

Índice

1.	Justificación.....	2
2.	Agradecimientos.....	3
3.	Licencia del documento	4
4.	Objetivos del proyecto.....	5
5.	Fases del proyecto.....	6
5.1	Planteamiento	6
5.2	Búsqueda de soluciones.....	7
5.3	Planificación y organización del trabajo.....	7
5.4	Diseño UI/UX.....	7
5.5	Desarrollo.....	8
5.6	Testing de la aplicación.....	8
5.7	Implantación/puesta en marcha.....	8
6.	Desarrollo del proyecto.....	9
6.1	Planteamiento	9
6.2	Búsqueda de soluciones.....	16
6.2.1	Tipos de aplicaciones móviles.....	16
6.2.2	Frameworks para el desarrollo de aplicaciones híbridas.....	22
6.2.3	Plan de empresa: estudio de la viabilidad del proyecto.....	24
6.3	Planificación y organización del trabajo.....	25
6.4	Diseño UI/UX.....	39
6.5	Desarrollo.....	54
6.5.1	Frontend.....	54
6.5.2	Backend.....	55
6.6	Testing de la aplicación.....	116
6.7	Implantación/puesta en marcha.....	117
6.7.1	Publicación de la app en Google Play	117
6.7.2	Promoción y puesta en marcha.....	121
7.	Conclusiones.....	122
8.	Mejoras o alternativas propuestas.....	123
9.	Bibliografía.....	124

Anexo I Frameworks para el desarrollo de aplicaciones híbridas

Anexo II Plan de empresa NFU

Anexo III Índice de Figuras

Anexo IV Índice de tablas

1. Justificación

El presente Trabajo Fin de Curso (TFC) se desarrolla principalmente por dos particularidades.

La primera de ellas, debido a los intereses y conocimientos de la autora en diferentes temáticas, como son: el deporte y las nuevas tecnologías, en concreto la las aplicaciones móviles.

La segunda de ellas está relacionada con el trabajado que se llevó a cabo en el primer trimestre, en el cual había que desarrollar una página web. En mi caso, se realizó una página web para conectar deportistas llamada "**NFU Nos Falta Uno**", los cuales podían realizar una búsqueda de deportistas para poder formar equipo y poder llevar a cabo una partida. Se observa al final del proyecto, que para que los usuarios hagan uso de **NFU** es necesario entrar continuamente en la página web, siendo un proceso tedioso, en el caso de acceder desde un móvil o tablet.

Como conclusión, uniendo las dos particularidades anteriores, se desarrolla en el presente TFC "*Estudio de las diferentes plataformas multidispositivo y selección de las más adecuada para el desarrollo de la aplicación móvil NFU, Nos Falta Uno*".

2. Agradecimientos

A mis gatos, Simón, Joplin y Tino por quedarse a mi lado en mis horas de dedicación al ciclo.

A mis perras Matilde y Maya, por hacerme levantar de la silla, cuando ya era suficiente el estudio de cada día.

A mis profesores por motivarme y enseñarme en tan poco tiempo tanto.

Al “Nostre”, Vicent Esparza, por compartir tantos momentos, hacerme reir en cada uno de ellos y por ser un gran amigo.

A mis padres, por enseñarme a no tener límites.

Y en especial a mi compañero de vida, Vicent, agradecerle el estar ahí, motivarme y ayudarme durante estos dos años. Y no olvidar su gran paciencia....

“Nunca digas nunca, porque los límites, como el miedo, son a menudo una ilusión.”

Michael Jordan

3. Licencia del documento

Esta obra está bajo una [licencia de Creative Commons Reconocimiento-CompartirIgual 4.0 Internacional.](#)



Este obra está bajo una [licencia de Creative Commons Reconocimiento-CompartirIgual 4.0 Internacional.](#)

4. Objetivos del proyecto

El objetivo principal es la realización de una aplicación híbrida, utilizando para dicha labor el framework Ionic¹, siendo necesario conocer y comprender las características y el funcionamiento de éste, indagando sus posibilidades y ventajas frente a otras alternativas. Proporcionando a los usuarios una aplicación móvil basada en la búsqueda de deportistas para poder formar equipo y poder llevar a cabo una partida.

Otro de los objetivos ligado al desarrollo de una aplicación de este tipo es el de estudiar y utilizar diferentes tecnologías de desarrollo web, como son HTML, CSS y AngularJS.

Por último, se estudiarán y se pondrán en práctica técnicas de diseño que permitan implementar una aplicación cómoda e intuitiva, cuidando la experiencia de usuario y haciéndola independiente del dispositivo en el que se ejecute, sin que éste suponga límites para su utilización.

¹ <http://ionic.io/>

5. Fases del proyecto

Para desarrollar un proyecto informático es necesario definir el ciclo de vida dicho proyecto, teniendo en cuenta sus aplicaciones, parámetros y desarrollo, obteniendo de esta manera un perfecto funcionamiento y planificación durante el desarrollo. Al implementar el ciclo de vida en el proyecto, es necesario tener en cuenta las etapas o fases de desarrollo. Cada una de las fases o etapas que a continuación se exponen, marcan las pautas para el correcto desarrollo, pudiendo formar cada una parte de la siguiente o ejecutarse varias fases a la vez.

Como se puede observar en la siguiente imagen, el proyecto se divide en siete fases, que a continuación se desarrollan.

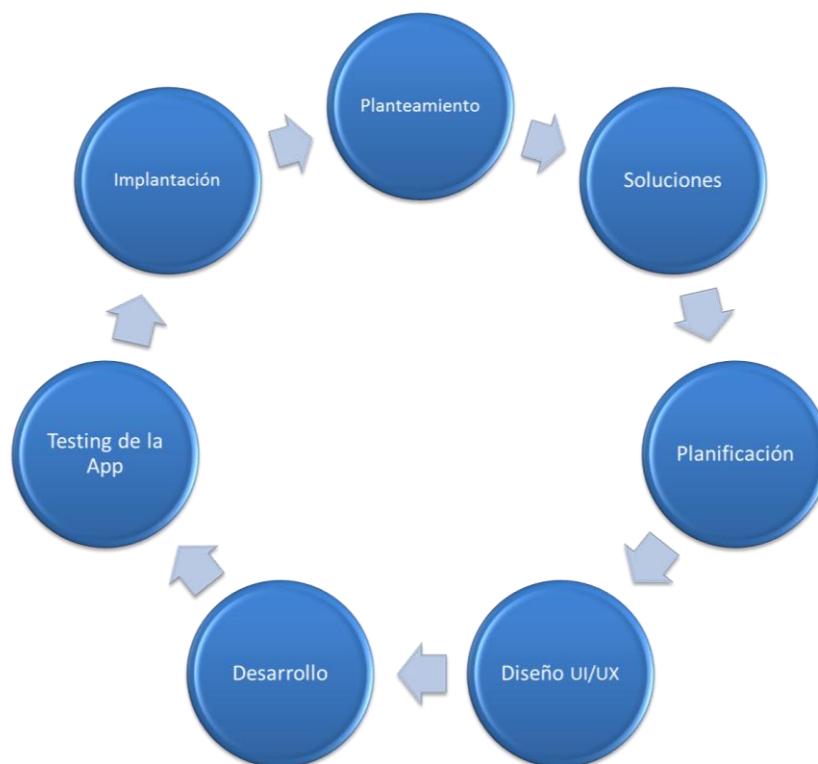


Fig. 1 Fases del proyecto. Fuente: elaboración propia.

5.1 Planteamiento

La primera fase de un proyecto, consiste en analizar la propuesta a desarrollar, es decir, qué soluciones se van a llevar a cabo para solucionar el problema que se cuestiona. Es también de estudio en esta fase, el análisis de las aplicaciones similares que se disponen en el mercado, que cubren parcial o totalmente las necesidades cuestionadas.

En definitiva:

- Análisis de la propuesta.
- Búsqueda de información de aplicaciones que cubren las necesidades de la propuesta.

5.2 Búsqueda de soluciones

La segunda fase, consiste en estudiar los diferentes frameworks para programar la aplicación, seleccionando aquel que mejor se adapte a las necesidades del proyecto. También es de estudio en esta fase, evaluar la viabilidad del proyecto.

Como conclusión:

- Estudio de los diferentes frameworks para programar la aplicación.
- Elección del framework.
- Evaluación de su viabilidad. Idoneidad técnica y funcional.

5.3 Planificación y organización del trabajo

Es la tercera fase del desarrollo del proyecto. Consiste en tener un programa de trabajo con un desglose de todas las actividades que se van a realizar (desde el diseño hasta las pruebas finales), estimando el número de horas que se le va a dedicar cada una de ellas y estableciendo los medios humanos que se van a utilizar para alcanzar los objetivos que se hayan propuesto. En este proceso, que ha de ser continuo, se han de reflejar:

- Elección del software y hardware.
- Planificación del trabajo.

5.4 Diseño UI/UX

Previo al desarrollo de la aplicación es necesario tener totalmente definido el diseño estructural de la app y su comportamiento. Para ello se, desarrollarán los mockups de las diferentes interfaces de la aplicación y posteriormente se llevará a cabo el diseño.

El diseño consiste tanto en la confección del aspecto y usabilidad como en la correcta aplicación de las guidelines² de diseño de aplicaciones de Android, además de la correcta adaptación a todas las densidades de pantallas y su tratamiento para que sean aptas para la programación.

El diseño de aplicaciones móviles es una de las fases más importantes pues nada tiene que ver con el diseño para programas de escritorio Windows o incluso diseños web y es lo que lo hace especialmente interesante pues se ha de hacer específico para cada app ya que cada una de estas tiene un propósito diferente.

En definitiva:

- Elaboración de los mockups de la interfaces
- Diseño de la aplicación.

² <https://www.google.com/design/spec/style/icons.html#>

5.5 Desarrollo

La quinta fase del proyecto, consiste en la programación de la aplicación, en función de la tecnología que se haya decidido emplear para cada plataforma de programación. También se debe tener en cuenta la parte del backend. Es decir, en esta fase, se llevará a cabo:

- Programación parte Servidor/BD, backend Reconstrucción del servidor.
- Programación frontend.

5.6 Testing de la aplicación

Una vez desarrollada la app es necesario hacer un testing profundo de todas las partes del mismo. El testeo se puede dividir en:

- Testeo funcional: para asegurar que la aplicación trabaja como debería y sigue todos los flujos debidos.
- Testeo de rendimiento: para comprobar que el comportamiento de la aplicación bajo ciertas condiciones (múltiples peticiones de acceso simultáneas, poca cobertura, poca batería...) es el correcto.

5.7 Implementación/puesta en marcha

A la finalización del desarrollo la app será apta para darse a conocer y comercializarse y el último paso será subirlo a los markets de aplicaciones correspondientes. Para este último paso habrá que firmar digitalmente la app con la cuenta de desarrollador, crear certificados para notificaciones push en Apple (si la app lleva esta funcionalidad implementada), compilar el paquete y subirlo a Google Play, App Store, Windows Marketplace... así como preparar el resto de requisitos necesarios tales como las imágenes, logos, descripciones etc. requeridos por los markets de apps.

Uno de los pasos más importantes dentro de la puesta en marcha y que conviene hacerlo antes de finalizar el desarrollo y una vez está publicada, es la promoción de la app. La promoción son todas las medidas que van a causar impacto para el lanzamiento de la app y que ayudarán a aumentar la visibilidad y por tanto las descargas, ya que aumentando la visibilidad se aumentan las descargas y las posiciones en los rankings, y esto a su vez atrae más descargas. Para ello es importante trabajar las redes sociales, foros y blogs para hacerse notar y mejorar en los rankings.

En definitiva:

- Publicación de la app en los markets de aplicaciones
- Promoción

6. Desarrollo del proyecto

En este apartado, se desarrollan minuciosamente las fases anteriormente mencionadas. Para hacer más fluida la lectura del proyecto y no ser un tanto tediosa, se hacen referencias a anexos, dónde el lector podrá disfrutar de unos contenidos más amplios.

6.1 Planteamiento

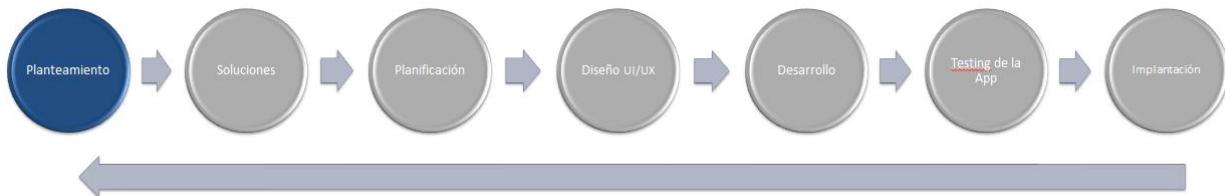


Fig. 2 Fase planteamiento. Fuente: elaboración propia.

Muchas veces, a lo largo de nuestra vida, nos hemos encontrado en situaciones en las que no hemos jugado un partido de fútbol, baloncesto o cualquier otro deporte, por dos motivos principalmente:

- 1) Porque a última hora uno o varios jugadores no puedan asistir al encuentro o,
- 2) Porque queremos jugar un partido y no disponemos de la suficiente gente para poder jugarlo.

Por esa razón, nace **NFU**, una comunidad de deportistas en la que los usuarios pueden interactuar para poder llevar a cabo un partido de su deporte favorito.

El planteamiento inicial de este proyecto, es el desarrollo de una aplicación para móvil, cuyo objetivo principal es la conexión de deportistas mediante la creación de partidas. Las funcionalidades principales que enriquecen a la aplicación, se explicarán después de exponer las apps similares que hay disponibles en el mercado.

6.1.1 Aplicaciones móviles de búsqueda de deportistas

De la información hallada, surge primeramente, la necesidad de establecer y organizar cuáles son las aplicaciones móviles utilizadas en el ámbito de la búsqueda de deportistas, tanto a nivel nacional como internacional. A continuación se exponen las aplicaciones encontradas en las diferentes tiendas de aplicaciones móviles.

6.1.1.1 Timpik

La aplicación **Timpik**³ está desarrollada por la empresa española Timpik Technologies S.L., siendo publicada en las tiendas de aplicaciones desde el año 2011.



Timpik es una plataforma que conecta deportistas, disponible en formato web y en aplicación móvil, en la cual existen unas ochenta disciplinas deportivas, en las cuales el usuario puede adherirse a un partido ya creado o crear su propio partido.

Fig. 3 Logo App
Timpik. Recuperado
de
<https://play.google.com/store/apps/details?id=com.timpik&hl=es>

El funcionamiento es sencillo, el usuario puede realizar las siguientes funciones dentro de la app:

- Encontrar partidos y eventos que se juegan cerca del usuario
- Organizar partidos
- Apuntarse a los partidos que mejor le vengan al usuario
- Gestionar los mensajes
- Visitar perfiles de otros jugadores
- Crear comunidades o grupos



Fig. 4 Perfil usuario Timpik.
Recuperado de
<https://play.google.com/store/apps/details?id=com.timpik&hl=es>

Para acceder a **Timpik**, el usuario se tiene que registrar en la app o en la web, mediante un registro o utilizando la cuenta de Facebook,

³ <http://www.timpik.com/>

accediendo posteriormente a la página del perfil para rellenar los campos obligatorios.

Una vez se ha dado de alta el usuario en Timpik, éste puede buscar partidas que los usuarios hayan creado, u organizar una propia partida.

Después de jugar un partido, los usuarios pueden elegir al mejor jugador del evento y valorar al organizador.

Cabe destacar que los formularios para crear un evento o los campos a llenar dentro del perfil, son bastante extensos. También que los niveles que tienen los usuarios, no son del todo ciertos, según opiniones del usuarios en las tiendas de aplicaciones.

6.1.2 Fubles

La aplicación **Fubles**⁴ está desarrollada por la empresa italiana Fubles S.L.R., siendo publicada para iPhone en el año 2009 y para Android en el año 2012.

Fubles es una plataforma que conecta deportistas, disponible en formato web y en aplicación móvil. A diferencia de Timpik, en Fubles sólo existe una disciplina deportiva, el fútbol, en la cual el usuario puede adherirse a partidos creados o crear su propio partido.

Fubles permite al usuario realizar las siguientes funciones dentro de la app:

- Encontrar partidos de fútbol que se juegan cerca del usuario
- Organizar partidos
- Crear partidos para que otros jugadores se unan.



Fig. 5 Logo App Fubles.
Recuperado de
<https://play.google.com/store/apps/details?id=it.android.fubles&hl=es>



Fig. 6 Pantalla inicio Fubles.
Recuperado de
<https://play.google.com/store/apps/details?id=it.android.fubles&hl=es>

⁴ <http://www.fubles.com/>

El funcionamiento es similar a Timpik, para poder acceder necesitamos realizar el registro mediante la cuenta de Facebook.

Una vez registrados, se accede a la pantalla principal de la aplicación, en la cual se puede acceder a crear un partido, búsqueda de partidos, perfil y próximos partidos.

Cabe destacar, que Fubles a diferencia de Timpik, es más específica, ya que sólo se centra en un solo deporte. Esto puede ser una ventaja o un inconveniente, ya que hay comentarios de usuarios que prefieren Timpik al tener más variedad deportiva y más partidos disponibles.

6.1.3 Decathlon Sport Meeting

La aplicación **Decathlon Sport Meeting**⁵ está desarrollada por la empresa Decathlon España, S.A.U., siendo publicada en las tiendas de aplicaciones desde el año 2013.

Decathlon Sport Meeting es la red social deportiva donde cualquier persona puede encontrar gente para hacer deporte, buscar deportistas, compartir experiencias, y disfrutar de más de 30 deportes de manera sencilla y gratuita.

Las principales funcionalidades de la app son las siguientes:

- Crea propuestas detalladas para salir a practicar deporte y descubrir nuevos usuarios o crea actualizaciones para subir tus fotos, logros y metas que podrán ver otros deportistas.
- Compartir propuestas deportivas y actualizaciones a través de la cuenta de WhatsApp, Facebook, Twitter o e-mail.
- Utiliza el buscador para conocer qué deportistas y buscar propuestas deportivas que están cerca de usuario.
- Crea los lugares de práctica de deportes en el mapa para añadirlos a tus propuestas.
- Utiliza los mensajes privados para chatear con otros deportistas.

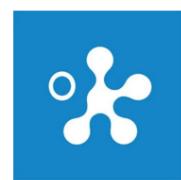


Fig. 7 Logo
Decathlon Sport
Meeting.
Recuperado de
<https://play.google.com/store/apps/details?id=com.decathlon.sportmeeting&hl=es>



Fig. 8 Perfil usuario Decathlon
Sport Meeting. Recuperado
de
<https://play.google.com/store/apps/details?id=com.decathlon.sportmeeting&hl=es>

⁵ <https://sportmeeting.decathlon.com/>

Cuadro resumen de las aplicaciones móviles para la búsqueda de deportistas:

Nombre	Logo	Empresa	Disponible en:	Fecha	Precio e instalaciones	Uso
Timpik		Timpik Technologies S.L.	iPhone y Android	2011	Gratis 50.000 – 100.000	80 disciplinas deportivas
Fubles		Fubles S.L.R.	iPhone y Android	2009	Gratis 50.000 – 100.000	Fútbol
Decathlon Sport Meeting		Decathlon España, S.A.U.	iPhone y Android	2013	Gratis 50.000 – 100.000	30 disciplinas deportivas

Tabla 1 Comparativa de las Apps de búsqueda de deportistas. Fuente: elaboración propia

Después de estudiar las diferentes aplicaciones móviles que existen en el mercado para la búsqueda de deportistas, es necesario indicar que mejoras se van a llevar a cabo en la aplicación de NFU.

La primera mejora, tiene que ver con la parte de seguridad. En las tres aplicaciones indicadas, un usuario se puede registrar y acceder a la aplicación con cualquier cuenta de correo electrónico, ya que ha dicho correo no llega la activación de la cuenta. NFU soluciona este aspecto, creando la activación mediante correo electrónico.

La segunda mejora a destacar de NFU, es la incorporación de notificaciones push. Estas notificaciones las pueden activar y desactivar los usuarios desde su perfil de usuario. Las notificaciones le llegan al usuario, cuando se crea una partida cerca su posición o cuando otro usuario le invita para jugar.

La tercera y última mejora, es la posibilidad de crear torneos de cualquiera de los deportes, por parte de un usuario organizador.

6.2 Búsqueda de soluciones

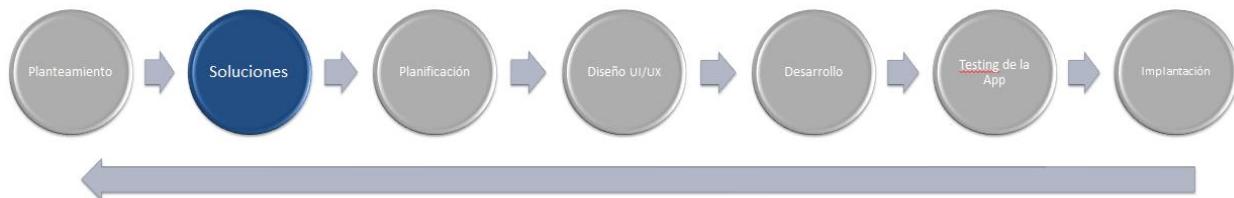


Fig. 9 Fase Soluciones. Fuente: elaboración propia

Antes de pasar a estudiar los diferentes frameworks disponibles para programar la aplicación, es necesario realizar una pequeña introducción para explicar los diferentes tipos de aplicaciones móviles existentes. En el último apartado de esta fase se expone el estudio de la viabilidad económica-financiera **“NFU Nos Falta Uno”**

6.2.1 Tipos de aplicaciones móviles

Existen tres tipos de aplicaciones móviles: nativas, web e híbridas. La elección de un tipo u otro vendrá determinado por la elección del desarrollador, en función de las necesidades y los requisitos del proyecto a desarrollar.

A continuación, se exponen, de forma resumida, los tres tipos de aplicaciones móviles:

a. Aplicaciones nativas

Una **aplicación nativa** es la que se desarrolla de forma específica para un determinado sistema operativo, llamado Software Development Kit o SDK. Cada una de las plataformas, **Adroid, iOS o Windows Phone**, tienen un sistema diferente, por lo que si se quiere tener una app disponible en todas las plataformas se deberán de crear varias apps con el lenguaje del sistema operativo seleccionado. Los lenguajes de programación, de los principales sistemas operativos, son los siguientes:

- Para iOS: Swift
- Para Android: Java
- Para Windows Phone: C#

Aplicaciones nativas	
Ventajas	Inconvenientes
Se ejecutan de forma muy rápida.	Coste de desarrollo elevado.
Acceso total a todo el hardware disponible (sensores de luminosidad, cámara de fotografía/vídeo, posicionamiento, brújula, etc.).	Requiere de aprobación por parte del proveedor de la tienda de Apps (Apple, Google, Microsoft,...).
Posibilidad de venta en tiendas de Apps.	Mantenimiento costoso y requiere de aprobaciones constantes para cada actualización.
Si tienen la información en local, no requieren de acceso a Internet.	Para cada tecnología hay que realizar y mantener aplicaciones distintas (iOS, Android, WindowsPhone, ...).

Tabla 2 Ventajas y desventajas aplicaciones nativas. Fuente: elaboración propia

b. Aplicaciones web

Una **aplicación web o webapp** es la desarrollada con lenguajes muy conocidos por los programadores, como es el **HTML, Javascript y CSS**. La principal ventaja con respecto a la nativa es la posibilidad de programar independiente del sistema operativo en el que se usará la aplicación. De esta forma se pueden ejecutar en diferentes dispositivos sin tener que crear varias aplicaciones.

Las aplicaciones web se ejecutan dentro del propio navegador web del dispositivo a través de una URL. Por ejemplo en Safari, si se trata de la plataforma iOS. El contenido se adapta a la pantalla adquiriendo un aspecto de navegación APP.

¿Puede considerarse esto una APP? En realidad la gran diferencia con una aplicación nativa (además de los inconvenientes que se muestran en la tabla) es que no necesita instalación por lo que no pueden estar visibles en app store y la promoción y comercialización debe realizarse de forma independiente. De todas formas se puede crear un acceso directo que sería como “instalar” la aplicación en el dispositivo.

Las apps web móviles son siempre una buena opción si nuestro objetivo es adaptar la web a formato móvil.

Aplicaciones web	
Ventajas	Inconvenientes
Se desarrollan como páginas web, por lo que es más fácil y sencillo su mantenimiento.	Se requiere conexión a Internet para funcionar.
Tiene un coste menor que el desarrollo de aplicaciones nativas.	La velocidad de ejecución depende directamente de la velocidad de conexión del usuario.
Sólo se requiere de una tecnología (HTML5 y CSS3) para todas las plataformas móviles.	El acceso directo al hardware de los dispositivos móviles no es viable.
No requiere estar disponible en ninguna tienda de Apps para acceder a la aplicación, ya que funciona con el navegador web.	Fluidez: Un interface de usuario desarrollado en HTML5 no es tan fluido como uno nativo.

Tabla 3 Ventajas y desventajas aplicaciones web. Fuente: elaboración propia

En la siguiente imagen se comparan las características entre una App Web y una App Nativa.



Fig. 10 Comparación aplicaciones web con nativas. Recuperado de <http://www.accensit.com/index.php/en/accensit-blog-en/150-mobile-platforms.html>

c. Aplicaciones híbridas

Una **aplicación híbrida** es una combinación de las dos anteriores, se podría decir que recoge lo mejor de cada una de ellas. Dentro de las aplicaciones híbridas, se distinguen dos vertientes: las aplicaciones híbridas basadas en web app y las aplicaciones híbridas interpretadas.

Las **apps híbridas basadas en web app** consisten en programar la aplicación como si de una aplicación web se tratara con HTML, CSS y javascript. El código web se embeberá dentro de una web view y será el propio motor del navegador el que ejecute el código de la app.

En el caso de las **apps híbridas interpretadas**, el lenguaje de programación empleado es javascript, que posteriormente será transformado y compilado antes de empaquetarlo en una app nativa. Este código javascript será ejecutado al abrir la aplicación en un engine javascript propio del Smartphone.

Ambas permiten su uso en diferentes plataformas, pero también dan la posibilidad de acceder a gran parte de las características del hardware del dispositivo. La principal **ventaja de las web apps** es que a pesar de estar desarrollada con HTML, Java o CSS, es posible agrupar los códigos y distribuirla en las app store, teniendo un look & feel distante de las aplicaciones nativas. La principal ventaja de las apps interpretadas es que se asemejan bastante a las apps nativas, pero la curva de aprendizaje, comparando con las web app, es más costosa.

Muchos de los frameworks utilizados para el desarrollo multiplataforma de aplicaciones híbridas basadas en web app, hacen uso de Cordova o PhoneGap. Los frameworks más populares son:

- Ionic
- Appery.io
- Onsen UI
- Framework 7
- jQuery Mobile

Los frameworks más utilizados para el desarrollo de aplicaciones híbridas interpretadas son:

- React Native

- Native Script
- Appcelerator
- Fuseteools

Aplicaciones web	
Ventajas	Inconvenientes
Se distribuye mediante los respectivos "stores".	La velocidad de ejecución depende directamente de la velocidad de conexión del usuario.
Minimizamos el código específico: La mayor parte del código puede utilizarse para el resto de plataformas. Solo se utiliza código nativo para aquellos aspectos que lo requieran.	Sigue requiriendo del proceso de aprobación por parte de proveedores de tiendas de Apps.
Menor coste de mantenimiento al ser la mayor parte del código común a todas las plataformas.	Look & Feel distante de las aplicaciones nativas
Una aplicación Híbrida puede acceder a los recursos del dispositivo móvil prácticamente como una nativa.	
Pueden funcionar online y offline	

Tabla 4 Ventajas y desventajas aplicaciones web. Fuente: elaboración propia

En la siguiente imagen se puede observar una comparativa entre las características de una App Híbrida y una App Nativa.

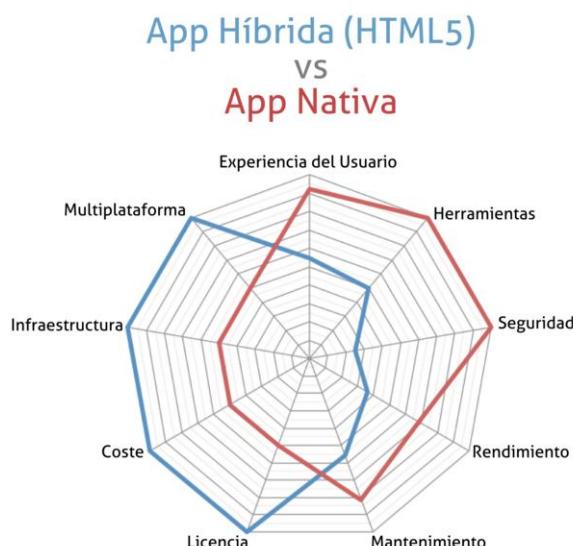


Fig. 11 Comparación app híbrida y con app nativa

En la siguiente imagen se muestra una comparación entre los tres tipos de aplicaciones móviles.

	Nativa	Web App	Híbrida
Características de la aplicación			
Gráficos	API Nativa	HTML, Canvas, SVG	HTML, Canvas, SVG
Rendimiento	Muy rápido	Lento	Rápido
Look & Feel	Nativo	Emulado	Emulado
Distribución	Store	Web	Store
Acceso dispositivo			
Cámara	Sí	No	Sí
Notificaciones	Sí	No	Sí
Contactos, calendario	Sí	No	Sí
Almacenamiento offline	Almacenamiento seguro de ficheros	El que permita el navegador: Cache, IndexedDb, SQL compartida	Almacenamiento seguro de ficheros, SQL compartida.
Geolocalización	Sí	Sí	Sí
Conectividad	Online y offline	Mayormente Online	Online y offline
Conocimientos necesarios	ObjectiveC, Java	HTML5, CSS, Javascript	HTML5, CSS, Javascript

Tabla 5 Comparación final de los tres tipos de aplicaciones móviles. Fuente: elaboración propia

Después de analizar las diferentes características de las distintas aplicaciones móviles, se ha decidido desarrollar la aplicación para el proyecto **“NFU Nos Falta Uno”** de forma híbrida. Los principales motivos de la elección son:

- Crear la aplicación para diferentes plataformas, realizando la programación con HTML, CSS y Javascript.
- Uso de GPS, notificaciones y de la cámara del móvil.
- Presencia en las tiendas de aplicaciones de los principales sistemas operativos.
- Rendimiento de la aplicación rápido.

6.2.2 Frameworks para el desarrollo de aplicaciones híbridas

En el Anexo I del presente proyecto, se exponen de manera detallada los diferentes frameworks para el desarrollo de aplicaciones híbridas, haciendo la distinción entre dos grandes grupos: las aplicaciones híbridas basadas en web app y las aplicaciones híbridas interpretadas.

Una vez realizado el estudio de los diferentes frameworks del Anexo I para el desarrollo de aplicaciones híbridas, y estudiado las ventajas y desventajas de cada uno, podemos concluir que el framework seleccionado para el desarrollo de la aplicación “**Nos Falta Uno**” es **Ionic**.

Ionic, permite trabajar y aprovechar todas las capacidades del terminal móvil mediante la inclusión de plugins desarrollados por la comunidad. El programador puede disponer de un plugin que acceda a las propiedades de la cámara, un plugin que permita leer códigos QR, plugins que te permitan enviar documentos a imprimir a una impresora... Hay infinidad de plugin disponibles. En esta página se pueden ver muchos de los plugin de los que se comenta anteriormente. <http://plugins.cordova.io/>

Ionic aprovecha todos los plugin de Cordova, gracias a eso hay actualmente una cantidad ingente de plugins con los que poder trabajar, cosa que es de agradecer porque pese a que quizá todavía la comunidad es pequeñita, es una magnifica comunidad.

A parte, dispone de un CLI (Command Line Interface) con el que desde el terminal del programador podrá crear nuevos proyectos a su gusto, por ejemplo bien empezando con un proyecto en blanco o con alguna pequeña estructura determinada acorde a sus necesidades o necesidades del cliente. También se puede compilar el proyecto, emular la app mediante un emulador o hacer que funcione nuestra app en un teléfono mediante un solo comando. Es todo muy sencillo.

Si bien es cierto que para mi gusto a la documentación de **Ionic** le faltan ejemplos y podría ser un poco más extensa, con un poco de investigación puedes conseguir hacer todo lo que uno se proponga.

Las razones principales de la selección de Ionic son:

- Plataforma gratuita
- Alto rendimiento: la velocidad es importante. Tan importante que sólo se nota cuando no está en tu app. Ionic está construido para ser

rápido gracias a la mínima manipulación del DOM, con cero jQuery y con aceleraciones de transiciones por hardware.

