javascript* (tiene integración específica con *angularjs*).

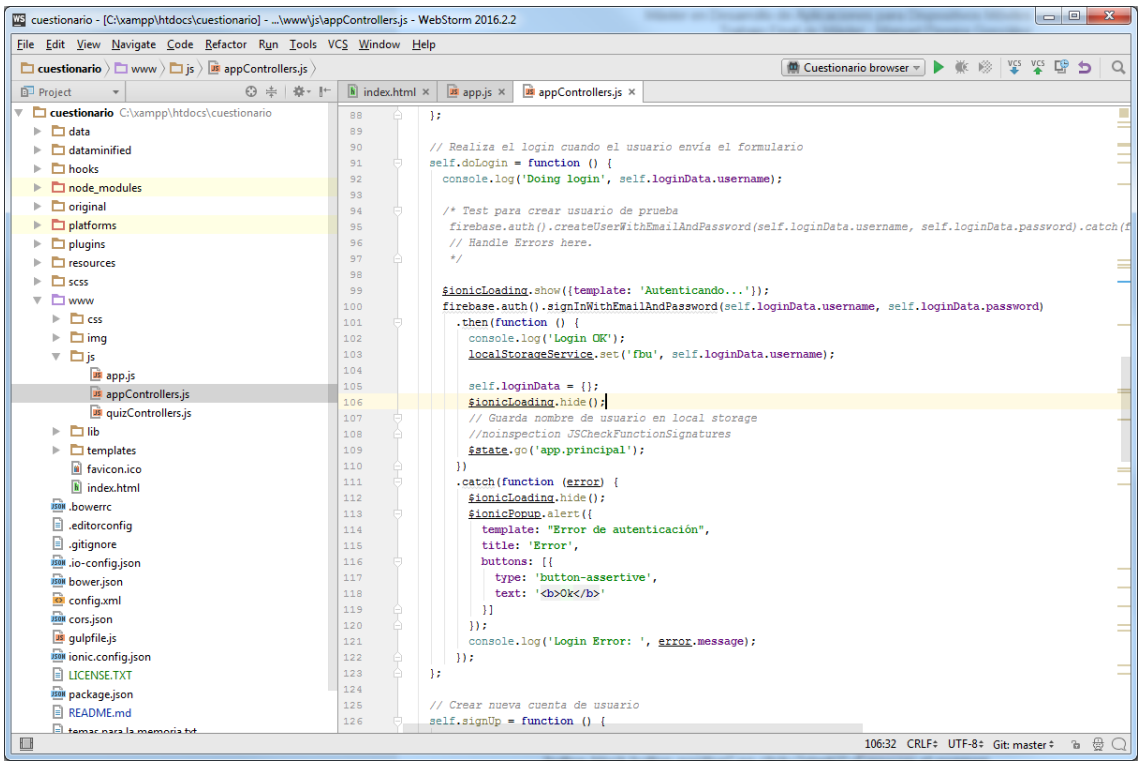


Figura 18: Entorno de desarrollo WebStorm

Inicialmente se utilizó una licencia de evaluación de WebStorm que permitía su uso por un periodo de 30 días, y posteriormente se ha utilizado una licencia de estudiante obtenida utilizando la dirección de correo electrónico de la UOC.

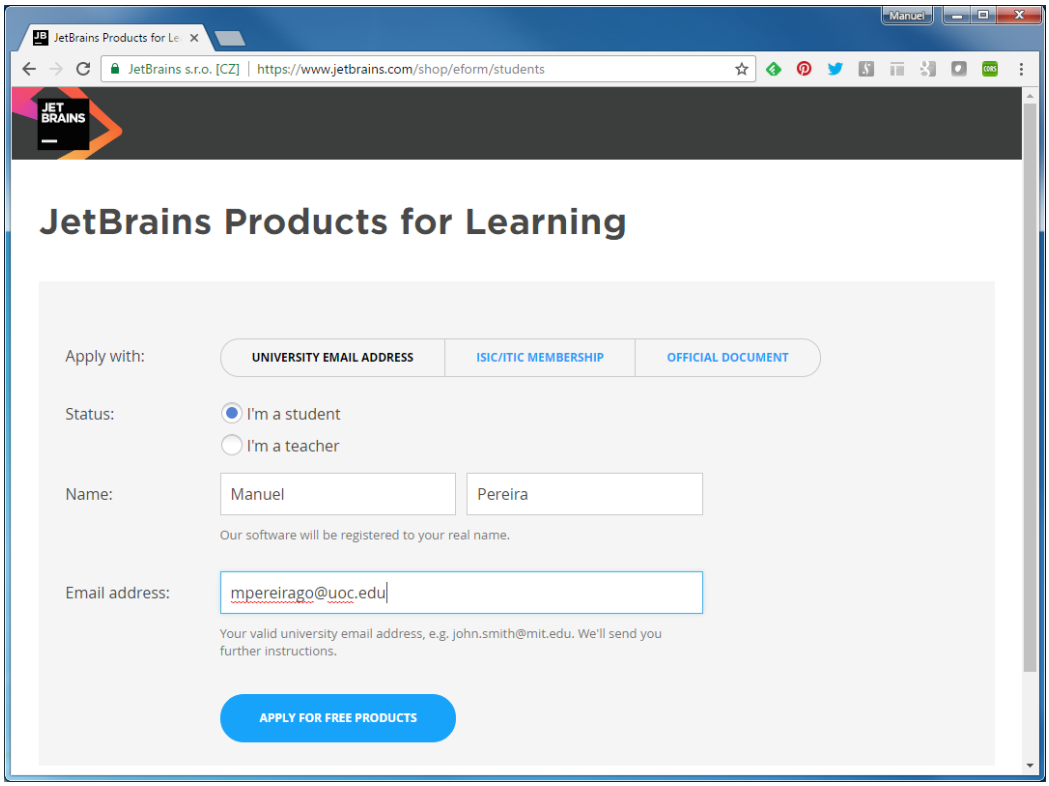


Figura 19: Adquisición de licencia de estudiante de WebStorm

Como respositorio de código se ha utilizado GitLab, todo el código del proyecto se encuentra almacenado en la siguiente URL:

