UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Máster en Ingeniería Informática

Trabajo Fin de Máster

La Perspectiva del Sabio: aplicación móvil para la investigación sociológica del envejecimiento activo

José Alberto Andújar Espinosa

Septiembre, 2017

UNIVERSIDAD DE EXTREMADURA

ESCUELA POLITÉCNICA

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Máster en Ingeniería Informática

Trabajo Fin de Máster
La Perspectiva del Sabio: aplicación móvil para la investigación sociológica del envejecimiento activo

Autor: José Alberto Andújar Espinosa

Tutor: Dr. Pablo Bustos García de Castro

Co-Tutor: Dr. Luis Vicente Calderita Estévez

Tribunal Calificador

Presidente: Dr. José Moreno del Pozo

Secretario: Dr. Pilar Bachiller Burgos

Vocal: Dr. Pedro Miguel Núñez Trujillo

ÍNDICE GENERAL DE CONTENIDOS

Índice de Figuras

Agradecimientos

Me gustaría dar las gracias a mi familia, a mi madre y a mi padre, por todo el soporte recibido durante estos años de formación. Siempre he sentido su apoyo y confianza, incluso en los primeros años de carrera cuando los resultados no me acompañaron. Ahora este TFM les da la razón y demuestra públicamente que su cariño y paciencia han dado sus frutos.

En segundo lugar, tengo que agradecerle también mucho a mi hermano mayor. Él ha sido un espejo en el que mirarme, y su ejemplo de tesón y trabajo ha sido fundamental para conseguir la motivación necesaria para terminar este máster.

Por supuesto, a Elena, quien siempre está a mi lado ayudándome a superar los retos no solo de este proceso sino de nuestro día a día. Además, su ayuda ha sido decisiva para aumentar la calidad del texto de esta memoria.

En otro plano distinto de agradecimiento, a mis tutores, por guiarme durante el proceso y, sobre todo, por conseguir que me esforzara un poco más para alcanzar sus expectativas.

Por último, y no por ello menos importante, a la futura doctora Lidia Domínguez, mi socióloga de cabecera, por permitirme conocer un mundo tan alejado de la informática, por lo reconfortante que es hacer un software que servirá para algo y, sobre todo, por lo bien que nos lo hemos pasado (y lo que nos queda) durante la creación de la Perspectiva del Sabio.

Resumen

En la sociedad actual se está produciendo un fenómeno que preocupa a científicos y políticos: el envejecimiento del envejecimiento. Esta etapa de la vida debe poder desarrollarse en igualdad de condiciones y oportunidades que el resto. Nuestros mayores deben seguir formando parte activa de la sociedad y, por tanto, deben envejecer activamente para alcanzar una integración y desarrollo pleno en la comunidad.

El problema aparece cuando desconocemos qué factores son los condicionantes para estimular un envejecimiento activo ¿Puede el entorno condicionar el resto de variables? En nuestras ciudades o en nuestro barrio, simplemente encontramos políticas y decisiones basadas más en la buena fe y la intuición del gobernante que en estudios y datos que avalen esas actuaciones.

Por tanto, La Perspectiva del Sabio aspira a ser una herramienta tecnológica novedosa que permita, de una parte, a los propios protagonistas recabar información de su entorno más próximo: su barrio. De otra parte, debe permitir a la investigadora en Sociología Lidia Domínguez Párraga, estudiar, analizar y comprender la visión que de su entorno tienen nuestros mayores, con la esperanza, de que pueda determinar qué condiciones favorecen un envejecimiento activo.

Para conseguir este reto tecnológico se ha desarrollado un ecosistema compuesto por una aplicación móvil para recoger la información, un servidor central para almacenar los datos y una aplicación web para que nuestra experta pueda analizarlos. Para lograr este desafío, ha sido necesario usar las tecnologías más vanguardistas y prometedoras del mercado.

Abstract

In today's society, there is a phenomenon that is worrying scientists and politicians: the ageing of ageing. This stage of life must be able to be developed in equal conditions and opportunities than the rest. Our elders must continue to be an active part of society and, therefore, they must actively age to achieve a full integration and development in the community.

The problem arises when we do not know what factors are decisive to encourage an active ageing. Can the environment condition the rest of variables? In our cities or in our neighborhood, we simply find policies and decisions based more on the good faith and intuition of the officeholder than studies and data that support those actions.

Therefore, the wise's view aims to be a novel technological tool that allows, on the one hand, the protagonists themselves to gather information from their nearest environment: their neighborhood. On the other hand, it should allow us to study, analyze and understand the vision that our elders have about their environment, in the hope that our sociologist Lidia Domínguez Párraga can determine which conditions favor an active ageing.

To achieve this technological challenge, we have developed an ecosystem composed of a mobile application to get the information, a central server to store the data and a web application to analyze them. To achieve it, it has been necessary to use the most cutting-edge and promising technologies in the market.

Palabras clave

Aplicación híbrida, aplicación móvil, aplicación web, MongoDB, NoSql, JavaScript, Diseño Web, JSON, NodeJS, API REST, Ionic, AngularJS, Android, iOS, App Store, Play Store, Envejecimiento Activo, Entorno Urbano, Cuarta Edad, Personas Mayores.

Keywords

Hybrid Application, Mobile Application, Native Application, Web Application, MongoDB, NoSql, JavaScript, Web Design, JSON, NodeJS, API REST, Ionic, AngularJS, Android, iOS, App Store, Play Store, Active Ageing, Urban Environment, Fourth Age, Elderly People.

Motivación

Siempre he querido trasladar, de alguna manera, los conocimientos adquiridos durante la carrera y el máster a la sociedad. Cuando mi amiga Lidia Domínguez me explicó el propósito de su Tesis y me manifestó los problemas que se encuentran en su campo para recabar información, comprendí que debía centrar mis esfuerzos en brindarle un conjunto de herramientas informáticas que le ayudaran a realizar su investigación.

Además, cuando el motivo de su investigación es estudiar el envejecimiento activo de las personas mayores y su eje principal de investigación ansía descubrir los factores que les permiten una mejor calidad de vida (con la esperanza natural que tenemos todos de vivir muchos años y, tal vez, con el pequeño egoísmo de poder beneficiarme también de las conclusiones o recomendaciones que arroje su Tesis), me decido a poner a su servicio mis recursos temporales y profesionales.

Coincide el inicio de nuestras conversaciones sobre su investigación con un acontecimiento personal que a priori nada tenía que ver con su ámbito de trabajo. A través de mi tutor, el profesor Pablo Bustos, conozco una sugerencia de otro profesor, Fernando Pulido, quien le ha propuesto la idea de utilizar el teléfono móvil para recoger información en el campo aprovechando la cantidad de sensores que incorporan y utilizando a la persona para que decida qué información captura.

Pablo me seduce con la propuesta y comenzamos a trabajar en el proyecto, a pensar, definir y buscar qué soluciones tecnológicas son las más indicadas para, al menos, plantear la viabilidad de la tarea. Por suerte, los proyectos tecnológicos suelen conseguir algo de financiación, y entre eso, y los recursos de mi empresa, conseguimos avanzar en su construcción y ha derivado hacia el desarrollo de EcoGram.

En las primeras fases de este proceso advierto que, aunque sean campos muy opuestos, puedo usar parte de esos conocimientos y de esa tecnología

para construir un ecosistema de herramientas que ayuden a la socióloga Lidia Domínguez en su investigación, concretamente una aplicación móvil y una aplicación web a la que hemos llamado “la Perspectiva del Sabio”, que permitirá a un conjunto de mayores **-sabios-** recabar información de la ciudad y su entorno para ser analizados, desde la aplicación web, por ella.

Soy consciente desde el primer momento que el resultado final de los desarrollos de EcoGram y la Perspectiva del Sabio serán muy diferentes y que implicarán un esfuerzo extra por mi parte, ya que podía haber usado los esfuerzos y resultados de EcoGram como Trabajo Fin de Máster, como así me lo han expresado mis tutores, otros profesores y otros compañeros de máster.

Sin embargo, me seduce la idea y el reto de que un buen software, sobre una adecuada tecnología, bien diseñada y bien estructurada, pueda ser adaptado fácilmente entre ámbitos tan diferentes en la realidad. Obviamente, deben compartir características comunes, pero es fundamental un proceso de abstracción que nos permita maximizar los elementos comunes y definir un modelo de datos flexible que nos permita, si se me permite la analogía, conseguir un tronco común del que derivar ramas concretas específicamente diseñadas para el problema en cuestión.

Sin olvidar que mi motivación, la informática, y parafraseando al conocido personaje de Tolkien, el mago Gandalf, ha sido hacer algo bueno con el tiempo que nos ha sido dado, y probablemente, inspirado en movimientos como Hackforgood, donde los estudiantes de ingenierías y máster ponen al servicio de la sociedad sus conocimientos, creando, diseñando e implementando aplicaciones que tenga un beneficio social. Decido contribuir a la investigación sociológica de Lidia, ya que, -insisto-, las ciencias sociales tienen problemas para conseguir fondos.

De otra parte, y con la conciencia tranquila respecto a EcoGram, puesto que ha conseguido financiación para seguir desarrollándose, me he sumergido en convertir la Perspectiva del Sabio en una realidad. Ambos proyectos

deberán agradecer la elegante idea de Fernando Pulido, y siempre compartirán una raíz común en su concepción primigenia: usar el teléfono móvil como un multi-sensor y al ser humano como su software de ejecución.

1. Introducción

La informática ha supuesto una revolución tecnológica, entre otras razones, porque ha tenido la capacidad de codificar la realidad. El desarrollo de aplicaciones y soluciones software a medida para problemas concretos ha generado una época de grandes cambios. Así, por ejemplo, ¿quién haría ahora una presentación usando diapositivas clásicas? El ordenador personal y los portátiles cambiaron drásticamente la forma en que la sociedad realizaba muchas de sus tareas y la llegada de los dispositivos móviles no ha hecho sino aumentar la demanda de soluciones tecnológicas a problemas cotidianos, como la gestión de cuentas bancarias, la contabilidad, la gestión de los stocks, etc.

Obviamente, existe otra vertiente de aplicaciones que han surgido como fin en sí mismo; ejemplo de ello pueden ser las redes sociales o los videojuegos, aunque no serán abordadas en este trabajo. Nuestro ideal tiene un matiz más romántico que recuerda más a aquel ingeniero que tenía como objetivo facilitar la vida de las personas, la de aquellos pioneros que escuchaban al experto, extraían la información básica necesaria para concebir las estructuras de datos y sus relaciones, y que, a fin de cuentas, proporcionaban una solución tecnológica a la demanda del cliente.

En la siguiente sección estudiaremos el problema al que se enfrenta nuestra investigadora, las necesidades para llevar a cabo su investigación y conoceremos someramente el ámbito de su estudio.

1.1 Planteamiento del problema.

El planteamiento del problema tiene en este TFM dos vertientes; por un lado, el problema sociológico, y por otro el problema tecnológico. A continuación trataremos en primer lugar el aspecto sociológico de forma que podamos situar al lector en el contexto.

1.1.1 Planteamiento del problema desde el punto de vista de la sociología[1]

Una de las principales características de las sociedades modernas es el envejecimiento de su población. Este fenómeno demográfico tiene un alcance global en occidente. Cada vez hay más personas mayores y estas a su vez viven más años que las generaciones anteriores. Los cambios culturales y sociales entre los cuales destaca la liberación de la mujer, sumado a los avances tecnológicos y médicos han tenido como resultado un aumento de la esperanza de vida y un descenso de la natalidad (Alves, 2013; Caprara y López, 2014; Rodríguez, Rodríguez, Castejón y Morán, 2013). Nos encontramos en una situación nunca antes vivida, la cual preocupa tanto a políticos como a científicos y ciudadanos: el envejecimiento del envejecimiento (Chao, 2005; Cordero del Castillo, 2006). Actualmente, la jubilación ya no es sinónimo de retiro, sino de oportunidad para continuar participando en la sociedad, así como para seguir aprendiendo y disfrutando (Sancho, Andújar y Rodríguez 2015).