- Programación de un mismo código para Android e iOS
- AngularJS & Ionic: Ionic utiliza AngularJS con el fin de crear un marco más adecuado para desarrollar aplicaciones ricas y robustas. Ionic no sólo se ve bien, sino que su arquitectura central es robusta y seria para el desarrollo de aplicaciones. Trabaja perfectamente con AngularJS.
- Gran cantidad de plugins para acceder a las características del dispositivo
- Centro nativo: Ionic se inspira en las SDK de desarrollo móviles nativos más populares, por lo que es fácil de entender para cualquier persona que ha construido una aplicación nativa para iOS o Android. Lo interesante, como sabéis, es que desarrollas una vez, y compilas para varios.
- Bonito diseño: limpio, sencillo y funcional. Ionic ha sido diseñado para poder trabajar con todos los dispositivos móviles actuales. Con muchos componentes usados en móviles, tipografía, elementos interactivos, etc.
- Lo que programas se puede visualizar al instante con Ionic View
- Por mis conocimientos de HTML, CSS y AngularJS.
- Alto rendimiento de la aplicación, ya que los móviles de hoy en día cuentan con 2 o 3 GB de memoria RAM y procesadores de 4 u 8 núcleos.
- Con cambiar un 10% de código, se puede convertir la app en una página web.
- Un potente CLI: con un sólo comando podrás crear, construir, probar y compilar tus aplicaciones en cualquier plataforma.

6.2.3 Plan de empresa: estudio de la viabilidad del proyecto

Antes de pasar a las siguientes fases de planificación, diseño y desarrollo, es necesario realizar el plan de empresa del proyecto para saber si es viable o no. Al igual que en el apartado anterior, el plan de empresa completo de NFU se encuentra disponible en el Anexo II del presente documento.

De la realización del plan de viabilidad se extrae la conclusión de que existe un lugar en el mercado para el desarrollo del negocio planteado.

En el análisis del entorno se concluye por un lado, que existe una tendencia social hacia la práctica de los deportes de equipos; y por el otro, se ve claramente que se está produciendo un aumento en la penetración y el uso de internet, especialmente en el ámbito de las redes sociales, motivado entre otras cosas por el compañerismo y el aspecto social. Los consumidores han tomado conciencia y buscan la mejor relación calidad-precio a la hora de practicar un deporte.

Del plan de marketing se concluye por una parte que existe un amplio sector de la población que muestra interés en este sector, especialmente el público masculino y femenino de edad comprendida entre 16 y 60 años. Por otra parte, es importante conferir una imagen de especialización para crearse un hueco en el mercado a la vez que se debe mantener una línea de precios muy competitiva.

En referencia a la actividad de la empresa no se prevé que los procesos internos sufran cambios durante sus primeros años, pero quedarán a disposición de la evolución llevada a cabo. El organigrama de la empresa se deberá ir modificando acorde al conjunto de operaciones a realizar y puede verse alterado por el volumen de ventas experimentado. También, se aconseja considerar la posibilidad de expandir la cartera de productos y servicios además de abrirse a mercados internacionales una vez se vayan asentando las bases del negocio lo que supondría grandes cambios en la estructura de la empresa.

Por último, se concluye que la constitución de esta empresa es viable a nivel económico según las previsiones realizadas en el plan económico financiero. Para financiar los primeros meses de actividad, son necesarios 1.000 € de capital social. Hasta el segundo mes de actividad no se empiezan a generar beneficios, momento a partir del cual la empresa comenzará a autofinanciarse.

6.3 Planificación y organización del trabajo

Esta tercera fase se centra en desglosar todas las actividades que se van a realizar, desde el planteamiento e identificación del problema, hasta la

puesta en marcha. A continuación se exponen todas las fases planteadas en este apartado, indicando detalladamente las actividades que se van a realizar en cada una de ellas, los medios humanos que se van a utilizar, las horas que se van a emplear y el software y hardware que se va a necesitar.

Para tener una visión global de la planificación del proyecto, se muestra la siguiente tabla. Se puede observar que el proyecto se ha realizado entre el 14 de marzo y el 31 de mayo.

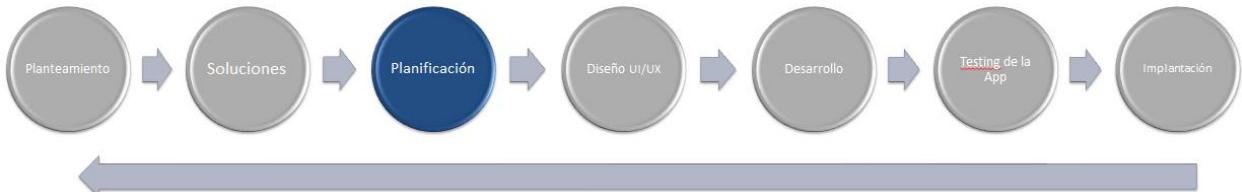


Fig. 12 Fase Planificación. Fuente: elaboración propia

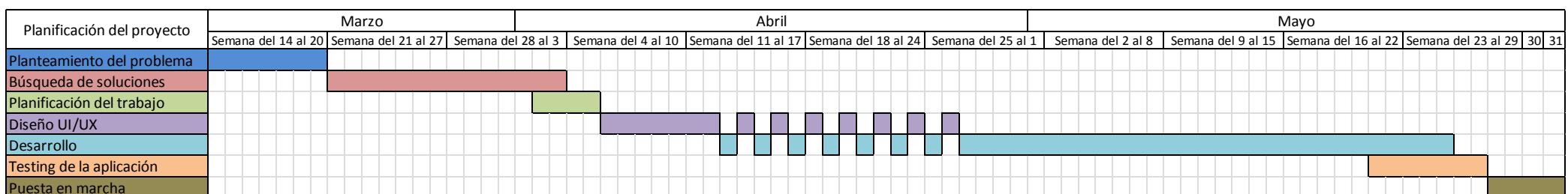


Fig. 13 Diagrama de Gantt de todas las fases del proyecto. Fuente: elaboración propia

En todas las fases se utiliza el mismo hardware, en este caso un ordenador portátil de la marca ASUS con procesador Intel Core i7, 8GB de memoria RAM, tarjeta gráfica nVidia GeForce GT820M con 2GB DDR3L VRAM y 1TB de disco duro.

Fase 1: Planteamiento e identificación del problema		
Descripción: consiste en analizar la propuesta a desarrollar, es decir, qué soluciones se van a llevar a cabo para solucionar el problema que se cuestiona. Es también de estudio en esta fase, el análisis de las aplicaciones similares que se disponen en el mercado, que cubren parcial o totalmente las necesidades cuestionadas.	Duración (semanas): 	
Actividades: <ul style="list-style-type: none"> - Búsqueda de soluciones al problema planteado, apuntando las ideas en una libreta. - Búsqueda en Google Play y análisis de las apps similares en las tiendas de aplicaciones móviles. 	Tiempo invertido (horas): 	
Conocimientos:	Recursos utilizados:	
Informática e Internet	Google Chrome Google Play	Portátil con conexión a Internet

Tabla 6 Fase 1: Planteamiento e identificación del problema. Fuente: elaboración propia

Fase 2 : Búsqueda de soluciones	
Descripción: consiste en estudiar los diferentes frameworks para programar la aplicación, seleccionando aquel que mejor se adapte a las necesidades del proyecto. También es de estudio en esta fase, evaluar la viabilidad del proyecto.	Duración (semanas): 
Actividades: <ul style="list-style-type: none"> - Búsqueda en Internet y análisis posterior de los frameworks para desarrollar una aplicación móvil híbrida. - Elaboración del plan de empresa con los siguientes apartados: presentación de la empresa, estudio de mercado, plan de marketing, plan de producción, infraestructuras, organización y recursos humanos, plan económico-financiero, seguridad y salud laboral y trámites para el inicio de la actividad. 	Tiempo invertido (horas): 
Conocimientos:	Recursos utilizados:
Informática e Internet	Google Chrome Word y Excel
	Portátil con conexión a Internet

Tabla 7 Fase 2: Búsqueda de soluciones. Fuente: elaboración propia

Fase 3 : Planificación y organización del trabajo	
Descripción: es en la fase en la que nos encontramos. Consiste en tener un programa de trabajo con un desglose de todas las actividades que se van a realizar, estimando el número de horas que se le va a dedicar cada una de ellas y estableciendo los medios humanos que se van a utilizar para alcanzar los objetivos que se hayan propuesto.	Duración (semanas): 
Actividades: - Descripción de cada una de las fases, estimando el tiempo invertido, y el software y hardware necesario.	Tiempo invertido (horas): 
Conocimientos:	Recursos utilizados:
Informática y capacidad de organización del trabajo.	Google Chrome Word y Excel

Tabla 8 Fase 3: Planificación y organización del trabajo

Fase 4 : Diseño UI/UX	
Descripción: previo al desarrollo de la aplicación es necesario tener totalmente definido el diseño estructural de la app y su comportamiento. Para ello se, desarrollarán los mockups de las diferentes interfaces de la aplicación y posteriormente se llevará a cabo el diseño.	Duración (semanas): 
Actividades: <ul style="list-style-type: none">- Elaboración de los mockups- Diseño de la aplicación	Tiempo invertido (horas): 
Conocimientos: De informática y de diseño, siguiendo la guía de estilo de Google.	Recursos utilizados: Google Chrome Photoshop CS6 (Diseño) Balsamiq (Mockups)

Tabla 9 Fase 4: Diseño UI/UX

Adobe Photoshop⁶ es un editor de gráficos rasterizados desarrollado por Adobe Systems Incorporated. Usado principalmente para el retoque de fotografías y gráficos, su nombre en español significa literalmente "taller de fotos". Es líder mundial del mercado de las aplicaciones de edición de imágenes y domina este sector de tal manera que su nombre es ampliamente empleado como sinónimo para la edición de imágenes en general.

Todo el diseño de la aplicación se ha realizado con este programa.

Balsamiq Mockups⁷ es una aplicación AIR pensada para ayudar a crear de manera fácil y rápida los wireframes. Lo consigue a través de un sistema de drag-and-drop de componentes predefinidos. Todo es tan



Fig. 14 Logo Photoshop.
Recuperado de
https://commons.wikimedia.org/wiki/File:Adobe_Photoshop_CS6_icon.svg



Fig. 15 Logo Balsamiq mockups.
Recuperado de
<http://informatica.desecondaria.com/>

⁶ <http://www.adobe.com/es/products/photoshop.html>

⁷ <https://balsamiq.com/>

fácil como buscar el elemento que necesitamos (un botón, un scroll, una imagen) y arrastrarlo al canvas. En pocos minutos se pueden tener las pantallas definidas. El set de componentes es bastante amplio y, en general, se puede construir cualquier tipo de interfaz.

Fase 5 : Desarrollo	
Descripción: programación de la aplicación siguiendo las pautas de diseño y funcionalidad establecidas en las fases anteriores, para conseguir alcanzar los objetivos planteados.	Duración (semanas):  5
Actividades: <ul style="list-style-type: none">- Programación frontend- Programación parte Servidor/BD, backendRecostrucción del servidor.	Tiempo invertido (horas):  75
Conocimientos: De informática y de Internet. También del uso de Ionic, SASS, AngularJS, PHP y MySQL.	Recursos utilizados: Google Chrome Framework Ionic Apache Cordova SASS AngularJS PHP Bower Gulp npm MySQL Atom NetBeans Workbench

Tabla 10 Fase 5: Desarrollo. Fuente: elaboración propia

Ionic es un framework propiedad de la empresa Drifty. La característica principal de Ionic es que aporta a HTML un conjunto de controles para la interfaz gráfica que no están incluidas en este pero que sí son comunes en las aplicaciones móviles. Estos componentes están construidos con una combinación de CSS, HTML y JavaScript, y se comportan como los controles nativos que se usan normalmente.

Integra Cordova directamente, lo que permite que con una sola instalación ya se pueda compilar directamente la aplicación a la plataforma que se desee o instalar los plugins de Cordova para utilizarlos en Ionic.

Además Ionic va a aportar un ecosistema de herramientas que en la actualidad son completamente gratuitas y que van a facilitar en gran medida las labores de desarrollo. Entre ellas encontramos:

Ionic.io Platform: Conjunto de herramientas para gestionar la aplicación, entre ellas las que serían más útiles para el proyecto se encuentran las siguientes:

- **Ionic Creator:** herramienta de desarrollo grafica que permite realizar las primeras fases de implementación de la aplicación de una forma gráfica, arrastrando y soltando componentes y creando páginas de forma automática.
- **Ionic Deploy:** Ha aparecido recientemente y es la solución back-end de ionic que va a permitir realizar actualizaciones a la aplicación.
- **Ionic Analytics:** Servicio de analíticas para la aplicación.
- **Ionic View App:** Aplicación móvil para realizar pruebas en terminales.

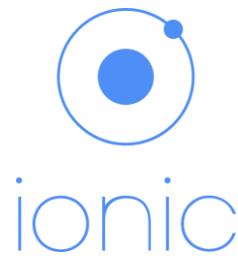


Fig. 16 Logo Ionic.
Recuperado de
<http://asktechi.blogspot.com.es/2015/08/how-to-add-app-logo-in-header-using.html>

- **Ionic Lab:** Herramienta para testear la aplicación en múltiples tamaños de pantalla y sistemas operativos.

Ionic funciona sobre Angular JS. Angular JS es un framework de JavaScript de código abierto propiedad de Google que proporciona a los desarrolladores web la posibilidad de escribir aplicaciones completas de forma rápida y proporciona una buena estructura de la aplicación.

Ionic incorpora un nuevo e importante conjunto de mejoras en aplicaciones híbridas que otras herramientas (como jQuery Mobile) no han sido capaces de proporcionar. Hasta hace poco, los dispositivos móviles eran relativamente lentos y sólo una aplicación nativa podía ofrecer el rendimiento y la experiencia al usuario que muchos desarrolladores querían o necesitaban.

En la actualidad la gran potencia que desarrollan los dispositivos móviles permiten ejecutar aplicaciones móviles híbridas a través del navegador web ofreciendo una calidad prácticamente nativa. Ionic está demostrando que es capaz de aprovechar esta potencia y alcanzar un alto rendimiento en sus aplicaciones.

Cordova⁸ es el framework que aloja el navegador web y que sirve de



comunicador entre el terminal y el código. Apache Cordova es de distribución libre que pueden ser utilizados libremente en cualquier aplicación sin necesidad de atribución o licencias de ningún tipo. Este compilador realiza el proceso de creación de la aplicación nativa a partir de nuestro código HTML, CSS y JavaScript. En las últimas versiones la distribución ha reducido su tamaño hasta el mínimo posible ya que el peso es un hándicap muy importante en el sector móvil. Esta reducción ha sido a costa de extraer todas las funcionalidades, los plugins, que permiten interactuar con el dispositivo

Fig. 17 Logo Apache Cordova. Recuperado de <http://palebluedot.tv/mobile-app-development-belfast/>

⁸ <https://cordova.apache.org/>

móvil. En la actualidad estos plugins se añaden manualmente en función de las necesidades específicas de cada proyecto.

Sass⁹ (Syntactically Awesome Style Sheets) es un preprocesador CSS, esto significa que se puede usar como lenguaje de script con elementos de programación sacados de Ruby y que ayudan a manipular el resultado de tus archivos css. Puedes usar variables, hacer operaciones matemáticas básicas, definir y llamar a funciones, manipular colores, etc.

Escribimos sintaxis sass dentro de ficheros con extensión .scss, que posteriormente compilamos para dar lugar a los ficheros .css.



Fig. 18 Logo Sass. Recuperado de <http://maquetando.com/sass-y-less-el-presente-de-css>

AngularJS¹⁰, o simplemente Angular, es un framework de JavaScript de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles.

La biblioteca lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript. Los valores de las variables de JavaScript se pueden configurar manualmente, o recuperados de los recursos JSON estáticos o dinámicos.

PHP¹¹ es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en



Fig. 19 Logo AngularJS.
Recuperado de
<http://www.marioperez.com.mx/cursos-online/>



Fig. 20 Logo Php.
Recuperado de:
<http://www.taringa.net/posts/ebooks-tutoriales/19201586/Php-o-C.html>

⁹ <http://sass-lang.com/>

¹⁰ <https://angularjs.org/>

¹¹ <http://www.php.net/>

el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

PHP se considera uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hasta el día de hoy[cita requerida], lo que ha atraído el interés de múltiples sitios con gran demanda de tráfico, como Facebook, para optar por el mismo como tecnología de servidor.

Bower¹² es un complemento ideal para el desarrollo web. Es un sencillo programa que nos sirve para tener al día las dependencias de un proyecto para la web, en lo que respecta al desarrollo frontend. Se trata de un programa basado en NodeJS que se ejecuta desde la consola y que tiene un sencillo API de comandos útiles para realizar tareas de mantenimiento y administración de paquetes necesarios para construir un proyecto web, concretamente la parte del lado del cliente.

Con Bower podemos descargar y actualizar todo tipo de librerías, frameworks, plugins, etc., pero sin tener que preocuparnos por descargarlos y subirlos a mano nosotros mismos. En este artículo aprenderemos a usar sus funciones más comunes.

Gulp.js¹³ es un build system(sistema de construcción) que permite automatizar tareas comunes de desarrollo, tales como la minificación de código JavaScript, recarga del navegador, compresión de imágenes, validación de sintaxis de código y un sin fin de tareas más.



Fig. 21 Logo Bower.
Recuperado de
<http://bower.io/docs/about/>



Fig. 22 Logo Gulp.
Recuperado de
[https://www.npmjs.com/
package/gulp](https://www.npmjs.com/package/gulp)

¹² <http://bower.io/>

¹³ <http://gulpjs.com/>

npm¹⁴ es un gestor de paquetes para javascript", es el predeterminado para node.js, cuando instalas node también se instala npm. ¿Y qué quiere decir esto? Pues que a través de npm podremos instalar y gestionar los paquetes para nuestras aplicaciones.



Fig. 23 Logo npm.
Recuperado de
<https://www.npmjs.com/>

MySQL¹⁵, es un sistema de gestión de base de datos relacional o SGBD. Este gestor de base de datos en multihilo y multiusuario, lo que le permite ser utilizado por varias personas al mismo tiempo, e incluso, realizar varias consultas a la vez, lo que lo hace sumamente versátil.



Fig. 24 Logo MySQL. Recuperado de
[http://logodatabases.com/mysql-
logo.html/mysql](http://logodatabases.com/mysql-logo.html/mysql)

Nació como una iniciativa de Software Libre y aún sigue ofreciéndose como tal, para usuarios particulares. Pero si se desea utilizarlo para promover datos en una empresa, se puede comprar una licencia, como un software propietario, que es autoría de la empresa patrocinante (Actualmente Oracle Corporation).

La mayor parte del código se encuentra escrito en lenguaje C/C++ y la sintaxis de su uso es bastante simple, lo que permite crear bases de datos simples o complejas con mucha facilidad. Además, es compatible con múltiples plataformas informáticas y ofrece una infinidad de aplicaciones que permiten acceder rápidamente a las sentencias del gestor de base de datos.

Atom¹⁶ es un editor de texto orientado al desarrollo de aplicaciones, desarrollado por Github. El editor de texto es el mejor compañero de los desarrolladores; es su herramienta fundamental para poder escribir las líneas de código que luego darán vida a una aplicación o a una página web. Un buen editor de texto puede ayudar mucho a los desarrolladores y, por ejemplo, nos puede ahorrar tiempo en la fase de depuración si el editor nos

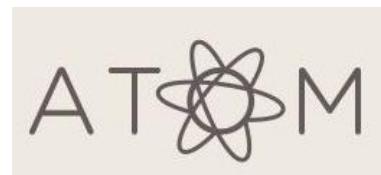


Fig. 25 Logo Atom. Recuperado
de
[http://www.josedomingo.org/pl
edin/2014/10/mi-experiencia-
con-atom/](http://www.josedomingo.org/pledin/2014/10/mi-experiencia-con-atom/)

¹⁴ <https://www.npmjs.com/>

¹⁵ <https://www.mysql.com/>

¹⁶ <https://atom.io/>

ofrece ayuda con la sintaxis del lenguaje de programación con el que estemos trabajando.

NetBeans¹⁷ es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.



Fig. 26 Logo NetBeans. Recuperado de <http://www.logotypes101.com/logo/netbeans>

MySQL Workbench¹⁸ es una herramienta visual de diseño de bases de datos que integra desarrollo de software, Administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de base de datos MySQL.



Fig. 27 Logo MySQL Workbench. Recuperado de <http://www.moodlebites.com/login/index.php>

¹⁷ <https://netbeans.org/>

¹⁸ <https://www.mysql.com/products/workbench/>

Fase 6 : Testing de la aplicación	
Descripción: testing profundo de todas las partes de la aplicación, para saber si funciona correctamente toda la parte funcional y también saber si el rendimiento es adecuado.	Duración (semanas):  1
Actividades: - Utilización de la aplicación para detectar problemas funcionales y de rendimiento.	Tiempo invertido (horas):  15
Conocimientos: De Ionic.	Recursos utilizados: Google Chrome Framework Ionic
	Portátil con conexión a Internet

Tabla 11 Fase 6: Testing de la aplicación. Fuente: elaboración propia

Fase 7 : Implantación/puesta en marcha	
Descripción: a la finalización del desarrollo la app será apta para darse a conocer y comercializarse y el último paso será subirlo a los markets de aplicaciones correspondientes.	Duración (semanas):  0,5
Actividades: <ul style="list-style-type: none">- Firmar digitalmente de la app con la cuenta de desarrollador.- Compilación de la app con Ionic y Phonegap- Preparación de imágenes y descripción para la Google Play.	Tiempo invertido (horas):  7,5
Conocimientos: De informática y de diseño. También de como publicar aplicaciones móviles.	Recursos utilizados: Google Chrome Framework Ionic Photoshop CS6 Portátil con conexión a Internet

Tabla 12 Fase 7: Implantación/puesta en marcha. Fuente: elaboración propia

6.4 Diseño UI/UX

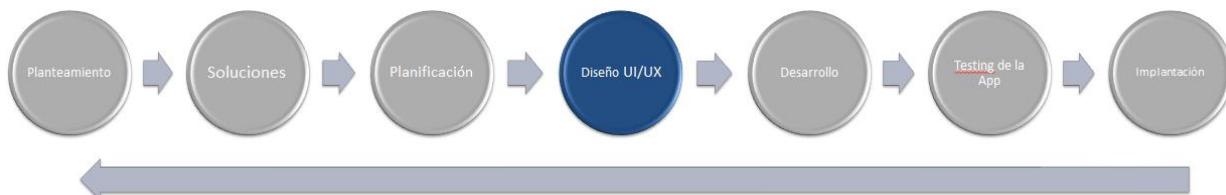


Fig. 28 Fase de Diseño UI/UX. Fuente: elaboración propia

Partiendo de la información obtenida en las fases previas, se crea la interfaz de la aplicación, ofreciendo al usuario la posibilidad de interactuar con la totalidad de las funcionalidades de la aplicación de la forma más cómoda e intuitiva posible.

Para elaborar este diseño se han elegido imágenes y tipografías gratuitas y de libre acceso, que no comprometan la legalidad de la aplicación, obtenidas principalmente de Freepik¹⁹. Así mismo, se han creado estilos propios para mantener uniforme la estética del proyecto.

A continuación se muestran los mockups y el diseño de las diferentes ventanas de las que consta la aplicación:

1. Pantalla de carga o inicio
2. Pantalla de Login
3. Pantalla de partidas
 - a. Pantalla de información de la partida
 - b. Pantalla de jugadores participantes
 - c. Pantalla de geolocalización de la partida
 - d. Pantalla de comentarios
4. Pantalla de mapa de partidas
5. Pantalla de instalaciones
6. Pantalla de mi perfil
 - a. Crear partida
 - b. Gestionar partidas
 - c. Invitaciones
 - d. Modificar perfil
 - e. Settings

¹⁹ <http://www.freepik.com/>

6.4.1 Pantalla de carga o inicio



Fig. 29 Mockup y diseño de la pantalla inicial. Fuente: elaboración propia

Al iniciar la aplicación, el usuario podrá observar el logo y la imagen inconfundible de **Nos Falta Uno**, mientras se carga el contenido de ésta.



Fig. 30 Logo NFU. Fuente: elaboración propia

El logo de NFU surge de la semejanza con un escudo de un equipo deportivo, ya sea de baloncesto, fútbol, balonmano, etc. Se han utilizado los colores verde y azul, después de realizar un estudio de los siguientes parámetros:

- El contenido deportivo de la aplicación, y
- El público principal: en este caso personas del sexo masculino. La razón es que este sexo realiza más deporte. Según una encuesta realizada en el 2014, los puntos destacables por los que las mujeres realizan menos deporte son: falta de interés, más obligaciones familiares con lo que lleva menos tiempo libre. Los colores fríos, son los favoritos del público masculino.

En el centro destacan las siglas NFU (Nos Falta Uno) con la tipografía **“Zoloft”**. La parte superior del logotipo tiene cinco estrellas que simbolizan éxito y gloria. La parte baja se decora con una especie de cinta donde se inserta el nombre de la aplicación web para resaltar aún más el nombre de la web.

6.4.2 Pantalla de Login



Fig. 31 Mockup y diseño de la pantalla login. Fuente: elaboración propia

La segunda pantalla con la que se va a encontrar el usuario es con el login, el acceso a la aplicación se puede realizar de dos formas:

- Mediante el uso del nombre de Usuario y Password, o
- Mediante el uso de las redes sociales.

Una vez logueados, se recibirá un correo para confirmar el alta en la aplicación.

6.4.3 Pantalla de partidas

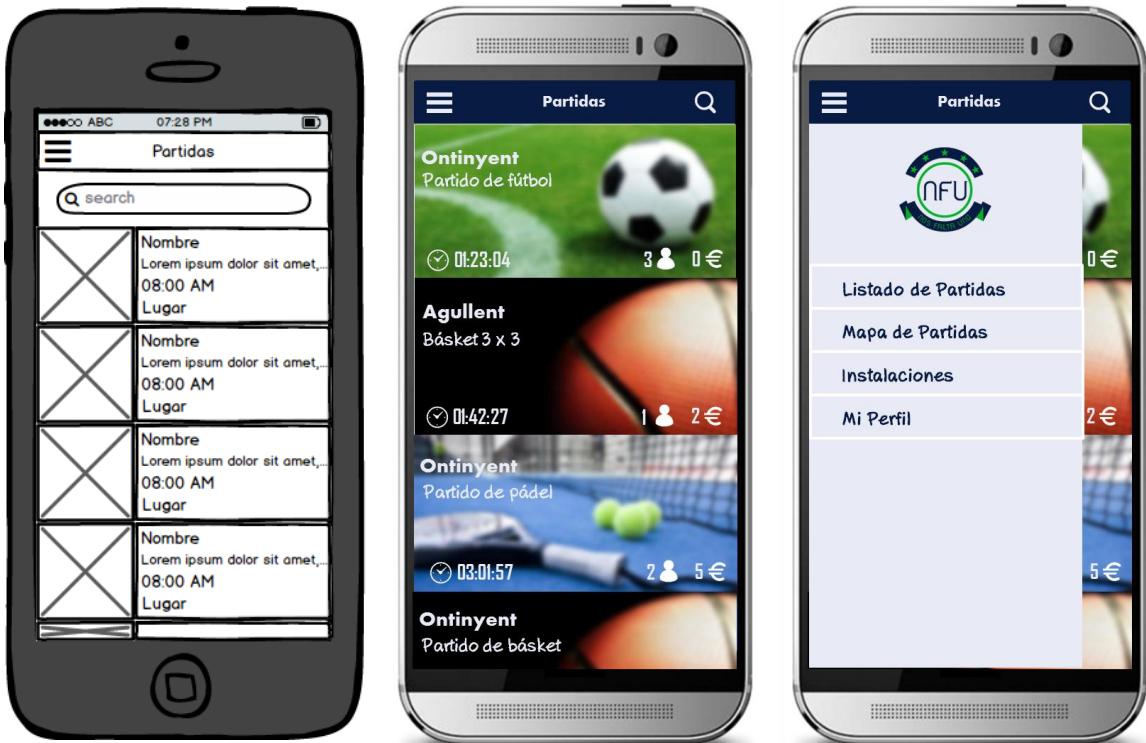


Fig. 32 Mockup y diseño de la pantalla partidas y del menú lateral. Fuente: elaboración propia

Una vez se accede a la aplicación, la primera pantalla que va a ver el usuario es la pantalla de partidas. En ella, el usuario puede observar las partidas que otros usuarios han dado de alta y que están cerca de su posición, ya que la aplicación geolocaliza al usuario.

En cada partida se indica, en la parte superior, el municipio donde se va a jugar la partida y el nombre de ésta. En la parte inferior, se observan tres iconos, indicando: el tiempo que falta para que se inicie la partida, el número de jugadores que faltan para completar los equipos y el coste de la partida.

Para mostrar las partidas, se ha optado por mostrar una lista, utilizando una imagen inconfundible para cada uno de los deportes. Para ofrecer un contraste visual entre la imagen, y los textos y los iconos se ha utilizado el color blanco.



En la parte superior, se utiliza un header, que se repite a lo largo de la aplicación, del mismo color azul que el logo. Conteniendo el icono  , desde el cual se puede realizar una búsqueda, y el icono del menú  . Al seleccionarlo, éste se desliza en la pantalla, mostrando los diferentes apartados:

- Listado de partidas
- Mapa de partidas
- Instalaciones
- Mi perfil

La parte del menú se explica en el apartado 6.4.4.

Si el usuario selecciona una de las partidas, se le mostrará la información de ésta en los siguientes cuatro subapartados:

a) Pantalla de información de la partida



Fig. 33 Mockup y diseño de la pantalla información de la partida. Fuente: elaboración propia

La pantalla de estos cuatro apartados se configura con el mismo header y el mismo footer, este último realizado con un navbar.

En el primer ícono del navbar, se muestra la información de la partida. La información se estructura en dos partes, en la parte superior se utiliza la imagen del deporte, y los iconos de coste y de número de jugadores. En la parte inferior se muestra la información del partido, las normas y el botón de “*unirse a la partida*”. Ambas partes están separadas por una franja gris, la cual muestra la ubicación y el nombre la partida.

b) Pantalla de jugadores participantes

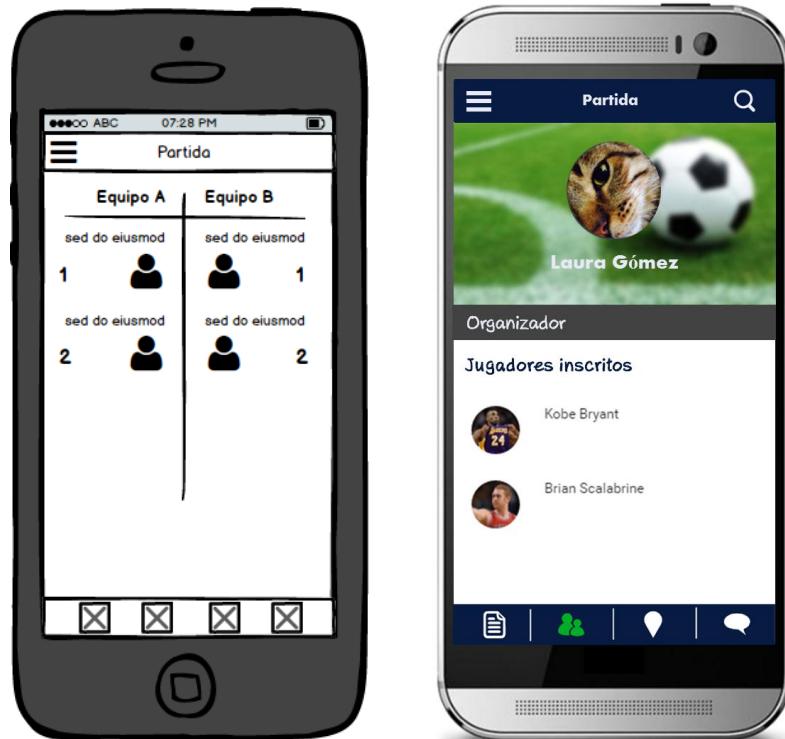


Fig. 34 Mockup y diseño de la pantalla jugadores. Fuente: elaboración propia

El segundo ícono del navbar, es la pantalla de jugadores participantes. Al igual que en el apartado anterior a), la información se estructura en dos partes, en la parte superior se utiliza el avatar del organizador sobre la imagen del deporte. En la parte inferior se muestra a los jugadores inscritos, indicando el avatar y el nombre del usuario. Ambas partes están separadas por una franja gris, la cual hace de encabezado, mostrando la palabra organizador.

c) Pantalla de geolocalización de la partida

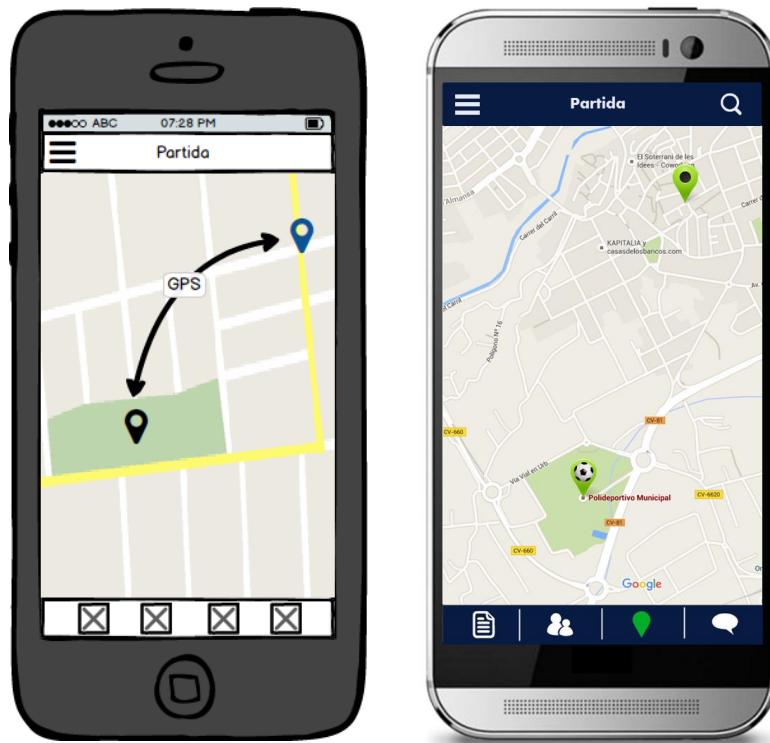


Fig. 35 Mockup y diseño de la pantalla ubicación de la partida. Fuente: elaboración propia

El tercer ícono del navbar, es la pantalla de la ubicación de la partida. En esta pantalla, se muestra la ubicación del usuario mediante la geolocalización, y la ubicación de la partida. Si el usuario pulsa sobre el ícono de la partida, se le muestran las indicaciones hasta llegar a ésta.

d) Pantalla comentarios



Fig. 36 Mockup y diseño de la pantalla comentarios. Fuente: elaboración propia

El cuarto y último ícono del navbar, muestra la pantalla de comentarios. En esta pantalla, los usuarios pueden escribir mensajes para comunicarse entre ellos o con el organizador, para poder aclarar algunos aspectos o configurar equipos.