https://gitlab.com/manuelpereira/cuestionario.git

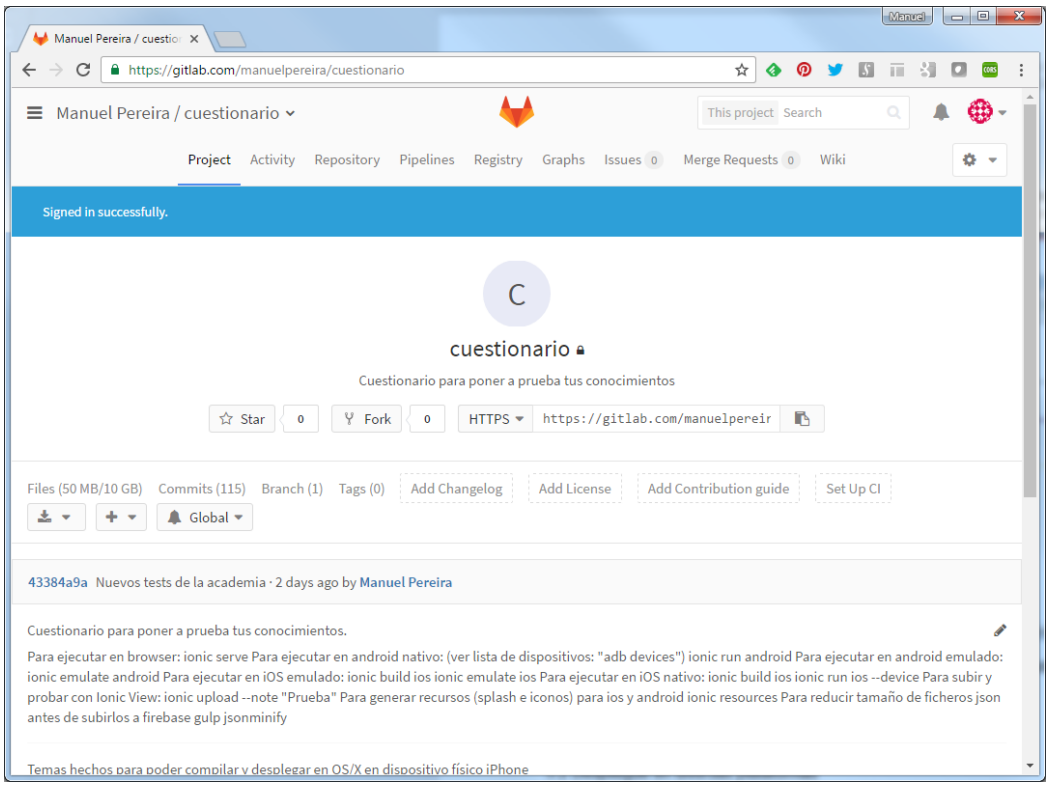


Figura 20: Respositorio de código GitLab

Para el diseño básico de las pantallas de la aplicación se ha utilizado ***Ionic Creator***. Se trata de una herramienta online de diseño visual de pantallas que permite la generación automática del código fuente asociado a dichas pantallas:

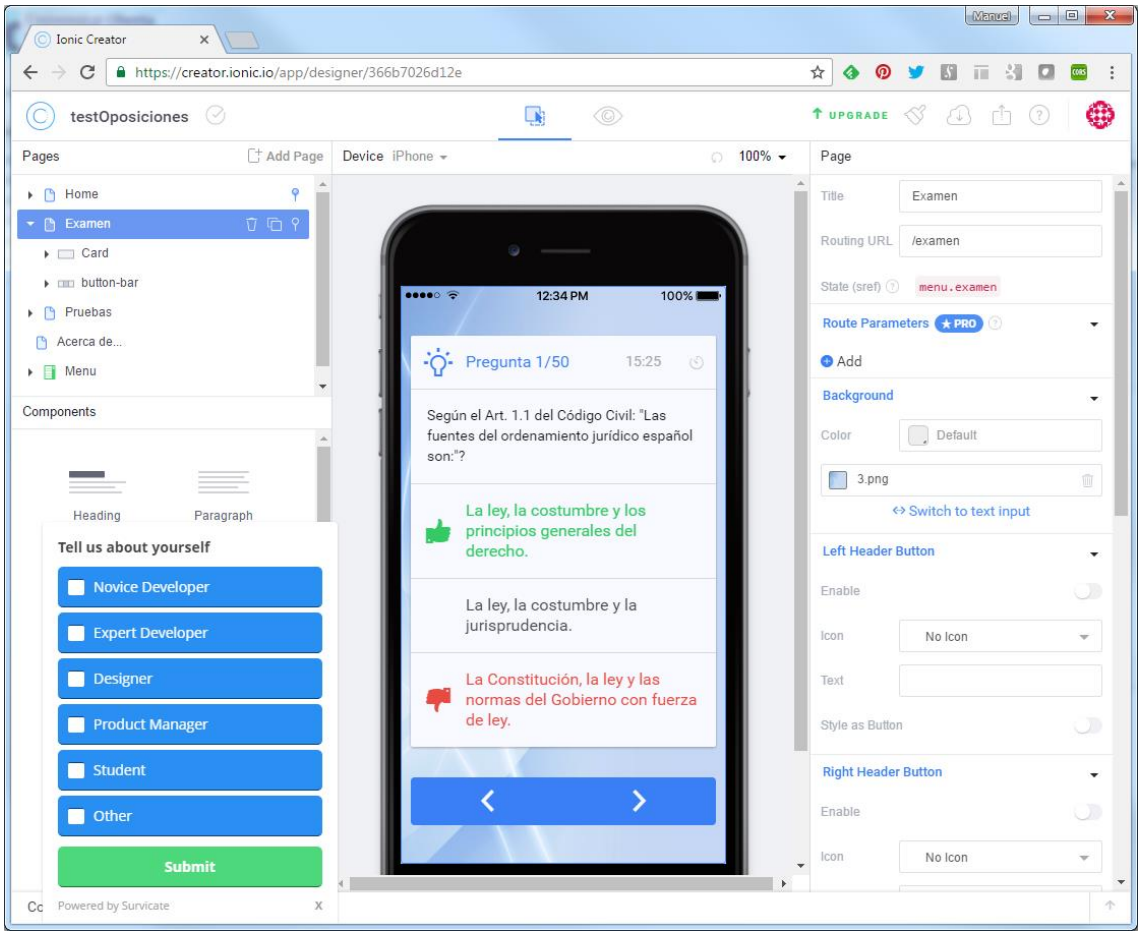


Figura 21: Ionic Creator

5.2 Despliegue en diversas plataformas

La aplicación se ha desarrollado utilizando **ionic framework**, que permite el desarrollo **híbrido** y por tanto su utilización y despliegue en diversas plataformas de dispositivos móviles. Se han utilizado y probado las siguientes plataformas:

1) **Web browser**: El desarrollo de todas las funcionalidades no nativas se ha realizado arrancando el servidor ionic en un **PC local** y accediendo directamente vía web browser (se ha probado con las últimas versiones de *Google Chrome*, *Internet Explorer*, Safari y *Firefox*).

Figura 22: Web Browsers utilizados

2) **Ionic View**: Esta App disponible en los markets de los distintos sistemas operativos, te permite visualizar tus proyectos sin necesidad de haberlos publicado. Esto permitirá probar la aplicación sin necesidad de tener que publicarla en los markets ni desplegarla en nativo (un proceso bastante lento).

Figura 23: Ionic View

3) **Android Emulado y Nativo**: Se han realizado pruebas de forma nativa tanto en el emulador de dispositivos de Android como en smartphones Androids. Se han utilizado los siguientes dispositivos:

i. Samsung Galaxy S5:

   http://www.samsung.com/es/consumer/mobile-devices/smartphones/galaxy-s/SM-G901FZWAAMO

ii. Huawei P8 Lite:

   http://www.huaweispain.com/microsites/huawei-p8lite/

4) **iOS Nativo**: Se han realizado pruebas de forma nativa tanto en el emulador de dispositivos de iOS como en smartphones y tabletas. La compilación, emulación y despliegue en dispositivos iOS se ha realizado utilizando un iMac. Se han probado los siguientes dispositivos:

i. Apple iPhone 6:

http://www.apple.com/es/iphone/

ii. Apple iPhone 6s:

http://www.apple.com/es/iphone/

iii. Apple iPad 2:

http://www.apple.com/es/ipad/

5.3 Pruebas con usuarios reales

Finalizada la primera versión Beta de la aplicación, se han realizado **pruebas con cuatro usuarios reales durante un periodo de un mes**, tanto en dispositivos Android como iOS. Todos los usuarios se encontraban estudiando la Oposición para el Cuerpo Administrativo de las Cortes Generales, por lo que **han realizado un uso extensivo de la aplicación durante dicho mes**.