En este contexto nos encontramos con cambios importantes a la hora de estudiar a las personas mayores. En primer lugar, el nuevo perfil de los mayores se caracteriza por un aumento en los ingresos económicos gracias a las pensiones -anteriormente no existía la prestación por jubilación-, un mayor nivel de estudios -la mayoría de las personas de edad eran analfabetas-, un conocimiento elemental de las nuevas tecnologías y la conciencia del autocuidado, tanto físico como mental (Bazo y Maiztegui, 1999; Requena, 2006; Sancho, Andújar y Rodríguez 2015). Este nuevo perfil conlleva a una nueva forma de vejez que se caracteriza por ser una vejez activa. Por otro lado, tenemos una sociedad y una serie de organizaciones que están asumiendo esta nueva imagen de los mayores modificando, poco a poco, la idea de vejez, y asumiendo que hay que cubrir nuevas necesidades.

[1] Para la elaboración de esta sección se agradece a Lidia Domínguez su supervisión, repaso y en muchos casos escritura.

Por tanto, el nuevo arquetipo de personas mayores poco o nada tiene que ver con el imaginario de los "abuelos/as" que pasaban el día en casa viendo la tele, cosiendo o jugando la partida de cartas con los amigos (Dias, 2012). Han dejado de ser pasivos para formar parte activa de la sociedad de múltiples maneras, por ejemplo, formando parte de organizaciones y asociaciones sin ánimo de lucro como voluntarios, cuidando de familiares dependientes -tanto niños como adultos- o incluso ayudando económicamente a los familiares (Rodríguez, Rodríguez, Castejón y Morán, 2013). Estas son solo algunas muestras de las actividades diarias que llevan a cabo los mayores en nuestro entorno, las cuales se ven, casi cada día, marcadas por la actuación de personas que superan los 65 años -edad común de jubilación-. La cuestión clave es que, al no pertenecer al mundo laboral -en una sociedad donde la productividad es el valor en alza-, su trabajo es invisible para la mayoría de las personas (Bazo, 2000; Delgado, 2003; Fernández y Kelh, 2001).

Al amparo de estas premisas surge la línea general de investigación de nuestra socióloga, Lidia Domínguez, que tiene como objetivo principal poder aportar una mejora en la vida de las personas mayores activas. Para ello, por un lado parte de la Teoría del envejecimiento activo (Havighurts, 1963), y por otro, del entorno arquitectónico como agente condicionante de la vida diaria de las personas. Ambos conceptos serán expuestos a continuación.

1.1.1.1 Envejecimiento activo

Para entender el concepto de envejecimiento activo y sus límites, debemos definir previamente el concepto de "cuarta edad". La cuarta edad se relaciona con lo que clásicamente se ha asociado con la imagen de "viejo", o lo que es lo mismo, con la enfermedad, la dependencia y la muerte, y actualmente se sitúa en torno a los 80 años (Chao, 2005; Cordero del Castillo, 2006; Requena, 2006).

El envejecimiento activo, también conocido como envejecimiento saludable o con éxito, es una etapa difícil de encuadrar, puesto que sus márgenes son

difusos. La socióloga Lidia Domínguez ha considerado que tiene lugar en el intervalo que comprende desde la jubilación a la cuarta edad, y las personas que se encuentran entre estos márgenes serán nuestro público objetivo, es decir, los usuarios de nuestra aplicación móvil. En consecuencia, la aplicación deberá ser diseñada teniendo en cuenta sus capacidades y aptitudes hacia la tecnología.

A pesar de que la Organización Mundial de la Salud (OMS) comenzó a interesarse por el estado de los mayores en 1995 gracias al programa "Envejecimiento y salud", no ha sido hasta el 2002 cuando ha definido el concepto de envejecimiento activo como "el proceso de optimizar las oportunidades de salud, participación y seguridad en orden a mejorar la calidad de vida de las personas que envejecen." (OMS, 2002). A partir de este momento se populariza la idea, y se comienza a empoderar a las personas mayores, aunque todavía se les asocie con algunos prejuicios negativos. Desde esta perspectiva, se entiende que todos los mayores persiguen un envejecimiento activo o con éxito dado que este fomenta una buena salud tanto física como mental y, por ende, facilita el bienestar.

1.1.1.2 El entorno como agente

Según Remy y Voyé (1976), el modo de vida de los individuos está condicionado por los elementos que lo rodean y su relación con el entorno. El hecho de mantener un envejecimiento activo no depende solo de la voluntad propia de la persona sino de muchos otros factores entre los que se encuentra el entorno en el que habita. La arquitectura que nos rodea puede limitar o fomentar nuestras actividades sean del tipo que sean (Eitler, McMahon, Thoerig y Building Healthy Places Initiative, 2013; Palomino, Grande y Linares 2014), por lo que llevar a cabo un envejecimiento con éxito vendrá delimitado parcialmente por el lugar de residencia de la persona. Tal como señala Fernández-Ballesteros (2011), el modo en que se envejece viene marcado por la genética y la historia de vida de cada uno, así como del ambiente social que le rodea.

Sobre estas ideas, la OMS publicó en 2007 una guía para aquellas ciudades que quisieran pertenecer a las "Ciudades amigables de las personas mayores". El objetivo es crear ciudades más accesibles para las personas mayores, de forma que puedan ser más independientes a la hora de realizar sus actividades diarias, es decir, construir lugares donde envejecer con calidad y dignidad, fomentando la participación de las personas mayores en ese proceso. Es en este punto donde mayor fuerza alcanza nuestra propuesta, ya que permite a las personas mayores informar, desde su perspectiva, de los requisitos y carencias de su entorno inmediato, con la esperanza de conseguir una ciudad que realmente se adapte a sus necesidades. Es sabido, y así lo confirman numerosos estudios (Franco, Diez-Roux, Glass, Lazo, Caballero y Brancati, 2008; Glass y Bilal 2016; Ronzi, Pope, Orton y Bruce, 2016; Rose, 1985), que el entorno puede afectar o favorecer la salud y el bienestar psicológico de las personas, así como a su forma de vida o sus hábitos. En el caso de las personas mayores, además, se ha descubierto que el hecho de poder seguir viviendo en su propia casa de forma independiente, y por tanto, permanecer en el barrio, tiene efectos positivos (Almeida, 2014; Beard y Bloom, 2015; Fernández-Ballesteros y Corraliza, 2004; Rowles, 1983; Wiles, Leibing, Guberman, Reeve y Allen, 2012).

Por tanto, y a la luz de lo anterior, parece claro que la ciudad en la que vivimos y su arquitectura es importante, pero ¿realmente nos afecta en su totalidad? En el caso de los mayores, y debido a su inevitable aumento de enfermedades y limitaciones, las barreras arquitectónicas se encuentran mucho más cerca que en la ciudad como ente. En primer lugar, en su propio hogar y en su edificio (Fernández-Ballesteros y Corraliza, 2004; Heart Healthy Hood Project, 2015), y en segundo lugar, en el barrio. Como personas jóvenes es difícil pensar en todas las limitaciones que pueden rodear a una persona mayor y, además, estas no siempre son físicas. Es evidente que, si un barrio tiene un acceso difícil, está plagado de escaleras, bordillos altos, aceras en mal estado..., va a dificultar el tránsito de la persona mayor, y es probable que repercuta también en una actividad menor. Sin embargo, no es tan evidente descubrir estas limitaciones cuando

no tienen que ver con las limitaciones arquitectónicas y su naturaleza es relativa a fenómenos intangibles como el número y tipo de establecimientos cercanos, o el nivel de seguridad percibido (Buffel, Phillipson y Scharf, 2012; Cohen, Mason, Bedimo, Scribner, Basolo y A. Farley 2003; Phillipson, 2011; Wiles, Leibing, Guberman, Reeve y Allen, 2012). Por tanto, las características de un barrio pueden conseguir que una persona mayor tenga una vida más activa.

A la luz de lo expuesto anteriormente, Lidia Domínguez pretende realizar un estudio cualitativo basado en entrevistas individuales y, en la parte que nos alcanza directamente, necesita de una herramienta tecnológica que le permita la recogida de información desde la perspectiva del mayor, al que nos gusta denominar “el sabio”.

El objetivo es conocer, a través de las actividades diarias, los barrios en los que viven las personas mayores. Para lograrlo, a parte de las entrevistas individuales, pretende conseguir datos cuantitativos y objetivos gracias a la aplicación, ya que los sabios le enviarán información continua sobre la zona en la que viven, respondiendo a preguntas sencillas que, además, han sido previamente acordadas en reuniones grupales. Por ejemplo, una pregunta para orientar esa búsqueda de información es: *¿Dónde suele comprar?* En este caso, se espera que la naturaleza de los datos recibidos incluya fotografías de las tiendas en las que van a comprar, ya sea en un momento puntual o de forma continua, y también se esperan comentarios sobre las razones de su preferencia. El propósito es averiguar si los barrios están preparados para administrar las necesidades básicas y si son del agrado de nuestros mayores.

Igualmente, con una pregunta más directa relacionada con el entorno como agente: “*¿Qué le impide/ayuda a moverse?”* se esperan fotos y comentarios de objetos, lugares, situaciones, etc. donde las personas mayores sienten un refuerzo positivo que les anima a realizar diferentes actividades (parques, el sol de media tarde, ir a misa, los amigos, una cafetería, abundancia de bancos para descansar...) y objetos o situaciones que, por contra, les

complican la actividad (aceras en mal estado, escaleras, coches mal aparcados, obras...).

Es por todo ello que necesita almacenar, procesar y visualizar los datos proporcionados por esos usuarios de forma útil. Para ello, en este TFM se ha creado también una aplicación web que le permite cotejar, repasar y tratar la información almacenada. A esta parte del sistema, y por su similitud con las funcionalidades de gestión habituales de los backend, la hemos denominado “panel de administración”.

Por tanto, nuestra investigadora tratará de estudiar y comprender, mediante el análisis de ambos conjuntos de datos (las entrevistas y los datos proporcionados por la app), el día a día de nuestros mayores en su entorno más próximo: su barrio.

1.1.2 Planteamiento del problema desde el punto de vista de la informática

El apartado anterior nos ha permitido situar al lector y ubicarlo en el contexto sociológico del envejecimiento activo. Los puntos claves para esta otra vertiente del problema, la tecnológica, y al igual que le ocurre a muchas empresas y freelance, residen en conocer y sintetizar los requisitos de nuestro cliente, así como traducir sus deseos y expectativas a la realidad de la tecnología actual, a los recursos disponibles y a la capacidad de trabajo del desarrollador o desarrolladores.

En nuestro caso particular contamos con un único desarrollador, el autor de este TFM, que tendrá que interpretar los diferentes roles del proceso de creación del software, abarcando desde el director de proyecto al diseñador gráfico. En consecuencia, una de las primeras tareas es reunirse con la investigadora y concretar las necesidades. Habitualmente, en el mundo empresarial las conclusiones derivadas de estas reuniones adoptan la forma de un contrato estipulado en un pliego de condiciones, aspecto formal que no será necesario en esta ocasión. Es frecuente también que a lo largo del

proyecto haya que redefinir algunas de esas condiciones, en ocasiones porque no se supo prever una determinada funcionalidad, en otras porque alguna no puede llevarse a cabo en la forma prevista. La minimización de este tipo de errores suele adquirirse con la experiencia, pero es interesante desde el principio desarrollar este tipo de habilidades para conseguir una optimización tanto temporal como económica en la construcción de software.

A continuación se mostrará este proceso de manera resumida y desde una perspectiva final (ya que el proceso abarcó varias reuniones), así como la validación por fases a lo largo del desarrollo (ver **Material y método**).