6.4.4 Pantalla de mapa de partidas

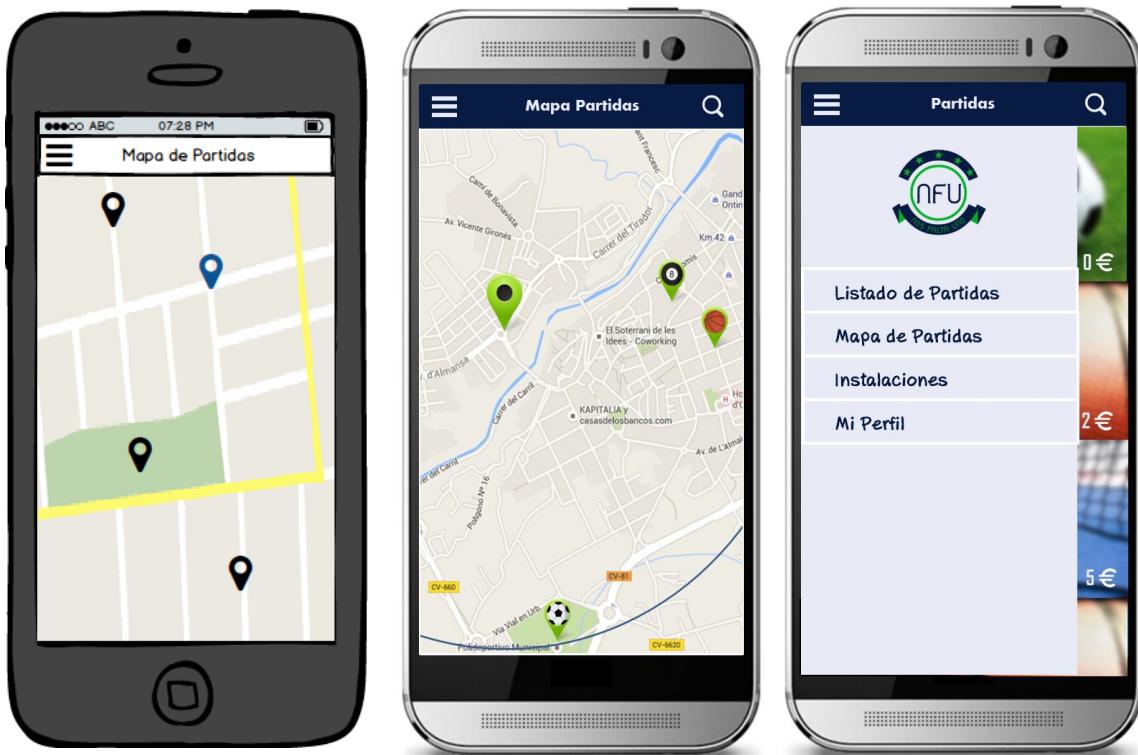


Fig. 37 Mockup y diseño de la pantalla mapa de partidas. Fuente: elaboración propia

En el menú, al seleccionar el botón “*Mapa de partidas*”, se muestran geolocalizadas en el mapa de Google Maps, las partidas más cercanas al usuario. Cada deporte se muestra con su ícono de geolocalización correspondiente, para evitar confusiones. Si el usuario pulsa sobre cualquier deporte geolocalizado, se le mostrará la información de la partida, del apartado 6.4.3. a. b. c. y d.

6.4.5 Pantalla de Instalaciones

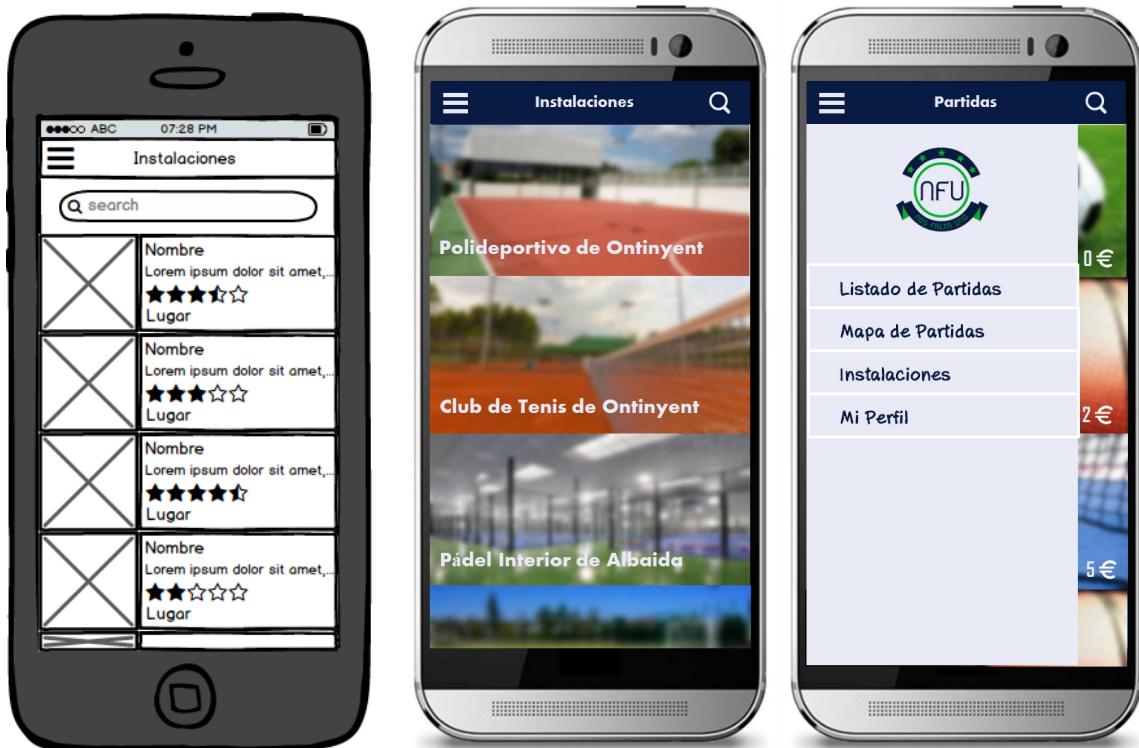


Fig. 38 Mockup y diseño de la pantalla instalaciones. Fuente: elaboración propia

Si seleccionamos “*Instalaciones*” en el menú, el usuario puede observar listadas las instalaciones más cercanas a él. Cada instalación muestra una imagen difuminada de ésta y el nombre correspondiente. Si el usuario selecciona una de las instalaciones de la pantalla, navegará a la siguiente pantalla.



Fig. 39 Mockup y diseño de la pantalla información de la instalación. Fuente: elaboración propia

En esta pantalla, el usuario obtiene una pequeña información de la instalación, la valoración de ésta y si selecciona el botón , obtendrá la ubicación e indicaciones para llegar a la instalación.

Al igual que el resto de la aplicación, la información se estructura en dos partes, en la parte superior se utiliza la imagen de la instalación. En la parte inferior se muestra la descripción de la instalación. Ambas partes están separadas por una franja gris, en la que se indica la valoración de la instalación, realizada por los usuarios.

6.4.6 Pantalla de mi perfil

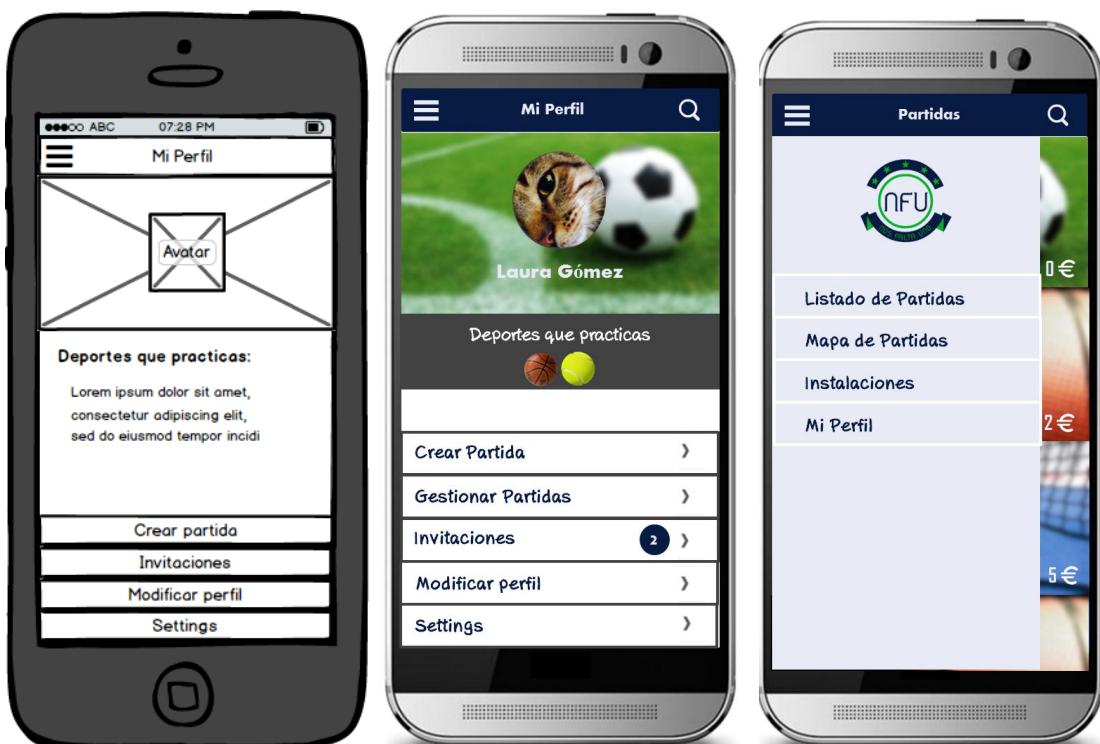


Fig. 40 Mockup y diseño de la pantalla Mi perfil. Fuente: elaboración propia

Si seleccionamos en el menú “Mi Perfil”, el usuario navega a su perfil. Esta pantalla, sigue la misma estructura que en el resto de la aplicación, dividiendo la información en dos partes. En la parte superior, se muestra el avatar del usuario sobre una imagen de fondo. En la parte inferior, se encuentra un submenú formado por:

- Crear partida
- Gestionar partidas
- Invitaciones, indicando el número de ellas
- Modificar perfil
- Settings

Ambas partes están separadas por una franja gris, en la que se muestran los deportes que practica el usuario.

A continuación se explica el submenú de “*Mi Perfil*”.



Fig. 41 Mockup y diseño de la pantalla crear partidas. Fuente: elaboración propia

Al seleccionar “*Crear Partida*”, se muestra un formulario que el usuario tiene que llenar, para poder dar de alta una partida. El formulario está compuesto de:

- Nombre y deporte
- ¿Dónde y Cuándo?
- Configuración: plazas y coste



Fig. 42 Mockup y diseño de la pantalla gestionar partidas. Fuente: elaboración propia

Si el usuario selecciona “Gestionar Partidas”, se muestran las partidas en las que el usuario se ha apuntado. El diseño es igual que el de la pantalla “Partidas”. Si el usuario selecciona alguna partida, se le mostrará la información de la partida, del apartado 6.4.3. a. b. c. y d, y podrá realizar las gestiones oportunas.



Fig. 43 Mockup y diseño de la pantalla invitaciones. Fuente: elaboración propia

Al seleccionar “*Invitaciones*”, se muestran las partidas en las que el usuario ha sido invitado. El diseño es igual que el de la pantalla “*Partidas*”. Si el usuario selecciona alguna partida, se le mostrará la información de la partida, del apartado 6.4.3. a. b. c. y d.

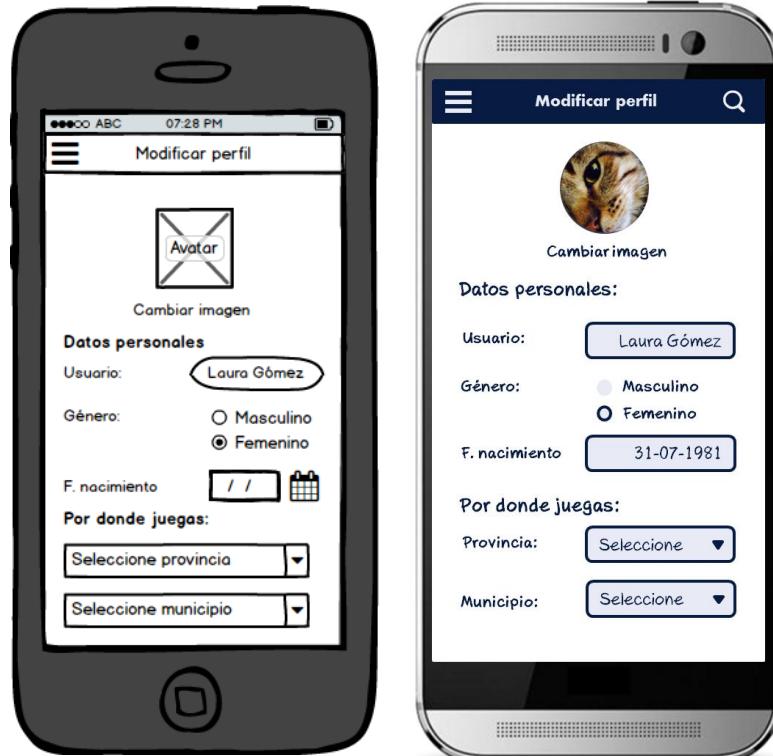


Fig. 44 Mockup y diseño de la pantalla modificar perfil. Fuente: elaboración propia

En la pantalla “*Modificar Perfil*”, el usuario puede modificar la siguiente información de su perfil:

- Imagen de perfil
- Género
- Fecha de nacimiento
- Municipio
- Teléfono
- Correo
- Deportes que practica, y
- Nivel

Por último si se selecciona el botón “*Settings*”, el usuario puede activar o desactivar las notificaciones de las partidas y las notificaciones de los usuarios. Es decir, que cuando un usuario cree una partida o te invite a una partida, puedes ver o no una notificación en el ícono de la aplicación.



Fig. 45 Mockup y diseño de la pantalla settings. Fuente: elaboración propia

6.5 Desarrollo

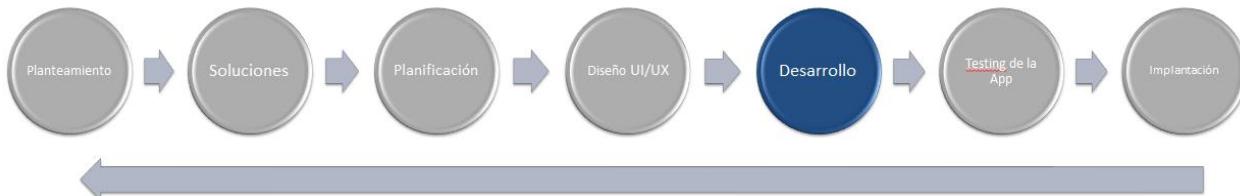


Fig. 46 Fase Desarrollo. Fuente: elaboración propia

En esta fase se explica detalladamente el desarrollo del frontend y backend de la aplicación NFU, siguiendo ambos una estructura MVC.

El **MVC** (Model-View-Controller o Modelo-Vista-Controlador), es un patrón de diseño que separa los datos, la lógica y las interfaces de usuario. Como su nombre indica, está separado en tres componentes: Modelo, Vista y Controlador. Está basado en la ideología de separación de conceptos y cumple perfectamente con los objetivos de los patrones de diseño.

- **Modelo:** es la capa encargada de los datos, es decir, la que se encarga de hacer peticiones a las bases de datos para enviar o recibir información. Estas bases de datos pueden estar alojadas de forma local en nuestra app o de forma remota en un servidor externo.

- **Vista:** se trata del código que nos permitirá presentar los datos que el modelo nos proporciona, como ejemplo podríamos decir que en una aplicación es el código HTML que nos permite mostrar la salida de los datos procesados.

- **Controlador:** es la capa que sirve de enlace entre la vista y el modelo. Envía comandos al modelo para actualizar su estado, y a la vista correspondiente para cambiar su presentación.

6.5.1 Frontend

La estructura de este apartado sigue el orden de desarrollo, es decir:

1º Prerrequisitos e instalación de Ionic

2º Ionic Creator

3º Estructura carpetas

4º Arquitectura del proyecto

5º Añadir plataforma

6º Compilar y ejecutar el proyecto

7º Trabajar con SASS

8º Plugins instalados

9º Panel de control (Dashboard)

1º Prerrequisitos e instalación de Ionic

Para que funcione el framework Ionic hace falta una serie de prerrequisitos, como la instalación de Nodejs y el npm para que funcionen los comandos de Ionic y se añadan sus librerías.

Por tanto el primer paso es instalar Nodejs en el Sistema Operativo de nuestro ordenador, en mi caso la última versión más estable de Ubuntu.

Primero actualizo:

```
$ sudo apt-get update
```

Instalo nodejs:

```
$ sudo apt-get install nodejs
```

Posteriormente instalo npm (Node Package Manager). Cuando usamos Node.js rápidamente tenemos que instalar módulos nuevos (librerías) ya que Node al ser un sistema fuertemente modular viene prácticamente vacío. Así que para la mayoría de las operaciones deberemos instalar módulos adicionales. Esta operación se realiza de forma muy sencilla con la herramienta npm (Node Package Manager).

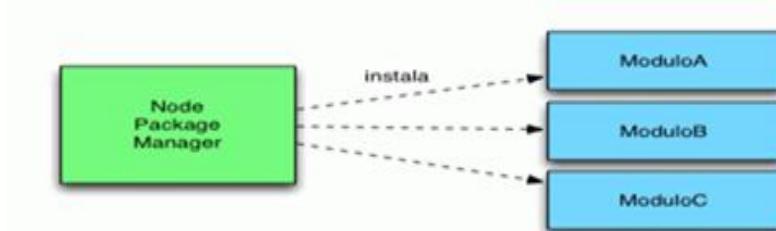


Fig. 47 Estructura de npm. Recuperado de http://i.blogs.es/d470fc/650_1000_npm1-1/450_1000.gif

```
$ sudo apt-get install npm
```

Para buscar componentes podemos usar el repositorio general de Node (npmjs.org) el cual almacena todos los Node Package Modules que han sido

publicados.

A continuación Ionic se instala igual que PhoneGap o Cordova. Aunque trabaje bajo ellos, Ionic tiene su propio instalador:

```
$npm install -g cordova ionic
```

2º Ionic Creator

Una vez realizado estos pasos, Ionic tiene una plataforma online en la cual, de forma sencilla puedes crear la estructura básica de la Aplicación y descargártela.

Cuando te registras entras en la siguiente pantalla:

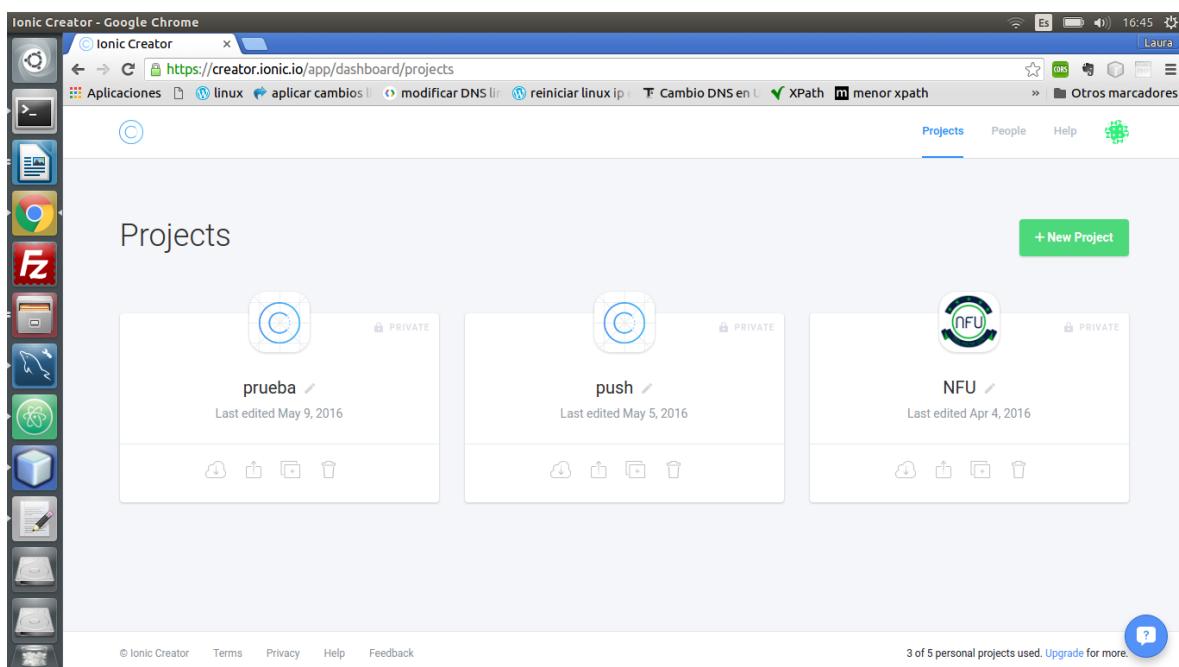


Fig. 48 Pantalla Projects de Ionic Creator. Fuente: elaboración propia

Creamos el proyecto NFU en el botón “+new project”:

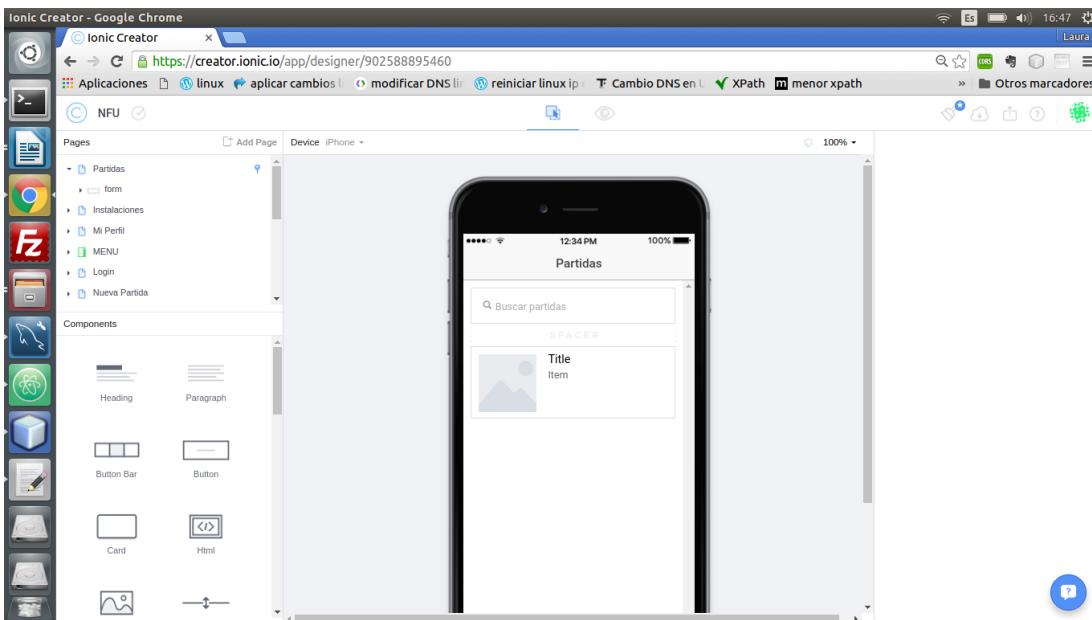


Fig. 49 Pantalla de NFU en Ionic Creator. Fuente: elaboración propia

Y desde aquí de forma muy intuitiva puedes crear la estructura básica de la aplicación.

Una vez construida la estructura. Ionic te facilita mediante un comando y un número de aplicación el descargártela en tu ordenador, a continuación en la siguiente imagen se muestra en que botón se descargártela en tu ordenador, a continuación en la siguiente imagen se muestra en que botón se debe pulsar para que te muestre los parámetros de descarga.

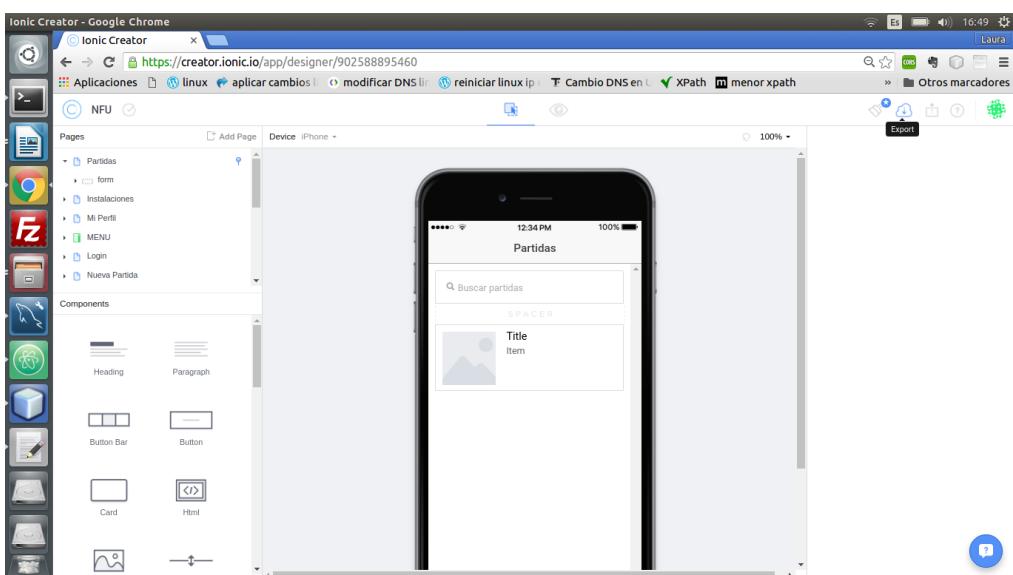


Fig. 50 Exportar proyecto Ionic Creator. Fuente: elaboración propia

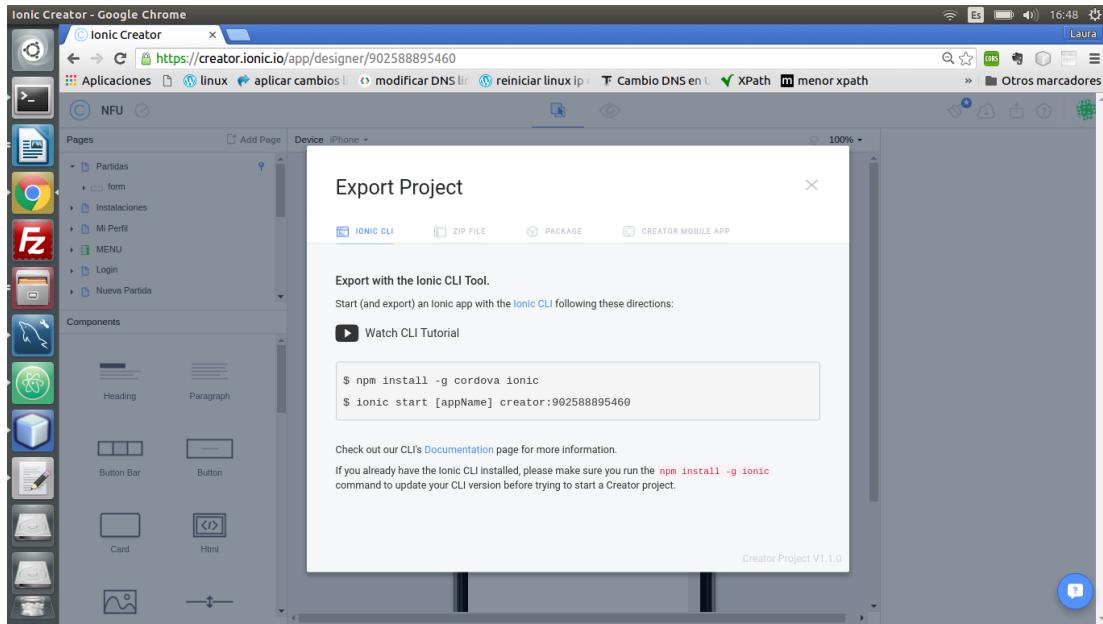


Fig. 51 Parámetros de descarga Ionic Creator. Fuente: elaboración propia

```
$ionic start NFU creator:902588895460
```

Cuando se descarga la aplicación en tu ordenador, Ionic te da la opción de crear una cuenta en Ionic.io account para poder enviar notificaciones y usar Ionic View App, esto último se explicará más adelante.

```
laura@laura-X550LD:~$ ionic start NFU creator:16565986a72b
Creating Ionic app in folder /home/laura/NFU_ based on creator:16565986a72b project
Downloading: https://github.com/driftyco/ionic-app-base/archive/master.zip
[=====] 100% 0.0s
Downloaded Creator Project: https://creator.ionic.io/api/v1/creator/16565986a72b/download-start/cordova?sid=08wo2hlmx1afre3mll7epr5wjw5fs3th
Updated the hooks directory to have execute permissions
Update Config.xml
Initializing cordova project
Your Ionic project is ready to go! Some quick tips:
* cd into your project: $ cd NFU_
* Setup this project to use Sass: ionic setup sass
* Develop in the browser with live reload: ionic serve
* Add a platform (ios or Android): ionic platform add ios [android]
  Note: iOS development requires OS X currently
  See the Android Platform Guide for full Android installation instructions:
  https://cordova.apache.org/docs/en/edge/guide_platforms_android_index.md.html
* Build your app: ionic build <PLATFORM>
* Simulate your app: ionic emulate <PLATFORM>
* Run your app on a device: ionic run <PLATFORM>
* Package an app using Ionic package service: ionic package <MODE> <PLATFORM>
For more help use ionic --help or ionic docs
Visit the Ionic docs: http://ionicframework.com/docs
Create an ionic.io account to send Push Notifications and use the Ionic View app?
(Y/n):
```

Fig. 52 Descargar proyecto desde la consola. Fuente: elaboración propia

Aceptamos y nos crea una cuenta en <https://apps.ionic.io/apps> con un panel de control donde podemos manejar una serie de parámetros útiles para el desarrollo de la aplicación. También nos permite utilizar la aplicación móvil Ionic View app, que posteriormente se explica.

Para poder optar a utilizar el panel, se debe instalar un plugin:

```
$ionic add ionic-platform-web-client
```

3º Estructura de carpetas

Como hemos visto en el apartado anterior, al generar un nuevo proyecto de Ionic se nos creará una estructura predefinida de carpetas y ficheros que nos permitirán organizar el código de nuestro proyecto. A continuación vamos a ver para que sirve cada una de estos elementos:

- hooks/- Esta carpeta se utiliza para añadir scripts que se ejecutarán cuando se produzcan determinados eventos, como por ejemplo antes o después, de la compilación, etc. En la propia carpeta se incluye un fichero con instrucciones para su utilización.
- Platforms/ - Contiene el código específico de las plataformas móviles para las cuales se va a compilar, como por ejemplo Android, iOS, etc. El código de esta carpeta es generado y no se ha de modificar manualmente.
- Plugins/ - Contiene los plugins o módulos instalados para nuestra aplicación, los cuales se utilizan para añadir funcionalidad como el acceso a las características nativas del móvil.