Los usuarios han validado la funcionalidad de la aplicación, confirmando que **les era de gran utilidad**, y sugiriendo multitud de ideas.

Como resultado de estas pruebas con usuarios reales también se han detectado diversas incidencias que se han procedido a solucionar, según se detalla en el siguiente apartado.

5.4 Manual de compilación

> **Nota Importante**: Junto con esta memoria no se hace público el código fuente de la aplicación, puesto que, como se ha comentado, se pretende realizar una monetización futura del producto

```
********************************************************************************
INSTRUCCIONES DE COMPILACIÓN Y EJECUCIÓN
********************************************************************************

Para compilar y generar la aplicación para las distintas plataformas es necesario
instalar "ionic framework". Instrucciones de instalación disponibles en:
http://ionicframework.com/docs/guide/installation.html

********************************************************************************

Para ejecutar en browser:
  ionic serve
Para ejecutar en android nativo:
  (ver lista de dispositivos: "adb devices")
  ionic run android
Para ejecutar en android emulado:
  ionic emulate android
Para ejecutar en iOS emulado:
  ionic build ios
  ionic emulate ios
Para ejecutar en iOS nativo:
  ionic build ios
  ionic run ios --device
Para subir y probar con Ionic View:
  ionic upload --note "Prueba"
Para generar recursos (splash screen e iconos) para ios y android
  ionic resources
Para reducir tamaño de ficheros json antes de subirlos a firebase
  gulp jsonminify

********************************************************************************

Acceso a la consola de Firebase:
  https://console.firebase.google.com/project/cuestionario-9996f/overview

********************************************************************************

Para poder compilar en OS/X y desplegar en dispositivos físicos iPhone/iPad realizar:

sudo npm install -g cordova
sudo npm install -g ionic
npm rebuild node-sass
ionic platform rm ios
ionic platform add ios
cordova platform add ios
cordova platform add android
cordova platform add browser
sudo npm install -g bower
sudo bower install ngCordova --allow-root
sudo npm install -g cordova-browser
sudo npm install -g ios-deploy --unsafe-perm=true
cordova plugin add https://github.com/EddyVerbruggen/Insomnia-PhoneGap-Plugin.git
cordova plugin add cordova-plugin-vibration

********************************************************************************
```

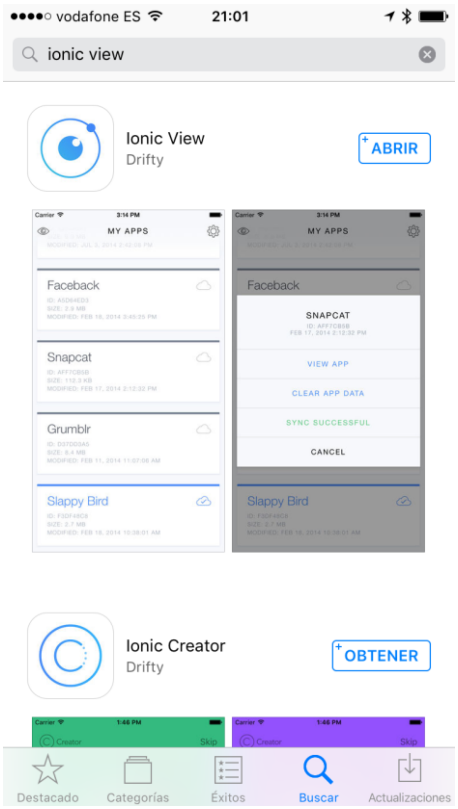
5.4 Manual de instalación (iOS y Android)

Debido a que la aplicación nativa no está subida a los principales markets de Android e iOS (*Play Store* y *App Store* respectivamente), la forma más rápida de poder probar la aplicación en un dispositivo Android o iOS es utilizar la herramienta ***Ionic View***, según se muestra a continuación.

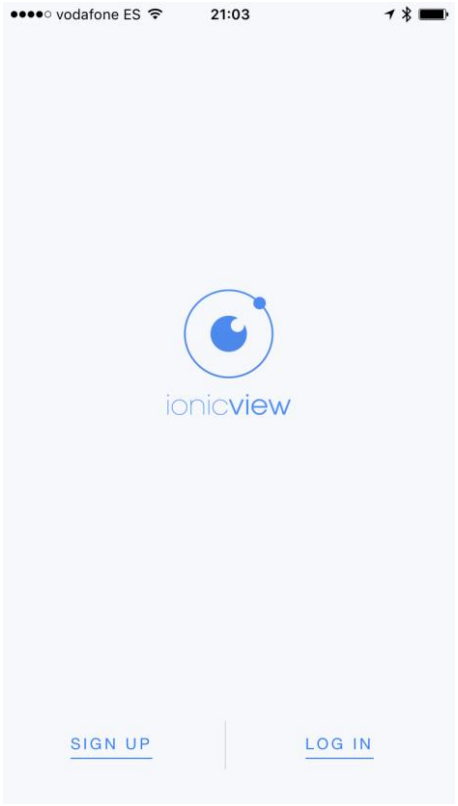
Manual de instalación para iOS

A continuación se muestran los pasos a seguir para instalar la aplicación *ionic view* en un dispositivo iOS y poder ejecutar la App "TestIt Oposiciones" dentro de ella:

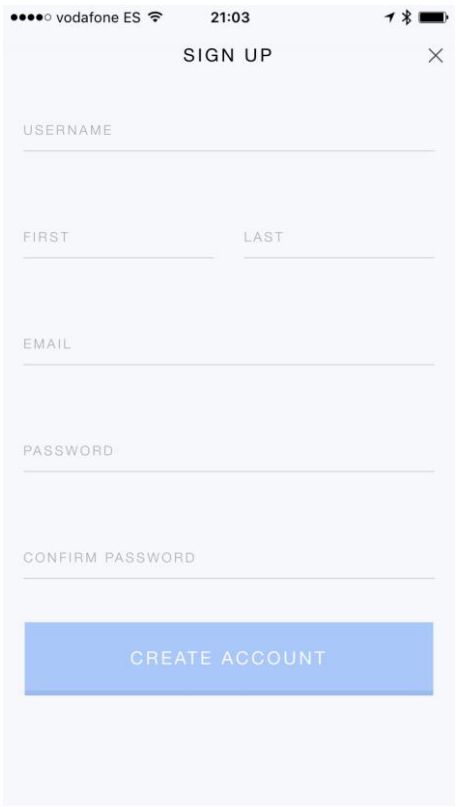
1. Abrir en su dispositivo iOS el App Store, buscar la aplicación "ionic view" e instalarla:



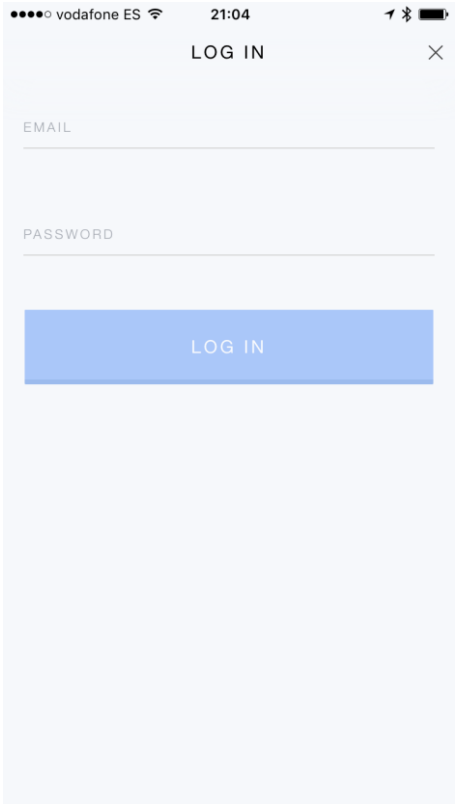
2. Abrir la aplicación “ionic view” recién instalada



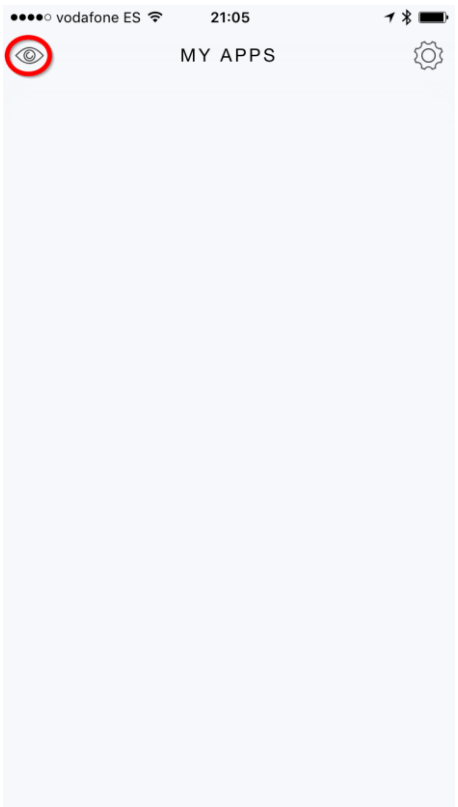
3. Si no se dispone de una cuenta de ionic, crear una utilizando la opción “Sign Up”:



4. Acceder con la cuenta creada, utilizando la opción “Log In”:



5. En la zona de “My Apps” no aparece ninguna aplicación, pulsamos sobre el icono con forma de ojo de la zona superior izquierda:



6. En la pantalla “Preview Shared App” introducimos **el código de aplicación “XXXXXXXX”** (no se muestra el código real, pues se pretende monetizar la app en el futuro), y pulsamos sobre “LOAD APP”.

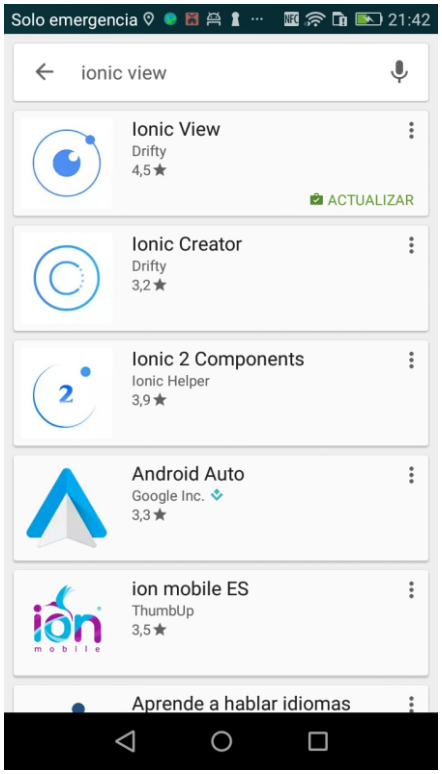
7. Tras cargarse la aplicación, se muestra la pantalla de login. Introducimos los datos de la **cuenta de prueba a@a.com (contraseña: *aaaaaa*)**:

8. Tras introducir la cuenta de prueba, accedemos al menú principal de la aplicación:

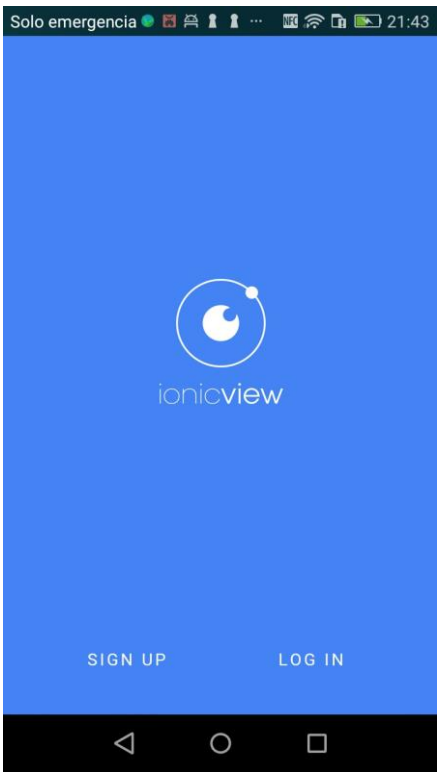
Manual de instalación para Android

A continuación se muestran los pasos a seguir para instalar la aplicación *ionic view* en un dispositivo Android y poder ejecutar la App "TestIt Oposiciones" dentro de ella:

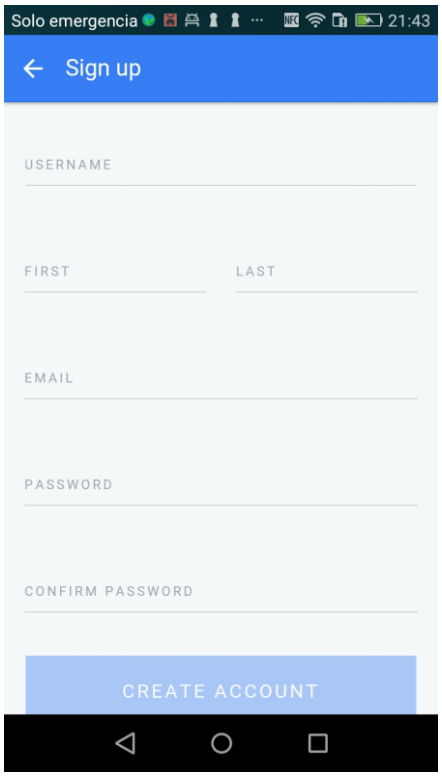
1. Abrir en su dispositivo Android el Play Store, buscar la aplicación "ionic view" e instalarla:



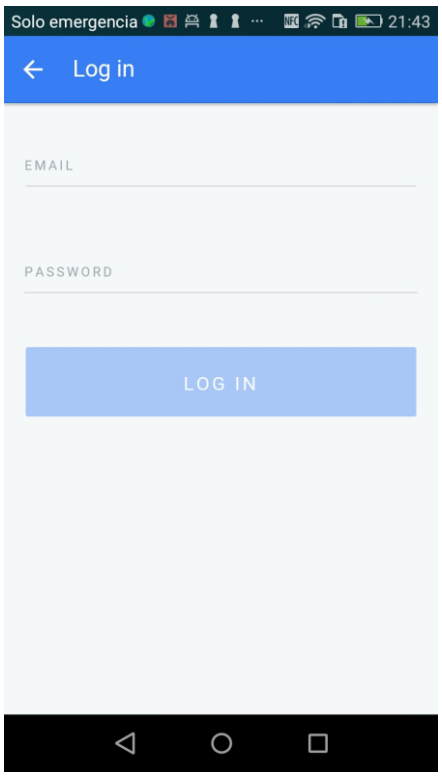
2. Abrir la aplicación “ionic view” recién instalada