Del contexto del apartado anterior y de las conversaciones con la investigadora, se extrae la idea básica y general desde el punto de vista tecnológico (ya comentada en la sección de Agradecimientos): ***usar el Smartphone como un multisensor y al usuario como la inteligencia que decide qué sensorizar***. En consecuencia, nos encontramos con que los usuarios de nuestra aplicación van a capturar información de su barrio y nos la van a enviar, la almacenaremos, y mediante una app web, nuestro cliente (nuestra investigadora) podrá analizarla. La Figura 1: Idea general de la Perspectiva del Sabio muestra esta idea general.

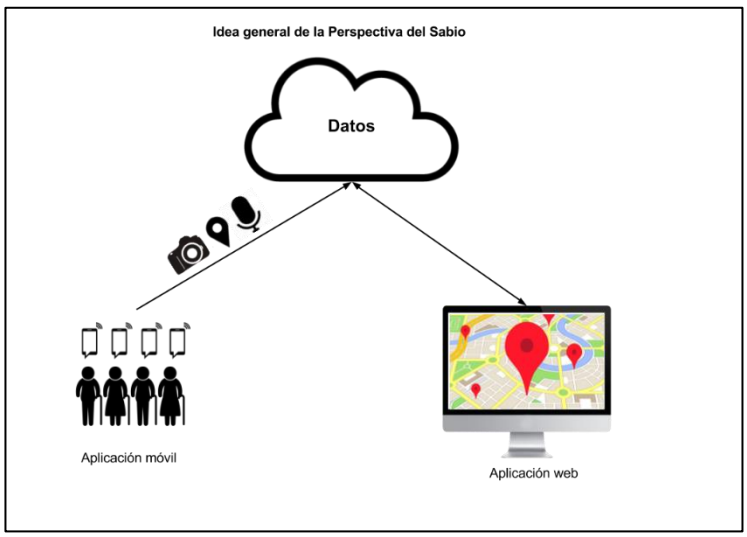

Figura 1: Idea general de la Perspectiva del Sabio

Por tanto, nuestro sistema tiene tres partes claramente diferenciadas, a saber: la aplicación móvil, desde donde el usuario nos proporcionará información; el servidor, donde almacenaremos y procesaremos dicha información; y la aplicación web, desde donde nuestra investigadora tendrá una vista especial para supervisar, editar y trabajar con la información almacenada.

1.1.2.1 Aplicación móvil

Esta pieza del sistema es crucial para el desempeño correcto de las otras, ya que es la fuente principal de información. Básicamente, nuestra investigadora aspiraba a que se pudiera subir información del entorno de manera sencilla. El objetivo era categorizar el barrio y obtener información relevante del entorno, permitiéndole localizar puntos de interés para su estudio.

Su interés inicial reside en obtener una fotografía del lugar en cuestión y añadirle la información geográfica mediante el uso de las coordenadas GPS para reprsentar, así, la perspectiva que el mayor tiene de su entorno. La investigadora también necesita conocer la fuente del dato (el usuario) que envía la foto, por lo que es necesario un proceso de login en la aplicación móvil para almacenar para cada usuario sus fotos.

Desde el punto de vista del hardware se determinó que ambos sensores, el de ubicación y la cámara, estaban disponibles en la mayoría de los smartphones, y que además eran sencillos de usar, convirtiéndose en candidatos ideales[2]. El paquete de datos que contiene la información anterior será enviado por el usuario, finalmente, con un simple clic.

La investigadora también observó, durante las entrevistas individuales que había realizado, que la proporción de personas mayores que tienen iPhone y Android era similar a la del resto de la población, lo que ha condicionado el

[2] Para conseguir una mayor precisión en la catalogación de la información, se añade automáticamente la fecha y hora de captura de la imagen.

tipo de aplicación realizada para cubrir estas dos plataformas (**3.2.2 Hybrid applications**).[3]

También nos gustaría señalar, y aunque ha quedado fuera del proyecto, una futura característica descrita en el capítulo **7, Trabajos Futuros**. Durante el desarrollo del proyecto nos pareció necesario y útil la inclusión de un chat de todos los usuarios con la investigadora (no entre los usuarios) que les permita solventar las dudas a la hora de capturar la información. Además, podría convertirse en el medio para transmitir la pregunta inicial planteada por la investigadora, consiguiendo agilidad y dinamismo a la hora de capturar la Perspectiva del Sabio.

Otro aspecto fundamental es el front-end de la aplicación móvil, pues teníamos que conseguir un diseño adecuado a nuestro público objetivo. Para ello, estudiamos las aplicaciones que usan las personas mayores en nuestro entorno familiar, así como la de los participantes en las entrevistas individuales. La conclusión obtenida es que debía ser un diseño simple y directo, con un tamaño de letra adecuado, en colores claros y limpios, con botones de suficiente dimensión, iconos adecuados y con espacio para texto y, sobre todo, que condujera unívocamente al propósito de obtener y enviar la información requerida. Estas razones originan que se dejen a un lado las opciones de personalización que abundan en otro tipo de aplicaciones, los iconos de desbordamiento, las pestañas, menús desplegables, los ajustes parametrizados, etc.

En consecuencia, cada pantalla tiene el mínimo número de elementos posibles y siempre contiene un solo elemento de información a capturar. La primera vez que se usa la aplicación se necesitan solo cuatro pantallas y cuatro clics para enviar un paquete de datos completo. La segunda y posteriores solo tres, ya que en el proceso de login se recuerdan los datos

[3] En la sección **4 Material y método** observaremos que esta concepción inicial sufrió cambios en el tiempo. La más relevante tuvo lugar en la fase tres (**4.1.3 Tercera fase beta abierta**) del desarrollo, y fue la inclusión de una grabación de audio que acompaña a la captura de la fotografía.

del usuario. En nuestra opinión, se ha alcanzado una alta usabilidad y sencillez.

1.1.2.2 Servidor

Obviamente, esta parte del sistema es en la que menos interviene la visión y opinión de nuestra investigadora. Desde el punto de vista técnico, sin embargo, debe tener los suficientes recursos para soportar la carga de datos y las necesidades del conjunto. Al mismo tiempo, es importante elegir el Sistema Operativo adecuado e instalar y configurar la máquina de acuerdo a las tecnologías que elijamos en las siguientes secciones. En general, se valora un acceso remoto ágil, una suficiente capacidad de memoria y una velocidad de cómputo razonable para la carga estimada. Afortunadamente, las empresas de hosting actuales ofrecen una amplia gama de configuraciones adaptadas a todos los bolsillos y necesidades. Además, gracias a los nuevos sistemas de virtualización es posible crear máquinas con bajas prestaciones para las primeras fases de un proyecto y aumentarlas cuando sea necesario.

En nuestro caso se ha elegido Microsoft Azure (Azure.microsoft.com, 2017) debido a mi experiencia anterior en su utilización y, fundamentalmente, a que tiene un coste cero gracias a que soy beneficiario del programa BizSpark (Bizspark.microsoft.com, 2017) para empresas innovadoras.

1.1.2.3 Aplicación web

El último de los grandes bloques que conforman nuestro ecosistema es la aplicación web, a la que hemos denominado simplemente “panel de administración” o “panel de control”, ya que nos permite el acceso y administración de los datos. Nos proporciona una vista especial de la información recabada por la aplicación móvil que está orientada a las especificaciones y necesidades de nuestra investigadora y, al mismo tiempo,

permite las operaciones comunes de edición, inserción, borrado… sobre los datos almacenados.

Tras las reuniones pertinentes con la socióloga, detectamos que necesitaba un diseño funcional a la vez que compacto que le permitiese tener en la pantalla de su navegador el máximo de información posible. Concluimos que se necesitaba dividir la GUI en tres columnas. (Ver Figura 2: GUI en tres columnas)

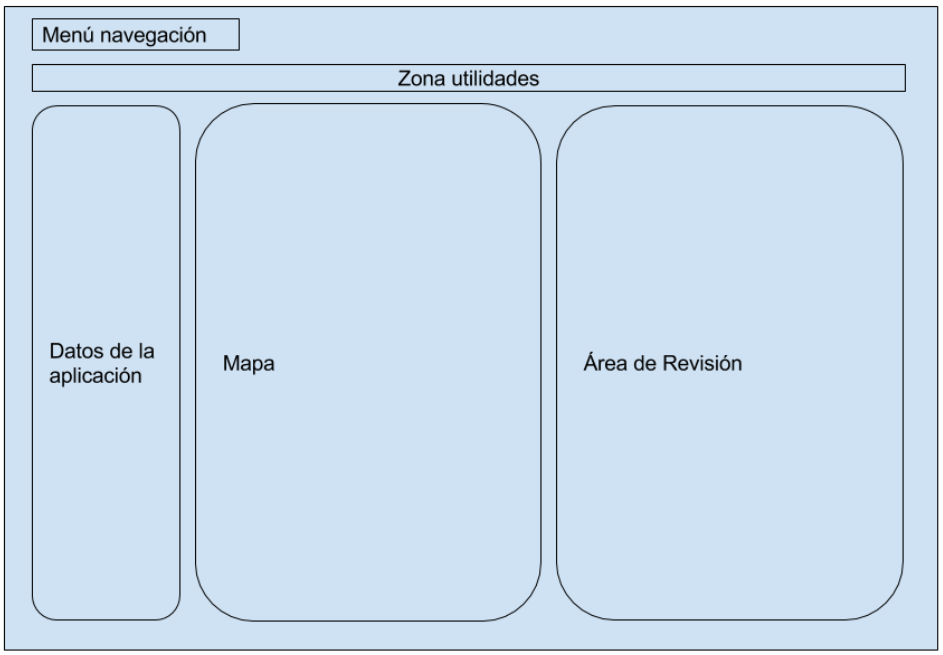

Figura 2: GUI en tres columnas

El área de la izquierda contendría todos los datos almacenados, el área central contendría un mapa que ofreciera una visión global y rápida del lugar desde donde se han tomado esos datos, y el área de la derecha sería el lugar de revisión y edición de dichos datos.

La proporción del ancho de cada columna sería relativa a la información contenida, siendo la columna central la de mayor ancho. El diseño debía ser *responsive* para ordenadores portátiles y de sobremesa ajustando el ancho total al de la pantalla. La proporción final es de 25%, 45% y 30% respectivamente. Intencionadamente dejábamos a un lado el aspecto que

pudiese tener desde el navegador web del Smartphone, puesto que se determinó que era una vía de acceso que superaba el alcance de este TFM.

Reservamos la parte superior para los clásicos menús de navegación y para la zona de utilidades (Figura 2: GUI en tres columnas). Respecto a la primera, actualmente hay una sola pantalla, pero es una forma de diseñar bien el proyecto desde el principio y prepararlo para escenarios futuros. Respecto a la segunda, y en honor a la verdad, no se predijo su lugar ni necesidad en los inicios del proyecto, sino que fue el avance en el desarrollo el que determinó que se necesitaban nuevas funcionalidades. Esto suele ser habitual, y debemos usar un software que se pueda adaptar fácil y rápidamente, como ocurre en este TFM.

La primera de las funcionalidades incorporadas y la más importante es la posibilidad de exportar todos los datos almacenados a un formato compatible con el software ArcGIS (esri España, 2017). ArcGIS es un programa ampliamente usado en diferentes áreas de conocimiento que, además, nuestra investigadora domina. Esto le permitirá utilizar las herramientas de este software para completar más fácilmente su análisis geográfico del barrio. Para lograrlo, se colocó un botón con esa funcionalidad en la zona de utilidades, concretamente en la parte superior de la columna de la izquierda (Ver Figura 46: Resultado final aplicación web).

La segunda consistió en la incorporación de un buscador, también en la zona de utilidades, que permite la búsqueda por usuario y por rango de fechas (Ver Figura 46: Resultado final aplicación web). Si bien la primera característica, la búsqueda por usuario, ya se había manifestado como necesaria en las reuniones mantenidas con anterioridad, la opción de delimitar la búsqueda por fechas ha sido muy útil para la posterior exportación de datos no repetidos al formato de fichero compatible con ArcGIS.