Se pueden añadir plugins de ionic desde la siguiente dirección:

<https://market.ionic.io/plugins>

o de Cordova desde:

<https://cordova.apache.org/plugins/?platforms=cordova-android>

- Resources/ - Recursos específicos de las plataformas. En esta carpeta podremos colocar aquellos assets que sean únicos o dependientes de una plataforma en concreto.
- scss/ - Código SCSS que será compilado a la carpeta www/css/
- www/ - Contiene el código fuente principal de nuestra aplicación web: HTML, CSS, JavaScript, imágenes, etc. Esta carpeta es donde tendremos que desarrollar la aplicación web de forma centralizada y después utilizarla para la compilación para las distintas plataformas.
- bower.json - Listado de dependencias y paquetes de Bower.
- config.xml - Contiene la configuración de Cordova (o PhoneGap) con las opciones específicas para cada plataformas de compilación.
- gulpfile.js - Listado de tareas de Gulp.
- ionic.project - Configuración del proyecto de Ionic.
- package.json - Dependencias y paquetes de NodeJS.

Al crear un nuevo proyecto todas las carpetas se encontrarán vacías o incluirán algunos ficheros de ejemplo. Las carpetas Platforms y Resources no vienen por defecto sino que se crearán una vez añadamos la primera plataforma de compilación. A continuación se explica cómo se añade la plataforma Android.

La carpeta www incluye a su vez una serie de subcarpetas y algo de código de ejemplo. Como hemos dicho, esta carpeta es donde tenemos que desarrollar el código principal de nuestra aplicación así que tenemos que conocer bien su estructura:

- css/ - Aquí colocaremos todas las hojas de estilo que se usen en la aplicación.
- Img/ - En esta carpeta almacenaremos las imágenes de nuestro proyecto.

- js/- Contendrá todo el código JavaScript de la aplicación.
- lib/ - Aquí guardaremos todas las librerías que se usen en nuestro código. Por defecto ya viene incluido todo el código de la librería Ionic (Javascripts, CSS, etc.) para que lo podamos cargar desde nuestro proyecto.
- templates/- Esta carpeta viene preparada para almacenar las plantillas o vistas de la aplicación (en algunas versiones no se crea por defecto).
- index.html- Este es el fichero principal que se abrirá al cargar la aplicación. Aquí tendremos que cargar todo lo necesario y mostrar el contenido de la primera pantalla.

4º Arquitectura del proyecto

Ionic, al estar basado en Angular, utiliza el patrón conocido como Vista-Controlador (View-Controller). En este tipo de patrón las diferentes secciones de la interfaz se pueden dividir en distintas vistas hijas o incluso podrían ser vistas hijas que contengan a su vez otras vistas hijas. Los controladores están asociados a estas vistas y se encargan de proporcionar los datos necesarios y la funcionalidad de los diferentes elementos.

En la siguiente imagen se puede ver un esquema de la arquitectura completa que sigue Ionic y Angular:

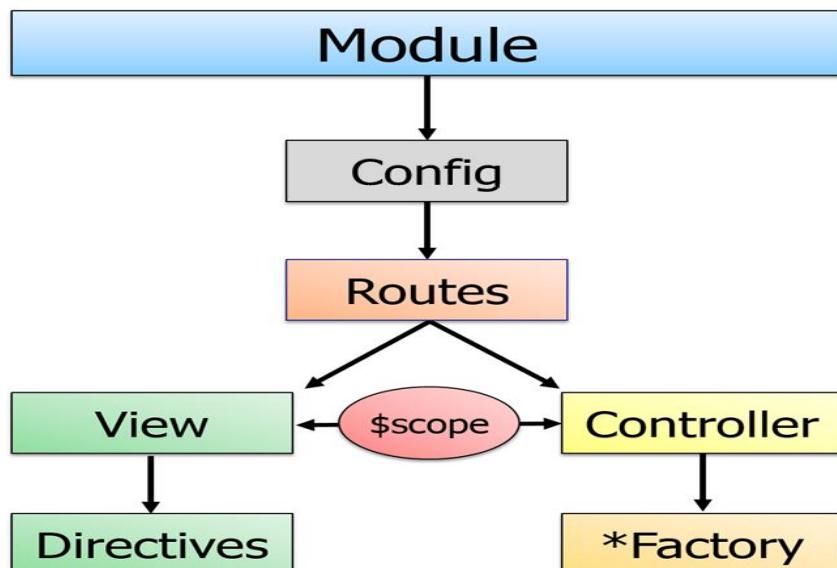


Fig. 53 Arquitectura del proyecto. Recuperado de <https://ajgallego.gitbooks.io/ionic/content/arquitectura.html>

En la arquitectura de una aplicación intervienen muchos tipos de componentes además de las Vistas y Controladores, iremos viendo cada uno de ellos poco a poco, pero principalmente los que nos interesan son las Vistas, Controladores, Servicios o Factorías, y la Configuración y Rutas.

Intuitivamente, la tarea de cada uno de estos componentes en una aplicación con Ionic es la siguiente:

- Los controladores obtienen los datos de uno o varios Servicios o Factorías y lo envían a una vista o template a través de la variable `$scope`.
- Las vistas o templates contienen la descripción visual de una pantalla (o de una parte de una pantalla) y obtienen los datos a mostrar de la variable `$scope`.
- La configuración y las rutas de la aplicación permiten enlazar los controladores con las vistas o templates correspondientes.
- Las directivas permiten crear y usar componentes con aspecto y comportamiento personalizado.

Inicializando la aplicación

El punto de inicio de una aplicación Ionic es el fichero `index.html`, el cual como mínimo deberá tener el siguiente contenido:

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="initial-scale=1, maximum-
scale=1, user-scalable=no, width=device-width">
  <title></title>

  <script src="lib/ionic/js/ionic.bundle.js"></script>
  <script src="lib/ionic-platform-web-
client/dist/ionic.io.bundle.min.js"></script>

  <!-- cordova script (this will be a 404 during development) --
>
  <!-- Cordova is bootstrapped by ionic-platform-web-client,
```

```
uncomment this if you remove ionic-platform-web-client... -->
<!-- <script src="cordova.js"></script> -->
<script src="js/ng-cordova.min.js"></script>

<!-- compiled css output -->
<link href="css/ionic.app.css" rel="stylesheet">
<script src="lib/ionic-timepicker/dist/ionic-
timepicker.bundle.min.js"></script>

<script src="js/app.js"></script>
<script src="js/controllers.js"></script>
<script src="js/routes.js"></script>
<script src="js/services.js"></script>
<script src="js/directives.js"></script>
<script src="js/apiconnector.js"></script>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.
min.js"></script>

<script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBwWRE31IO
QHBkXtanf-dA5bWyxDSoA-Yo&libraries=geometry"
type="text/javascript"></script>

<!-- Only required for Tab projects w/ pages in multiple tabs
<script src="lib/ionicuirouter/ionicUIRouter.js"></script>
-->

</head>

<body ng-app="app" animation="slide-left-right-ios7">
<div>
  <div>
    <!--<ion-nav-bar class="bar-stable">
      <ion-nav-back-button class="button-icon icon ion-
ios-arrow-back">Back</ion-nav-back-button>
    </ion-nav-bar>-->
```

```

<ion-nav-view></ion-nav-view>
</div>
</div>

</body>

</html>

```

En primer lugar en la cabecera del código se cargan las dependencias de la aplicación: hojas de estilo, librería JavaScript de Ionic (la cual incorpora Angular), librería de Cordova y los ficheros JavaScript de nuestra aplicación (`js/app.js`).

A continuación en el `body` se indica el módulo de Angular a utilizar con `ng-app="app"`. Esta línea inicia la carga del módulo llamado `app` que estará definido en el fichero `js/app.js` de la forma:

```
angular.module('app', ['ionic'])
```

Esta es la forma de crear un módulo con Angular, como primer parámetro indicamos el nombre del módulo (`app`) y a continuación las dependencias, que en este caso solo se carga la librería de Ionic. A continuación podremos indicar la configuración, los controladores y servicios que componen el módulo de la forma:

```
angular.module('starter', ['ionic'])
```

```
.config( /* ... */ )
```

```
.controller( /* ... */ )
```

```
.factory( /* ... */ )
```

4.1º Configuración y rutas

La configuración nos permite establecer las rutas o estados (states) que va a tener la aplicación y enlazar cada uno de ellos con una ruta, un controlador y un template (la vista). Además se tendrá que especificar también una ruta inicial o por defecto.

A continuación se incluye un ejemplo de cómo especificar la configuración de un módulo:

```
angular.module('app.routes', [])

.config(function($stateProvider, $urlRouterProvider) {

    // Ionic uses AngularUI Router which uses the concept of
states
    // Learn more here: https://github.com/angular-ui/ui-
router
    // Set up the various states which the app can be in.
    // Each state's controller can be found in controllers.js
$stateProvider
    .state('login', {
        url: '/',
        templateUrl: 'templates/login.html',
        controller: 'loginCtrl'
    })
    .state('mENU.games', {
        url: '/games',
        views: {
            'side-menu21': {
                templateUrl: 'templates/games.html',
                controller: 'gamesCtrl'
            }
        }
    })
    ///página por defecto

    $urlRouterProvider.otherwise('/')

});
```

En este ejemplo se establecen dos rutas o estados, aunque en la aplicación hay muchos más. La primera llamada login, que tendrá la URL “/” y usará el template login.html y el controlador loginCtrl. Y la segunda ruta llamada games que tendrá la URL /games, usará el template games.html y el controlador gamesCtrl. Además se establece que por defecto (en la primera llamada y cuando la ruta no exista) se usará la ruta “/”, que se corresponde con el estado login.

Si decimos que incluya una plantilla que no existe o nos equivocamos en el nombre del fichero html no mostrará ningún error, simplemente no funcionará.

Al realizar una consulta a la aplicación en primer lugar se creará el módulo con esta configuración y a continuación se cargará la ruta solicitada o la ruta por defecto. Si por ejemplo se solicita el estado login se llamará al controlador indicado para preparar los datos y a continuación se cargará el template login.html, se sustituirán los valores de la plantilla con los indicados en el controlador y por último se mostrará la vista al usuario.

Enlaces

Para crear un enlace que nos lleve a un estado indicado en la configuración simplemente tenemos que usar el atributo `ui-sref` con el nombre del estado deseado. Por ejemplo, para volver al login podríamos hacer:

Volver al inicio

Si ponemos un enlace a una dirección que no existe o a un state que no existe no mostrará un error. Al pulsar intentará cambiar de pantalla pero volverá a mostrarse la misma donde estaba.

Rutas con parámetros

También podemos generar rutas con parámetros para pasar valores entre vistas. Por ejemplo, podemos tener un listado de usuarios y al pulsar sobre un elemento de la lista abrir una nueva pantalla con la vista detalle del usuario. Para esto es necesario que el enlace indique el índice del elemento pulsado para así poder abrir la vista con los datos del usuario correspondiente.

Para añadir parámetros a las rutas simplemente tenemos que indicarlos anteponiendo dos puntos (:) al nombre del parámetro en la url, por ejemplo:

```

        controller: 'gameCtrl'
    }
}
}
}
)
)
```

En el ejemplo anterior se definen una ruta o estado. Game recibe un parámetro (`id`) a partir del cual podremos obtener los datos del usuario solicitado.

También podemos definir los parámetros de las rutas usando llaves en lugar de dos puntos. Por ejemplo, para el último estado del ejemplo anterior podríamos haber puesto `/game/{Id}`.

Para generar un enlace a una ruta con parámetros usaremos también el atributo `ui-sref` y simplemente tenemos que añadir al nombre del estado un objeto entre paréntesis con los valores. A continuación se muestra un ejemplo:

```
<a ui-sref="game({Id: id })">Detalles del juego</a>
```

4.2º Controladores

Los controladores podríamos decir que son el equivalente al cerebro de la aplicación, ya que son los que gestionan el flujo de los datos. Al mostrar una página de la aplicación en primer lugar se llama a un controlador, este controlador usará una vista (o template) para generar dicha página y además cargará los datos necesarios mediante servicios (services o factores, que veremos después). El controlador envía los datos necesarios a la plantilla a través de la variable `$scope`, de esta forma desde la vista solo tenemos que mostrar los datos recibidos dándoles el formato adecuado.

Por ejemplo, al acceder a la página `#login` se llamará automáticamente al controlador que tendrá como nombre `loginCtrl`. Este controlador está configurado para usar la plantilla llamada "login.html" con el siguiente contenido:

```

<ion-view title="Login" id="page5" class="photo_login">
    <ion-content padding="true" has-header="false">
        <form class="list" novalidate="novalidate"
id="form_login" name="form_login">
            <a style="display:inline-block;" href="#"
onclick="window.open('http://51.254.97.198/NFU/#/', '_system',
'location=yes'); return false;">
                <div style="text-align:left;">
                    
                </div>
            </a>
            <div class="spacer" style="width: 285px; height:
25px;"></div>
            <div>
                <!--<p>
                    <b class="positive letter">INICIAR SESIÓN:</b>
```

```

        </p>-->
    </div>
    <div class="spacer" style="width: 285px; height: 25px;"></div>
    <div class="spacer" style="width: 285px; height: 25px;"></div>

        <label class="item item-input">
            <input type="text" placeholder="USUARIO" ng-model="login.user" id="user_login" name="user_login" required autofocus="">

            </label>
            <span class="warning" ng-show="form_login.user_login.$error.required && (form_login.user_login.$dirty || form_login.user_login.$touched)">Este campo es requerido</span>
            <span class="warning" ng-show="form_login.user_login.$error.pattern">Usuario no válido</span>
            <span class="text-danger" ng-show="AlertMessage" ng-model="login.user_error">{{login.user_error}}</span>
            <ion-list></ion-list>
            <label class="item item-input">
                <input type="password" placeholder="PASSWORD" ng-model="login.password" name="password_login" id="password_login" required>

                </label>
                <span class="warning" ng-show="form_login.password_login.$error.required && (form_login.password_login.$dirty || form_login.password_login.$touched)">Este campo es requerido</span>
                <span class="warning" ng-show="form_login.password_login.$error.pattern">Password no válido</span>
                <span class="text-danger" ng-show="AlertMessage" ng-model="login.password_error">{{login.password_error}}</span>
                <span class="text-danger" ng-show="AlertMessage" ng-model="login.activar_error">{{login.activar_error}}</span>
                <div class="spacer" style="width: 285px; height: 25px;"></div>

                    <a id="login-button1" class="button button-positive button-block" ng-click="login_submit()" ng-disabled="form_login.$invalid">Iniciar sesion</a>

                    <div class="row">
                        <div class="col"> <a class="facebook-sign-in button button-block button-social" ng-click="facebookSignIn()"><i class="icon ion-social-facebook"></i></a></div>
                        <div class="col"> <a class="facebook-sign-in button button-block button-social" ng-click="facebookSignIn()"><i class="icon ion-social-twitter"></i></a></div>

```

```
        </div>

        <ion-item class="positive " ui-sref="signup">Crear
Cuenta</ion-item>
        <div class="spacer" style="width: 285px; height:
17px;"></div>
        <ion-list>
            <ion-item class="positive " ui-
sref="recover">Recuperar Contraseña</ion-item>
        </ion-list>
    </form>
</ion-content>
</ion-view>
```

Para definir el controlador asociado loginCtrl lo podríamos realizar de la forma:

```
.controller('loginCtrl', function($scope, $localStorage,
$http, $state, $timeout, services, $cordovaGeolocation,
$cordovaFacebook, $window, $ionicPopup, $ionicLoading) {
  $scope.login = {
    user: "",
    password: ""
  }
  var appID = 93713001640368;
  var version = "v2.0"; // or leave blank and default is
v2.0
  $scope.facebookSignIn = function() {
    var jwt =
'eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJqdGkiOiJjZjEzzjEwMS03OD
I3LTQ4N2QtOTI5Zi1iNDQyNGI2MmUxNzQifQ.Vv1PortMkNtSwghUQiup_DszTUD
MY-1KTcz20tpqH64';
    var tokens = $window.localStorage['token'];
    var profile = 'tester';

    // Build the request object
    var req = {
      method: 'POST',
      url: 'https://api.ionic.io/push/notifications',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': 'Bearer ' + jwt
      },
      data: {
        "tokens": tokens,
        "profile": profile,
        "notification": {

          "android": {
            "message": "Hello Android!",
            "sound": "android-sound.wav",
          }
        }
      }
    }
    services.pushNotification(req).then(function(response) {
      $ionicLoading.hide();
      $ionicPopup.alert({
        title: 'Success',
        template: 'Push notification sent successfully'
      });
    }, function(error) {
      $ionicLoading.hide();
      $ionicPopup.alert({
        title: 'Error',
        template: 'There was an error sending the push notification'
      });
    });
  }
})
```

```

        "icon": "ionitron.png",
        "icon_color": "#rrggbb",
        "collapse_key": "foo",
        "tag": "bar"
    },
    "ios": {
        "title": "Howdy",
        "message": "Hello iOS!"
    }
}
};

// Make the API call
$http(req).success(function(resp) {
    alert(resp.data.config.notification.android.message);

    console.log("Ionic Push: Push success", resp);
}).error(function(error) {
    // Handle error
    console.log("Ionic Push: Push error", error);
});
}

var posOptions = {
    timeout: 10000,
    enableHighAccuracy: false
};
$cordovaGeolocation
.getCurrentPosition(posOptions)
.then(function(position) {
    //console.log(position);
    var lat = position.coords.latitude;
    var long = position.coords.longitude;
    $window.localStorage['lat'] = lat;
    $window.localStorage['long'] = long;

}, function(err) {
    alert("No se ha podido localizar su ubicación")
});

var watchOptions = {
    timeout: 3000,
    enableHighAccuracy: false // may cause errors if true
};

var watch =
$cordovaGeolocation.watchPosition(watchOptions);
watch.then(
    null,
    function(error) {
        switch (error.code) {
            case error.PERMISSION_DENIED:
                $scope.error = "User denied the request for
Geolocation."

```

```

        break;
    case error.POSITION_UNAVAILABLE:
        $scope.error = "Location information is
unavailable."
        break;
    case error.TIMEOUT:
        $scope.error = "The request to get user location
timed out."
        break;
    case error.UNKNOWN_ERROR:
        $scope.error = "An unknown error occurred."
        break;
    }
},
function(position) {
    var lat = position.coords.latitude
    var long = position.coords.longitude
});

watch.clearWatch();

$scope.myRegex_name = /^[a-z ñáéíóú]{2,60}$/i;
$scope.myRegex_password = /(?=.*\d)(?=.*[a-z])(?=.*[A-
Z]).{6,}/;
$scope.myRegex_email = /^[a-zA-Z0-9_\.\-]+@[a-zA-Z0-9\-
]+\.[a-zA-Z0-9\-\.\.]+$/;

///////////////////////////////login
user
/////////////////////////////login user
$scope.login_submit = function() {

$ionicLoading.show({
    content: 'Loading',
    animation: 'fade-in',
    showBackdrop: true,
    maxWidth: 200,
    showDelay: 0
});
var SingINuser = JSON.stringify($scope.login);
//alert(SingINuser);
// $window.localStorage['user'] = $scope.login.user;

services.post('login', 'signin', SingINuser)
.then(function(data) {

    if (data.data.success) {
        $ionicLoading.hide();
        /* $scope.login.msg = data.data.message */

        $localStorage.setObject('user', data.data.users);
        var authProvider = 'basic';
        var authSettings = {

```

```

        'remember': true
    };

var loginDetails = {
    'email': data.data.users.email,
    'password': $scope.login.password,
};

$scope.authSuccess = function() {
    // replace dash with the name of your main
state
    $state.go('mENU.games');
};

$scope.authFailure = function(errors) {
    for (var err in errors) {
        // check the error and provide an appropriate
message
        // for your application
    }
};

Ionic.Auth.login(authProvider, authSettings,
loginDetails)
    .then($scope.authSuccess, $scope.authFailure);

// $window.localStorage['user'] =JSON.stringify(
data.data.users);

// console.log(data.data.users);
/*
authService.SetCredentials(data.data.users);*/

/*console.log($cookies.getObject('user'));*/
// 

} else {

$ionicLoading.hide();
$scope.AlertMessage = true;
//console.log(data.data);
$timeout(function() {

    $scope.AlertMessage = false;

}, 5000)

// $scope.login.msg = data.data.error.message;
// alert(data.data.error.user);
$scope.login.user_error = data.data.error.user;
$scope.login.password_error =
data.data.error.password;

$ionicPopup.alert({

```

```

        title: data.data.error.activar,
        template: 'No olvide darlo de alta en su
bandeja de correo'
    );
}
);
}
)
)
```

Los parámetros que recibe la función del controlador serán inyectados por Angular en la llamada. Por lo tanto podemos usar estos parámetros para cargar servicios o clases que necesitemos. En este ejemplo se inyecta la variable `$scope` que como hemos visto nos permite pasar datos a la llamada post para loguearnos en la aplicación.

La clase `$stateParams` nos permite recoger el valor de los parámetros de entrada proporcionados a una ruta. Por lo tanto, en el controlador tenemos que añadir esta clase a sus argumentos (para que Angular la inyecte) y después simplemente accederemos a los parámetros como si fueran propiedades de este objeto. Por ejemplo, si hemos definido en la configuración una ruta para mostrar la vista detalle de un usuario:

```

.config(function($stateProvider, $urlRouterProvider) {
    $stateProvider
        .state('tabs.game', {
            url: '/game/:id',
            views: {
                'game': {
                    templateUrl: 'templates/game.html',
                    controller: 'gameCtrl'
                }
            }
        })
    )
})
```

El parámetro `Id` que recibe esta ruta lo podríamos recoger en el controlador de la forma:

```

.controller('gameCtrl', function($scope, services,
$localStorage, $stateParams, $cordovaAppAvailability,
$ionicLoading, $window, $localStorage, $ionicPopup, $state) {
    //console.log($stateParams);
    $ionicLoading.show();
    services.get('games', 'idGame',
    $stateParams.id).then(function(data) {
        if (data.data.success) {
            $ionicLoading.hide();

            game = data.data.game;
            console.log(game);
            $scope.image = game.deporte;
            $scope.level = game.bajo;
```

```

        $scope.hour = game.hora;
        $scope.day = game.dia;
        $scope.inscrip = game.inscripcion;
        $scope.name_install = game.nombre;
        $scope.name_game = game.nombre_game;
        $scope.seats = game.plazas;
        $scope.location = game.ubicacion;
        $scope.user = game.usuario;
        var players = data.data.jugadores;
        // console.log(jugadores);
        var user = $localStorage.getObject('user').nombre;

        for (property in players) {
            console.log(user);
            console.log(players[property])
            if (players[property] === user) {
                $scope.player = true;
            } else {
                $scope.player = false;
            }
            if (players[property] !== user) {
                $scope.delete_player = true;
            } else {
                $scope.delete_player = false;
            }
        }

    }

$window.localStorage['id'] = game.id;
$localStorage.remove('facility');
console.log($localStorage.getObject('facility'));
$localStorage.setObject('facility', game);
console.log($localStorage.getObject('facility'));
// $localStorage.setObject('jugadores', jugadores
);
} else {
    $ionicLoading.hide();

    $ionicPopup.alert({
        title: data.data.mensaje,
    })
}

});

$scope.play = function() {
    $ionicLoading.show();

    var data = {
        nombre: $localStorage.getObject('user').nombre,

```

```

        id: $stateParams.id,
    }
    //alert(data.id)
    services.post('users', 'play',
data).then(function(data) {

    if (data.data.success) {

        $ionicLoading.hide();
        console.log(data);
        console.log();
        var confirmPopup = $ionicPopup.confirm({
            title: data.data.mensaje

        });

        confirmPopup.then(function(res) {
            if (res) {
                // console.log('ok');
                $state.go('mENU.games');
            } else {
                //console.log('You are not sure');
            }
        });
    } else {
        $ionicLoading.hide();

        $ionicPopup.alert({
            title: data.data.error,
        });
        console.log(data);
    }
});

$scope.Delete = function() {
    $ionicLoading.show();

    var data = {

        nombre: $localStorage.getObject('user').nombre,
        id: game.id,
    }
    services.post('users', 'play_delete',
data).then(function(data) {

        if (data.data.success) {

            $ionicLoading.hide();
            console.log(data);
            console.log();
            var confirmPopup = $ionicPopup.confirm({
```

```

        title: data.data.mensaje

    });

confirmPopup.then(function(res) {
    if (res) {
        // console.log('ok');
        // $state.go('mENU.myprofile');
    } else {
        //console.log('You are not sure');
    }
});
} else {
$ionicLoading.hide();

$ionicPopup.alert({
    title: data.data.error,
});

console.log(data);
}
});
}

})

```

De esta forma podemos acceder a todos los parámetros pasados a través de la ruta. Tenemos que tener cuidado de que el nombre de variable sea el mismo y que esté definido, de otra forma obtendríamos el valor `undefined`.

4.3º Vistas

Las vistas en una aplicación de Ionic se refieren al concepto de "*templates*" según la librería Angular. Por este motivo todas las vistas se guardarán dentro de la carpeta "templates/" en ficheros separados con extensión .html.

Las vistas contendrán la descripción de la parte gráfica de una pantalla, por lo que serán básicamente código HTML mezclado con CSS y también podremos usar otras directivas o componentes de Ionic, como botones, barras, etc. A continuación se incluye un ejemplo de una vista que guardaremos en el fichero /templates/games.html:

```

<ion-view title="Partidas" id="page1">
<ion-content overflow-scroll="true" class="has-header">
<form class="list">
<label class="item item-input" name="SearchGame">
    <i class="icon ion-search placeholder-icon"></i>
    <input type="search" ng-model="filtro"
placeholder="Buscar partidas ">
</label>
<ion-item href="#/tabs/game/{{item.id}}" class="item"
back-img="{{item.deporte}}" ng-repeat="item in items |
filter:filtro| limitTo:listlength">

```

```

<!--
<h2 class="games_color">{{item.ubicacion}}</h2>
<h2 class="games_color">{{item.nombre_game}}</h2>
<div class="row row-center row-bottom">
    <div class="col">
        <h2 class="games_color ion-android-time">
{{item.hora}}</h2></div>
    <div class="col">
        <h2 class="games_color ion-android-person">
{{item.plazas}}</h2></div>
    <div class="col">
        <div class="row row-center row-bottom">
            <div class="col">
                <h2 class="games_color">
{{item.inscripcion}}</h2></div>
                <div class="col">
                    <h2 class="games_color ion-social-euro">
{{item.inscripcion}}</h2>
                </div>
            </div>
        </div>
    </div>
</ion-item>
</form>
<ion-infinite-scroll ng-if="listlength<=items.length"
on-infinite="loadMore()" distance="3%">
</ion-infinite-scroll>
</ion-view>

```

Contenedor o página principal

Pero ¿dónde se muestran estas vistas y el título indicado? Al inicio de esta sección, al incluir el código del fichero `index.html`, dentro de la sección `<body>` añadimos dos directivas para esto:

```

<body ng-app="app" animation="slide-left-right-ios7">
    <div>
        <div>
            <ion-nav-bar class="bar-stable">
                <ion-nav-back-button class="button-icon icon ion-ios-arrow-back">Back</ion-nav-back-button>
            </ion-nav-bar>
            <ion-nav-view></ion-nav-view>
        </div>
    </div>
</body>

```

Como se puede intuir, la directiva `<ion-nav-bar>` mostrará la barra de navegación y además incluirá el título indicado en la vista. Y la directiva `<ion-nav-view>` es donde se incluirá el contenido de la plantilla de la vista actual.

El botón de la barra de título indicado mediante la directiva `ion-nav-back-button` permitirá volver a la página anterior y solo se mostrará cuando se pueda pulsar.

Sustitución de variables

Dentro de la vista o plantilla podemos acceder a los datos proporcionados por el controlador. Para mostrarlos tenemos que usar una plantilla con el formato `{ {user} }` que se sustituirá por el valor de dicha variable. Por ejemplo:

```
<h4 class="games_color grey_game game_margin">{ {user} }</h4>
```

donde `{ {user} }` es una variable que se ha asignado al `$scope` en el controlador de la forma:

```
$scope. user = "Juan";
```

Por lo que al mostrar la vista y una vez sustituidas las variables se podrá leer el texto "Juan".

Es importante que nos fijemos que en la vista tenemos que usar directamente el nombre de la variable mientras que en el controlador las tenemos que añadir al objeto `$scope`.

Bucles

Con Angular es posible crear bucles directamente en la plantilla para repetir un trozo de código. Para esto usaremos el atributo `ng-repeat` en la etiqueta que queremos que se repita. Es importante que nos fijemos en que la etiqueta la hemos de colocar en el mismo atributo a repetir y no fuera.

```
<ion-item href="#/tabs/game/{{item.id}}" class="item" back-
img="{{item.deporte}}" ng-repeat="item in items | filter:filtro |
limitTo:listlength">
    <!---->
    <h2 class="games_color">{{item.ubicacion}}</h2>
    <h2 class="games_color">{{item.nombre_game}}</h2>
    <div class="row row-center row-bottom">
        <div class="col">
            <h2 class="games_color ion-android-time">
                {{item.hora}}</h2></div>
        <div class="col">
            <h2 class="games_color ion-android-person">
                {{item.plazas}}</h2></div>
        <div class="col">
            <div class="row row-center row-bottom">
                <div class="col">
                    <h2 class="games_color">
                        {{item.inscripcion}}</h2></div>
                <div class="col">
                    <h2 class="games_color ion-social-euro"></h2>
                </div>
            </div>
        </div>
    </div>
</ion-item>
```

En este código creamos un bucle a partir de la variable `items`, la cual

contendrá un array de objetos con los datos de las partidas. Esta variable se habrá asignado al `$scope` en el controlador correspondiente de la forma:

```
services.post('users', 'load_coordinates_near_map', data)
  .then(function(data) {
    console.log(data);
    var items = new Array();
    var items = data.data.game;
    if (data.data.success) {
      //console.log(data.data.game);
      //  var items = new Array();

      for (var i = 0; i < items.length; i++) {

        var first_point = new
google.maps.LatLng($window.localStorage['lat'],
$window.localStorage['long']);
        var second_point = new
google.maps.LatLng(items[i].latitud, items[i].longitud);
        var distancia =
google.maps.geometry.spherical.computeDistanceBetween(first_point,
second_point);
        //console.log(items[i]);
        if (distancia < 5000) {
          items[i].i = i + 1;
          //$scope.items.push({ id: $scope.items.length});
          $scope.items.push(items[i]);
          //  console.log($scope.items);
        }
      }

    } else {
      console.log(data);
    }
  });
});
```

Condiciones

Además de sustituir variables y crear bucles Angular también permite crear condiciones mediante el uso de `ng-if/ng-show`. Ambos atributos realizarán la misma acción: mostrar u ocultar el contenido de la etiqueta en la que se encuentren dependiendo de la condición. Entonces, ¿cuál es la diferencia? Pues muy sencilla: con `ng-if` la etiqueta se eliminará del DOM si la condición es falsa, mientras que con `ng-show` la etiqueta simplemente se ocultará (pero permanecerá en el DOM).

El valor pasado a `ng-if/ng-show` puede ser una variable, una expresión o

una llamada a una función, por ejemplo:

```
<span class="text-danger" ng-show="AlertMessage" ng-model="login.activar_error">{{login.activar_error}}</span>
```

Y en el controlador podríamos añadir dicho método al \$scope simplemente haciendo:

```
$Scope.AlertMessage=true; si se quiere añadir.
```

```
$Scope.AlertMessage=false; para no enseñar el valor.
```

4.4º Servicios (Factories/Services)

La capa de datos en una aplicación Ionic o Angular se encarga de proporcionar, como su propio nombre indica, los datos desde un almacenamiento local o desde un servicio externo. Esta capa de datos la proporcionan las clases conocidas como *Servicios o Factories*. Ambas clases son muy similares y nos referiremos a ellas como capa o clase de datos. La diferencia estriba en el valor devuelto, los servicios siempre tienen que devolver un objeto (ya que al injectarlos se les llamará con new), mientras que los *Factories* son más versátiles y podrán devolver lo que queramos ya que simplemente se ejecutará la función que lo define.

Como hemos visto el proceso seguido en una petición es el siguiente: El controlador solicita los datos a la capa de datos para prepararlos y pasárselos a la vista. La capa de datos normalmente definirá una serie de métodos para el acceso a los datos. A continuación se incluye un ejemplo sencillo de una capa de datos:

```
.factory('$localStorage', ['$window', function ($window) {
    return {
        set: function (key, value) {
            $window.localStorage[key] = value;
        },
        get: function (key, defaultValue) {
            return $window.localStorage[key] || defaultValue;
        },
        setObject: function (key, value) {
            $window.localStorage[key] = JSON.stringify(value);
        },
        getObject: function (key) {
            return JSON.parse($window.localStorage[key] || '{}');
        },
        remove: function (key) {
            $window.localStorage.removeItem(key);
        }
    }
}])
```

En este caso el servicio \$localStorage define varios métodos unos modificarán el objeto que se almacenará en localStorage y otros devolverán los parámetros guardados en el localStorage. Siguiendo este esquema podemos añadir todos los métodos que queramos al servicio y definir una API de consulta.

Para usar un servicio desde un controlador en primer lugar hay que solicitarlo en los parámetros para que Angular lo inyecte. Posteriormente, desde dentro del código de la función ya lo podemos usar y acceder a los valores o funciones que hayamos definido en la clase de datos:

```
.controller('gamesCtrl', function($scope, $localStorage) {  
})
```

Consultas asíncronas

Lo más común es que tengamos que consultar esos datos desde un servicio remoto o desde una base de datos. Si utilizamos un servicio que pueda tardar en devolver la respuesta tendremos que trabajar de forma asíncrona para no bloquear la aplicación.

Al realizar una petición asíncrona la función del servicio no podrá devolver el valor directamente, sino que usará la función `then` para ayudarnos con la respuesta asíncrona. Por ejemplo, para hacer una consulta HTTP a una API externa y devolver la respuesta podríamos escribir el siguiente código en el servicio:

```
services.get('installation', 'list_installation')  
.then(function(data) {  
  
    if (data.data.success) {  
        var items_fa = new Array();  
  
        var items_fa = data.data.nom_install;  
  
        console.log(items_fa.length);  
        for (var i = 0; i < items_fa.length; i++) {  
  
            var first_point = new  
google.maps.LatLng($window.localStorage['lat'],  
$window.localStorage['long']);  
            var second_point = new  
google.maps.LatLng(items_fa[i].latitud, items_fa[i].longitud);  
            var distancia =  
google.maps.geometry.spherical.computeDistanceBetween(first_point,  
second_point);  
  
            if (distancia < 5000) {  
                console.log(items_fa[i]);  
                items_fa[i].i = i + 1;  
                $scope.items_fa.push(items_fa[i]);  
            }  
        }  
  
        $scope.currentPage = 1; //current page  
        $scope.entryLimit = 3; //max no of items to display in a  
page  
        $scope.filteredItems = $scope.list.length; //Initially  
for no filter  
        $scope.totalItems = $scope.list.length;*/
```

```
        } else {
            $scope.messageFailure = data.data.type_error;
        }
    });
}
```

4.5º Directivas

Según la definición que da Angular: las directivas son marcadores sobre los elementos del DOM (como los atributos, los nombres de elementos o etiquetas, o las clases CSS) que indican al compilador HTML de Angular (`$compile`) que adjunte un determinado comportamiento a dicho elemento del DOM o incluso que lo transforme por un bloque completo de contenido.

Por clarificarlo aún más, las directivas permiten transformar el aspecto y comportamiento del código HTML por otro que definamos nosotros. En Ionic, todas las etiquetas que empiezan con `ion-` son directivas de Angular que tienen un comportamiento y un aspecto asociado. Por ejemplo, la etiqueta:

```
<ion-list>
```

Esta etiqueta se procesa mediante una directiva de Angular y genera el código y comportamiento correspondiente a un listado. Y para que esto funcione, en el código de Ionic hay una directiva que está configurada para gestionar cualquier elemento con el nombre `ion-list`.

En el ejemplo el nombre de la directiva indica `ionList`, pero Angular lo transformará y procesará tanto el nombre `ionList` como `ion-list`. Otro ejemplo, si creamos una directiva para `ionTab` se tendrá en cuenta tanto `ionTab` como los elementos con nombre `ion-tab`.

5º Añadir plataforma

En Ionic (con la ayuda de Cordova) permite generar código para multitud de plataformas, entre ellas están Android, iOS, Amazon Fire OS, Blackberry 10, navegador, Firefox OS, Ubuntu, WebOS, Windows Phone 8 y Windows. Las posibles plataformas para las que podemos generar dependerán del sistema operativo que utilicemos, por ejemplo, para iOS solo podremos compilar desde un Mac en nuestro caso hemos elegido que la aplicación se desarrolle para Android.

Por tanto añadiremos la plataforma Android al proyecto.