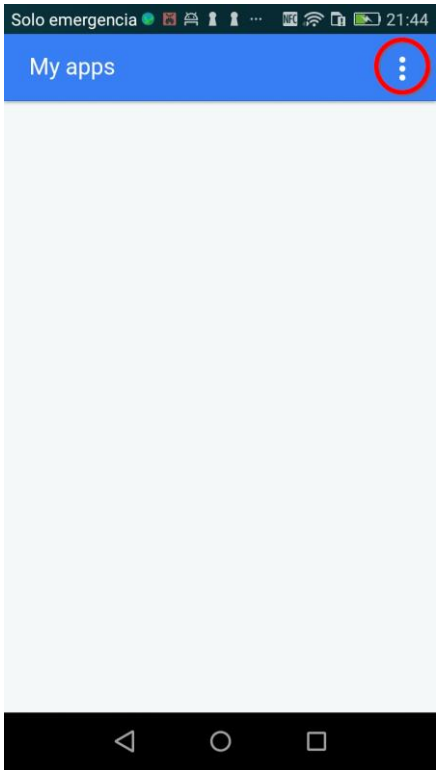
3. Si no se dispone de una cuenta de ionic, crear una utilizando la opción “Sign Up”:

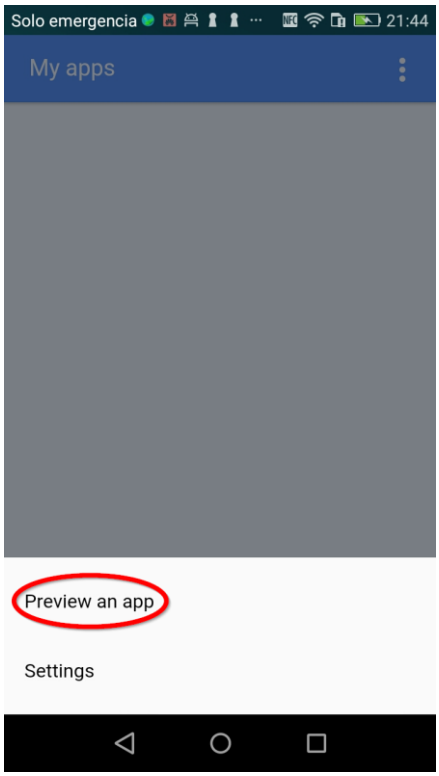


4. Acceder con la cuenta creada, utilizando la opción “Log In”:



5. En la zona de “My Apps” no aparece ninguna aplicación, pulsamos sobre los tres puntos verticales de la zona superior derecha, y después sobre “Preview an app”:

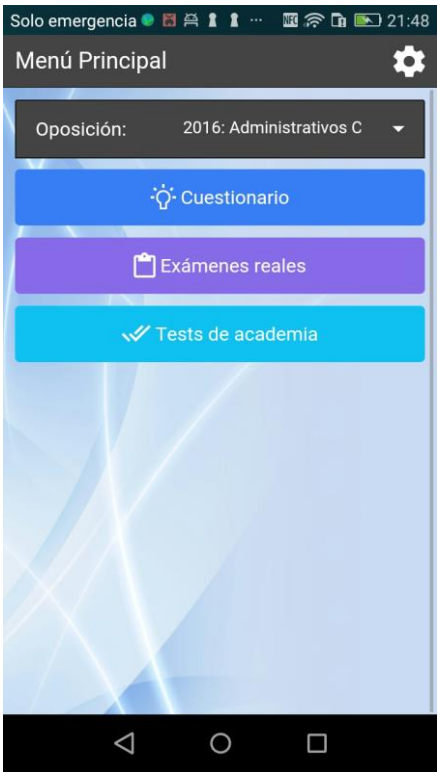




6. En la pantalla “Preview Shared App” introducimos **el código de aplicación “XXXXXXXX”** (no se muestra el código real, pues se pretende monetizar la app en el futuro), y pulsamos sobre “LOAD APP”.

7. Tras cargarse la aplicación, se muestra la pantalla de login. Introducimos los datos de la **cuenta de prueba a@a.com (contraseña: *aaaaaa*)**:

8. Tras introducir la cuenta de prueba, accedemos al menú principal de la aplicación:



5.5 Manual de usuario

El manual de usuario de la aplicación se encuentra en un documento independiente entregado junto con esta memoria.

5.6 Decisiones tomadas y problemas durante la implementación

En este apartado se incluyen algunos problemas que se han ido encontrando y las distintas decisiones que se han ido tomando durante el desarrollo de la aplicación:

- **Ralentización en la respuesta a eventos del usuario**: Todas las cargas de datos se hacen **asíncronas** para evitar paradas en el interfaz. Inicialmente no se hacía así, y si por ejemplo el usuario accedía a la pantalla de cuestionarios e intentaba ir hacia atrás cuando todavía no se habían cargado todos los datos, el sistema no respondía.
  Para cuidar la usabilidad al máximo, también se ha mostrado un símbolo de "Cargando…" siempre que se hacen llamadas a Firebase:

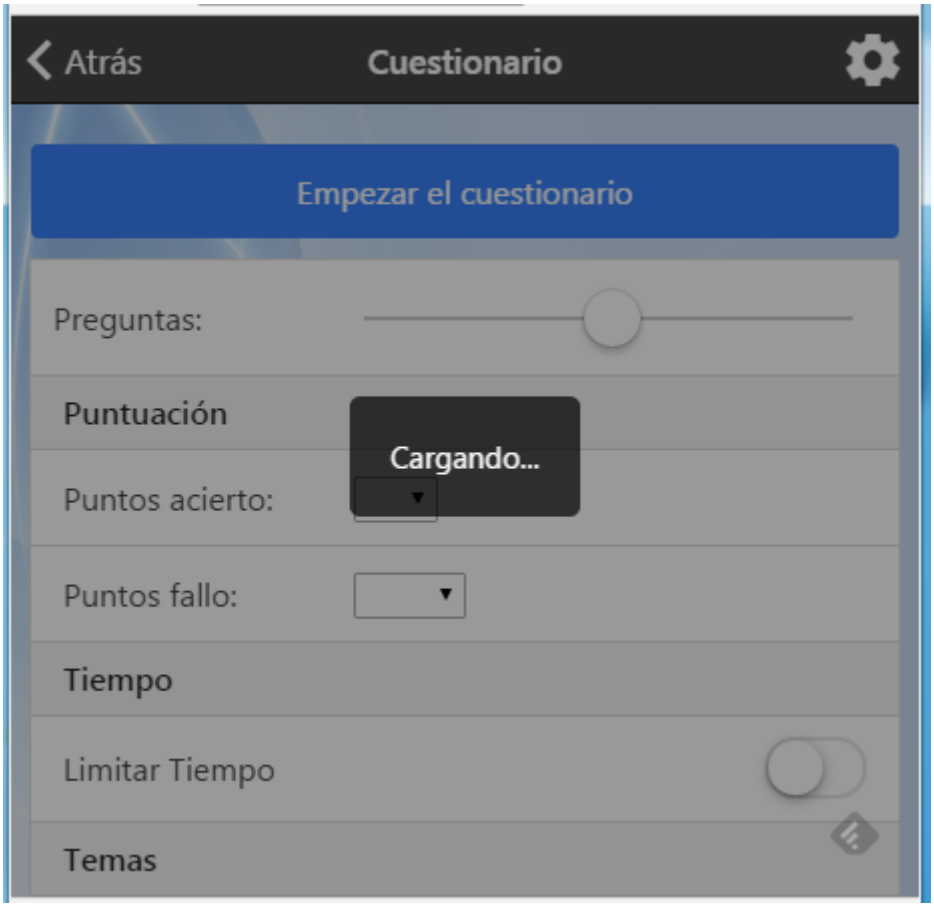


Figura 24: Símbolo "Cargando…" mientras se obtienen datos de Firebase

- **No volver a pedir usuario y contraseña**: En las aplicaciones para dispositivos móviles no es usual el que cada vez que el usuario abra la aplicación se le soliciten sus credenciales (usuario y contraseña). Para poder recordar las credenciales del usuario y así sólo pedírselas la primera vez que abre la aplicación, se ha utilizado la funcionalidad de HTML5 **LocalStorage**, que permite almacenar información en el propio dispositivo asociado al WebView. Concretamente se ha utilizado el **LocalStorageModule de angularjs** (https://github.com/grevory/angular-local-storage).

```
angular.module('starter.controllers', ['firebase', 'LocalStorageModule'])

.controller('AppCtrl', function ($scope, $rootScope, $state, localStorageService,

  var self = $scope;
  self.debugMode = false;

  // Referencia a firebase para almacenamiento de datos
  var storage = firebase.storage();
  var storageRef = storage.ref();

  // Carga el modal para registro de nuevos usuarios
  $ionicModal.fromTemplateUrl('templates/signup.html', {
    scope: self,
    animation: 'slide-in-up'
  }).then(function (modal) {
    self.modal = modal;
  });

  // Inicializa
  self.initLogin = function () {
    // Mira si el usuario ya está autenticado
    self.loginData = {};
    var currentUser = localStorageService.get('fbu');
    if (currentUser == null) {
      console.log('User not authenticated');

      // Auto-login for debugging
      if(self.debugMode) {
        self.loginData.username = 'a@a.com';
        self.loginData.password = 'aaaaaa';
        self.doLogin();
      }
    }
    else {
      console.log('User already authenticated');
      self.goToMainPage();
    }
  };
```

Figura 25: Uso de LocalStorage para recordar al usuario autenticado

- **Recordar opciones seleccionadas previamente**: Al acceder en distintas ocasiones a una pantalla, por ejemplo la realización de un cuestionario, se mostraban siempre las opciones por defecto. Esto generaba una mala experiencia de usuario, porque tenía que volver a seleccionar los parámetros siempre desde cero.
  Para recordar los parámetros de realización de cuestionarios entre distintas sesiones se ha utilizado también **LocalStorage**, donde se almacenan las opciones seleccionadas cada vez que se comienza un nuevo test.

```
// Busca parámetros en local storage
var tmp = localStorageService.get('cparams');
if (tmp == null) {
  self.cuestionario.parametros = {};
  self.cuestionario.parametros.numeroPreguntas = 50;
  self.cuestionario.parametros.puntosacierto = 1;
  self.cuestionario.parametros.puntosfallo = 0.5;
  self.cuestionario.parametros.limitarTiempo = false;
  self.cuestionario.parametros.tiempoLimite = 60;
  self.cuestionario.parametros.seleccionarTemas = false;
  self.cuestionario.parametros.temasSeleccionados = new Array(self.temas.length);
  for (var i = 0; i < self.cuestionario.parametros.temasSeleccionados.length; ++i) {
    self.cuestionario.parametros.temasSeleccionados[i] = false;
  }
}
else {
  self.cuestionario.parametros = tmp;

  // Para evitar posibles errores
  if (!self.cuestionario.parametros.temasSeleccionados || self.cuestionario.parametros.
    self.cuestionario.parametros.temasSeleccionados = new Array(self.temas.length);
    for (var i = 0; i < self.cuestionario.parametros.temasSeleccionados.length; ++i) {
      self.cuestionario.parametros.temasSeleccionados[i] = false;
    }
  }

  // Actualiza algunos parámetros para evitar errores al des-serializar (ver https://fc
  if (self.cuestionario.parametros.limitarTiempo == "true") {
    self.cuestionario.parametros.limitarTiempo = true;
  }
  else if (self.cuestionario.parametros.limitarTiempo == "false") {
    self.cuestionario.parametros.limitarTiempo = false;
  }

  if (self.cuestionario.parametros.seleccionarTemas == "true") {
    self.cuestionario.parametros.limitarTiempo = true;
  }
  else if (self.cuestionario.parametros.seleccionarTemas == "false") {
    self.cuestionario.parametros.limitarTiempo = false;
  }
}
```

Figura 26: Uso de LocalStorage para recuperar datos del cuestionario

- **Errores de cross-domain para probar en local contra Firebase**: Inicialmente hubo muchos problemas para realizar pruebas en un PC y browser local de la integración con Firebase. Para evitar errores de cross-domain con Firebase para poder probar en local se ha tenido que instalar la utilidad *gsutil* de *Phyton 2.7* y crear el fichero *cors.json* con el siguiente contenido:

```
[
{
"origin": ["http://localhost:8100"],
"method": ["GET"],
"maxAgeSeconds": 3600
}
]
```

Desde la línea de commandos, posicionandose en el directorio que contienen el fichero creado, se ha ejecutado: "*python gsutil cors set cors.json gs://cuestionario-9996f.appspot.com*"

- **El dispositivo se bloquea mientras estás haciendo un cuestionario**: En las pruebas con los usuarios finales, estos se quejaban de que, en mitad de la realización de un cuestionario, si estaban un tiempo pensando sobre la respuesta a una pregunta (y por tanto sin tocar la pantalla) el dispositivo se les bloqueaba. Para evitar dicho bloqueo se ha utilizado el plugin ***insomnia*** de cordova que hace uso de la funcionalidad nativa de cada dispositivo.

```
// Evita que se bloquee la pantalla cuando hay un cuestionario abierto
self.$on('$ionicView.enter', function () {
  // Si se paró el timer, lo re-crea por donde fuese
  if (timerPaused) {
    timerPaused = false;
    self.cambiarTiempo();
  }
  if (window.cordova) {
    $cordovaInsomnia.keepAwake();
  }
});

// Para el contador de tiempo si se sale de la pantalla
// Evita que se bloquee la pantalla cuando hay un cuestionario abierto
self.$on('$ionicView.leave', function () {
  if (timeCounter != null) {
    //noinspection JSCheckFunctionSignatures
    $interval.cancel(timeCounter);
    timeCounter = null;
    timerPaused = true;
  }

  if (window.cordova) {
    $cordovaInsomnia.allowSleepAgain();
  }
});
```

Figura 27: Uso del plugin insomnia para evitar el bloqueo de la pantalla

- **Usabilidad en la transición cuando se acaba el tiempo**: Durante el periodo de pruebas de la aplicación con usuarios reales, algunos de ellos se quejaron de que cuando finalizaba el tiempo disponible para hacer un cuestionario o test desaparecía la pantalla de forma muy brusca, mostrando directamente los resultados de la prueba. Para hacer más usable esta transición, se decidió que al finalizar el tiempo también vibrase el dispositivo para avisar. Para ello se ha utilizado el plugin ***vibration*** de cordova.