Finalmente, también advertimos la necesidad obvia del acceso a este panel mediante login (usuario y contraseña) para preservar la integridad de los datos. Ella es, de momento, el único usuario con permisos de acceso.

1.1.2.4 Comunicación

Existe una parte tecnológica más y que está intrínsecamente relacionada a las otras tres: la forma en que se comunican. En nuestra opinión, esta decisión debe estar en consonancia con los requisitos del cliente, además de con la naturaleza de los datos, y debe ser una solución flexible que pueda adaptarse, tan rápida y fácilmente como sea posible, a modificaciones imprevistas y a la supuesta escalabilidad del sistema.

Para nuestra propuesta necesitamos un tipo de intercambio de datos lo más sencillo y versátil posible para soportar los diferentes formatos dentro del paquete. La aplicación móvil es la generadora de información, y se debe permitir un modelo de operaciones orientado a eventos que nos permita lanzar el envío como respuesta a la acción de un botón. Es deseable también que las partes intervinientes en la comunicación, el entorno de ejecución y la estructura de intercambio del dato sean ligeras y eficientes.

De la misma forma, en aras de la coherencia y elegancia es recomendable, siempre que sea posible, seleccionar la misma tecnología de comunicación en todas las partes de nuestro sistema. Es decir, si utilizamos los mismos fundamentos en la comunicación del servidor con la aplicación web que los que usamos entre el servidor y la aplicación móvil, estaremos simplificando su desarrollo y posterior mantenimiento, tal y como sucede en este TFM.

Como consecuencia a todo lo expuesto en las secciones precedentes, se estudiarán y describirán en la sección **3 State of the art** las tecnologías más relevantes y novedosas que han intervenido en el desarrollo del sistema. Por eso, se omiten algunas como por ejemplo HTML o CSS, por considerarlas fuera del ámbito de aprendizaje del TFM o por considerarlas

suficientemente conocidas por el lector. Antes de llegar a ese punto definamos los objetivos a alcanzar con la realización de este TFM.

2. Objetivos

Una vez conocido el problema al que nos enfrentamos, y como si fuera una situación real en la que un cliente ficticio (nuestra investigadora) pide a una empresa (el autor del TFM) una aplicación, sintetizamos (tras conocer los problemas a los que se enfrenta la investigadora y los requerimientos de la misma y, por supuesto, sin olvidar la parte académica que subyace en este tipo de trabajos) en los siguientes ítems los objetivos de este TFM:

- Desarrollar una aplicación móvil y una aplicación web usando las tecnologías más novedosas y prometedoras del mercado actual.
- Estudiar y seleccionar las tecnologías web necesarias para conseguir el objetivo anterior, así como, los diferentes framework y herramientas de desarrollo.
- Aprender a diseñar una aplicación móvil basada en principios de usabilidad y sencillez para un público objetivo concreto: las personas mayores.
- Usar una metodología ágil para el desarrollo de aplicaciones en base a las especificaciones, no siempre estáticas, del cliente.
- Aprender y explorar técnicas de visualización de datos para crear una aplicación útil, compacta y adaptada a las necesidades de la investigadora
- Aprender el uso y funcionamiento tanto de Apple Store como de Play Store.
- Conocer la realidad de la tercera edad actual y la influencia de las infraestructuras, el medio ambiente y el entorno en su calidad de vida.
- Desarrollar una herramienta que pueda servir a mejorar la sociedad.

3. State of the art

3.1 Web applications

A web application is a computer application that runs on the web browser, this means that uses the browser as the execution environment. Nowadays, the big companies trend is to use webapps, sometimes they are imitations of the desktop applications, and in others cases, they are totally new applications, without version of desktop, like PlayMyHit. Some reasons for this trend may be the increase of the cloud services, derived from a dizzying increase in Internet speed. In addition the users demand ubiquity in their software tools. Fact, that in our opinion, it is a paradox and a return to the old idea of several low-performance devices connected to a local server, where we interact with the server through several input/output peripherals, such as screen, keyboard and mouse. The server runs the applications since it has the machine resources and offers us a personalized view accessed by login.

On the other hand, an interesting feature is that web applications solve, at least in part, the problems derived from the creation of specific applications for each platform and operating system.

Our researcher expressed this preference from the beginning. She wants to access from her laptop and computer, in the same way that she does it with GoogleDocs. She needs to access from several locations, since she needs to show the stored information to her advisors and collaborators. So we decided to create a web app to build the admin panel. The design should be responsive, but only focused to laptop and computers.

3.2 Mobile applications

A mobile application can be defined as: "a computer application that has been designed to be run on a mobile device," such as a smartphone or a tablet.

Mobile applications do not appear in our society until 2008. Mobile applications are available through distribution platforms similar to application repositories of free operating systems such as Debian, Ubuntu or Fedora, in this case they are known as "stores" and they are basically an app store. The user chooses one app from the app store and the app will be installed on the device. Typically, the app store is controlled by the owner of the mobile operating system. Although there are independent platforms like F-droid (En.wikipedia.org, 2017) or Cydia (En.wikipedia.org, 2017).

Usually we refer to them with the diminutive "app", and although the term sounds us really familiar, this term did not became popular until seven years ago. According to the American Dialect Society, the term app was one of the words of the year 2010. This allows us to have a perspective, about the speed with which technology and computer science continues to advance.

3.2.1 Native applications

Native applications are developed specifically for one platform, and can take full advantage of all the device features. (Nngroup.com, 2017) Typically, they have been built using the software provided for each mobile operating system and known as the Software Development Kit (SDK). If an application pretends to be available in several stores, to cover, for example, a greater number of users, it involves an extra effort on the part of the developers to learn and to master the particular conditions of each platform and SDK, factor that must be taken into account in the design stage and thus anticipate the scope of its deployment, since it will be the moment to decide what type of mobile application is intended to be made.

Another common feature of native applications is that their interfaces follow the standards and norms of the platform, and to be able to publish them in the corresponding store is necessary to obtain a developer license (Figura 40: Membership details Apple Developer Program).

3.2.2 Hybrid applications

Hybrid applications have almost all the features of the other two types of applications, web apps and native apps, combining the advantages and disadvantages of both.

An important feature of hybrid apps regarding to web apps is the overcoming about the limitations of hardware access. However, hybrid apps do not reach the rates of performance that native apps get.

It could be said that its main advantage of hybrid apps lies on the use of the development languages for web applications which means a fast development, as well as, the packaging or compilation through a wrapper or native container, which makes them work as if it were a native application (Ibarra, 2015).

The nature of these types of apps, as well as the way they are developed, allow them to be called: multiplatform applications, because with the same code we can build different applications. For example, for Android and iOS, and distribute them in each of their stores. From the user point of view, there is no difference with the native applications. Once in the corresponding store, hybrid apps could be installed and used in the same way that native apps.

On the other hand unlike web applications, hybrid apps allow access to the sensors through libraries (Cuello y Vitonne, 2015). Also, hybrid apps could have a visual design less similar to the operative system in which they run, but fortunately, today there is ways to use controllers and native buttons of each platform to get more level of aesthetic integration.

We would like to note that, we can found slight differences in their definitions in the literature. It is common in this kind of taxonomies. For instance, and according to Ramírez (2015), hybrid apps are native web mobile apps, and he defines them, as the set of applications that are not web apps themselves, nor pure native apps, but they have a native component that delegates the

run task in a embedded web browser, achiving with this some of native apps advantages.

3.2.3 Comparison

The following Figure tries to synthesize and summarize the features studied above (Figure 3: Summarize the features studied above (Gutierrez, Gutierrez and Gutierrez, 2017)):

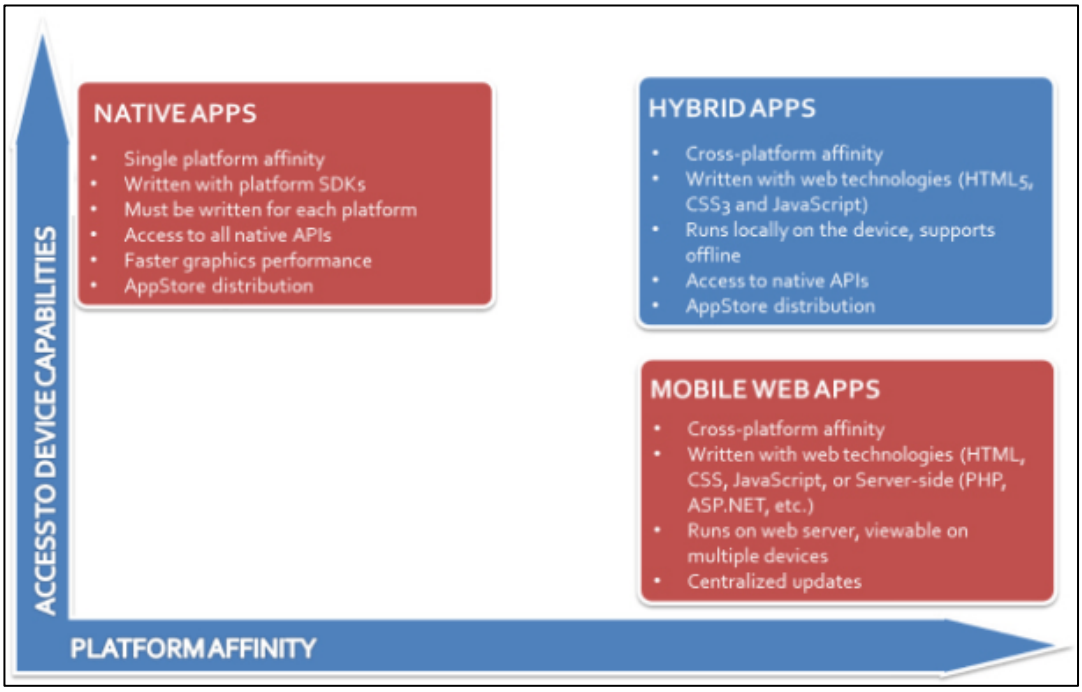

Figure 3: Summarize the features studied above (Gutierrez, Gutierrez and Gutierrez, 2017)

As you can see, the axes show two key factors. On one hand, the abscissa shows the degree of platform affinity and the ordinate shows the degree of access to device capabilities. It is observed that the natives make better use of the resources of that platform, and that both hybrids and web apps stand out for their multiplatform capability.

This implies that the first have better performance, especially in rendering graphics, at the same time need to be written in the platform programing language. On the contrary hybrid apps are developed in web languages,

which, in general, allows to more developers can become developers of this kind of apps.
Finally we would like to point out how the hybrid applications try to combine the best of the two worlds, and their properties could be summarized in three key aspects: hybrid apps not to have to be written in the programming language of each platform, hybrid apps can use some of the device's native APIs and hybrid apps could be uploaded to the store. Therefore, develop the mobile application of this TFM as an hybrid application is the best choice.

3.3 JavaScript

It is very important to choose a good programming language. In our ecosystem JavaScript is the best option as we will see in the next sections, due to JavaScript complies with our requirements. Our target it was that the same programming language was used in all parts of the system, so that the learning in each one of them was useful to the rest. Another target is to achieve that the student gets the market trend, and definitely, JavaScript is one of the most popular programing language nowadays. Next, we will briefly describe this programing language.

JavaScript is an untyped interpreted programming language with object-oriented capabilities. JavaScript resembles C, C++ and Java, from a syntactic point of view, but it is very different in other many aspects. JavaScript is a loosely typed language, which means that variables do not need to have a type specified. Objects in JavaScript map property names to arbitrary property values. The Object-Oriented inheritance mechanism of JavaScript is prototype-based, this is quite different from inheritance in C++ and Java. It supports numbers, strings, boolean as primitive data types, also includes support for array, date and regular-expression objects. (Flanagan, 2006, Flanagan, 2011).

Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production. It is used to make webpages interactive and provide online programs. The web browser runs scripts

written in JavaScript embedded within HTML web pages. The scripts are run by the client computer instead of the web server by moving the load to the client-side.