```
$ionic platform add android
```

6º Compilar y ejecutar el proyecto en un navegador, emulador o dispositivo real.

Para ejecutar el proyecto tenemos tres opciones: abrirlo en el navegador, emularlo o ejecutarlo en un terminal real. Según lo que queramos hacer nos puede interesar una opción u otra. Abrirlo en una navegador sería la opción más rápida y la más recomendable durante el desarrollo. Sin embargo, si usamos características nativas del dispositivo y queremos probarlas no podremos usar el navegador y tendremos que abrirlo en un emulador o dispositivo real.

6.1º Abrir en el navegador

Para abrir nuestro proyecto en un navegador para poder ver y depurar lo que vamos haciendo, simplemente tenemos que ejecutar el siguiente comando en un terminal en la raíz del proyecto:

```
$ ionic serve
```

Este terminal lo tenemos que dejar abierto mientras que estemos trabajando para que funcione el servidor. Al ejecutarlo nos mostrará el siguiente texto por pantalla:

```
Running dev server: http://localhost:8100
Ionic server commands, enter:
  restart or r to restart the client app from the root
  goto or g and a url to have the app navigate to the
given url
  consolelogs or c to enable/disable console log
output
  serverlogs or s to enable/disable server log output
  quit or q to shutdown the server and exit
```

Como se puede ver en las instrucciones el servidor se creara en la URL <http://localhost:8100/>.

Automáticamente se nos tendría que abrir una ventana del navegador en esta dirección, si no se abre también podemos poner la dirección manualmente.

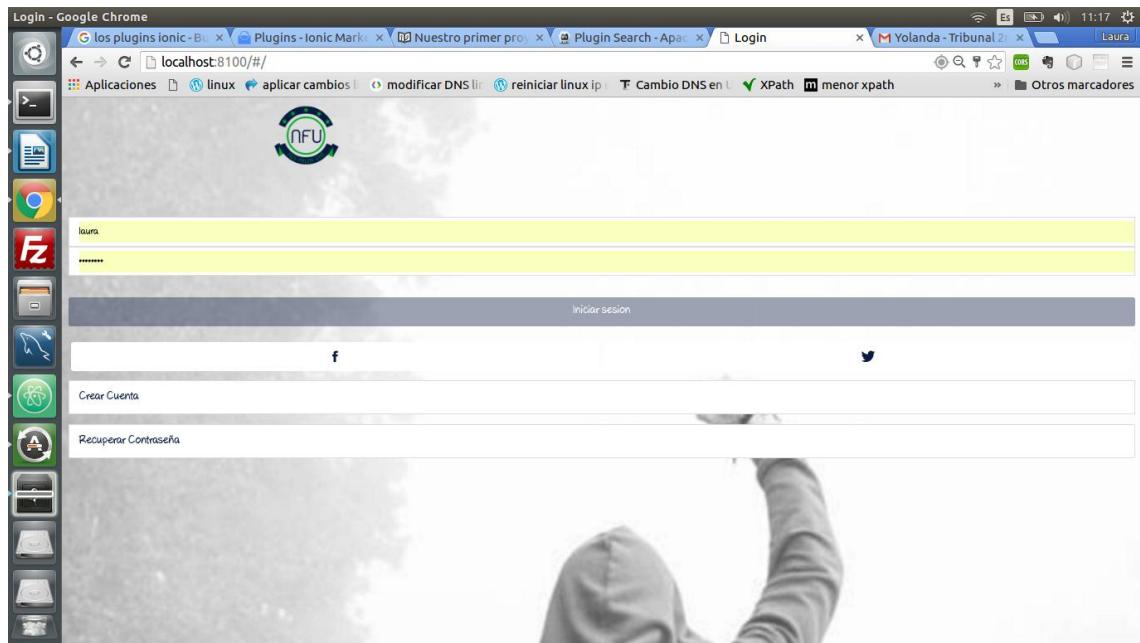


Fig. 54 Pantalla navegador con Ionic Serve. Fuente: elaboración propia

Con cada cambio que hagamos en el código del proyecto se recargará automáticamente el contenido del navegador. Si no funcionara por cualquier razón podéis escribir `restart` o `r` en el terminal.

Para parar el servidor simplemente tendremos que escribir `quit` o `q` en el terminal.

Como opción podemos añadir el parámetro `-lab` en la llamada (`$ ionic serve --lab`) que nos mostrará como quedaría la aplicación a la vez para Android y para iOS.

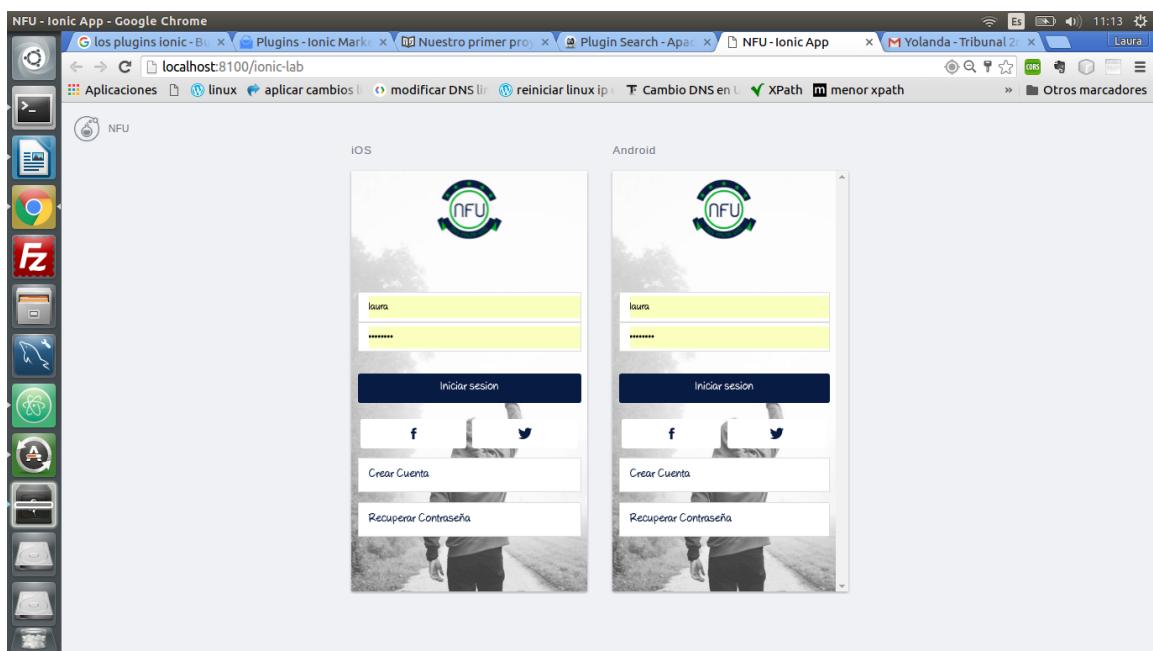


Fig. 55 Pantalla navegador con Ionic Serve - -lab. Fuente: elaboración propia

6.2º Emular o instalar en un dispositivo real

Una de las opciones que nos da ionic es poder ver la aplicación mediante un emulador.

Para ello primeramente deberemos instalar Oracle JDK 8:

```
$sudo apt-get install python-software-properties
```

Añadimos las librerías al repositorio

```
$sudo add-apt-repository ppa:webupd8team/java
```

```
$sudo apt-get update
```

```
$sudo apt-get install oracle-java8-installer
```

Después se debe instalar SDK android

Descargo el sdk android para linux de la siguiente página:

<http://developer.android.com/sdk/index.html#Other>

Descomprimo el archivo,

y en cd /android-sdk/tools/

ejecutamos el archivo ./android

Y seleccionamos la api que queremos instalar etc.

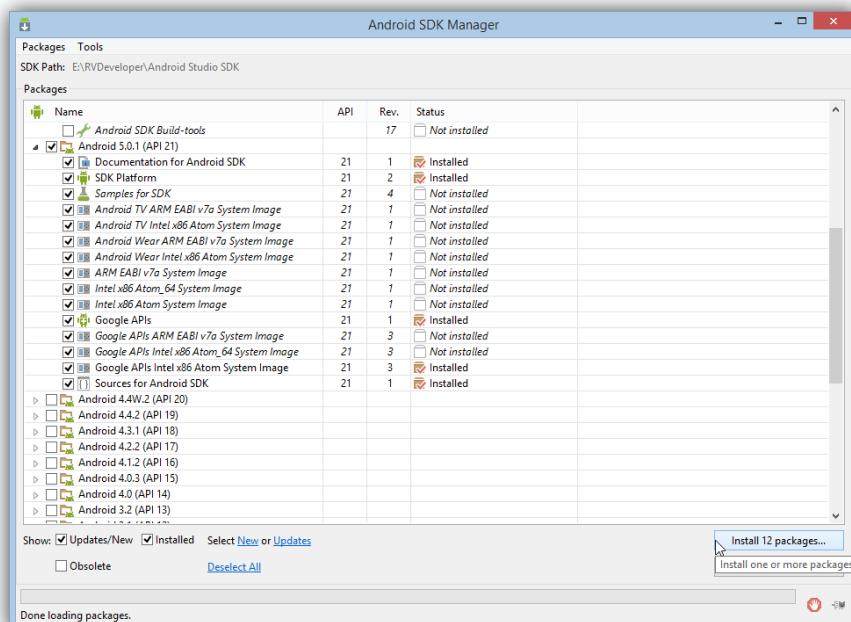


Fig. 56 Selección de API a instalar. Fuente: elaboración propia

Una vez realizamos estos pasos emulamos la aplicación

```
$ ionic emulate android
```

O

```
ionic run android
```

Uno de los errores que he obtenido al crear la aplicación mediante este último método era que el SDK no obtenía bien la dirección de la carpeta tools dentro del sdk.

Una corrección fue añadiendo la siguiente expresión:

```
export ANDROID_HOME=/home/laura/android-sdk-linux
```

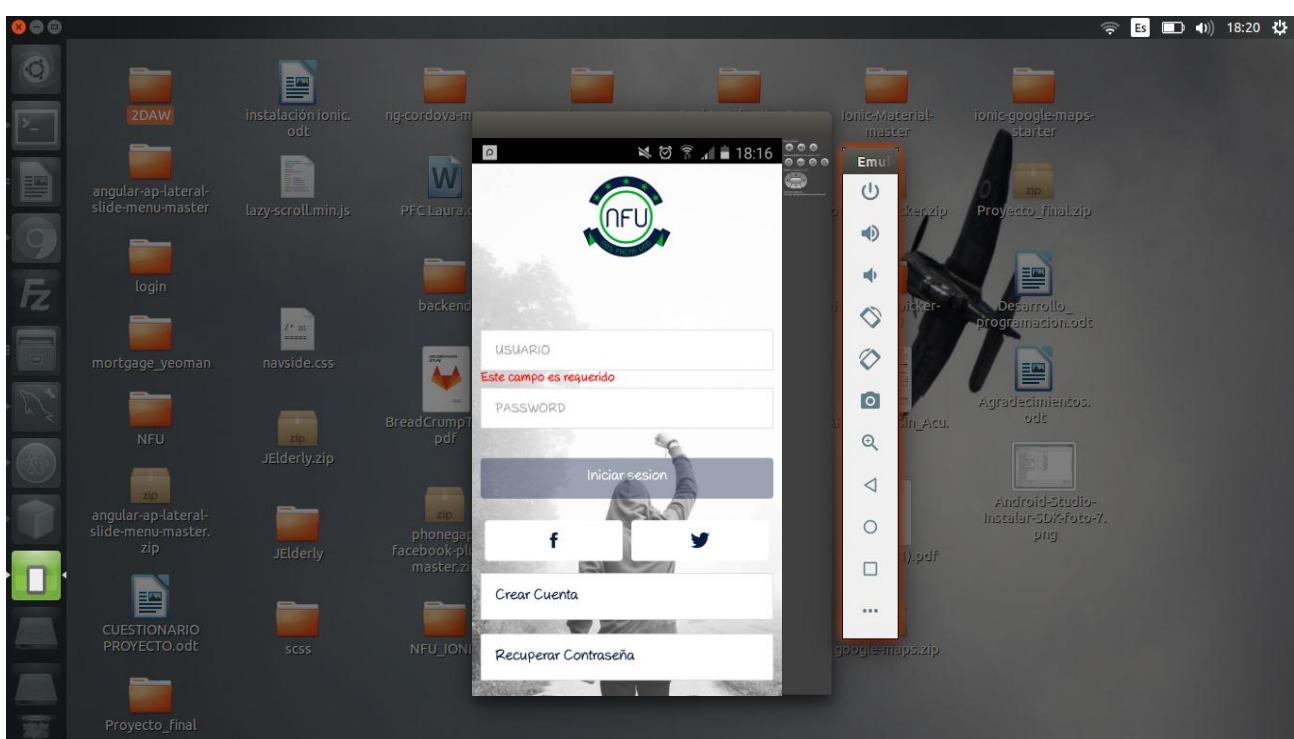


Fig. 57 Emulador. Fuente: elaboración propia

7º Trabajar con SASS

Se ha decidido trabajar en sass para ello se debe indicar en el proyecto mediante el siguiente comando.

```
$ionic setup sass
```

Ejemplos de sass en el proyecto.

A continuación se explica un ejemplo de uso de Sass en el proyecto, se añade una font-family.

Sass trabaja en ionic con un archivo `_variables.scss` donde se asignan todas las variables que se utilizarán para el diseño de la aplicación.

Este archivo lo podemos encontrar en `libs/ionic/scss/`

Fuera de la carpeta `/www` se encuentra otra carpeta `/scss` donde se asignan los valores que esas variables van a utilizar.

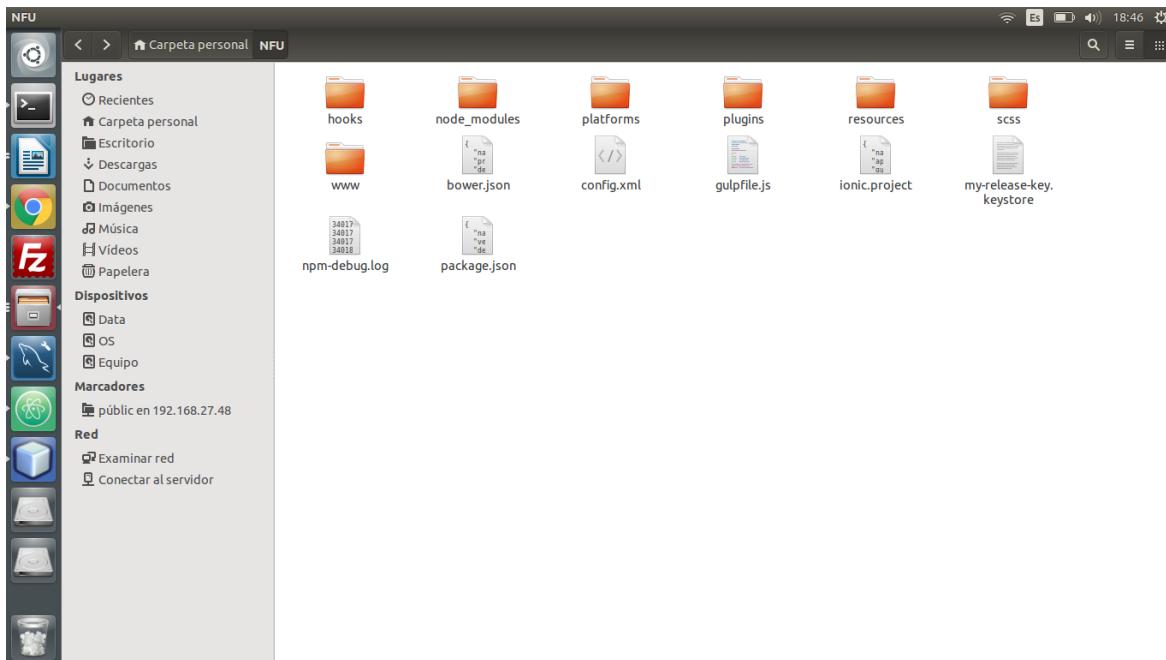


Fig. 58 Directorio. Fuente: elaboración propia

```
$font-family-axure: "../lib/ionic/fonts" !default;  
@font-face {  
    font-family: axure;  
    font-style: normal;  
    font-weight: normal;  
    src: url("../lib/ionic/fonts/axure.ttf");  
}
```

Se añade el font-famliy axure, para ello se indica donde se encuentra la fuente en la variable \$font-family-axure.

```
ionic.app.scss x

Slight:           #ffff !default;
Sstable:          #f8f8f8 !default;
Spositive:        #387ef5 !default;
Salm:             #1c1c1f3 !default;
Sbalanced:        #33cd5f !default;
Senergized:       #ffcc900 !default;
Sassertive:       #e6f473a !default;
Sroyal:           #886aea !default;
Sdark:            #4444 !default;
*/
Spositive: #081B42 !default;
Sstable: #00AF27 !default;
Slight: #fff !default;
Sbutton-stable-text:#ffff !default;
$energized:red !default;

// The path for our ionicons font files, relative to the built CSS in www/css
$ionicons-font-path: "../lib/ionic/fonts" !default;
$font-family-axure: "../lib/ionic/fonts" !default;
$font-family-zoloft: "../lib/ionic/fonts" !default;
@font-face {
    font-family: Zoloft;
    font-style: normal;
    font-weight: normal;
    src: url("../lib/ionic/fonts/zolofn_.ttf");
}

@font-face {
    font-family: axure;
    font-style: normal;
    font-weight: normal;
    src: url("../lib/ionic/fonts/axure.ttf");
}

// Include all of Ionic
@import "www/lib/ionic/scss/ionic";
```

Fig. 59 Archivo .scss. Fuente: elaboración propia

Nos desplazamos al archivo `_variables.scss` y le asignamos a la variable que utiliza las fuentes en la aplicación la variable de la fuente que le asignamos:

```
$font-family-sans-serif: $font-family-axure!default;
```

```
$font-family-light-sans-serif: $font-family-axure!default;
```

```
$font-family-monospace: monospace !default;
```

\$font-family-wp-base: font-family-axure!default;

`$font-family-base: $font-family-sans-serif`

```

variables.scss — /home/laura/NFU/www — Atom
└── _variables.scss
    ├── controllers.js
    ├── apiconnector.js
    ├── yourgames.html
    ├── routes.js
    ├── newGame.html
    ├── login.html
    ├── _backdrop.scss
    ├── game.html
    ├── games.html
    └── facility.html

17 // -----
18
19 $font-family-sans-serif:           $font-family-axure !default;
20
21 $font-family-light-sans-serif:     $font-family-axure !default;
22 $font-family-monospace:           monospace !default;
23 $font-family-wp-base: "axure" !default;
24 $font-family-base:                $font-family-sans-serif !default;
25 $font-size-base:                 14px !default;
26 $font-size-large:                18px !default;
27 $font-size-small:                11px !default;
28
29 $line-height-base:               1.428571429 !default; // 20/14
30 $line-height-computed:           floor($font-size-base * $line-height-base) !default; // ~20px
31 $line-height-large:              1.33 !default;
32 $line-height-small:              1.5 !default;
33
34 $headings-font-family:          $font-family-base !default;
35 $headings-font-weight:           500 !default;
36 $headings-line-height:           1.2 !default;
37
38 $base-background-color:          #fff !default;
39 $base-color:                     #000 !default;
40
41 $link-color:                    $positive !default;
42 $link-hover-color:              darken($link-color, 15%) !default;
43
44 $content-padding:               10px !default;

```

No results found for 'signup'

signup

Replace in current buffer

lib/ionic/scss/_variables.scss 21:62

Finding with Options: Case Insensitive

no results Find . * Aa ⌘F Replace ⌘R Replace All ⌘H

LF UTF-8 SCSS ⌘P master ⌘S+5,5 ⌘U 3 updates

Fig. 60 Archivo _variables.scss. Fuente: elaboración propia

Y así asignamos a toda la aplicación mediante una variable la font-family seleccionada.

8º Plugins instalados

Cordova Whitelist

Es un modelo de seguridad que controla el acceso a dominios externos sobre los que su aplicación no tiene control. Córdoba ofrece una política de seguridad configurable para definir a qué sitios externos se puede acceder. Por defecto, las nuevas aplicaciones están configuradas para permitir el acceso a cualquier sitio. Antes de trasladar la aplicación a la producción, se debe formular una lista blanca y permitir el acceso a los dominios y subdominios específicos de la red.

Para Android y iOS (a partir de sus versiones 4.0), la política de seguridad de Cordova requiere de este plugin para acceder a un recurso externo. Por defecto en las últimas versiones de ionic cualquier petición está bloqueada.

Al descargar Ionic, el plugin Cordova Whitelist, viene instalado.

Geolocalización

Este plugin permite obtener la posición GPS actual del usuario en cuestión. La línea de código para su instalación es la siguiente:

```
$Cordova plugin add org.apache.cordova.geolocation
```

Este plugin de Cordova/Phonegap provee información útil sobre la localización física del dispositivo, por ejemplo, la latitud o la longitud. El funcionamiento es el

siguiente:

El api se basa en tres métodos del objeto geolocation:

- geolocation.
- geolocation.watchPosition
- geolocation.clearWatch.
-

Con `getCurrentPosition()` se obtienen las coordenadas del dispositivo desde el que se lanza. Este método recibe tres parámetros: una función callback en caso de que haya salido bien; otra para cuando sale mal y el tercero son las opciones (que son opcionales).

Las opciones que tenemos disponibles son:

- enableHighAccuracy: valor booleano, true o false, para indicar si se quiere o no que sea lo más preciso posible, lo cual ralentiza o acelera el proceso.
- timeout: valor en milisegundos que debe aguardar esperando una respuesta hasta que dé error en caso de no haberla recibido.
- maximumAge: Con valor 0, se calcula la posición cada vez que se pide; con un valor en milisegundos se coge de la caché si hay alguna que no sea más antigua que el valor que se indica aquí.

FileTransfer

El FileTransfer proporciona una manera de cargar archivos usando HTTP mediante POST o PUT, y también para descargar archivos.

```
$ cordova plugin add org.apache.cordova.file-transfer
```

Métodos:

- upload: Envía el archivo al servidor.
- download: Descarga el archivo del servidor.
- abort: Aborta el proceso de descarga o envío al servidor

CordovaImagePicker

Plugin que te permite escoger una imagen o varias de la carpeta imágenes donde se almacenan.

```
$ cordova plugin add https://github.com/wymsee/cordova-imagePicker.git
```

Método:

```
getPictures(options);
```

\$cordovaCamera

Se utiliza para poder tomar fotos desde la propia cámara del móvil.

nota: El api de la cámara solo funciona en el móvil no funciona en el emulador.

```
$ cordova plugin add cordova-plugin-camera
```

Método:

```
getPicture(options);
```

\$cordovaKeyboard

Viene por defecto instalado en el ionic.

Es para acceder al teclado iOS, por tanto a nosotros no nos afecta.

InAppBrowser

Para poder registrarnos por twitter o facebook necesitamos este plugin.