- **Los ficheros JSON con las preguntas de los tests son muy grandes y tardan mucho en cargar:** Durante el periodo de pruebas de la aplicación con usuarios reales se detectó que, a la hora de realizar un cuestionario sin filtrar por ningún tema concreto, el cuestionario tardaba mucho en empezar (y la aplicación consumía muchos datos móviles). Esto era debido a que se cargaba secuencialmente un fichero en formato JSON bastante grande para cada tema del temario de la oposición. Se tomaron dos medidas:

  1. **Excesivo consumo de datos**: Para mitigar el efecto del excesivo consumo de datos, se utilizó una tarea de *gulp* que permite comprimir los ficheros en formato *JSON* antes de subirlos a la plataforma *Firebase*. Se ha incluido la siguiente configuración en el fichero *gulpfile.js*:

```
gulp.task('jsonminify', function () {
  return gulp.src(['./data/*.json'])
    .pipe(jsonminify())
    .pipe(gulp.dest('./dataminified'));
});
```

Figura 28: Tarea de gulp para comprimir los ficheros JSON

  Puede ejecutarse dicha tarea posicionándose con una línea de comandos sobre el directorio con las fuentes de la aplicación y ejecutando "*gulp jsonminify*".

  2. **Tarda mucho en cargar**: Para paliar el efecto de la carga en serie de todos los ficheros JSON con las preguntas de cada uno de los temas de la oposición, se ha realizado una **carga asíncrona en paralelo** de dichos temas. Esta carga se ha realizado utilizando la variable *$q* de *angularjs*, y un *array de promises* (puede obtenerse más información sobre $q de angularjs en https://docs.angularjs.org/api/ng/service/$q).

- **Bug en plataforma iOS al pulsar sobre un botón antes de terminar de seleccionar en combo:** Los usuarios de prueba que utilizaban sistema operativo iOS detectaron un problema al seleccionar una opción en las listas desplegables y pulsar sobre un botón. Si se seleccionaba el valor en la lista y se pulsaba sobre el botón "Aceptar", el formulario no obtenía correctamente el valor que se había seleccionado en la lista desplegable. Para solucionarlo se ha utilizado la opción data-tap-disabled="true" del componente button de ionic framework (más información en http://ionicframework.com/docs/api/page/tap/):

```
<button data-tap-disabled="true" id="startQuiz" class="button button-block button-positive"
        ng-click="start()">Empezar el examen
</button>
```

Figura 29: Botón de ionic con data-tap-disabled activado

- **Problema con el logotipo al poner la pantalla en horizontal:** El logotipo de la aplicación se deformaba en ciertos dispositivos en función de su resolución, y al colocar la pantalla en horizontal en algunas tabletas. Para solucionar este problema, se utilizó el siguiente estilo en la css de la aplicación:

```
.login-logo{
  margin: auto;
  margin-top: 15px;
  display: block;
  height: auto;
  max-width: 100%;
}
```

Figura 30: Estilo CSS para imagen del logotipo

- **Centrado vertical del menú principal en la pantalla:** Inicialmente se posicionó el menú principal de la aplicación en la zona superior de la pantalla, junto a la barra de navegación. Sin embargo, para mejorar la experiencia de usuario finalmente se decidió centrar dicho menú en la pantalla. Para ello se utilizó el siguiente estilo:

```
.vcenter {
  display: flex;
  flex-direction: column;
  justify-content: center;
}
```

Figura 31: Estilo CSS para centrar el menú principal

- **Problema de scroll al cambiar de pantalla**: Uno de los usuarios finales detectó que si en la pantalla de iniciar un cuestionario hacía scroll hasta la parte inferior de la pantalla (con el listado de temas desplegado) y pulsabas sobre “Empezar” en el botón de abajo, la pantalla se quedaba abajo y por tanto no se veía la primera pregunta del cuestionario. Para arreglarlo se tuvo que forzar un refresco del scroll manual, invocando al siguiente método:

```
// Refresca el scroll para evitar lags de tiempo
self.refrescarScroll = function () {
  $ionicScrollDelegate.resize();
  $ionicScrollDelegate.scrollTop();
};
```

Figura 32: Refresco del scroll manual para evitar errores

6. Conclusiones

La realización de este trabajo de final de máster me ha permitido **poner en práctica todo lo aprendido** durante el máster. Se realizó una **planificación inicial** con una serie de hitos que en su mayoría se han cumplido con puntualidad. La propia planificación ha servido para imponerme algo de presión y no dejar todo el trabajo para el final, ha sido un **trabajo constante y continuado**. La planificación se ha tenido que modificar ligeramente durante la fase de implementación y pruebas, pues las pruebas con usuarios reales han derivado en la **introducción de algunas modificaciones funcionales** (como la posibilidad de repetir el mismo examen realizado, o sólo las preguntas no acertadas). Una de las **lecciones aprendidas** es que estas modificaciones podrían haberse evitado de haber realizado **una maqueta previa** al desarrollo de la aplicación para validarla con los usuarios.

La **metodología** seguida ha sido una combinación de las aprendidas en diversas asignaturas del máster. Se han utilizado las técnicas aprendidas en la asignatura *Diseño Avanzado de Aplicaciones para Dispositivos Móviles* para realizar la **ficha de usuarios potenciales y el árbol de navegación**. Por otro lado, se ha utilizado una **metodología ágil** para la realización de diversos ciclos o *sprints* iterativos de desarrollo y validación funcional de dos semanas de duración.

En general me encuentro **muy satisfecho con el resultado obtenido**. Aunque la lista de funcionalidades de la aplicación no es muy extensa, se ha cuidado hasta el último detalle en cuanto a **usabilidad**. Las pruebas con los usuarios finales (opositores reales) demuestran que **se trata de una aplicación muy útil** para ellos, lo que me ha hecho plantearme seriamente la posibilidad de **monetizarla**, según se describe en el apartado del estudio de mercado y retorno de la inversión. Este es uno de los motivos por los que también he invertido bastante tiempo en la introducción de preguntas reales para la oposición “Cuerpo Administrativo de las Cortes Generales” (**más de 2.000 preguntas introducidas** a la fecha de entrega de este TFM).

A primera vista no aparenta ser una aplicación técnicamente compleja. Sin embargo, tal como se ha descrito en el apartado 5.3, han surgido **multitud de problemas técnicos** que han tenido que ir solventándose, en ocasiones utilizando workarounds para corregir bugs del propio framework ionic. La mayor parte de los problemas han venido derivados de que en muchas ocasiones la aplicación funcionaba correctamente en su versión web, pero al desplegarla nativamente en un dispositivo **iOS o Android el funcionamiento no era el esperado**. Por este motivo las pruebas finales se han realizado exhaustivamente en dispositivos de ambos sistemas operativos, combinando smartphones y tabletas con orientaciones tanto vertical como horizontal.