Initially only implemented client-side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases, and in non-web programs such as word processors and PDF software, and in runtime environments that make JavaScript available for writing mobile and desktop applications, including desktop widgets (En.wikipedia.org, 2017).

In our TFM we use JavaScript in each part of him, we use it in the server-side through NodeJS, and to develop our mobile and web app. Due to JavaScript, could be considered a robust and efficient general-purpose language, and according to Flanagan (Flanagan, 2011), the latest version of the language define new features for serious large-scale software development.

Finally, we would like to write this advice from Flanagan: “Web developers must learn: HTML to specify the content of web pages, CSS to specify the presentation of web pages, and JavaScript to specify the behavior of web pages”.

3.4 Hybrid Application Development Framework

Once we have decided that our implementation will be of the hybrid type, we must decide what technology to use to carry it out. One of the objectives of this TFM and the works of this nature is to get the student to acquire skills in the latest emerging technologies, as it will provide an extra point of competitiveness in the labor market.

My advisers advise me of the existence of Ionic (Ionic Framework, 2017), a very complete framework for the development of hybrid mobile applications, and that by its experience, it had the key requirements to be adopted by the

developer community, at the same time, that it had a great projection. Its forecast has been right (and took place at the end of the year 2015), and currently, and according to the information shown in its website, it is used by 5 million of developers in the worldwide. Also more than 4 million applications have been built using Ionic. These data are amazing, but it is still more amazing, if we take into account that it was born in 2013.

Therefore, and for the reasons above, Ionic was the framework chosen to develop our mobile application.

3.4.1 Ionic

Ionic Framework (En.wikipedia.org, 2017) is a complete open-source SDK for hybrid mobile app development. The original version was released in 2013 and built on top of AngularJS (**3.5.1 AngularJS Section**) and Apache Cordova **(3.4.2 Cordova)**. The more recent releases, known as Ionic 2 or simply "Ionic", are built on Angular. Angular is the underlying framework that powers Ionic.

Ionic (Ionic Framework, 2017) enables developers to build performant, high-quality mobile apps using familiar web technologies (HTML, CSS, and JavaScript). Apps can be built with these Web technologies and then distributed through native app stores to be installed on devices by leveraging Cordova.

In the words of one of its creator: Adam Bradle: “Ionic is that missing piece when creating native apps with web standards”. (The Official Ionic Blog, 2017) Ionic is focused mainly on the look and feel, or the UI interaction, of an app. Ionic tries to simplify one big part of the app development process: the front-end.

Actually uses Angular, however, in the future, Ionic plans to become more agnostic in order to support a broader variety of JavaScript frameworks.
Ionic is composed of the following elements:

3.4.1.1 CLI

The CLI (Ionic Framework, 2017), or command line interface, is a tool that provides a number of helpful commands to Ionic developers. In addition to installing and updating Ionic, the CLI comes with a built-in development server, build and debugging tools, and much more.

3.4.1.2 Components

Components in Ionic are reusable UI elements that serve as the building blocks for your mobile app. Components are made up of HTML, CSS, and sometimes JavaScript. Every Ionic component adapts to the platform on which your app is running. This is known as Platform Continuity.

3.4.1.3 Theming

Themes are sets of styles that get applied to an app. Ionic uses a light theme by default, but it also comes with a dark theme. In addition to theming, Ionic's Platform Continuity enables components to have platform-specific styles. This means the app's styles will change based on the platform (iOS, Android, etc.) on which it's being run, offering your users an experience with which they're familiar.

3.4.1.4 Navigation

Navigation works like a stack: push a page to the stack to navigate to it, and pop to go back. Modals and alerts can also be displayed by pushing them onto the navigation stack.

3.4.2 Cordova

Apache Cordova is a framework used for developing cross-platform mobile apps. Apache Cordova enables software programmers to build applications for mobile devices using CSS3, HTML5, and JavaScript instead of relying on platform-specific SDKs like those in Android, iOS, or Windows Phone (En.wikipedia.org, 2017). It enables wrapping up of CSS, HTML, and JavaScript code depending upon the platform of the device. It extends the features of HTML and JavaScript to work with the device. The resulting applications are hybrid.

The core of Apache Cordova applications use CSS3 and HTML5 for their rendering and JavaScript for their logic. Apache Cordova can be extended with native plug-ins, allowing developers to add more functionalities that can be called from JavaScript, making it communicate directly between the native layer and the HTML5 page. These plugins allow access to the device's accelerometer, camera, GPS, microphone... (En.wikipedia.org, 2017)

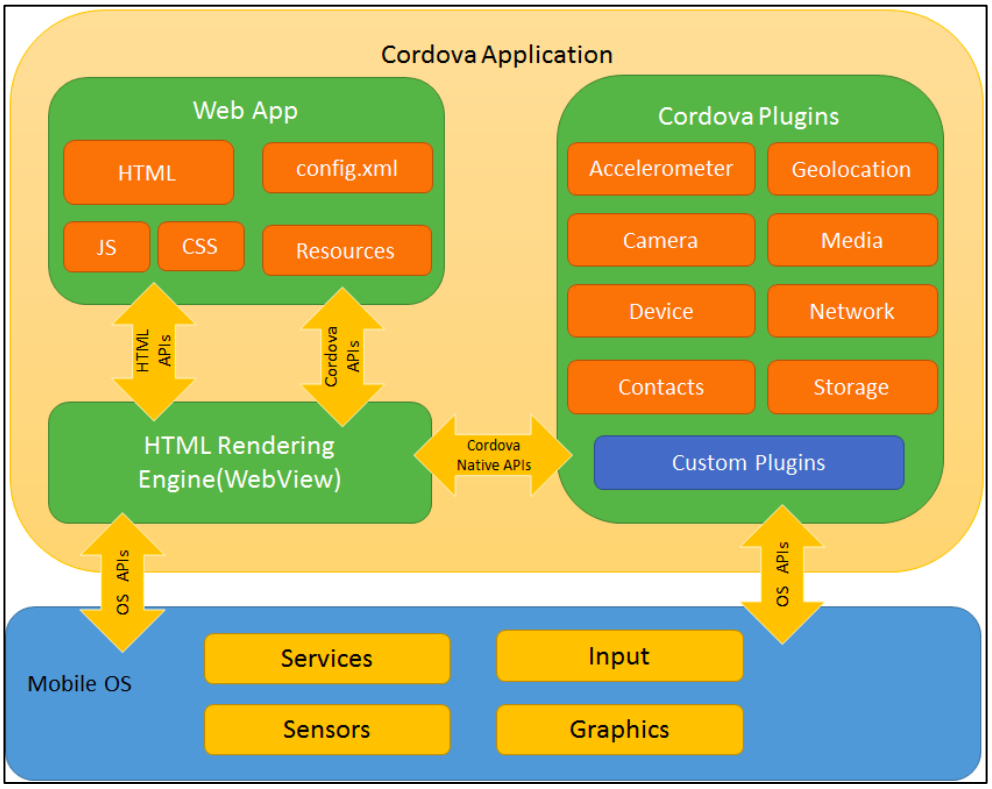

Figure 4: high-level view of the Cordova application architecture (Cordova.apache.org, 2017)

In the Figure 4: high-level view of the Cordova application architecture (Cordova.apache.org, 2017) you can see the Web App module, it is the part where our application code resides. This code has been built using Ionic (Seccion XXX). It is rendering in a WebView, and thanks to the Cordova and third-party plugins provide our application to access device capabilities.

Finally, Cordova provides us two basic workflows to create a mobile app. Cross-platform (CLI) workflow and Platform-centered workflow (Cordova.apache.org, 2017).

The first one builds an app that can run on different mobile operating systems such as Android or IOS, with little need for platform-specific development. The second one builds an app for a single platform and need to be able to modify it at a lower level. Obviously, we choose the first option in our TFM.

3.5 Web Application Development Framework

3.5.1 AngularJS

AngularJS (commonly referred to as "Angular.js" or "AngularJS 1.X") is a JavaScript-based open-source front-end web application framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single-page applications (En.wikipedia.org, 2017).

The architecture of AngularJS is shown in the Figure 5: Architecture of AngularJS (Weblogs.asp.net, 2017):

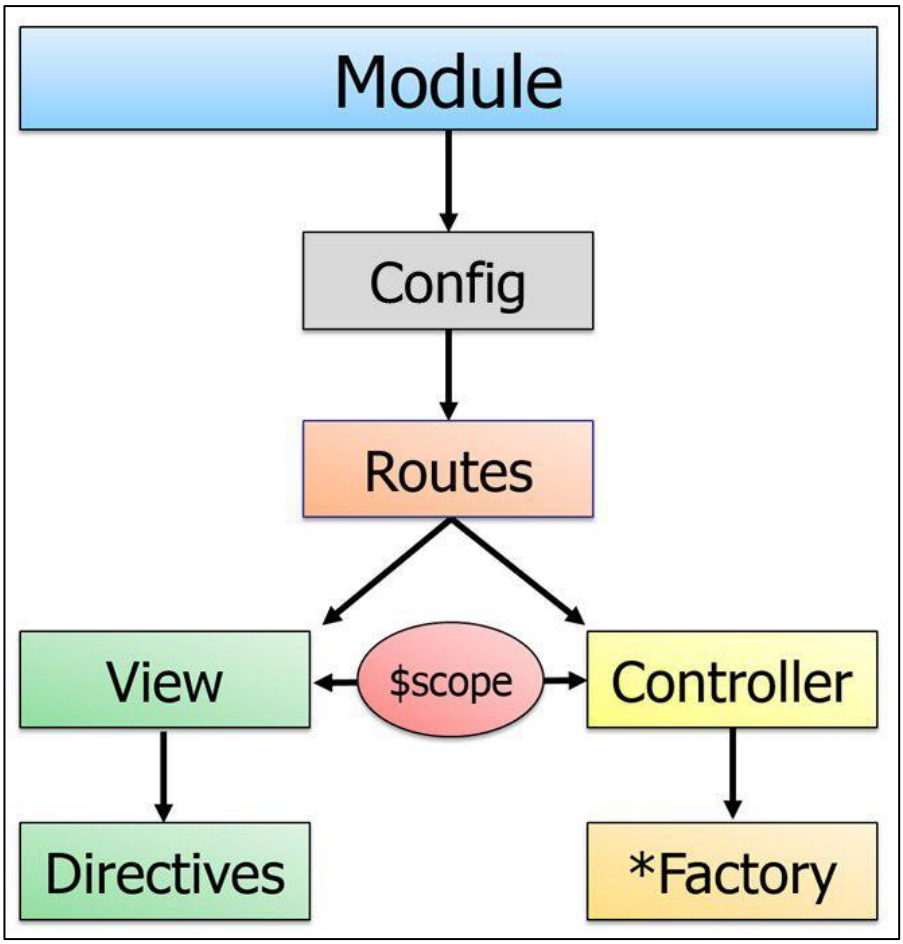

Figure 5: Architecture of AngularJS (Weblogs.asp.net, 2017)

- **View**, is the visual description of the screen or of part of her. It is written in HTML and CSS.
- **Controllers** manage the data flow and they implements the logic of the program.
- $**scope**, its function is to bind the controller with the view.
- **Directive**, allow us to create custom looking components directly on the view.
- **Factory** are objects that perform a given function that can be reused.
- **Routes** module helps your application to become a Single Page Application to navigate to different pages in your application, with no page reloading (W3schools.com, 2017).
- **Config** are files of configuration of our components AngularJS.
- **Modules** are containers for AngularJS components (Falace, 2017).

The JavaScript components complement Apache Cordova, the framework used for developing cross-platform mobile apps. It aims to simplify both the development and the testing of such applications by providing a framework for client-side model–view–controller (MVC) and model–view–viewmodel (MVVM) architectures, along with components commonly used in rich Internet applications. In 2014, the original AngularJS team began working on Angular (Application Platform) (En.wikipedia.org, 2017).