```
$cordova plugin add cordova-plugin-inappbrowser
```

9º Panel de control (DASHBOARD)

A continuación pasamos a describir el panel de control que nos ofrece Ionic.

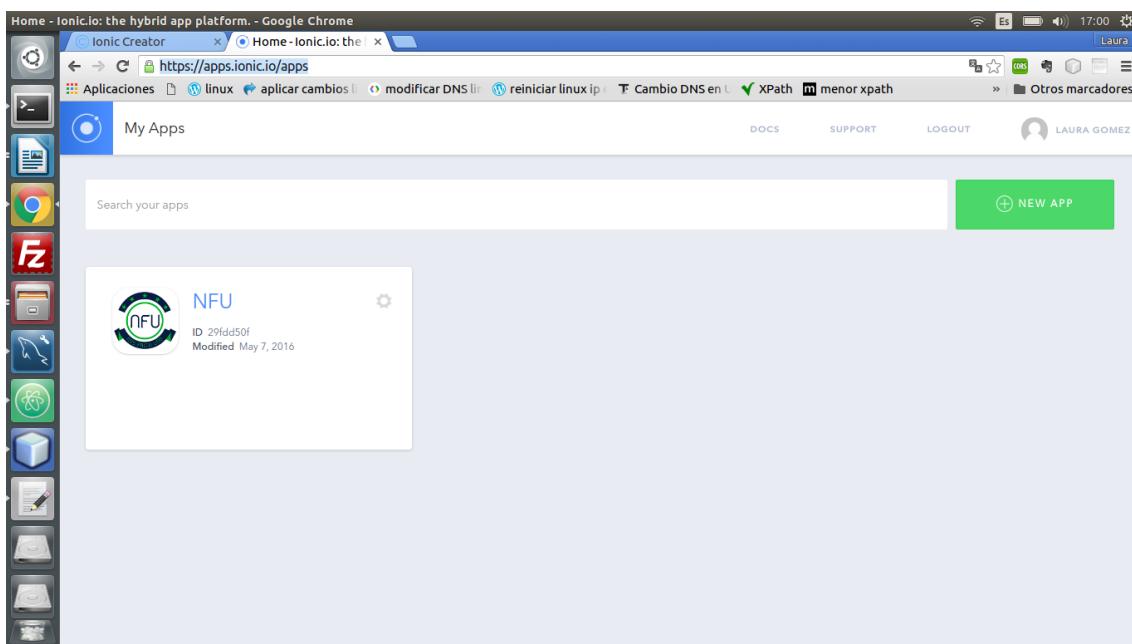


Fig. 61 Panel de control de Ionic Account. Fuente: elaboración propia

Clickamos en NFU y entramos al panel de control:

En el menú lateral podemos encontrar:

- Notificaciones Push.
- Analytics
- Deploy
- Package
- Users
- Settings

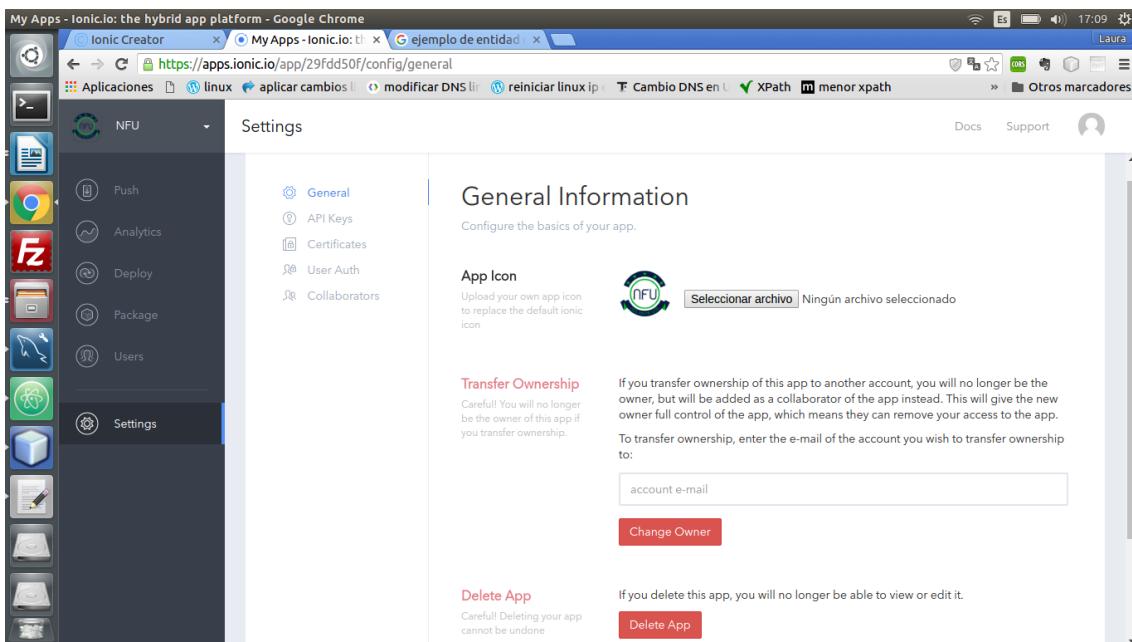


Fig. 62 Settings. Fuente: elaboración propia

9.1º Settings

General

Desde Settings General podemos borrar una aplicación, cambiar de propietario o añadir un ícono para la aplicación.

Api Keys

Api Keys, genera Token se utiliza para el envío de notificaciones push Ionic crea una api Key pública y secreta por defectos.

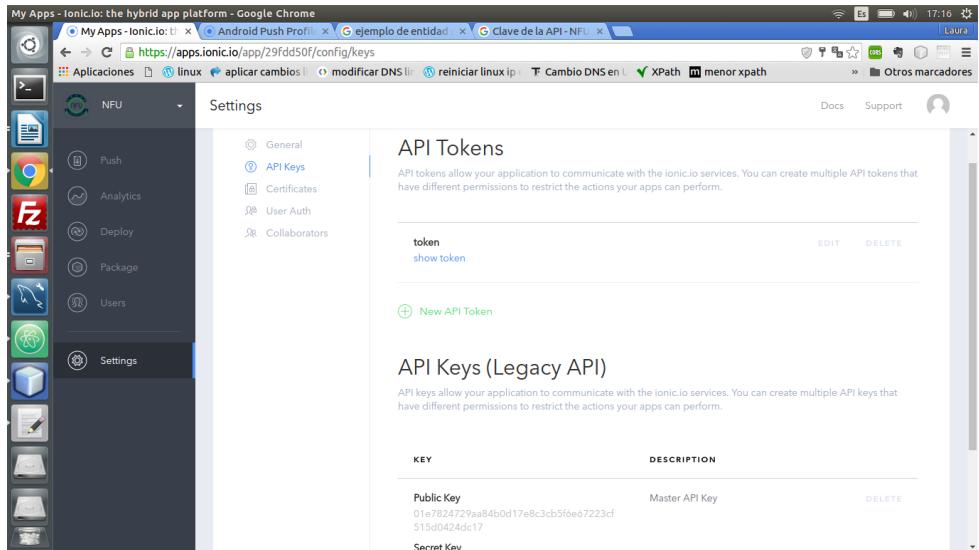


Fig. 63 API Keys. Fuente: elaboración propia

Certificados

Los certificados se pueden crear de producción o desarrollo.

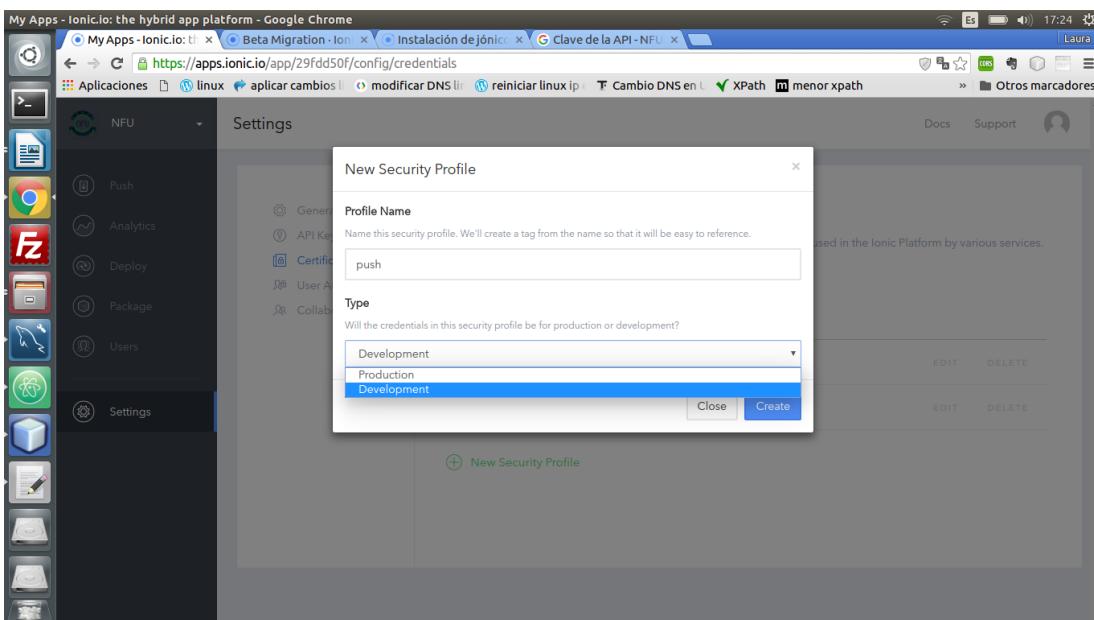


Fig. 64 Creación de certificado. Fuente: elaboración propia

Este certificado también se crea para poder enviar notificaciones push:

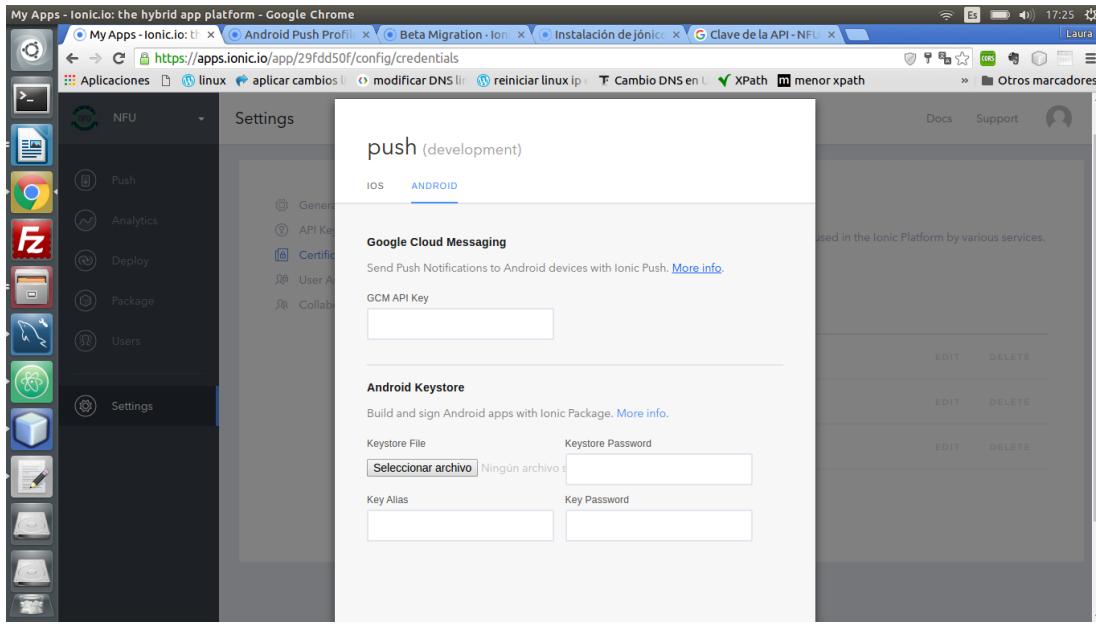


Fig. 65 Certificados. Fuente: elaboración propia

Como se puede observar se necesita la api key GCM de google.

A continuación se muestran los pasos para configurar El GCM.

PASO 1: CONFIGURACIÓN DE GCM

1.1: Creación de un proyecto API Google

- Abra la Consola para Google.
- Si no ha creado un proyecto de API todavía, haga clic en **Crear proyecto**.
- Suministrar un nombre de proyecto y el ID, haga clic en **Crear**.

Una vez que se ha creado el proyecto, aparecerá una página que muestra la identificación del proyecto y el número de proyecto como se ve a continuación. Copiar abajo de su número de proyecto . Lo utilizará más adelante como el ID del remitente GCM.

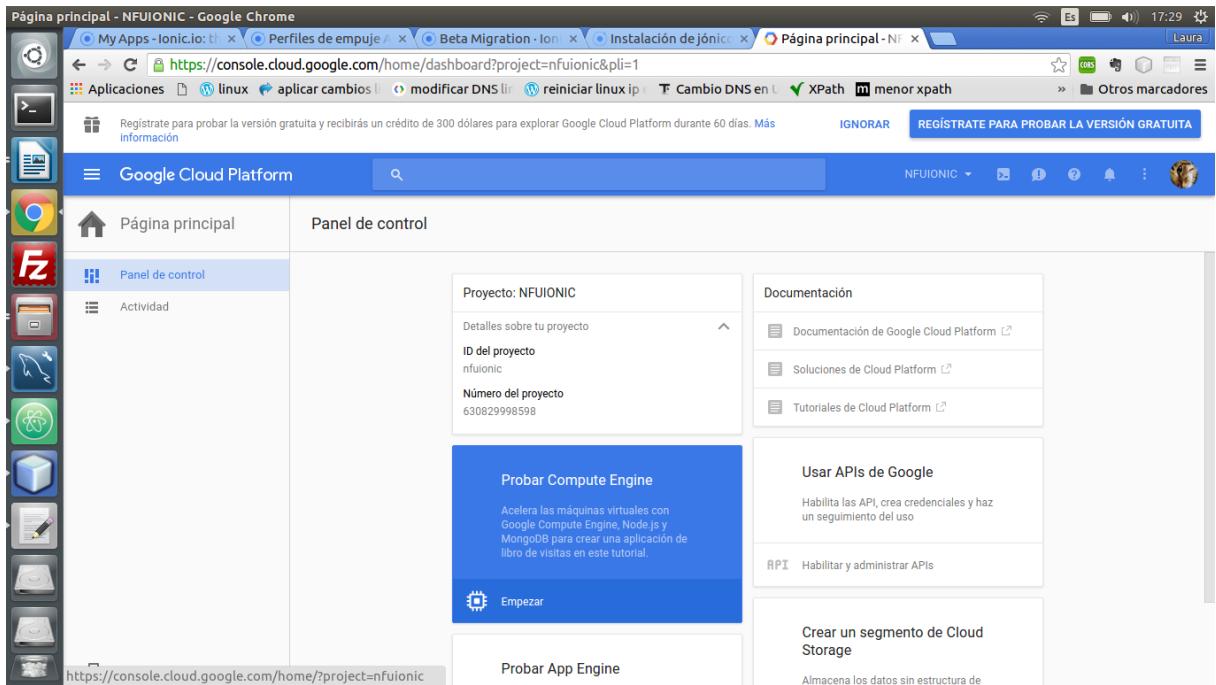


Fig. 66 Panel de control de la consola de Google. Fuente: elaboración propia

1.2: Habilitar el servicio GCM

- En el de uso Google API sección, seleccione Habilitar y gestionar API.
- En la lista que aparece de APIs, encontrar la mensajería en la nube Google para Android enlace.
- Haga clic en Habilitar API.

Una vez que se ha creado el proyecto, aparecerá una página que muestra la identificación del proyecto y el número de proyecto como se ve a continuación. Copiar abajo de su número de proyecto. Lo utilizará más adelante como el ID del remitente GCM.

1.3: La obtención de una clave de API

- Navegue hasta la Credenciales sección en el menú lateral de la izquierda.
- Seleccione Crear credenciales, a continuación, elija clave de API, a continuación, servidor de claves.
- El nombre de su clave de API y dejar las Aceptar peticiones de... campo en blanco.

- Haga clic en Crear. Debe ver a su clave de API de nueva creación como a continuación.
- **Guardar su clave de API** Vas a necesitar más adelante para conectar Ionic.io.

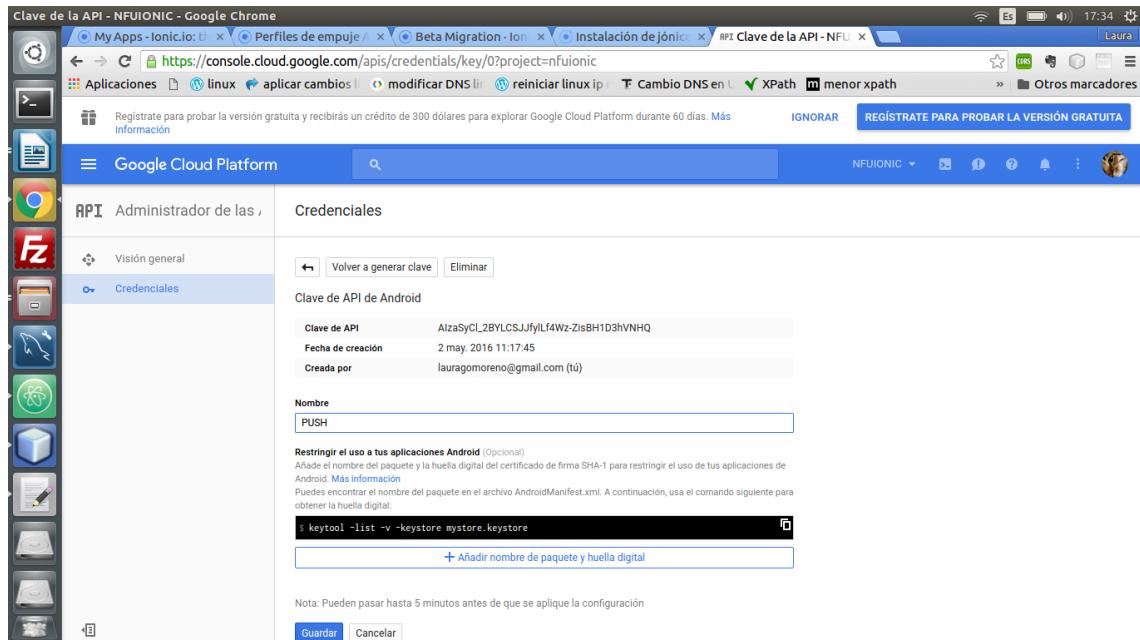


Fig. 68 API de Google. Fuente: elaboración propia

PASO 2: CONEXIÓN DE IONIC.IO

Lo primero es lo primero, es necesario configurar el número de proyecto dentro de su ionic aplicación.

```
$ionic config set gcm_key<630829998598>
```

Ahora, haga clic en el Android ficha y encontrar la sección marcada Google mensajería en la nube, introduzca la clave de API que ha generado en el Google consola de desarrollo, a continuación,

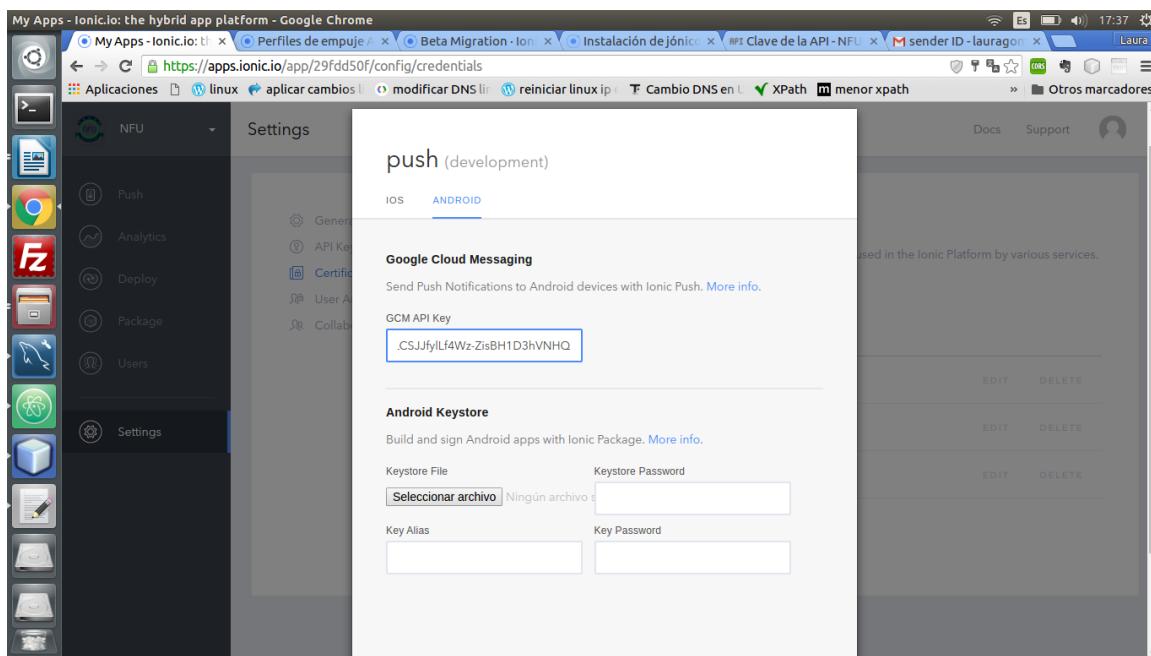


Fig. 69 Creación de certificado Android. Fuente: elaboración propia

User Auth

User Auth, en la versión Ionic 2 la cual se está migrando ahora requiere una autenticación por parte del usuario.

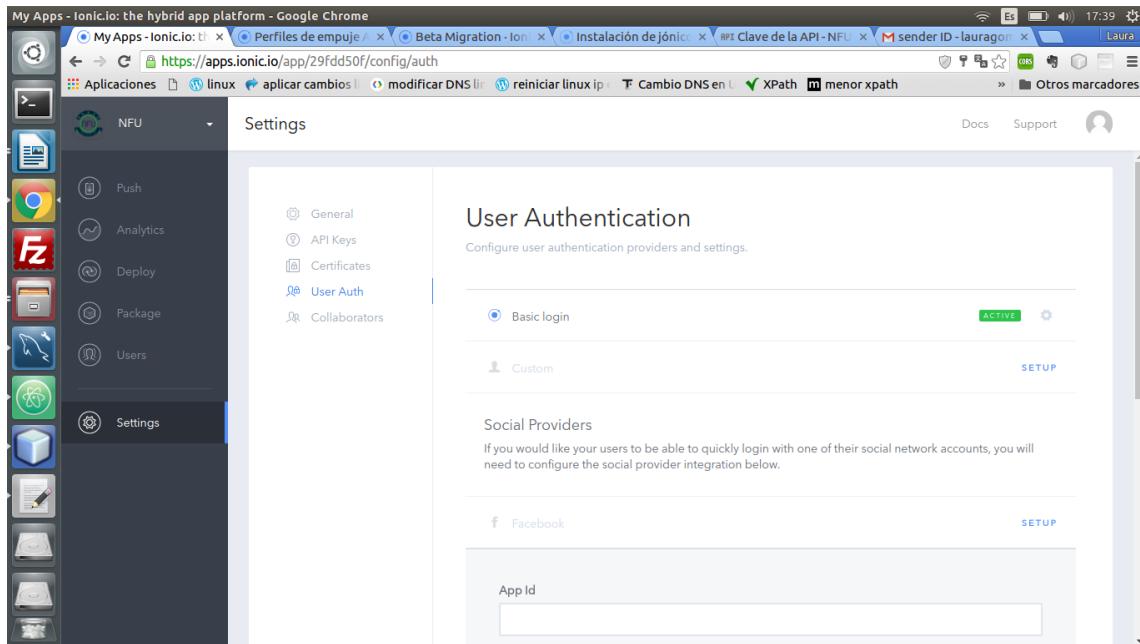


Fig. 70 User Auth. Fuente: elaboración propia

Para luego poder ver todos los usuarios registrados, si clickamos en el menú y seleccionamos Users.

A screenshot of the 'Users' list page from the Ionic.io platform. The left sidebar includes the 'Users' icon. The main area displays a table with two rows of user information. The columns are 'USER' and 'CREATED'. The first row shows a user named 'pepe' with an email 'boxnia@gmail.com' and a creation date of '2016-05-18 12:36 PM'. The second row shows a user named 'laura' with an email 'lauragomreno@gmail.com' and a creation date of '2016-05-10 5:42 PM'. Each row has 'VIEW' and 'DELETE' buttons at the bottom right.

Fig. 71 Users. Fuente: elaboración propia

En esta pantalla podemos observar todo los usuarios registrados.

En la aplicación NFU hay dos registros, uno lo realizamos mediante Ionic, y el otro lo se realiza en nuestra base de datos.

Login user mediante Ionic:

```
var loginDetails = {  
    'email': data.data.users.email,  
    'password': $scope.login.password,  
};  
  
$scope.authSuccess = function() {  
    // replace dash with the name of your main state  
    $state.go('mENU.games');  
};  
  
$scope.authFailure = function(errors) {  
    for (var err in errors) {  
        // check the error and provide an appropriate  
        message  
        // for your application  
    }  
};  
  
Ionic.Auth.login(authProvider, authSettings,  
loginDetails)  
.then($scope.authSuccess, $scope.authFailure);
```

Logout de la aplicación mediante Ionic:

```
Ionic.Auth.logout();
```

Create user mediante Ionic:

```
var details = {  
    'name': $scope.signup.name,  
    'email': $scope.signup.email,  
    'password': $scope.signup.password  
}  
  
Ionic.Auth.signup(details);
```

También se puede comprobar si un usuario está autenticado mediante:

```
var user = Ionic.User.current();
```

Con `Ionic.User.current()`, obtenemos el usuario actual de la aplicación.

```
if (user.isAuthenticated()) {  
  
    / the user is already signed in  
} else {  
    // we need to show a login or sign up form  
}
```

9.2º Package

Hay dos formas de construir una apk.

La primera de ellas es mediante el panel de control introduciendo el siguiente comando:

```
$ionic package build android
```

Crea la apk y la puedes descargar desde el panel de control.

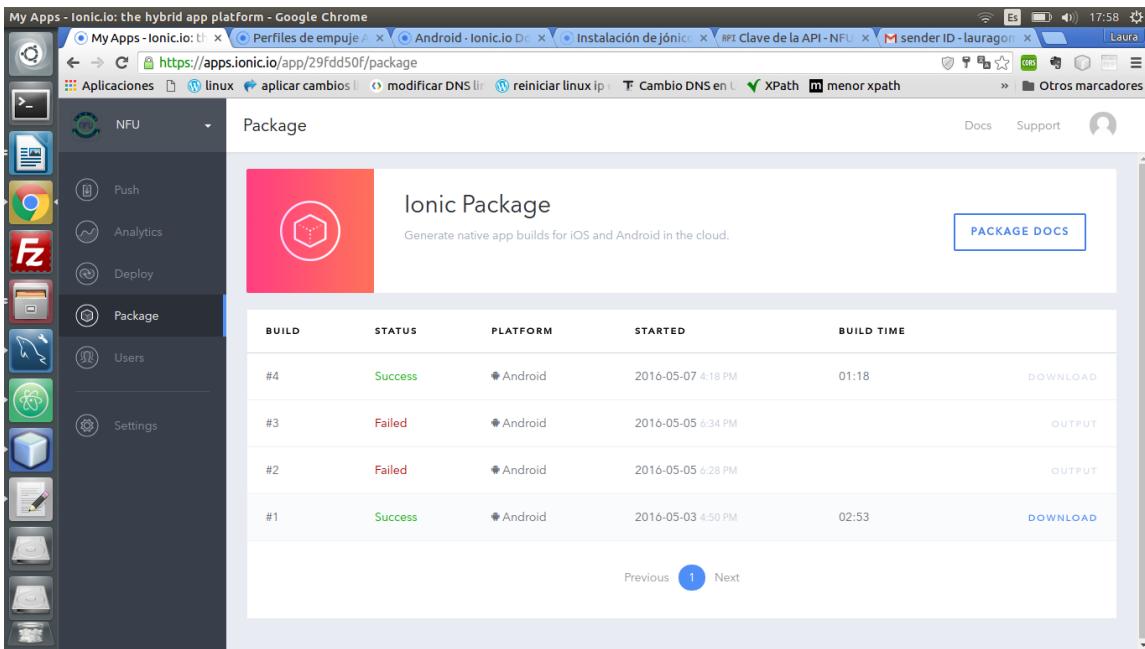


Fig. 72 Ionic Package. Fuente: elaboración propia

Y la segunda de ellas es construyendo directamente la apk:

```
$ionic build android
```

Y puedes encontrar la apk

```
cd /home/laura/NFU/platforms/android/build/outputs/apk/android-debug.apk
```

9.3º Deploy

Ionic tiene una aplicación “ionic view app” que se puede descargar del app store o del google play.

<http://view.ionic.io/>

El enlace para descargartela del google play:

<https://play.google.com/store/apps/details?id=com.ionic.viewapp>

En esta aplicación puedes subir cualquier aplicación creada en Ionic y comprobar toda su funcionalidad sin tener que instalar la apk en tu móvil mediante el siguiente comando.

```
$ionic upload
```

Cuando escribes el comando te solicita el email de registro y el password del ionic.account.es para subirlo a la sección deploy.

```

laura@laura-X550LD: ~/NFU
laura@laura-X550LD-$ cd NFU/
laura@laura-X550LD:~/NFU$ ionic upload
No previous login existed. Attempting to log in now.

To continue, please login to your Ionic account.
Don't have one? Create a one at: https://apps.ionic.io/signup

Email: lauragomoreno@gmail.com
Password:
Logged in! :)
Uploading app....
Saved app_id, writing to ionic.io.bundle.min.js...
Successfully uploaded (29fdd50f)

Share your beautiful app with someone:
$ ionic share EMAIL

Saved api_key, writing to ionic.io.bundle.min.js...
laura@laura-X550LD:~/NFU$ 

```

Fig. 73 Upload a Deploy. Fuente: elaboración propia

Una vez realizado estos pasos, puedes comprobar que también se ha registrado la subida de la aplicación en la sección Deploy y además que la aplicación se encuentra en Ionic view.

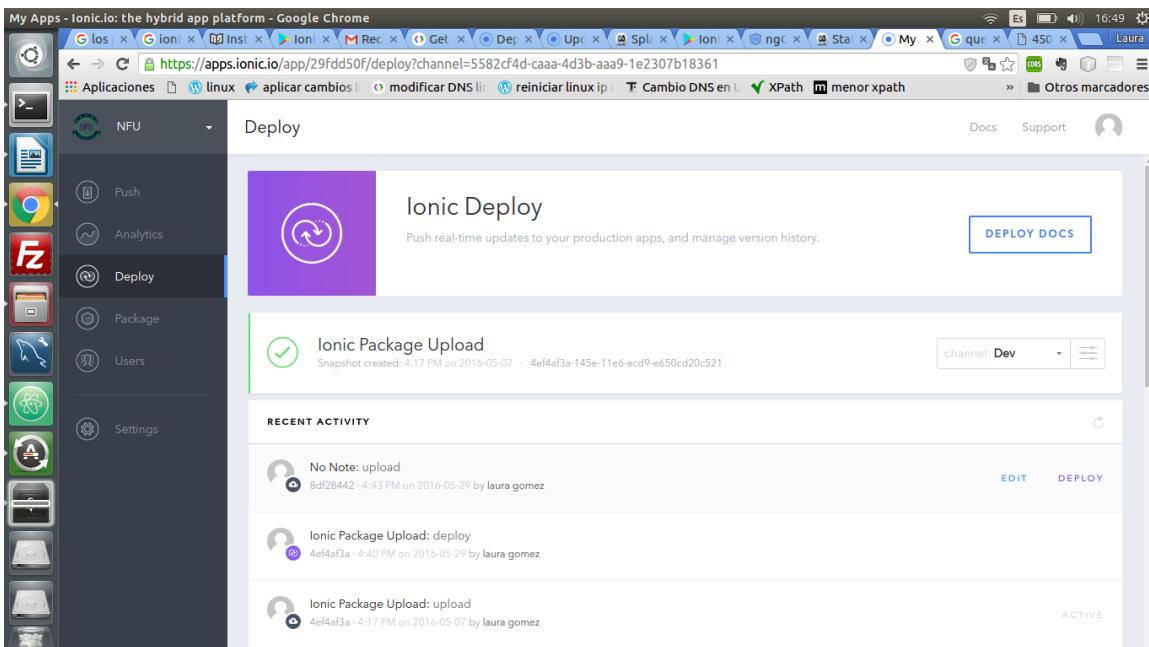


Fig. 74 Ionic Deploy. Fuente: elaboración propia

En edit, puedes añadir los cambios realizados en esa versión de la aplicación:

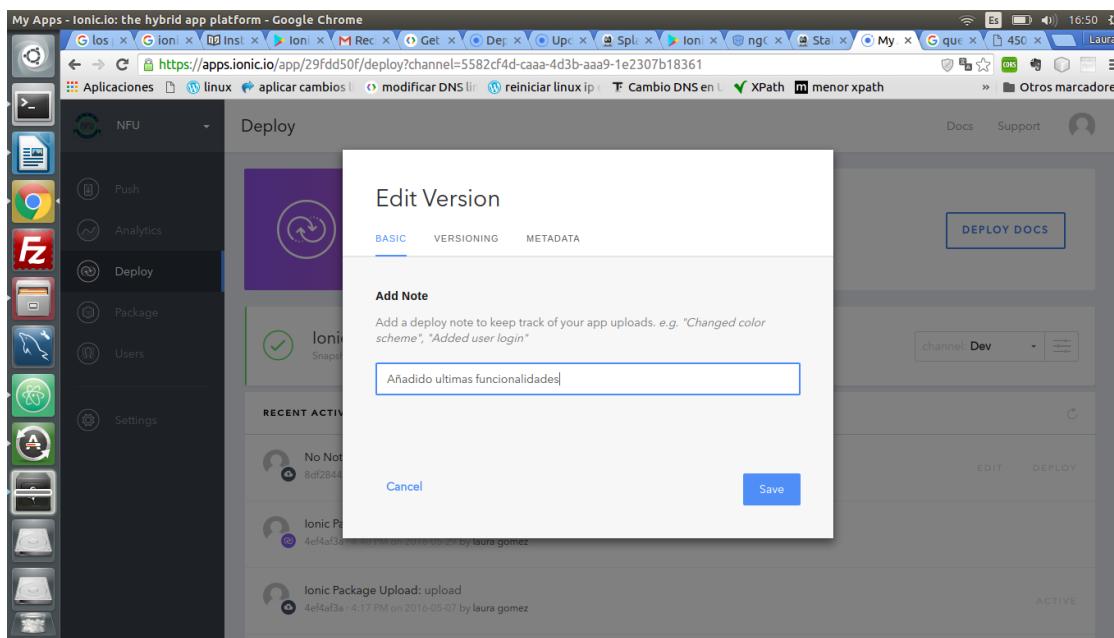


Fig. 75 Edit Version. Fuente: elaboración propia

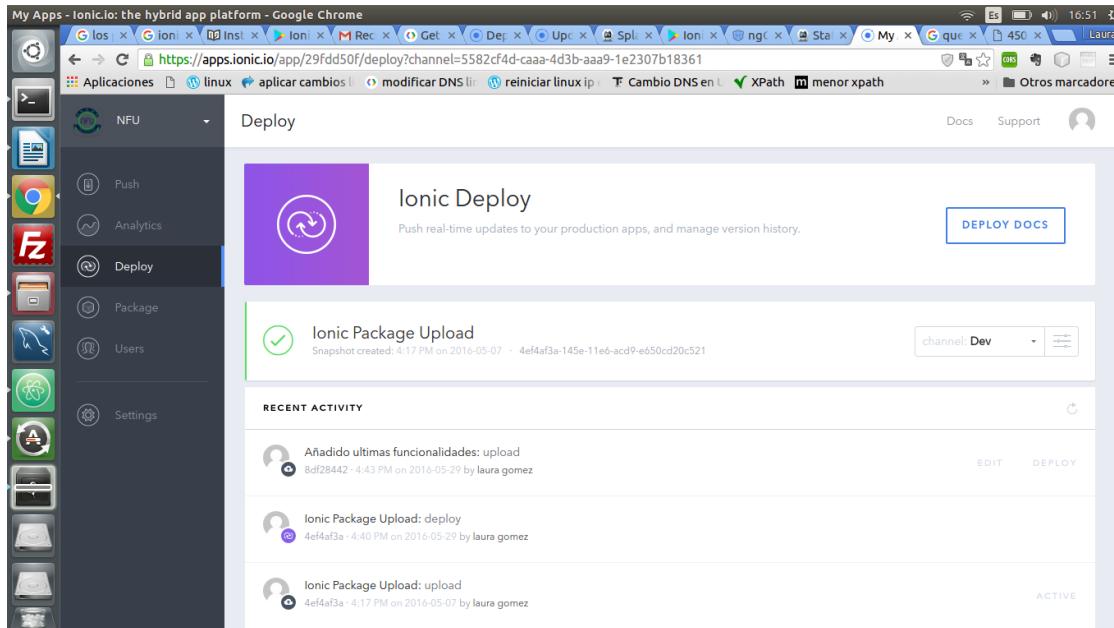


Fig. 76 Versiones Deploy. Fuente: elaboración propia

Así tenemos una lista de las últimas versiones y los últimos cambios que se han realizado. Que también nos permite recuperar una versión anterior.

En el Ionic view podremos usar la aplicación como si estuviera instalada en el propio móvil.

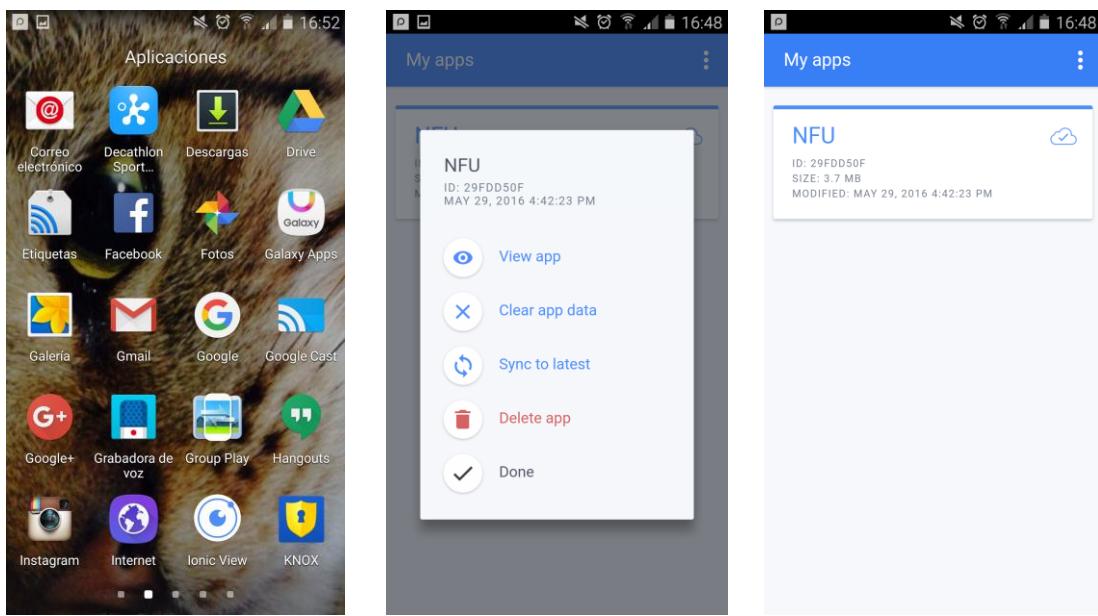


Fig. 77 Ionic View. Fuente: elaboración propia

9.4º Analytics

Gracias a ionic Analytics podemos saber si la aplicación está siendo utilizada, por quién y en qué fechas.

Para ello en el archivo `app.js`, se añade '`ionic.service.analytics`', un módulo de dependencia de '`ionic.service.core`', '`ionic.service.analytics`'

Y se añade el siguiente código también en el `app.js`:

```
$ionicAnalytics.register();
```

Así podemos desplazarnos al panel de control y obtener diferentes estadísticas de uso de la aplicación.

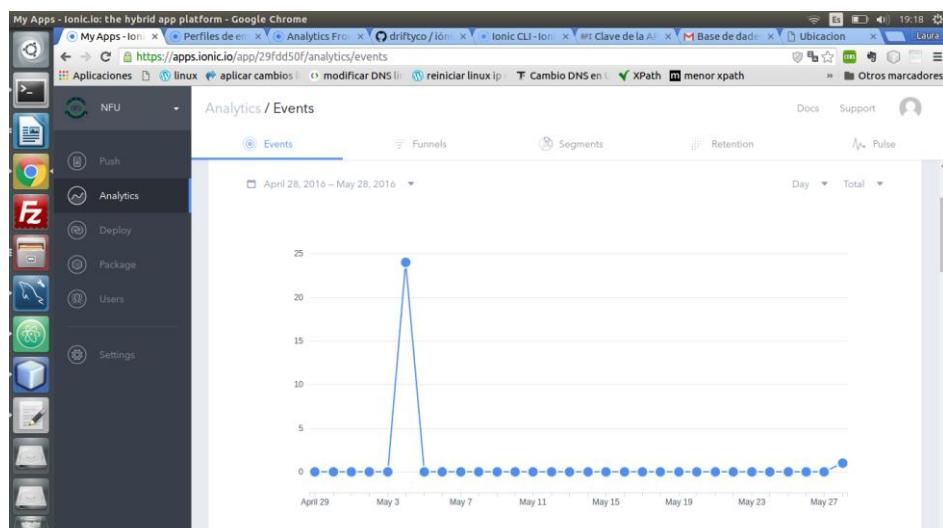
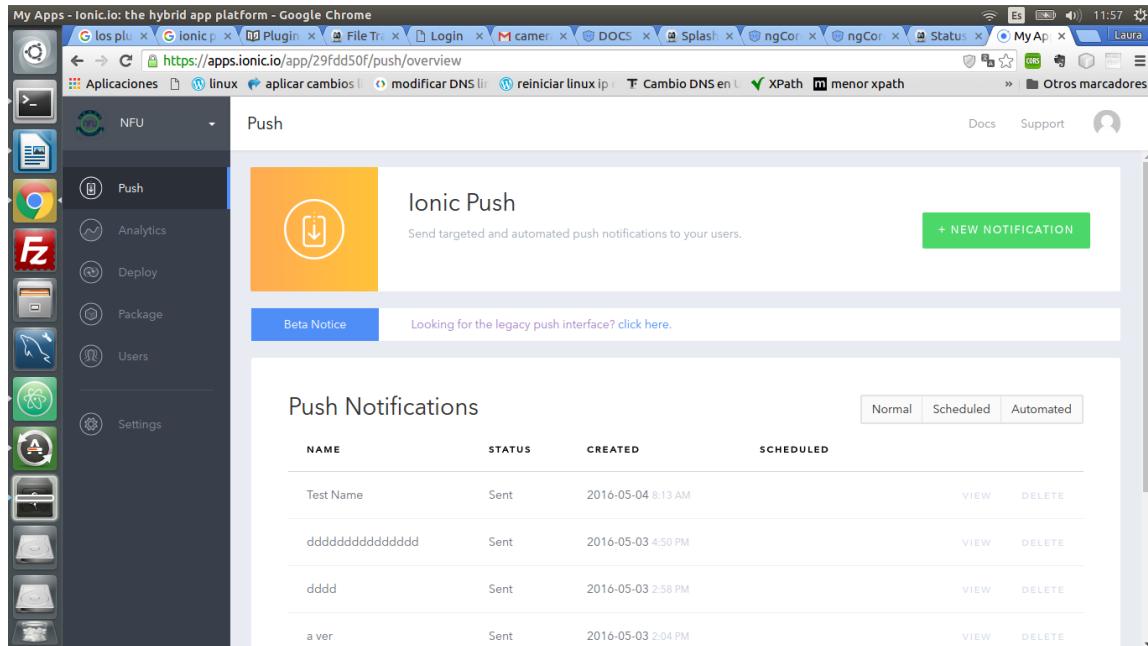


Fig. 78 Analytics. Fuente: elaboración propia

9.5º Analytics

Nos encontramos dos formas de enviar notificaciones push, la primera es mediante el panel de acceso.



The screenshot shows a web browser window titled "My Apps - Ionic.io: the hybrid app platform - Google Chrome". The URL is https://apps.ionic.io/app/29fdd50f/push/overview. On the left, there's a sidebar with icons for NFU, Push, Analytics, Deploy, Package, Users, and Settings. The main content area is titled "Push" and features a large orange button with a download icon labeled "Ionic Push" and the sub-instruction "Send targeted and automated push notifications to your users.". Below this is a "Beta Notice" button and a link to the legacy push interface. A green button labeled "+ NEW NOTIFICATION" is visible. The main section is titled "Push Notifications" and contains a table with four columns: NAME, STATUS, CREATED, and SCHEDULED. The table lists five notifications:

NAME	STATUS	CREATED	SCHEDULED
Test Name	Sent	2016-05-04 8:13 AM	VIEW DELETE
ddddddddd	Sent	2016-05-03 4:50 PM	VIEW DELETE
ddd	Sent	2016-05-03 2:58 PM	VIEW DELETE
a ver	Sent	2016-05-03 2:04 PM	VIEW DELETE

Fig. 79 Notificaciones Push. Fuente: elaboración propia

La segunda opción es mediante código dentro del proyecto Ionic:

```
var jwt = 'your-authorization-token'; // token que lo encontramos en settings/ api key

var tokens = ['your', 'target', 'tokens'];// token que se crea aleatoriamente al iniciar la sesión en ionic

var profile = 'your-security-profile-name';//certificado que he creado en settings /certificates

// Build the request object

var req = {

  method: 'POST',

  url: 'https://api.ionic.io/push/notifications', //api de notificaciones push de ionic

  headers: {

    'Content-Type': 'application/json',

    'Authorization': 'Bearer ' + jwt
```

```
        } ,  
        data: {  
            "tokens": tokens,  
            "profile": profile,  
            "notification": {  
                "title": "Hi",  
                "message": "Hello world!",  
                "android": {  
                    "title": "Hey",  
                    "message": "Hello Android!"  
                },  
            },  
        },  
    } ;
```

6.5.2 Backend

La estructura de este apartado sigue el orden de desarrollo, es decir:

1º Instalación y configuración del VPS

2º Instalación en VPS del servidor Apache2

3º Instalación de MySQL

1º Instalación y configuración del VPS

En primer lugar es necesario explicar qué es un Servidor VPS y para qué sirve, de esta forma el lector podrá comprender con mayor facilidad el resto de contenido.

Un Servidor Virtual o Servidor VPS es un servidor que se instala en una “partición” o fragmento de un servidor físico, pudiendo tener dentro de dicho servidor físico varios servidores virtuales que funcionan como servidores independientes unos de otros.

Un VPS es un servidor privado virtual (virtual private server) que permite alojar sitios web (sitio corporativo, e-commerce, contenidos, multimedia...) y/o aplicaciones de software, especialmente web (portal, extranet, soluciones de trabajo colaborativo, wiki, CRM...).

Al contrario que el alojamiento compartido, permite aislar varias aplicaciones en una misma máquina virtual dedicada al cliente. Dicha máquina virtual comparte recursos físicos (infraestructuras) con otras, pero tiene una parte de recursos dedicados.

En el presente proyecto se van a instalar las herramientas necesarias para desarrollar la aplicación web. Además de instalar lo necesario, ubicaremos nuestro código de backend en el VPS, y realizaremos las llamadas POST, GET, PUT; DELETE, al servidor, para recibir la información que solicitamos.

Características del VPS contratado

VPS SSD 1

- KVM OpenStack
- 1 vCore
- 2,4 GHz
- 2 GB RAM
- SSD 10 GB
- RAID 10 local

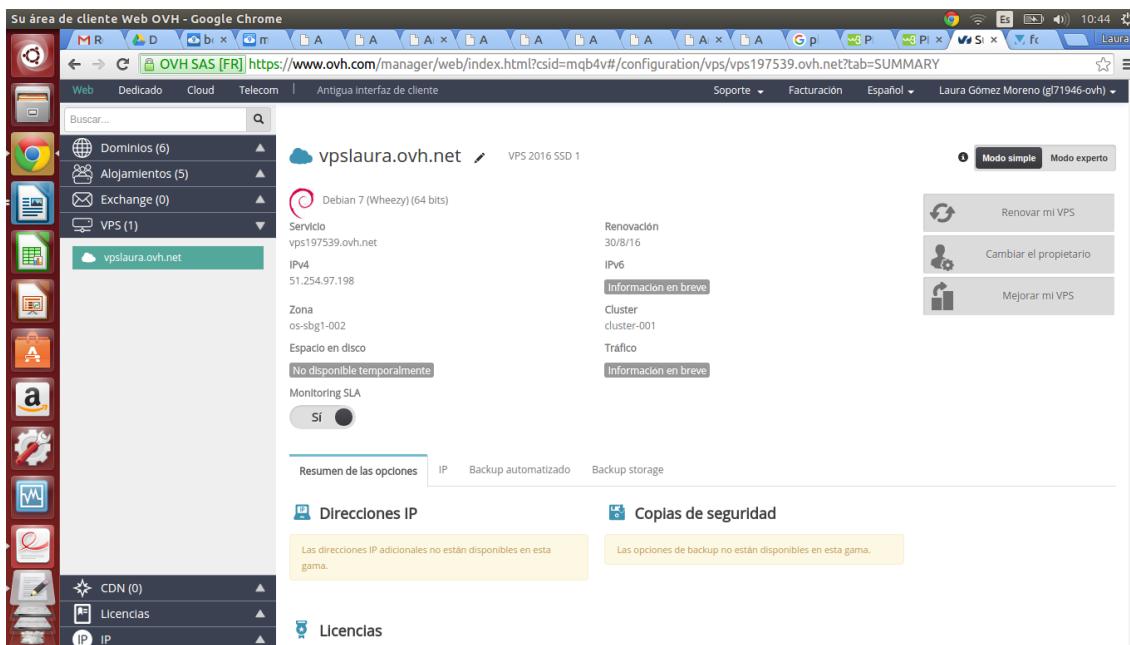


Fig. 80 Panel de control del VPS. Fuente: elaboración propia

El precio del VPS es de 2,99 €/mes + IVA (3,62 € IVA incl.)

Al contratar el VPS, en la empresa OVH²⁰, Viene por defecto instalado con Debian 7. Después dentro del panel de control que nos da acceso OVH, podemos instalar cualquier sistema operativo, excepto Windows que se tiene un coste adicional por utilizarlo. Utilizaremos Debian 7 porque es más estable que Ubuntu.

Al VPS nos conectaremos necesariamente por ssh, que es el nombre de un protocolo y del programa que lo implementa, y sirve para acceder a máquinas remotas a través de una red.

Si lo hacemos por cualquier SO de linux, podremos conectarnos por ssh con el siguiente comando. Previamente deberemos instalar el programa open ssh para poder establecer la conexión:

```
$sudo apt-get install openssh-server
```

```
$ssh -p 22 "root"@vpslaura.ovh.net
```

Una vez conectados al VPS se nos solicita el login y password, inicialmente entraremos como root.

1. Crear usuario con los privilegios de root.

```
$sudo adduser "laura"
```

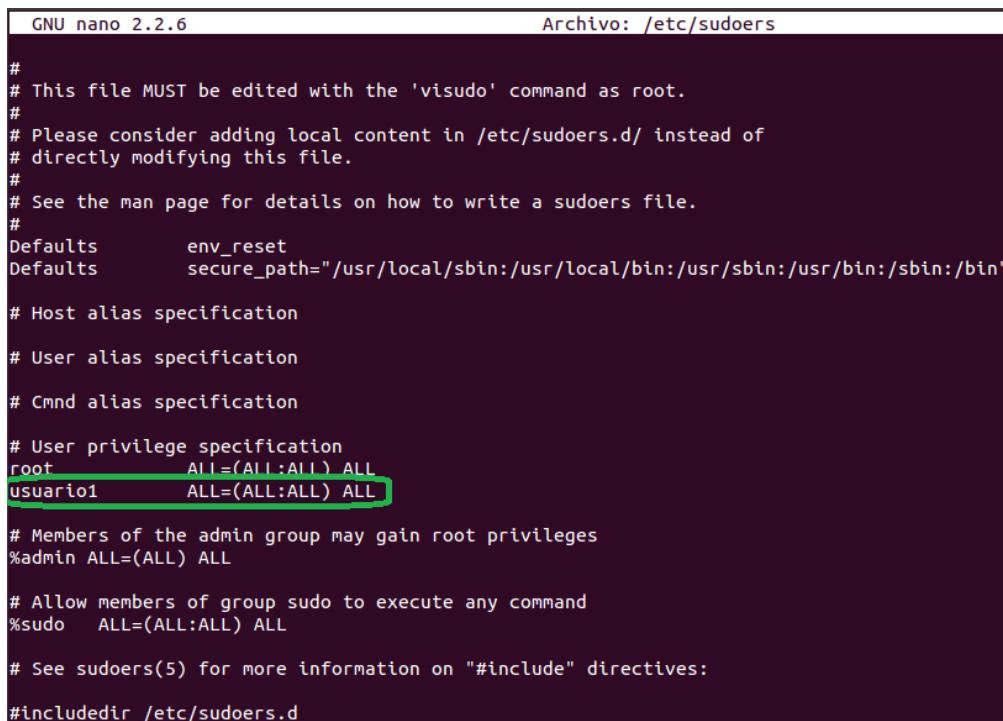
²⁰ www.ovh.es

2. Darle privilegios al nuevo usuario como root.

```
$sudo nano /etc/sudoers
```

Añadimos justo debajo de "**root ALL=(ALL=ALL) ALL**" la siguiente linea:

"laura" ALL=(ALL=ALL) ALL, como la de root, pero con el nombre del usuario con el que queramos ejecutar comandos igual que el root.



```
GNU nano 2.2.6                               Archivo: /etc/sudoers

#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults      env_reset
Defaults      secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root          ALL=(ALL:ALL) ALL
laura         ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin        ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo        ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:
#includeif /etc/sudoers.d
```

Fig. 81 Archivo sudoers modificado. Fuente: elaboración propia

3. Actualizamos.

```
$sudo apt-get update
```

4. Modificamos el archivo sshd_config, cambiamos el puerto por ejemplo 2250 para hacer nuestro vps más seguro, se aconseja que sea un puerto superior a 1024 y en el mismo archivo cambiamos el PermitRootLogin a NO, para que no se pueda logear como root.

```
$sudo nano /etc/ssh/sshd_config
```

```

GNU nano 2.0.9          File: /etc/ssh/sshd_config

# $OpenBSD: sshd_config,v 1.80 2008/07/02 02:24:18 djm Exp $

# This is the sshd server system-wide configuration file. See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/local/bin:/bin:/usr/bin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented. Uncommented options change a
# default value.

#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

# Disable legacy (protocol version 1) support in the server for new
# installations. In future the default will change to require explicit

^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text^T To Spell

```

Fig. 82 Archivo sshd_config. Fuente: elaboración propia

```

GNU nano 2.2.6          Fichero: /etc/ssh/sshd_config          Modificado

HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 768

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin no
StrictModes yes

^G Ver ayuda  ^O Guardar  ^R Leer Fich  ^Y Pág Ant  ^K CortarTxt  ^C Pos actual
^X Salir     ^J Justificar^W Buscar  ^V Pág Sig  ^U PegarTxt  ^T Ortografia

```

Fig. 83 Archivo ssdh_config. Fuente: elaboración propia

5. Reiniciar la conexión por ssh para que cambie la configuración que hemos establecido:

```
$/etc/init.d/ssh restart
```

6. Ahora nos conectamos por ssh:

```
$ssh -p 2250 laura@51.254.97.198
```

2º Instalación en VPS del servidor Apache2

Apache2 es un servidor HTTP de código abierto, para plataformas Unix(BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.12 y la noción de sitio virtual.

```
$sudo apt-get install apache2
```

Instalar php en apache:

```
$sudo aptitude install php5-mysql php5-curl php5-gd  
php-pear php5-imagick php5-imap php5-mcrypt php5-mhash php  
5-xsl php5-json php5-cli
```

3º Instalación de MySQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.

```
$sudo aptitude install mysql-server
```

Instalamos el siguiente módulo para acceder a MySQL y ponemos contraseña para instalarlo.

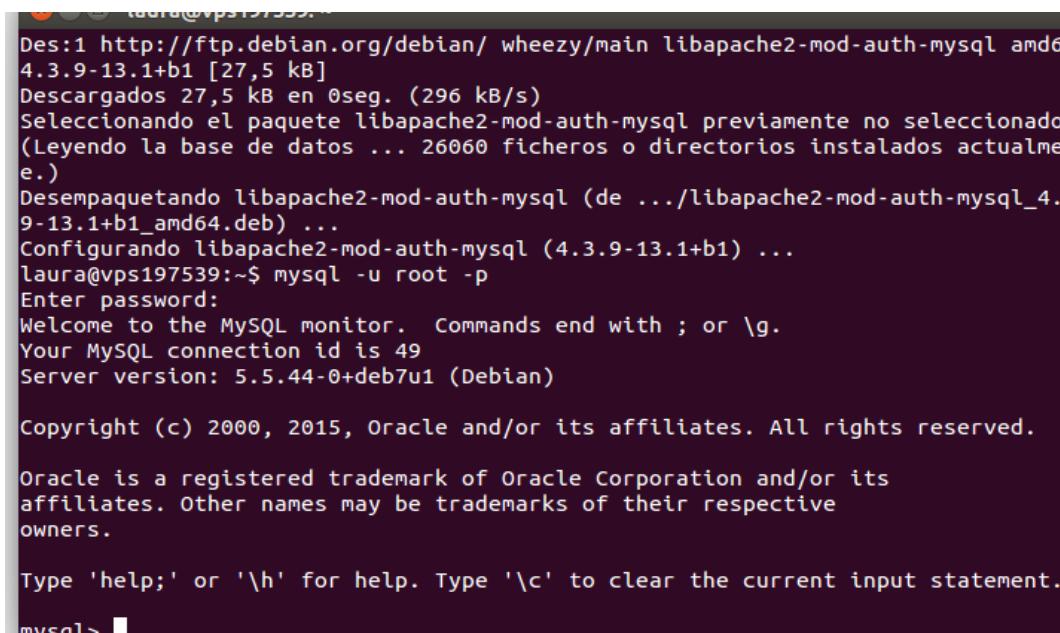
```
$sudo apt-get install mysql-server libapache2-mod-auth-mysql  
php5-mysql
```

```
laura@vps197539:~  
permitted by applicable law.  
Last login: Thu Sep 17 20:20:23 2015 from 193.144.127.13  
laura@vps197539:~$ sudo apt-get install mysql-server libapache2-mod-auth-mysql p  
hp5-mysql  
We trust you have received the usual lecture from the local System  
Administrator. It usually boils down to these three things:  
#1) Respect the privacy of others.  
#2) Think before you type.  
#3) With great power comes great responsibility.  
[sudo] password for laura:  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias  
Leyendo la información de estado... Hecho  
mysql-server ya está en su versión más reciente.  
php5-mysql ya está en su versión más reciente.  
Se instalarán los siguientes paquetes NUEVOS:  
libapache2-mod-auth-mysql  
0 actualizados, 1 se instalarán, 0 para eliminar y 0 no actualizados.  
Necesito descargar 27,5 kB de archivos.  
Se utilizarán 104 kB de espacio de disco adicional después de esta operación.  
;Desea continuar [S/n]? s
```

Fig. 84 Instalar módulo MySQL. Fuente: elaboración propia

Para acceder a MySQL desde comandos se puede acceder mediante:

```
$mysql -u root -p
```



The screenshot shows a terminal window with the following text:

```
laura@vps197539: ~
Des:1 http://ftp.debian.org/debian/ wheezy/main libapache2-mod-auth-mysql amd64
4.3.9-13.1+b1 [27,5 kB]
Descargados 27,5 kB en 0seg. (296 kB/s)
Seleccionando el paquete libapache2-mod-auth-mysql previamente no seleccionado
(Leyendo la base de datos ... 26060 ficheros o directorios instalados actualme
e.)
Desempaquetando libapache2-mod-auth-mysql (de .../libapache2-mod-auth-mysql_4.
9-13.1+b1_amd64.deb) ...
Configurando libapache2-mod-auth-mysql (4.3.9-13.1+b1) ...
laura@vps197539:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 49
Server version: 5.5.44-0+deb7u1 (Debian)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> 
```

Fig. 85 Acceder a MySQL. Fuente: elaboración propia

Para la crea de la base de datos y tablas se ha utilizado Workbench, ya que es una herramienta visual de diseño de bases de datos. Workbench integra desarrollo de software, administración de bases de datos, diseño de bases de datos, y creación y mantenimiento para el sistema de base de datos MySQL. Es una herramienta fácil de usar y muy intuitiva.

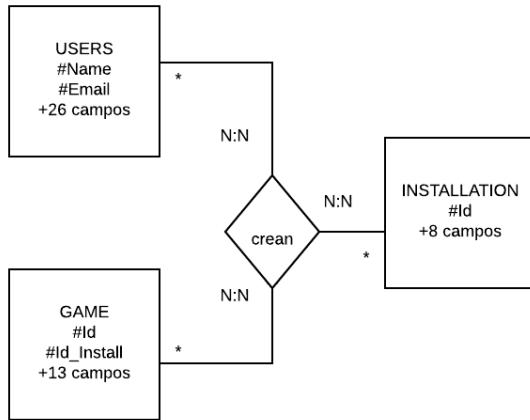
He creado una base de datos llamada nfu, con tres tablas, una de usuarios, otra de partidas y otra de instalaciones.

La de partidas e instalaciones tiene un foreign key entre id_install de partida e id de instalación. Esto se realiza para crear un vínculo entre las dos tablas cuando las columnas de una de ellas hacen referencia a las columnas de la otra que contienen el valor de clave principal. Esta columna se convierte en una clave externa para la segunda tabla.

La tabla “game” de la base de datos tiene un vínculo a la tabla “installation” porque existe una relación lógica entre partida e instalación. Ya que cuando se crea una partida va relacionada a una instalación.

A continuación adjunto modelo de Entidad Relación de la base de datos:

NFU



ENTIDAD/RELACIÓN

Fig. 86 Entidad/Relación NFU. Fuente: elaboración propia

Para desarrollar el backend he utilizado el lenguaje PHP, prácticamente tenía desarrollada la parte de backend ya que se ha realizado durante el curso, en el desarrollo de un framework y he seguido la estructura de éste. Aun así se ha reestructurado el código para optimizarlo y he añadido nuevas funcionalidades.

Uno de los problemas que se me planteó durante el desarrollo del backend, fue como guardar arrays en la base de datos. Ya que varios jugadores, dependiendo de las plazas solicitadas al crear la partida, pueden adherirse a ésta.

Por tanto para crear una base de datos óptima y poder guardar todos los jugadores que se añadan a la partida, se decidió guarda a éstos en un array.

Buscando en la documentación de PHP, encontré los métodos:

```
serialize();  
unserialize();
```

El método `serialize()` una forma muy sencilla de almacenar o transmitir cualquier tipo de valor (excepto el tipo resource) en forma de una cadena seriada. La serialización permite como hemos dicho, guardar un objeto o array en una base de

datos o transmitirlo entre aplicaciones remotas, para luego volver a convertirlo a su tipo y estructura original.

Ejemplo de los diferentes que se ha usado en el proyecto:

El siguiente array se serializa para guardarlo en mi base de datos:

```
$arrValueGame = array(  
    'id' => $_POST['id'],  
    'array' => $array,  
) ;  
  
$jugadores = serialize($arrValueGame);  
  
$sql = "UPDATE game SET jugadores='$jugadores' WHERE  
id ='$id';"  
  
return $db->ejecutar($sql);
```

La función serialize() devolverá un valor fácilmente almacenable en la base de datos:

```
a:2:{i:1;s:4:"pepe";i:2;s:5:"laura";}
```

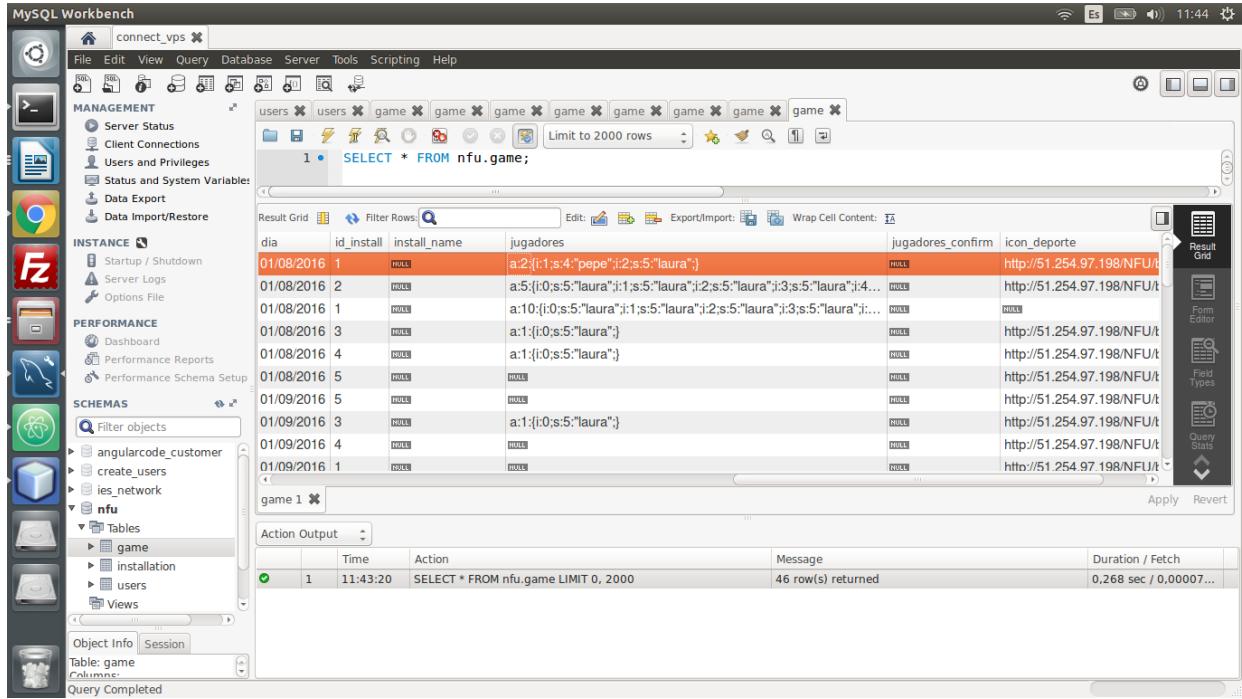


Fig. 87 Ejemplo de base de datos. Fuente: elaboración propia

Para recuperar el valor original en PHP a partir de la cadena serializada:

Hago la consulta a la base de datos y me retorna el valor \$arrValue:

```

set_error_handler('ErrorHandler');

try {

    $arrValue = loadModel(MODEL_USERS,
"users_model", "play", $_POST['id']);

} catch (Exception $e) {

    $arrValue = false;

}

```

\$arrValue me retorna el binario en formato string, y el método unserialize me lo convierte de nuevo en una array:

```
$array = (unserialize($arrValue[0]['jugadores']));
```

Se puede encontrar toda la documentación del Backend en el siguiente enlace:

<http://51.254.97.198/NFU/backend/output/>

6.6 Testing de la aplicación

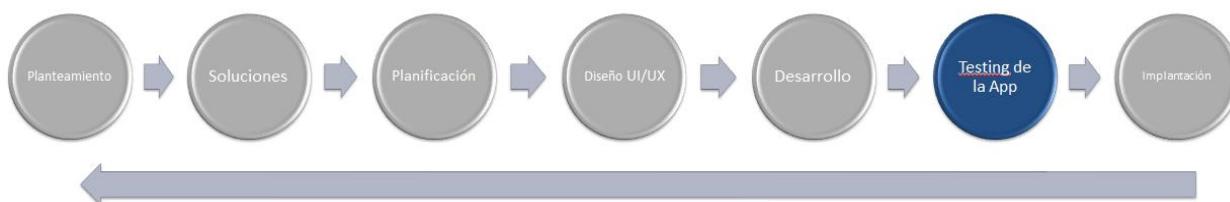


Fig. 88 Fase Testing de la App. Fuente: elaboración propia

Para conseguir una aplicación de calidad, es necesario realizar el testeo de la aplicación. Testear la app conseguirá evitar una mala experiencia de uso en determinadas situaciones o dispositivos, dada la variada naturaleza por la que están formados (memoria, procesador, pantalla...).

El testeo de una aplicación se puede realizar de dos maneras: de forma manual o automatizada. En este caso, se ha procedido a un testeo manual, ya que el tiempo de elaboración del proyecto completo ha sido ajustado. Como se puede observar en la planificación del proyecto del *apartado 6.3*, mientras se ultima la fase de programación de la app, a la vez se va testeando que todo funcione correctamente, es decir, que cada cosa realice lo que tiene que realizar. Para ello, se ha probado el funcionamiento de la app a fondo, creando jugadores y partidas, realizando peticiones de partidas a otros usuarios, configuración del perfil del usuario, etc.

Para que el lector tenga más información sobre cómo testear una aplicación, en el siguiente enlace²¹ se muestra como testear una aplicación de Android de forma automatizada. Para ello se utiliza la librería Android Testing Support library, la cual abastece de todo el framework necesario para testear la aplicación, incluyendo:

- **JUnit4** y un lanzador de tests compatible con los mismos (AndroidJUnitRunner).
- **Espresso**, para testear la interfaz gráfica.
- **UI Automator**, para la automatización de testeo de la interfaz gráfica.

²¹ <http://www.elandroidelibre.com/2016/03/como-testear-tu-aplicacion-en-android.html>

6.7 Implantación/puesta en marcha

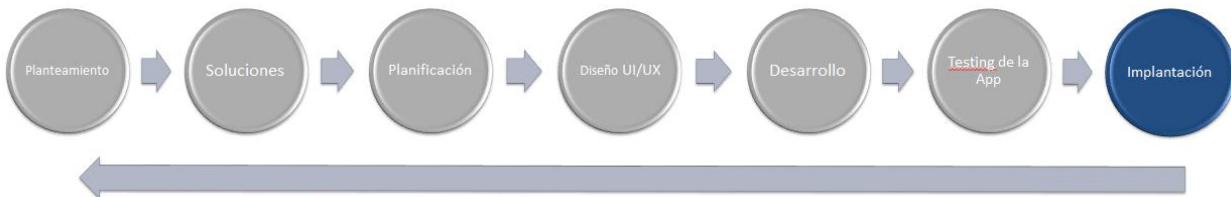


Fig. 89 Fase Implantación. Fuente: elaboración propia

Una vez realizadas todas las fases anterior, queda por último, realizar la publicación de la app “**NFU Nos Falta Uno**” en la tienda de aplicaciones Google Play. Para completar esta fase es necesario realizar por un lado, una parte de diseño y redacción, necesaria para subir la app a Google Play, y por otro lado, la promoción y puesta en marcha de la app.

6.7.1 Publicación de la app en Google Play

En primer lugar, antes pasar a crear el contenido que será publicado en Google Play, se deberá crear una cuenta de Google. Posteriormente esta cuenta de Google se tendrá que convertir en una cuenta de desarrolladores. Para ello, se accede a Google Play Developer Console, como se muestra en la siguiente imagen.

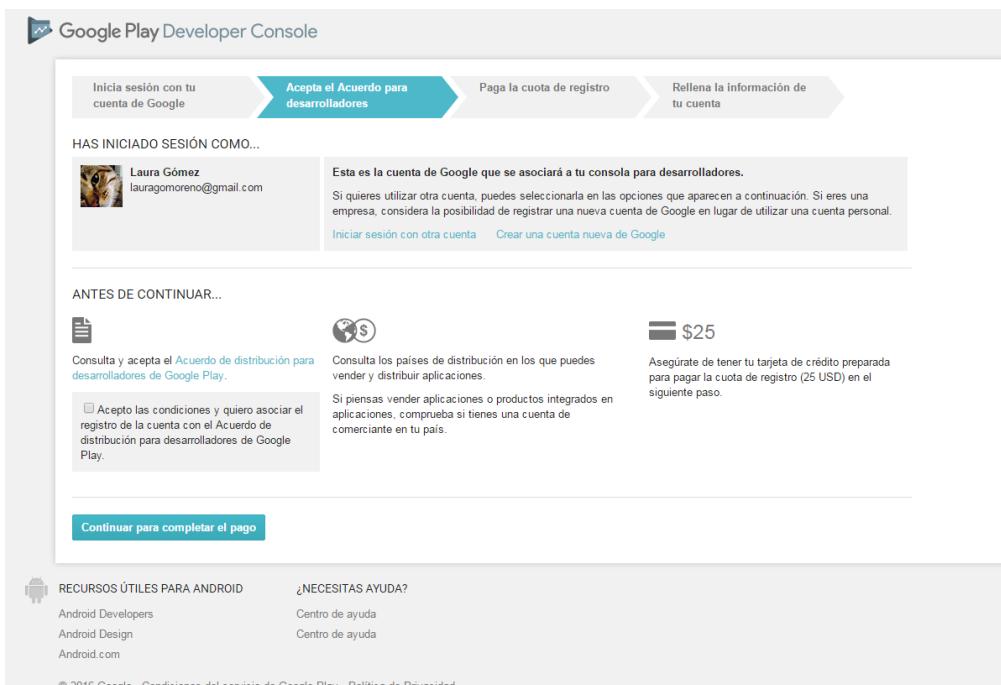


Fig. 90 Pantalla de pago de Google Play Developer Console. Fuente: <https://play.google.com/apps/publish/signup/>

Una vez realizado el pago de \$25 y completar la información de la cuenta, se accede al centro de gestión e información de las aplicaciones de Google Play. Para que el lector sepa cómo subir una aplicación paso a paso de inicio a fin, se adjunta el siguiente enlace²² en el cual se explica al detalle todo el proceso.

A continuación se exponen todos los campos que hay que llenar para poder completar con éxito la publicación de la aplicación en la tienda Google Play.

INFORMACIÓN DEL PRODUCTO

Título: NFU Nos Falta Uno

Descripción breve: En NFU puedes crear una partida para buscar compañeros o unirte a una ya creada.

Descripción completa: ¿Nunca puedes realizar tu deporte favorito porque no encuentras personas con quien practicarlo? :(

La app “Nos Falta Uno” te ayudará a encontrarla :) Simplemente configurando unos fáciles parámetros, “Nos Falta Uno” te facilitará información de personas que como tú, necesitan de un compañero de equipo.

- NFU te geolocaliza y te encuentra partidos y eventos que se juegan cerca de ti
- Organiza tus encuentros fácilmente
- Apúntate a los partidos y eventos que mejor te vengan
- Gestiona tus partidas

¿No sabes dónde está el centro deportivo donde vas a jugar la partida? no te preocupes, NFU te lleva hasta él.

Podrás activar o desactivar las notificaciones de otros deportistas cuando crean una partida, así ser el primero en apuntarte a la partida. O si lo prefieres, puedes crear una partida e invitar a tus compañeros de equipo favoritos.

Ahora ya no vale la excusa de que no se juega porque falta algún jugador, NFU te conecta con otros deportistas.

²² <http://www.elandroidelibre.com/2014/07/como-publicar-una-app-en-google-play-al-detalle.html>

ELEMENTOS GRÁFICOS

Capturas de pantalla:

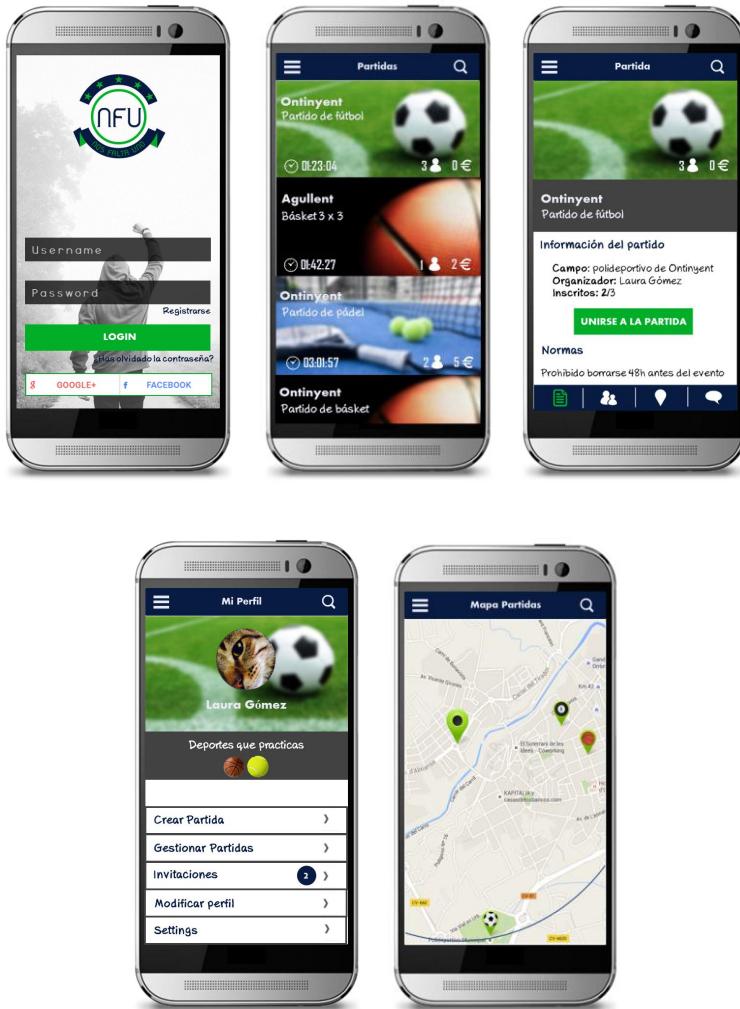


Fig. 91 Capturas de pantalla para publicación de la App. Fuente: elaboración propia

Icono de alta resolución:



Fig. 92 Icono App NFU. Fuente: elaboración propia

CATEGORIZACIÓN

Tipo de aplicación: Aplicaciones

Categoría: Deportes

Clasificación de contenido: para todos

DATOS DE CONTACTO

Sitio web: <http://www.nfunosfaltauno.com>

Correo electrónico: info@nfunosfaltauno.com

PRECIO Y DISTRIBUCIÓN

Esta aplicación es: gratuita

Países: Europa, Sudamérica y Estados Unidos

El resto de apartados a completar son consentimientos y política de privacidad.

6.7.2 Promoción y puesta en marcha

Uno de los aspectos más importantes después de la publicación de la app es la promoción de ésta. Para llevar a cabo esta promoción se realizarán los siguientes apartados:

- Construcción de un landing page sencilla y atractiva
- Creación de un blog para compartir contenidos
- Promoción de la app en redes sociales
- Reseñas de la aplicación en blogs especializados
- Creación de un video de la aplicación

7. Conclusiones

La aplicación NFU puede resultar de gran utilidad para los deportistas, frente al resto de aplicaciones semejantes que hay actualmente.

Ya que se ha tenido en cuenta a la hora de su desarrollo:

- el diseño, siguiendo los parámetros de usabilidad, para que le sea fácil al usuario navegar por la aplicación.
- y el uso nueva tecnologías punteras, que agilizan la carga de datos y la navegación
- y la viabilidad del proyecto, ya que se ha realizado un estudio en profundidad viendo la oportunidad de mercado que ofrece el desarrollo de esta aplicación.

Dos cosas que tendría en cuenta después de la realización del proyecto:

La primera, tendría en cuenta durante la elección de la plataforma un parámetro fundamental que no tuve al inicio, la estabilidad del proyecto IONIC, ya que durante la elaboración del proyecto la plataforma ionic estaba migrando a ionic 2, encontrándome documentación desfasada, métodos no implementados, o incluso páginas en blanco dentro de la documentación de ionic. Por tanto uno de los primeras cosas a tener en cuenta en el desarrollo del proyecto sería buscar la plataforma más estable.

Y la siguiente cosa a tener en cuenata antes de desarrollar el proyecto sería realizar un modelo ENTIDAD / RELACIÓN de la base de datos, con una buena estructura, creo que es imprescindible para que, tanto una aplicación móvil o web pueda crecer el proyecto sin que sea una limitación, como en este caso ha ocurrido.

Finalmente, y como conclusión personal, para mi este proyecto ha sido un reto personal, invirtiéndole todas las horas que podía permitirme y aunque falta la implementación de funcionalidad en la aplicación me quedo satisfecha con el resultado.

8. Mejoras o alternativas propuestas

Las mejoras propuestas para continuar el desarrollo de la aplicación NFU, son las siguientes:

- Que se pueda insertar anuncios por parte de los comercios registrados en la aplicación.
- Oferta del día. Los comercios podrán ofrecer su producto estrella por un día.
- Ranking de jugadores e instalaciones. Donde los jugadores podrán valorarse entre ellos y a las instalaciones.
- El usuario tendrá la opción de crear torneos y gestionarlos
- Insertar la disponibilidad de las pistas por parte de los centros.
- Además de terminar de implementar la funcionalidad planteada inicialmente.

9. Bibliografía

Anexo I

Frameworks para el desarrollo de aplicaciones híbridas

En este anexo se exponen los frameworks más populares para el desarrollo de aplicaciones híbridas, tanto web app como interpretadas. Una vez explicados el lector podrá observar una tabla comparativa.

Frameworks para el desarrollo de aplicaciones híbridas basadas en web app

Ionic

Ionic es un framework gratuito y open source para desarrollar aplicaciones híbridas multiplataforma que utiliza HTML5, CSS (generado por SASS) y Cordova como base. Es uno de los framework del momento por utilizar AngularJS para gestionar las aplicaciones, lo que asegura aplicaciones rápidas y escalables.



Fig. 93 Logo Ionic.
Recuperado de
<http://asktechiblogspot.com.es/2015/08/how-to-add-app-logo-in-header-using.html>

Ventajas

- Ofrece una amplia variedad de componentes.
- Permite utilizar Sass.
- Puede ejecutarse a través de ApacheCordova, PhoneGap o Trigger.io
- No requiere de Jquery para acceder al DOM.

Desventajas

- No tiene acceso a las funciones de cámara, geolocalización, etc., es decir, a los componentes hardware del dispositivo además de la lista de contactos (se soluciona mediante plugins).
- Puede ser lento para mapas como Google Maps.
- No presenta temas para las diferentes plataformas, es decir, el look & feel no simula al nativo.
- Está construido bajo AngularJs (por lo que es necesario conocer previamente AngularJs).

Appery.io

Appery.io²³ nos permite realizar aplicaciones bastante llamativas y profesionales ya que ofrece un entorno de desarrollo rápido e intuitivo basado en la nube, con servicios de back-end integrados. Combina la simplicidad de desarrollo visual con el poder de JavaScript para crear aplicaciones.



Fig. 94 Logo Appery.io.
Recuperado de
<https://plus.google.com/104276681162289155352/posts>

²³ <https://appery.io/>

Ventajas

- Es la única plataforma basada en la nube drag-and-drop.
- Aplicaciones compatibles con iOS, Android y Windows Phone
- Dispone de múltiples tutoriales y documentos
- Nivel de aprendizaje rápido

Desventajas

- No es gratuita
- Al ser una plataforma basada en la nube, hay veces que la velocidad de trabajo es lenta.

Onsen UI

Es un framework²⁴ open source que permite a los desarrolladores implementar aplicaciones con elementos que parecen nativos. Es simple de utilizar, puedes trabajar con o sin Angular JS, cuenta con una gran documentación que incluye montones de ejemplos para las estructuras de apps más comunes. Una desventaja de Onsen UI es que actualmente solo ofrece themes basados en iOS. Tal vez, las próximas versiones cuenten con soporte para Material Design.



Fig. 95 Logo Onsen UI. Recuperado de <https://onsen.io/>

Ventajas

- Funciona con elementos predefinidos
- Excelente documentación con ejemplos

Desventajas

- Builder de Phonegap/Cordova no incluido, pero soportado
- Sin soporte para Material Design

Framework 7

Lo que más llama de Framework 7²⁵ es que es totalmente un framework agnóstico (no cuenta con dependencias externas como Angular o React) y se las arregla para crear apps que simulen apps nativas, con componentes personalizados y animaciones propias de dichas apps. Alguien que programe en HTML, CSS y Javascript puede crear sin problema alguno una app híbrida mediante Framework 7. Lo peor es que no



Fig. 96 Logo Framework 7. Recuperado de <http://victorroblesweb.es/2016/03/05/installar-framework7/>

²⁴ <https://onsen.io/>

²⁵ <http://framework7.io/>

incluye ninguna herramienta para la emulación o el packaging de apps, por lo que necesitarás combinarlo con Cordova o Phonegap.

Ventajas

- Simple de utilizar
- Buen rendimiento
- Puede ser combinado con cualquier framework de Javascript que deseas

Desventajas

- Builder de Phonegap/Cordova no incluido, pero soportado

jQuery Mobile

JQueryMobile²⁶ es un Framework javascript para el desarrollo rápido y fácil de sitios webs optimizados para teléfonos móviles. Con este framework, aceleramos la velocidad de desarrollo de aplicaciones, encapsulando muchas tareas comunes que se realizan cuando usamos el lenguaje JavaScript.



Fig. 97 Logo jQuery Mobile. Recuperado de <http://logonoid.com/jquery-mobile-logo/>

Ventajas

- Facilidad. Si conocemos lenguajes Web (especialmente JavaScript), desarrollar una aplicación con PhoneGap nos resultará mucho más fácil que hacerlo a través de los SDKs asociados a cada plataforma.
- Velocidad de desarrollo. Con PhoneGap, codificamos la aplicación una sola vez y la adaptamos a múltiples plataformas,
- Aplicación nativa. Nuestra aplicación tendrá todas las ventajas de las aplicaciones nativas.
- Libre y gratuito

Desventajas

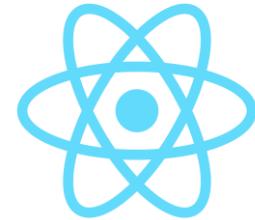
- Mal rendimiento. Muchos desarrolladores de PhoneGap descubren que sus aplicaciones tardan demasiado en cargar. Esto puede deberse a la pesada estructura del framework y la necesidad de usar JavaScript para implementar efectos visuales complejos.
- Limitaciones. PhoneGap no permite manipular las características del dispositivo con tanta libertad como el SDK correspondiente.
- Inconsistencias. Cada sistema operativo tiene su propio diseño. Y una aplicación nativa debería integrarse visualmente con el sistema operativo donde se ejecuta. Pero esto es muy difícil cuando la interfaz es la misma para todas las plataformas.

²⁶ <http://jquerymobile.com/>

Frameworks para el desarrollo de aplicaciones híbridas interpretadas

React Native

React Native²⁷ es un framework creado por Facebook que te permite crear aplicaciones completamente nativas usando javascript y ReactJS. El enfoque principal de React Native es ayudar a los desarrolladores a ser más eficientes al momento de crear una aplicación para múltiples plataformas aprendiendo un solo lenguaje para desarrollar en donde sea.



Ventajas

- Apariencia e interacción nativa
- Utiliza el concepto y la arquitectura nativa
- No requiere un WebView
- Excelente reporte de errores
- Gran comunidad

Fig. 98 Logo React Native. Recuperado de <http://angular.github.io/react-native-renderer/>

Desventajas

- Curva de aprendizaje pronunciada
- Código escrito para iOS no funcionará para Android y viceversa
- Actualmente sólo es compatible con dos plataformas principales , iOS y Android

Native Script

NativeScript²⁸ es un framework de código abierto de la empresa Telerik que permite usar JavaScript para construir aplicaciones móviles nativas para Apple iOS, Google Android y, en el futuro, Windows Universal.



Ventajas

- El enfoque: escríbelo una vez y úsallo donde quieras
- Gran documentación

Fig. 99 Logo Native Script. Recuperado de <https://github.com/NativeScript/NativeScript>

Desventajas

- Curva de aprendizaje pronunciada
- Comunidad pequeña

²⁷ <http://www.reactnative.com/>

²⁸ <https://www.nativescript.org/>

Appcelerator

Appcelerator Titanium²⁹ es una plataforma para desarrollar aplicaciones móviles y de escritorio utilizando tecnologías web. Fue desarrollado por Appcelerator Inc. y lanzado en diciembre del 2008. En junio de 2009 se añadió soporte para el desarrollo de aplicaciones móviles para iOS y Android, y posteriormente también para Blackberry y Windows.

Este framework proporciona una herramienta multiplataforma para crear aplicaciones nativas, de manera que los desarrolladores escriben código JavaScript abstrayéndose de la interfaz gráfica, que se implementa mediante componentes nativos, mejorando la experiencia de usuario en comparación con otras alternativas de desarrollo híbrido.



Fig. 100 Logo Appcelerator.
Recuperado de
<http://www.appcelerator.com/legal/trademark-policy/>

Ventajas

- Desarrolla aplicaciones móviles multiplataforma (iOS, Android, Blackberry y Windows Phone). Y también de escritorio (Windows, Mac y Linux).
- El aspecto y los controles son nativos, por lo que se obtiene mejor rendimiento.
- Es una plataforma gratuita, con soporte de pago.
- Reutilización del 60-90% de código en varias plataformas.
- La comunidad está en constante crecimiento.

Desventajas

- Los componentes visuales y los controles se definen manualmente mediante JavaScript.
- Las librerías JavaScript son diferentes dependiendo de la plataforma.
- Escasa compatibilidad entre los sistemas operativos.

²⁹ <http://www.appcelerator.com/>

Fusetools

Fusetools³⁰ es un framework de código abierto de la empresa Fusetools, que permite utilizar JavaScript para construir aplicaciones móviles nativas para Apple iOS y Google Android.

El framework de Fusetools es rápido, flexible e intuitivo, permitiendo la creación de apps para iOS y Android desde dentro de un entorno WYSIWYG. De este modo, Fusetools ahorra en tiempo de desarrollo, acelera el flujo de trabajo y reduce drásticamente el listón para la creación de aplicaciones visualmente atractivas que funcionan muy bien en todas las plataformas.



Fig. 101 Logo Fuse.
Recuperado de
<https://thefricky.wordpress.com/2015/11/21/fuse-podria-cambiar-las-reglas-del-juego-del-desarrollo-movil/>

³⁰ <https://www.fusetools.com/>

La siguiente tabla se muestra la comparativa de los frameworks para el desarrollo de aplicaciones híbridas interpretadas:

Framework	Logo	Native Look and Feel	Prerrequisitos	Comunidad	Sistemas operativos soportados	Documentación	Herramientas
React Native		9/10	ReactJS	8/10	 	5/10	React Developer Tools extensión para Chrome
Native Script		8/10	JavaScript	5/10	 	9/10	Libre CLI, otras opciones de pago
Appcelerator		9/10	JavaScript	8/10	 	8/10	Plataforma de pago
Fuseteam		8/10	JavaScript	7/10	 	7/10	Plataforma Fuseteam

Tabla 13 Comparación frameworks para el desarrollo de aplicaciones híbridas interpretadas. Fuente: elaboración propia

La siguiente tabla se muestra la comparativa de los frameworks para el desarrollo de aplicaciones híbridas basadas en web app:

Framework	Logo	Native Look and Feel	Prerrequisitos	Comunidad	Sistemas operativos soportados	Documentación	Herramientas
Ionic		7/10	AngularJS (opcional)	9/10	 	8/10	Potente CLI, Ionik SDK
Appery.io		7/10	HTML, CSS and JS	9/10	  	8/10	Cloud platform de pago
Onsen UI		6/10	AngularJS (opcional)	4/10	 	9/10	Monaca Cloud IDE (tiene plan free)
Framework 7		8/10	HTML, CSS and JS	6/10	 	8/10	Ninguna
jQuery Mobile		3/10	jQuery	8/10	  	5/10	Ninguna

Tabla 14 Comparación frameworks para el desarrollo de aplicaciones híbridas basadas en web. Fuente: elaboración propia

Anexo II

Plan de empresa de Nos Falta Uno

Anexo III

Índice de figuras

<i>Fig. 1 Fases del proyecto. Fuente: elaboración propia.</i>	6
<i>Fig. 2 Fase planteamiento. Fuente: elaboración propia.</i>	9
<i>Fig. 3 Logo App Timpik. Recuperado de https://play.google.com/store/apps/details?id=com.timpik&hl=es</i>	10
<i>Fig. 4 Perfil usuario Timpik. Recuperado de https://play.google.com/store/apps/details?id=com.timpik&hl=es</i>	10
<i>Fig. 5 Logo App Fubles. Recuperado de https://play.google.com/store/apps/details?id=it.android.fubles&hl=es</i>	11
<i>Fig. 6 Pantalla inicio Fubles. Recuperado de https://play.google.com/store/apps/details?id=it.android.fubles&hl=es</i>	11
<i>Fig. 7 Logo Decathlon Sport Meeting. Recuperado de https://play.google.com/store/apps/details?id=com.decathlon.sportmeeting&hl=es</i>	13
<i>Fig. 8 Perfil usuario Decathlon Sport Meeting. Recuperado de https://play.google.com/store/apps/details?id=com.decathlon.sportmeeting&hl=es</i>	13
<i>Fig. 9 Fase Soluciones. Fuente: elaboración propia</i>	16
<i>Fig. 10 Comparación aplicaciones web con nativas. Recuperado de http://www.accensit.com/index.php/en/accensit-blog-en/150-mobile-platforms.html</i>	18
<i>Fig. 11 Comparación app híbrida y con app nativa</i>	20
<i>Fig. 12 Fase Planificación. Fuente: elaboración propia</i>	25
<i>Fig. 13 Diagrama de Gantt de todas las fases del proyecto. Fuente: elaboración propia</i>	25
<i>Fig. 14 Logo Photoshop. Recuperado de https://commons.wikimedia.org/wiki/File:Adobe_Photoshop_CS6_icon.svg</i>	29
<i>Fig. 15 Logo Balsamiq mockups. Recuperado de http://informatica.desecundaria.com/</i>	29
<i>Fig. 16 Logo Ionic. Recuperado de http://asktechib.blogspot.com.es/2015/08/how-to-add-app-logo-in-header-using.html</i>	31
<i>Fig. 17 Logo Apache Cordova. Recuperado de http://palebluedot.tv/mobile-app-development-belfast/</i>	32
<i>Fig. 18 Logo Sass. Recuperado de http://maquetando.com/sass-y-less-el-presente-de-css</i>	33
<i>Fig. 19 Logo AngularJS. Recuperado de http://www.marioperez.com.mx/cursos-online/</i>	33
<i>Fig. 20 Logo Php. Recuperado de: http://www.taringa.net/posts/ebooks-tutoriales/19201586/Php-o-C.html</i>	33
<i>Fig. 21 Logo Bower. Recuperado de http://bower.io/docs/about/</i>	34
<i>Fig. 22 Logo Gulp. Recuperado de https://www.npmjs.com/package/gulp</i>	34
<i>Fig. 23 Logo npm. Recuperado de https://www.npmjs.com/</i>	35
<i>Fig. 24 Logo MySQL. Recuperado de http://logodatabases.com/mysql-logo.html/mysql</i>	35
<i>Fig. 25 Logo Atom. Recuperado de http://www.josedomingo.org/pledin/2014/10/mi-experiencia-con-atom/</i>	35
<i>Fig. 26 Logo NetBEans. Recuperado de http://www.logotypes101.com/logo/netbeans</i>	36
<i>Fig. 27 Logo MySQL Workbench. Recuperado de http://www.moodlebites.com/login/index.php</i>	36
<i>Fig. 28 Fase de Diseño UI/UX. Fuente: elaboración propia</i>	39
<i>Fig. 29 Mockup y diseño de la pantalla inicial. Fuente: elaboración propia</i>	40
<i>Fig. 30 Logo NFU. Fuente: elaboración propia</i>	40
<i>Fig. 31 Mockup y diseño de la pantalla login. Fuente: elaboración propia</i>	41
<i>Fig. 32 Mockup y diseño de la pantalla partidas y del menú lateral. Fuente: elaboración propia</i>	42
<i>Fig. 33 Mockup y diseño de la pantalla información de la partida. Fuente: elaboración propia</i>	43
<i>Fig. 34 Mockup y diseño de la pantalla jugadores. Fuente: elaboración propia</i>	44
<i>Fig. 35 Mockup y diseño de la pantalla ubicación de la partida. Fuente: elaboración propia</i>	45
<i>Fig. 36 Mockup y diseño de la pantalla comentarios. Fuente: elaboración propia</i>	46
<i>Fig. 37 Mockup y diseño de la pantalla mapa de partidas. Fuente: elaboración propia</i>	47

<i>Fig. 38 Mockup y diseño de la pantalla instalaciones. Fuente: elaboración propia</i>	48
<i>Fig. 39 Mockup y diseño de la pantalla información de la instalación. Fuente: elaboración propia</i>	48
<i>Fig. 40 Mockup y diseño de la pantalla Mi perfil. Fuente: elaboración propia</i>	49
<i>Fig. 41 Mockup y diseño de la pantalla crear partidas. Fuente: elaboración propia</i>	50
<i>Fig. 42 Mockup y diseño de la pantalla gestionar partidas. Fuente: elaboración propia</i>	51
<i>Fig. 43 Mockup y diseño de la pantalla invitaciones. Fuente: elaboración propia</i>	52
<i>Fig. 44 Mockup y diseño de la pantalla modificar perfil. Fuente: elaboración propia</i>	52
<i>Fig. 45 Mockup y diseño de la pantalla settings. Fuente: elaboración propia</i>	53
<i>Fig. 46 Fase Desarrollo. Fuente: elaboración propia</i>	54
<i>Fig. 47 Estructura de npm. Recuperado de http://i.blogs.es/d470fc/650_1000_npm1-1/450_1000.gif</i>	55
<i>Fig. 48 Pantalla Projects de Ionic Creator. Fuente: elaboración propia</i>	56
<i>Fig. 49 Pantalla de NFU en Ionic Creator. Fuente: elaboración propia</i>	57
<i>Fig. 50 Exportar proyecto Ionic Creator. Fuente: elaboración propia</i>	57
<i>Fig. 51 Parámetros de descarga Ionic Creator. Fuente: elaboración propia</i>	58
<i>Fig. 52 Descargar proyecto desde la consola. Fuente: elaboración propia</i>	58
<i>Fig. 53 Arquitectura del proyecto. Recuperado de https://ajgallego.gitbooks.io/ionic/content/arquitectura.html</i>	61
<i>Fig. 54 Pantalla navegador con Ionic Serve. Fuente: elaboración propia</i>	84
<i>Fig. 55 Pantalla navegador con Ionic Serve - -lab. Fuente: elaboración propia</i>	84
<i>Fig. 56 Selección de API a instalar. Fuente: elaboración propia</i>	85
<i>Fig. 57 Emulador. Fuente: elaboración propia</i>	86
<i>Fig. 58 Directorio. Fuente: elaboración propia</i>	87
<i>Fig. 59 Archivo .scss. Fuente: elaboración propia</i>	88
<i>Fig. 60 Archivo _variables.scss. Fuente: elaboración propia</i>	89
<i>Fig. 61 Panel de control de Ionic Account. Fuente: elaboración propia</i>	91
<i>Fig. 62 Settings. Fuente: elaboración propia</i>	92
<i>Fig. 63 API Keys. Fuente: elaboración propia</i>	93
<i>Fig. 64 Creación de certificado. Fuente: elaboración propia</i>	93
<i>Fig. 65 Certificados. Fuente: elaboración propia</i>	94
<i>Fig. 66 Panel de control de la consola de Google. Fuente: elaboración propia</i>	95
<i>Fig. 67 API habilitadas. Fuente: elaboración propia</i>	96
<i>Fig. 68 API de Google. Fuente: elaboración propia</i>	96
<i>Fig. 69 Creación de certificado Android. Fuente: elaboración propia</i>	97
<i>Fig. 70 User Auth. Fuente: elaboración propia</i>	98
<i>Fig. 71 Users. Fuente: elaboración propia</i>	98
<i>Fig. 72 Ionic Package. Fuente: elaboración propia</i>	101
<i>Fig. 73 Upload a Deploy. Fuente: elaboración propia</i>	102
<i>Fig. 74 Ionic Deploy. Fuente: elaboración propia</i>	102
<i>Fig. 75 Edit Version. Fuente: elaboración propia</i>	103
<i>Fig. 76 Versiones Deploy. Fuente: elaboración propia</i>	103
<i>Fig. 77 Ionic View. Fuente: elaboración propia</i>	104
<i>Fig. 78 Analytics. Fuente: elaboración propia</i>	104
<i>Fig. 79 Notificaciones Push. Fuente: elaboración propia</i>	105
<i>Fig. 80 Panel de control del VPS. Fuente: elaboración propia</i>	108
<i>Fig. 81 Archivo sudoers modificado. Fuente: elaboración propia</i>	109
<i>Fig. 82 Archivo sshd_config. Fuente: elaboración propia</i>	110
<i>Fig. 83 Archivo ssdh_config. Fuente: elaboración propia</i>	110
<i>Fig. 84 Instalar módulo MySQL. Fuente: elaboración propia</i>	111
<i>Fig. 85 Acceder a MySQL. Fuente: elaboración propia</i>	112
<i>Fig. 86 Entidad/Relación NFU. Fuente: elaboración propia</i>	113

<i>Fig. 87 Ejemplo de base de datos. Fuente: elaboración propia</i>	115
<i>Fig. 88 Fase Testing de la App. Fuente: elaboración propia</i>	116
<i>Fig. 89 Fase Implementación. Fuente: elaboración propia</i>	117
<i>Fig. 90 Pantalla de pago de Google Play Developer Console. Fuente:</i> <i>https://play.google.com/apps/publish/signup/</i>	117
<i>Fig. 91 Capturas de pantalla para publicación de la App. Fuente: elaboración propia</i>	119
<i>Fig. 92 Icono App NFU. Fuente: elaboración propia</i>	119
<i>Fig. 93 Logo Ionic. Recuperado de http://asktechis.blogspot.com.es/2015/08/how-to-add-app-logo-in-header-using.html</i>	126
<i>Fig. 94 Logo Appery.io. Recuperado de https://plus.google.com/104276681162289155352/posts</i>	126
<i>Fig. 95 Logo Onsen UI. Recuperado de https://onsen.io/</i>	127
<i>Fig. 96 Logo Framework 7. Recuperado de http://victorroblesweb.es/2016/03/05/instalar-framework7/</i>	127
<i>Fig. 97 Logo jQuery Mobile. Recuperado de http://logonoid.com/jquery-mobile-logo/</i>	128
<i>Fig. 98 Logo React Native. Recuperado de http://angular.github.io/react-native-renderer/</i>	129
<i>Fig. 99 Logo Native Script. Recuperado de https://github.com/NativeScript/NativeScript</i>	129
<i>Fig. 100 Logo Appcelerator. Recuperado de http://www.appcelerator.com/legal/trademark-policy/</i>	130
<i>Fig. 101 Logo Fuse. Recuperado de https://thefricky.wordpress.com/2015/11/21/fuse-podria-cambiar-las-reglas-del-juego-del-desarrollo-movil/</i>	131

Anexo IV

Índice de tablas

<i>Tabla 1 Comparativa de las Apps de búsqueda de deportistas. Fuente: elaboración propia</i>	14
<i>Tabla 2 Ventajas y desventajas aplicaciones nativas. Fuente: elaboración propia</i>	17
<i>Tabla 3 Ventajas y desventajas aplicaciones web. Fuente: elaboración propia</i>	18
<i>Tabla 4 Ventajas y desventajas aplicaciones web. Fuente: elaboración propia</i>	20
<i>Tabla 5 Comparación final de los tres tipos de aplicaciones móviles. Fuente: elaboración propia</i>	21
<i>Tabla 6 Fase 1: Planteamiento e identificación del problema. Fuente: elaboración propia</i>	26
<i>Tabla 7 Fase 2: Búsqueda de soluciones. Fuente: elaboración propia</i>	27
<i>Tabla 8 Fase 3: Planificación y organización del trabajo</i>	28
<i>Tabla 9 Fase 4: Diseño UI/UX</i>	29
<i>Tabla 10 Fase 5: Desarrollo. Fuente: elaboración propia</i>	31
<i>Tabla 11 Fase 6: Testing de la aplicación. Fuente: elaboración propia</i>	37
<i>Tabla 12 Fase 7: Implementación/puesta en marcha. Fuente: elaboración propia</i>	38
<i>Tabla 13 Comparación frameworks para el desarrollo de aplicaciones híbridas interpretadas. Fuente: elaboración propia</i>	132
<i>Tabla 14 Comparación frameworks para el desarrollo de aplicaciones híbridas basadas en web. Fuente: elaboración propia</i>	133