Aunque anteriormente ya había utilizado otros servicios de backend en la nube (como el extinto *Parse.com*), **nunca antes de este TFM había utilizado la plataforma Firebase**, y me ha sorprendido su facilidad de uso y versatilidad. Una vez definidas las estructuras de datos en formato JSON y los

nombres de los ficheros, actualizar las preguntas de la aplicación es tan fácil como subir un fichero nuevo con las preguntas modificadas a Firebase.

Finalmente destacar que la aplicación sólo se encuentra desarrollada en una primera versión con la lista de funcionalidades reducida, quedan muchas cosas por hacer. A continuación se listan algunas de las **funcionalidades aún no implementadas** sugeridas por los usuarios reales durante las pruebas:

- Cuando estás haciendo un cuestionario o examen, poder **hacer "slide" con el dedo a derecha e izquierda** para pasar las preguntas.
- Poder cambiar **tamaño de letra**.
- Incluir preguntas que contengan **imágenes**.
- Poder **agregar preguntas** desde la propia aplicación, crear tus preguntas personalizadas.
- Recibir **notificaciones push** para avisar sobre nuevas preguntas o exámenes añadidos a la aplicación.
- Poder ver el **histórico de preguntas** fallidas, para poder repetir las preguntas en las que se falló y analizar estadísticas de preguntas y temas.
- Poder notificar al desarrollador de la existencia de un **error en la respuesta** a una pregunta desde la propia aplicación.

A estas funcionalidades habría que añadir los desarrollos necesarios para llevar a cabo la **monetización** de la aplicación. Para ello se está evaluando la utilización del plugin InApp purchase para ionic disponible en la URL https://alexdisler.com/2016/02/29/in-app-purchases-ionic-cordova/

7. Glosario

- *TFM*: Trabajo de Final de Máster
- *App*: Aplicación para dispositivos móviles.
- *App híbrida*: Aplicación que funciona en distintas plataformas móviles, habitualmente basada en tecnologías HTML5 y CSS.
- *GIT*: Repositorio de código que permite el control de las distintas versiones, gestiona la seguridad de acceso, etc.
- *GitLab*: Proveedor de servicios de almacenamiento de código con estándar GIT.
- *HTML*: Lenguaje de marcado para la elaboración de páginas web.
- *Javascript*: Lenguaje de programación interpretado, que se utiliza comúnmente en el lado cliente como parte de un navegador web.
- *AngularJS*: Framework de desarrollo de aplicaciones Javascript.
- *Gulp.js*: Conjunto de herramientas para el desarrollo de Javascript que permite, entre otras cosas, la gestión del ciclo de vida del proyecto.
- *Ionic Framework*: Framework para el desarrollo de aplicaciones híbridas multiplataforma, basado en tecnologías como angularJS o apache cordova.
- *Ionic Market*: Tienda de componentes y plantillas de partida (templates) para ionic framework.
- *Ionic Creator*: Herramienta online de diseño visual de pantallas para ionic framework.
- *Ionic View App*: App disponible en los markets de los distintos sistemas operativos, que permite visualizar proyectos realizados con ionic framework sin necesidad de haberlos publicado.
- *JSON*: Acrónimo de JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos.
- *Firebase*: Plataforma que proporciona servicios de infraestructura de servidor para el desarrollo de aplicaciones para dispositivos móviles.
- *Parse.com*: Plataforma que proporciona servicios de backend adquirida por Facebook en 2013 y discontinuada en Enero de 2016.
- *Android*: Sistema Operativo para dispositivos móviles desarrollado por Google, basado en el núcleo de Linux.
- *iOS*: Sistema Operativo para dispositivos móviles Apple.
- *ROI*: Retorno de la Inversión, es una razón financiera que compara el beneficio en relación a la inversión realizada.

- *WebStorm*: Entorno de desarrollo que se integra perfectamente con ionic framework, y permite el desarrollo ágil de código en HTML y javascript (tiene integración específica con angularjs)
- *Local Storage*: Propiedad de HTML5 que permite almacenar datos en el navegador web, y que permanecerá almacenada sin importar que el navegador se cierre.
- *Apache Cordova*: Marco de desarrollo móvil de código abierto que permite utilizar las tecnologías estándar web como HTML5, CSS3 y JavaScript para desarrollo multiplataforma.
- *Insomnia*: Plugin de cordova que permite evitar el bloqueo automático de la pantalla.
- *Vibration*: Plugin de cordova que permite hacer vibrar el dispositivo móvil.
- *Notificaciones Push*: Notificaciones enviadas al dispositivo móvil desde la parte servidora.

8. Bibliografía y enlaces relevantes

Entorno de desarrollo WebStorm:
https://www.jetbrains.com/webstorm/
Repositorio git donde se almacena el código del proyecto:
https://gitlab.com/manuelpereira/cuestionario.git
Ionic Framework:
http://ionicframework.com/
Editor visual online Ionic Creator:
https://creator.ionic.io
Herramienta de pruebas Ionic View:
http://view.ionic.io/
Estructura de una aplicación con Ionic Framework:
http://ionicframework.com/docs/concepts/structure.html
http://mcgivery.com/structure-of-an-ionic-app/
Ionic framework tutorial:
https://thinkster.io/ionic-framework-tutorial
Guía de estilo de AngularJS:
https://angular.io/docs/ts/latest/guide/style-guide.html
Uso de LocalStorage en AngularJS:
https://github.com/grevory/angular-local-storage
Plugin ng-cordova para angujarjs (acceso a funcionalidades nativas):
http://ngcordova.com/
Uso de promises y $q en angularjs:
https://docs.angularjs.org/api/ng/service/$q
Documentación del componente *button* de ionic framework:
http://ionicframework.com/docs/api/page/tap/
Apache cordova:
https://cordova.apache.org/
Firebase:
https://firebase.google.com/
Integración de Firebase con Javascript y aplicación web:
https://firebase.google.com/docs/web/setup?hl=es
https://firebase.google.com/docs/database/web/start?hl=es
Facebook cierrar Parse:
http://manuelpereiragonzalez.blogspot.com.es/2016/02/facebook-cierra-parse-el-backend-como.html
Alternativas a Parse:
http://highscalability.com/blog/2016/2/2/the-big-list-of-alternatives-to-parse.html
Pautas de usabilidad y accesibilidad para dispositivos móviles:
http://olgacarreras.blogspot.com.es/2015/11/pautas-accesibilidad-usabilidad-movil.html
http://olgacarreras.blogspot.com.es/2014/01/responsive-design-y-accesibilidad.html
Nuevo modelo de suscripciones y prueba gratuita de iOS:
https://hipertextual.com/2016/09/app-store-iphone-7
Plugin InApp purchase para ionic:
https://alexdisler.com/2016/02/29/in-app-purchases-ionic-cordova/

Smartphone Samsung Galaxy S5:
http://www.samsung.com/es/consumer/mobile-devices/smartphones/galaxy-s/SM-G901FZWAAMO
Smartphone Huawei P8 Lite:
http://www.huaweispain.com/microsites/huawei-p8lite/
Smartphone Apple iPhone 6s:
http://www.apple.com/es/iphone/
Tableta Apple iPad 2:
http://www.apple.com/es/ipad/