The AngularJS framework works by first reading the HTML page, which has additional custom tag attributes embedded into it. Angular interprets those attributes as directives to bind input or output parts of the page to a model that is represented by standard JavaScript variables. The values of those JavaScript variables can be manually set within the code, or retrieved from static or dynamic JSON resources (Ipfs.io, 2017).

AngularJS is the frontend part of the MEAN stack, consisting of MongoDB database, Express.js web application server framework, Angular.js itself, and Node.js server runtime environment (En.wikipedia.org, 2017).

3.6 API REST

(BBVAOpen4U, 2017) REST (Representational State Transfer) was a new vision in the development of projects and web services. It was conceived and specified by Roy Fielding, who is considered an international reference and one of the parents of the HTTP protocol. Roy Fielding shown us this novel approach, which has changed the software engineering since 2000, in his PhD thesis: “Architectural Styles and the Design of Network-based Software Architectures” (Fielding y Taylor, 2000).

In our opinion a simple definition of REST could be the following: “REST is any interface between systems that uses HTTP to get data or generate operations over that data in all possible formats, such as XML and JSON (BBVAOpen4U, 2017) REST is a simple alternative to other data exchange protocols such as SOAP (Simple Object Access Protocol) (Es.wikipedia.org,

2017). SOAP has much more capacity but in the same way is much more complex. It is possible that REST simplicity has been one of the determining factors of its expansion.

The following is a brief summary of the most interesting features:

- Stateless client / server protocol: each HTTP request contains all the necessary information to execute it, which allows neither client nor server need to remember any previous state to satisfy it.
- The most important operations related to the data in any REST system and in the HTTP specification are four: POST, GET, PUT and DELETE.
- Objects in REST are always manipulated from the URI.The URI is the unique identifier of each resource in that REST system.
- Uniform interface: data transfer in a REST system applies specific actions (POST, GET, PUT and DELETE) on the resources. This makes easier the existence of a uniform interface that systematizes the process with the information.
- Layer system: hierarchical architecture between components. Each of these layers performs a functionality within the REST system.
- Hipermedia: the hypermedia concept explains the ability of an application development interface to provide to the client and to the user with the appropriate links to perform specific actions over the data.

Therefore, the advantages for development begin in the separation between the client and server. The REST protocol completely separates the user interface from the server and the data storage improving its portability to other kind of platforms and allowing an independent evolution of the project parts.

As a clear consequence of this division or independence, arise several advantages, such as, that any development team can scale the product without excessive difficulties. It is also possible to make changes in the

database or migrate to another type of technology as long as the requests are made in the right way and it is possible to have the frontend and backend on different servers with the consequent increase of flexibility.

However, its feature most relevant could be its adaptability due to its independence of the type of platform or programing language. We can have several servers in several technologies (PHP, Java, Python, Node.js). The only indispensable requirement is that the answers to the requests are always made in the same information exchange language. The most common are XML o JSON.

3.7 JSON

In our opinion, the best definition of JSON can be found on its website (www.json.org) and says: “JSON is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate”. Also, it is a minimal format for structuring data. It is used primarily to transmit data between a server and web application.
JSON is syntax of braces, brackets, colons, and commas that is useful in many contexts, profiles, and applications (Intenational, 2013). JSON was first presented to the world in 2001.

JSON means JavaScript Object Notation, it is an agnostic format to the programming language, but it maintains many similarities or conventions that are known by the communities of developers of the most known programming languages, such as C, C ++, C #, Java, JavaScript, Perl, Python, among others. JSON is based on a subset of the literal notation of JavaScript objects, although currently, and because of its widespread adoption as an alternative to XML, it is considered an independent language format. It has become a very popular language for data exchange due to the properties described above. JSON is syntax of braces, brackets, colons, and commas that is useful in many contexts, profiles, and applications.

Therefore, the most outstanding advantages offered by the representation of the data in JSON format could be due to its flexibility and simplicity, both for the realization of queries and for the storage of records. JSON also has a similar syntax to other programming languages and it's independent.

JSON avoids defining different types for numbers. Programming languages often define several types, such as floats or integers, and which usually make it difficult to exchange data between different programming languages. JSON instead offers only the representation of numbers that humans use: a sequence of digits.

On the other hand, JSON instead provides a simple notation for expressing collections of name/value pairs. Most programming languages will have some feature for representing such collections, which can go by names like record, struct, dict, map, hash, or object. JSON also provides support for ordered lists of values. Because objects and arrays can nest, trees and other complex data structures can be represented. By accepting JSON's simple convention, complex data structures can be easily interchanged between incompatible programming languages. (ECMA, 2013).

Finally, in our opinion, another feature that make JSON a good choice for the future, is that it is not expected that the JSON grammar will ever change. This gives JSON, as a foundational notation, tremendous stability.

This JSON is used in our implementation, and could be an illustrative example of a **JSON** object:

```
{
    "_id": ObjectId,
    "phoneNumber": "616197423",
    "profile": {
      "_id": ObjectId,
      "name": "Alberto Andújar"
    },
    "paquetes": []
  }
}
```

3.8 Data Storage

3.8.1 MongoDB

MongoDB is a multi-platform database oriented to documents. This new paradigm means that instead of storing the data in registers, as in classic databases, we store the data in documents and the documents in collections. (Ver Figure 6: Documents and collections in MongoDB) (Bermejo, 2015).

The document is stored in the BSON format. BSON is a binary representation of the JSON data exchange format. The benefits of this technique is that it is possible to better map objects, allowing easier, faster and more flexible integration in other applications, at the same time, that it simplifies their management and coding. BSON was designed to be lightweight, traversable an efficient (Bsonspec.org, 2017).

Therefore, the basic storage unit is the document. We could say, if we are allowed the comparison with SQL systems, that a document is "equivalent" to the row. In the same way, it could be said that a collection is "equivalent" to the table.

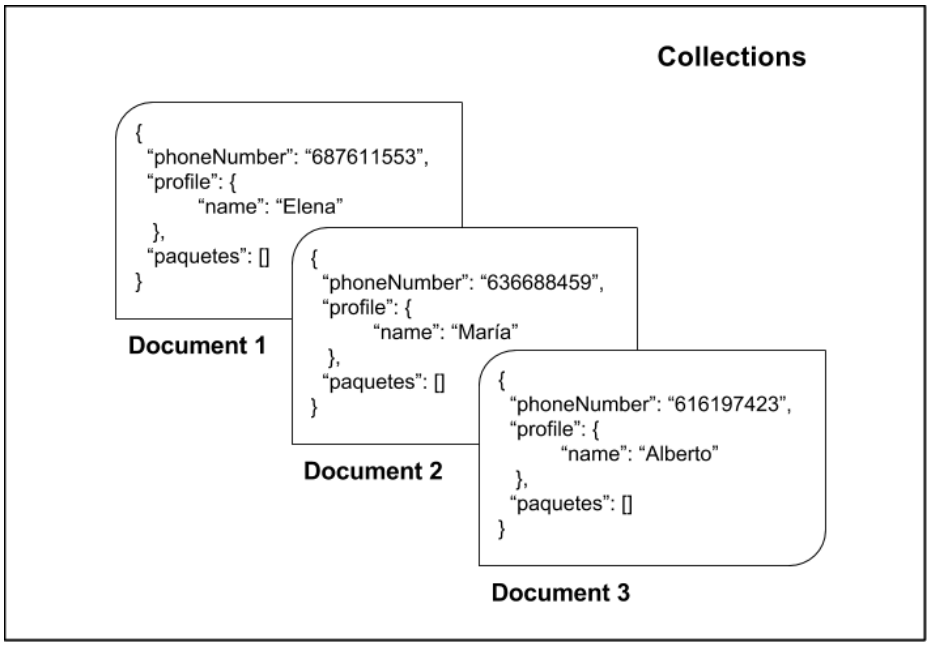

Figure 6: Documents and collections in MongoDB

One of the most important differences with respect to relational databases is that it is not necessary to follow a scheme. That is, documents from the same collection may have different schemes. In MongoDB, there is no standard scheme for working with data, but that does not mean that we can not relate it. Neither means that they do not have structure, it simply means that each document can have its own scheme.

Consequently, MongoDB is a documentary database where a unique key is used for each record. This kind of implementation allows, in addition to performing searches by key-value, to make more advanced queries about the content of the document and also a greater versatility in the queries itself.

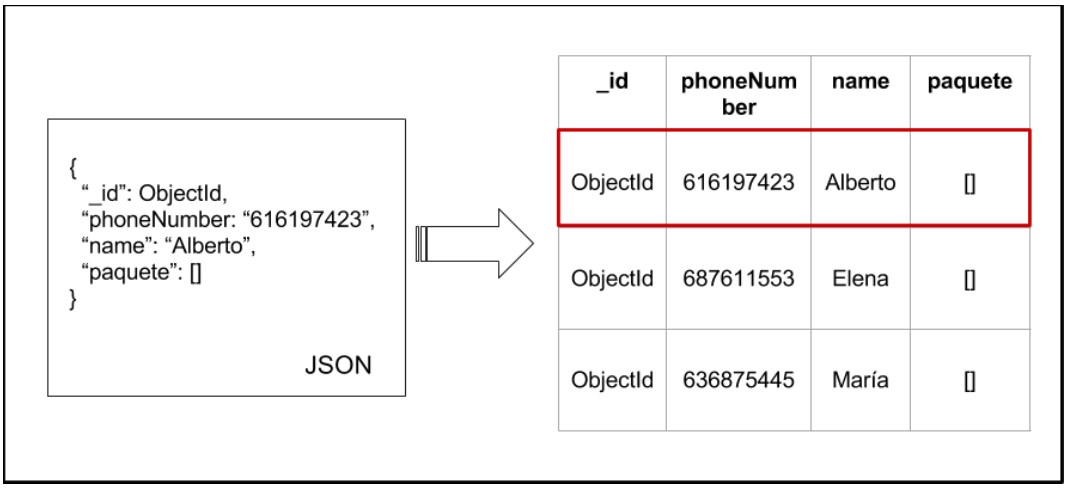

Figure 7: Key-value pairs in MongoDB

Figure 7: Key-value pairs in MongoDB shows that each key has a value and these values have different data types: strings, number, array. The keys are always string type. The key-value pairs are separated by commas and each key must be unique, being sensitive to the use of uppercase and lowercase characters. In addition, MongoDB assigns each document a unique identification number **(_id)** within its collection (Bermejo, 2015).

We insist that, the great advantage of the collections is their flexibility, since the documents do not have to have the same format. The collections are grouped in databases, being able to host several databases in the same

instance. By default, the size of the BSON documents is 16MB, if you want to exceed this amount you should use GridFS (En.wikipedia.org, 2017).

An interesting and especially useful feature, but not related to the above, is that MongoDB has a console. In the console, we can use the functions of MongoDB and many functions of JavaScript, we can also define variables, functions or use loops.

The reasons why we decided to use MongoDB instead of other traditional databases in this TFM are based on the facility to change the data schema of the stored documents, adapting them to the new requirements in a fast way.

The use of an API-REST in NodeJS is also a key factor, since it allows us to store almost directly the chosen one data exchange format. That is to say if we receive and read a JSON we can practically convert it without much effort to a document. We obtain a reduction of the development time, as well as a coherence in the design.

In addition, we can create indexes on all attributes thus accelerating searches, as well as maintaining a high availability and replication, features that can be very useful if the project scales.

4. Material y método

A la hora de desarrollar un proyecto software es imprescindible seguir una de las frases más conocidas de la programación: “divide and conquer”. Debemos conocer la naturaleza de los materiales y herramientas que vamos a utilizar, y seguir una metodología de desarrollo y aprendizaje que nos permita abarcar todo el espectro de las tecnologías necesarias. Nuestro sistema es complejo, tiene varias partes y va a necesitar de un desarrollo incremental de sus funcionalidades, así como una evaluación por fases que nos permita conocer si estamos en la dirección correcta, y a advertir los posibles errores lo antes posible.

En esta memoria nuestro punto de partida será una visión global de la arquitectura final del sistema y una división del diseño e implementación del sistema en tres fases o hitos, que pretende recoger, de manera más o menos fidedigna, el desarrollo temporal del proyecto.

No debemos olvidar que existe un primer estadio, bastante gris, desde el punto de vista del resultado final, y que suele pasar desapercibido en el desarrollo del proyecto, y que se podría resumir como la configuración y capacidad de uso del entorno de trabajo por parte del programador.

En primer lugar, es necesario instalar y aprender a usar los diferentes frameworks de desarrollo, así como adquirir destrezas en el lenguaje o los lenguajes de programación seleccionados. Para ello, el método general de aprendizaje ha sido utilizar los tutoriales proporcionados por los propios creadores de la tecnología, y que suelen estar disponibles en sus páginas webs corporativas. Por otro lado, se ha recurrido al aprendizaje personal a través de cursos online, tanto gratuitos como de pago, de las conocidas plataformas de *learning and teaching online* como CodeAcademy (Codecademy, 2017) o Udemy (Udemy, 2017).

Sirva de forma ilustrativa la Figura 8: Cursos realizados en Udemy[4]. Aunque hize un esfuerzo económico para realizar estos cursos, tuvieron una relevancia en mi aprendizaje[5].

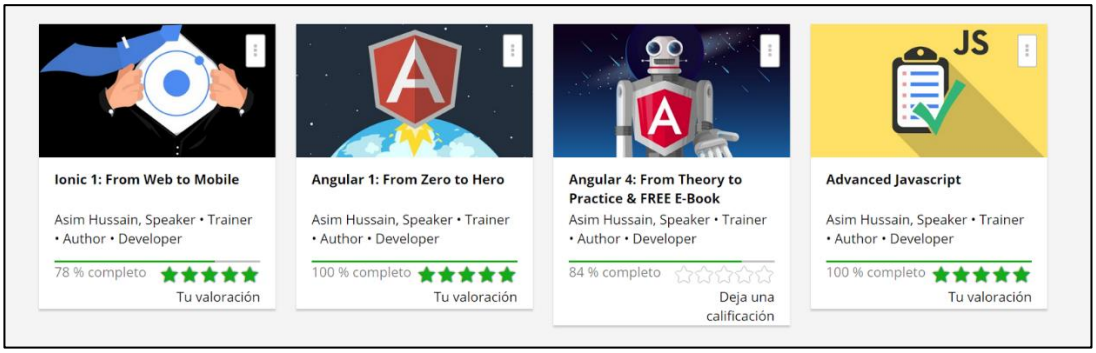

Figura 8: Cursos realizados en Udemy

En segundo lugar, y hasta alcanzar una familiarización con las tecnologías, lo habitual es trabajar de forma independiente con cada uno de las frameworks y tecnologías que lo componen, solventamos los problemas típicos de configuración, dependencias, plugins, entorno y aprendiendo la nomenclatura específica.

Una vez alcanzado ese nivel mínimo de destreza en cada una de las partes, el método habitual de trabajo es hacer un desarrollo en local, de forma que en un único ordenador tenemos instalado todos los elementos. En nuestro caso, tendríamos un servidor NodeJS, el framework de desarrollo de la aplicación móvil, herramientas asociadas (simulador del smartphone), y la aplicación web mediante el acceso a nuestro localhost. De esta manera todo el proceso, incluido el de la comunicación entre las partes, tiene lugar en la misma máquina.

El objetivo final es conseguir una réplica similar en local de la versión real online, lo que nos permite superar dificultades y familiarizarnos con los entornos consiguiendo una pseudo versión de desarrollo, que nos permitirá

[4] Como se puede observar en la Figura 8: Cursos realizados en Udemy, actualmente estoy aprendiendo la última versión de AngularJS.

[5] El total de horas invertidas en estos cursos supera las 100 horas.

realizar cambios y corregir errores antes de incorporarlos al sistema real o versión de producción.

A fin de cuentas, y como ocurrirá en los siguientes apartados, no son más que expresiones diferentes de la máxima divide y vencerás.

4.1 Diseño e Implementación

La Figura 9: Arqutectura de La Perspectiva del Sabio representa la arquitectura del sistema y define al más alto nivel sus partes más relevantes. Observamos en la Figura 9: Arqutectura de La Perspectiva del Sabio tres partes diferenciadas claramente: el servidor, donde se encuentra el almacenamiento de la información en MongoD, nuestra API REST en NodeJS, la aplicación móvil construida en Ionic, y la aplicación web construida en AngularJS. Representamos el intercambio de ficheros entre la aplicación móvil y el servidor con iconos y texto descriptivo, y el intercambio de paquetes de datos median JSON.

En el resto de secciones se describen para cada una de las tres fases: prototipo, fase beta y fase beta abierta, las tareas realizadas, así como lo más relevante respecto al desarrollo e implementación.

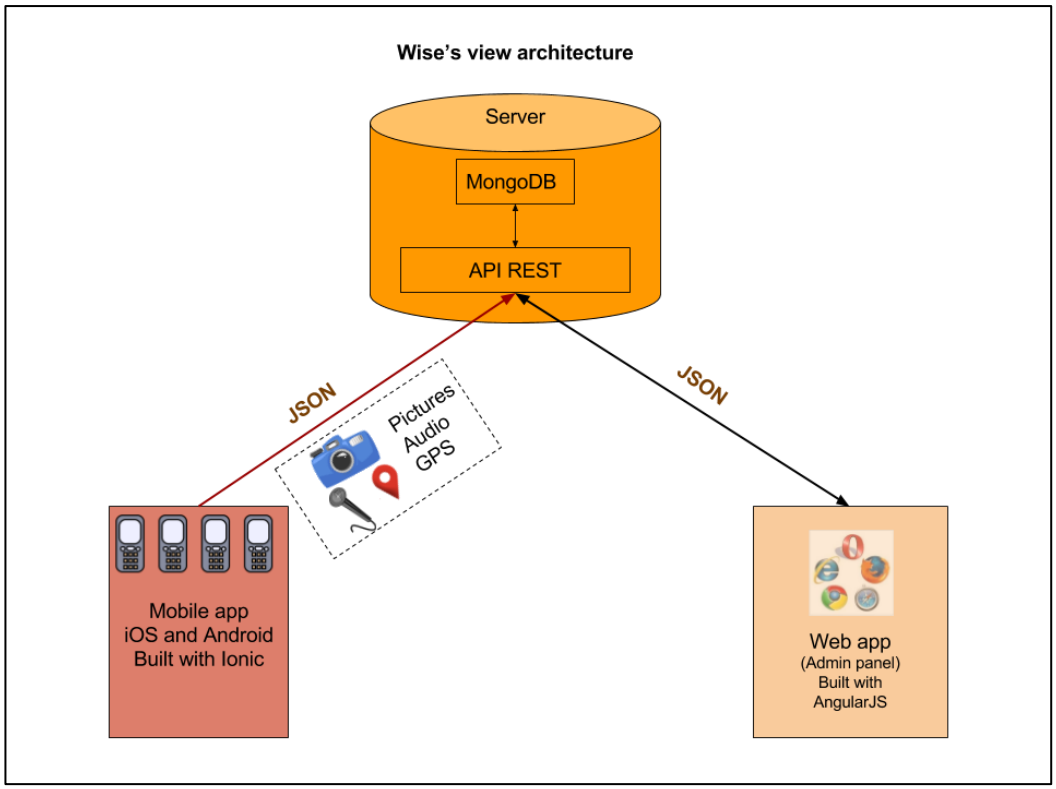

Figura 9: Arqutectura de La Perspectiva del Sabio

4.1.1 Primera fase prototipo

La hemos denominado prototipo porque al final de esta fase existe una funcionalidad mínima de todas las partes del sistema que puede ser ejecutada en real, pero que aún dista mucho de la versión final. Básicamente es una versión simplificada del resultado final, pero que utiliza todos sus elementos.

Este primer hito supera las dificultades técnicas de configuración e instalación en la versión de producción colocándonos en una posición de dominio y experiencia sobre el conjunto, desde donde será más sencillo ir añadiendo el resto de funcionalidades, así como mejorando el aspecto final del proyecto.

4.1.1.1 Servidor

Como dijimos el servidor se aloja en Microsoft Azure. Esta empresa perteneciente al grupo Microsoft, se define como una plataforma de cloud-computing para el desarrollo, testeo y despliegue de aplicaciones web.

Ofrece una amplia cantidad de servicios que la hacen adecuada para empresas pequeñas, medianas y grandes.

En nuestro caso particular aloja una máquina virtual del tipo DS1 Standard (Ver Figura 10: Principales máquinas virtuales de Microsoft Azure), sus características más relevantes son que tiene un único procesador, 3,5 GB de memoria RAM, y una capacidad de almacenamiento de 50GB. El sistema operativo instalado es un Ubuntu 16.04.2 LTS xenial.

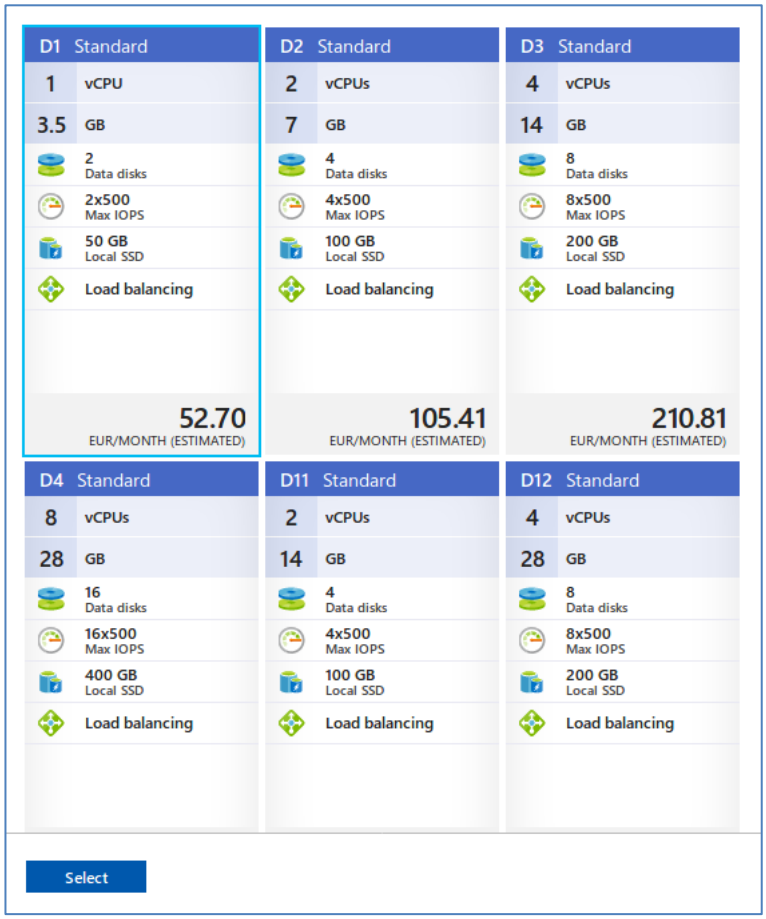

Figura 10: Principales máquinas virtuales de Microsoft Azure

De los diferentes aspectos que hemos configurado nos vamos a detener solo en los más relevantes. Uno de ellos ha sido la configuración de los puertos. A parte de los comunes, 80 (http) para ofrecer la web, y el 22 (ssh) para acceso remoto mediante consola para tareas de mantenimiento, necesitamos un puerto más para poder conectar el servidor con la aplicación web y con la aplicación móvil. El puerto que debe permitir la entrada y salida de datos es el 3000, y la selección de este puerto no corresponde a ningún

criterio específico, pero suele ser el seleccionado por la comunidad en sus ejemplos. Este puerto es escuchado por nuestra API REST NodeJS que se está ejecutando en el servidor para recibir la información enviada por la aplicación móvil. De la misma forma ocurre cuando interactuamos con la aplicación web y, por ejemplo, editamos la información. Lo relevante de este aspecto de la configuración es que Azure proporciona un cortafuego interno, y por defecto no tiene este puerto abierto, ni tampoco el destinado al ssh. Para conseguir esta conexión debemos hacerlo explícitamente en la sección de reglas del panel de control (ver Figura 11: Reglas de entrada de red en el panel de control de Microsoft Azure), es decir, no lo podemos configurar desde el Sistema Operativo. Al subir esta versión prototipo al servidor, no éramos capaces de conectar con él, y nos costó mucho tiempo y horas de depuración comprobar que el problema era externo a nuestro desarrollo, ya que no conocíamos esta peculiaridad del panel de administración de Azure.

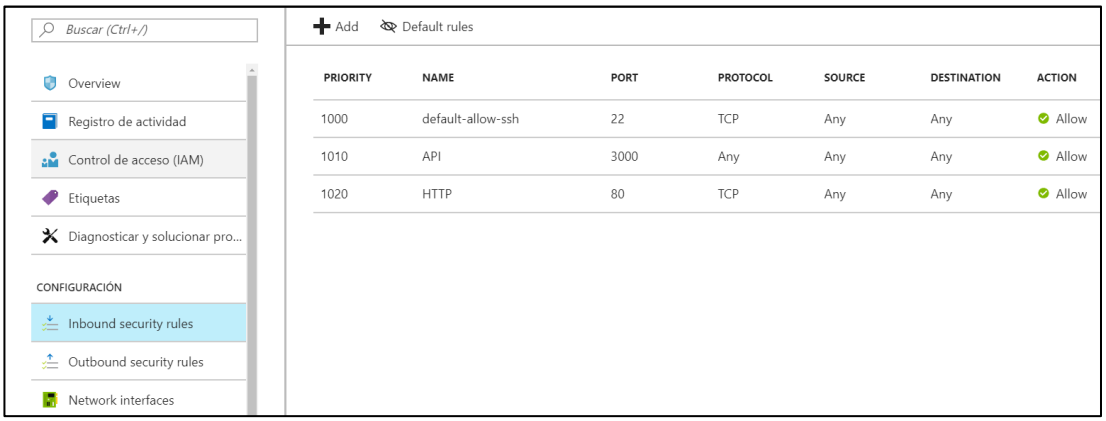

Figura 11: Reglas de entrada de red en el panel de control de Microsoft Azure

De otra parte, Microsoft Azure nos proporciona una IP pública (52.232.30.160), por la que accedemos a la web app. Finalmente, no se registro un nombre de dominio por la ausencia de presupuesto.

4.1.1.1.1 Instalación y configuración de NodeJS

NodeJS es el entorno de ejecución de nuestro código en javascript. De manera general podemos decir que nuestro código implementa una serie de funciones que definen nuestra API REST, así como un conjunto más amplio de funciones internas que implementan la lógica del programa y tratan con

los datos. Las funciones de la API pueden ser llamadas por las diferentes partes del sistema, como la aplicación móvil o la web, ya sea de forma manual o automática. De forma manual, por ejemplo, cuando accionamos un botón en una de ellas, o de forma automática, cuando procedemos visualizar la información al acceder al panel de administración. Las funciones de la API reciben un JSON como parámetros de entrada/salida. Esto permite el intercambio de datos entre las partes. A continuación se muestra parte de uno de los JSON de nuestro sistema:

```
{
  "_id" : ObjectId(),
  "phoneNumber" : "616197423",
  "profile" : {
      "_id" : ObjectId(),
      "name" : "Alberto Andújar"
  },
  "paquetes" : [
      {
          "_id" : ObjectId(),
          "location" : {
              "long" : "-3.593321857439841",
              "lat" : "37.20556118753419"
          },
          "created_at" : ISODate(),
          "data" : [
              {
                  "_id" : ObjectId("59b56b6ebd99c7f80a73c843"),
                  "name" : "6161974231505061742140.jpg",
                  "url" : "URL_Path6161974231505061742140.jpg",
                  "type" : "Image"
              },
              {
                  "_id" : ObjectId(),
                  "name" : "6161974231505061742651.wav",
                  "url" : "URL_Path/6161974231505061742651.wav",
                  "type" : "Audio"
              }
          ]
      }
    ],
  "upload_at" : ISODate()
}
```

La instalación de NodeJS puede realizarse de diferentes maneras. Una de ellas es utilizando el gestor de paquetes de Ubuntu, pero el problema que hemos detectado durante el desarrollo de este TFM es que NodeJS se actualiza muy frecuentemente y es bastante complejo instalar nuevas versiones, mantener las actuales y gestionar las antiguas. Debido a estos motivos, en nuestra opinión, la mejor forma de instalar NodeJS es utilizando NVM (Node Version Manager) (GitHub, 2017), una herramienta que nos permite gestionar más fácilmente las diferentes versiones de NodeJS. Además, su instalación en el servidor es sencilla y se realiza mediante la ejecución de un script desde el bash. El siguiente comando ejecutado curl (En.wikipedia.org, 2017) descarga y ejecuta dicho script:

```
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.4/install.sh | bash
```

Una de las ventajas de utilizar NVM es que nos permite la posibilidad de tener instaladas varias versiones de NodeJS en el sistema, además de permitimos, elegir una la versión que queremos ejecutar. La versión utilizada en este TFM es 6.10.3 LTS. Nos gustaría destacar aquí algunos de los comandos más útiles y usados de NVM, como por ejemplo:

- 'nvm ls-remote': Para ver las versiones disponibles para instalar.
- 'nvm install vX.X.X': Para instalar la versión de Node X.X.X.
- 'nvm use vX.X.X': Configura el sistema para utilizar la versión de Node X.X.X.

4.1.1.1.2 Instalación y configuración de Apache

Los servidores web o servidores HTTP son programas informáticos ejecutados en el lado del servidor que almacenan, procesan y sirven páginas web a los clientes. Sus funciones principales son procesar peticiones del cliente vía HTTP, generar respuestas, y realizar conexiones bidireccionales o unidireccionales con ellos, ya sean síncronas o asíncronas.

Existen diferentes servidores web como, por ejemplo, Apache, Zeus, Litespeed, Nginx o Lighttpd (Hostingdiario.com, 2017). De los anteriores, sin duda alguna, el servidor HTTP Apache es el servidor web más utilizado en el

mundo, aunque en los últimos años Nginx está experimentando una acelerada expansión y está siendo muy bien acogido por los desarrolladores, no obstante, lejos aún de Apache.

Por esta razón, en nuestro proyecto hemos decidido usar Apache como servidor web. Será el encargado de alojar nuestra aplicación web (panel de administración).

El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual (Es.wikipedia.org, 2017).

Apache está disponible dentro de los repositorios de software por defecto de la mayoría de las distribuciones Linux, incluida Ubuntu. Por tanto, es sencillo de instalar en nuestro servidor utilizando las herramientas habituales de administración de paquetes.

La instalación no requiere más de dos comandos. Como siempre, es recomendable actualizar el índice de paquetes por si hubiera algún cambio no reflejado en local, para posteriormente, instalar el paquete en cuestión: apache2.

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

Al final del proceso de instalación en Ubuntu 16.04 se inicia Apache automáticamente. Por tanto, el servidor web ya está en funcionamiento. Podemos comprobar que la instalación ha sido correcta accediendo a nuestro localhost o a nuestra IP remota, y debemos encontrar una página similar a la de la Figura 12: Apache inicial.

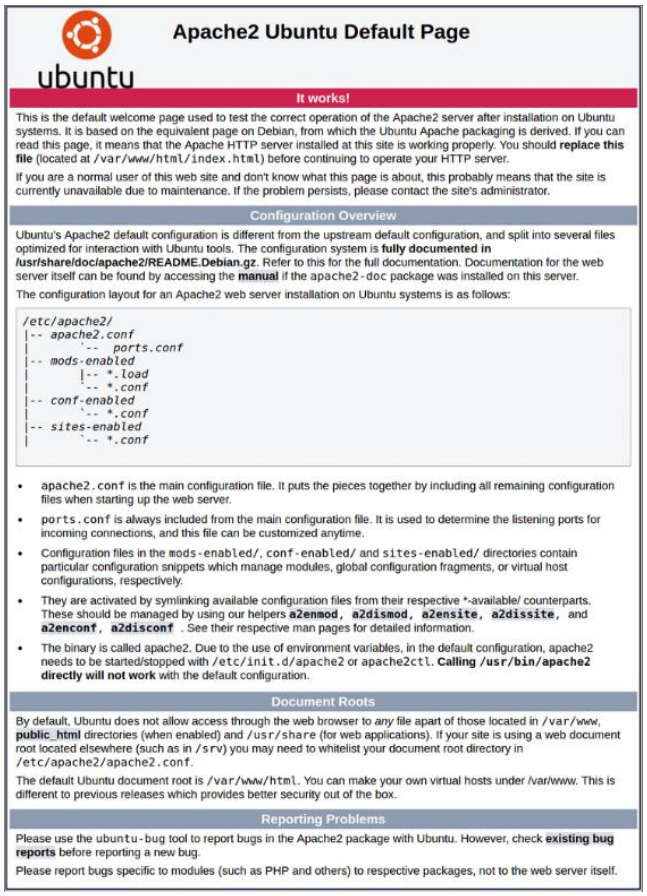

Figura 12: Apache inicial

4.1.1.1.3 Instalación y configuración de MongoDB

Aunque MongoDB (3.8.1 **MongoDB**) está actualmente incluido en el repositorio oficial de paquetes de Ubuntu, no nos proporciona su versión más actualizada. Por lo tanto, es recomendable instalar este software a través del repositorio mantenido por MongoDB. Su instalación es sencilla, y puede ser agregado como un nuevo repositorio a nuestro servidor. Mostramos a continuación el conjunto de comandos para completar la instalación usando la herramienta APT (Advanced Packaging Tool).

Primero importamos la llave para el repositorio oficial de MongoDB (Digitalocean.com, 2017):

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927
```

A continuación, debemos agregar los detalles del repositorio de MongoDB de tal manera que apt pueda saber de dónde descargar los paquetes. Ejecutando el siguiente comando creamos la lista para MongoDB:

```
$ echo "deb http://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.2 multiverse" |
sudo tee /etc/apt/sources.list.d/mongodb-org-3.2.list
```

Después de agregar los detalles del repositorio, debemos actualizar la lista de paquetes e instalar MongoDB:

```
$ sudo apt-get update
$ sudo apt-get install mongodb-org
```

Con la secuencia anterior de pasos tendríamos MongoDB listo para usar en nuestro servidor; no obstante, antes deberíamos configurarlo como un servicio de Ubuntu, por razones de eficiencia y estabilidad.

4.1.1.1.3.1 Configuración de MongoDB como servicio de Ubuntu

Un servicio o demonio es un programa que se ejecuta en segundo plano, fuera del control interactivo de los usuarios del sistema ya que carecen de interfaz con estos (Medium, 2017). El término demonio se usa fundamentalmente en sistemas UNIX y basados en UNIX, como GNU/Linux o Mac OS X (Es.wikipedia.org, 2017). En Windows y otros sistemas operativos se denominan servicios porque fundamentalmente son programas que ofrecen servicios al resto del sistema.

Los demonios o servicios se suelen iniciar al arranque del sistema, cuyo su cometido suele ser ofrecer servicios a otros programas, encargarse del hardware, atender peticiones del sistema, etc.

Las razones para configurar MongoDB como un servicio son relativas a la necesidad de que debe estar funcionando automáticamente cuando la máquina arranque, ya que deberá ofrecer los documentos y colecciones al resto del software. La configuración como servicio nos permitirá olvidarnos

de su puesta en marcha, por ejemplo, ante un reinicio inesperado o programado por Microsoft Azure[6].

Para lanzar apropiadamente MongoDB como un servicio de Ubuntu, hemos seguido los pasos recomendados por la conocida empresa de hosting Digital Ocean (Digitalocean.com, 2017).

Primeramente, creamos un archivo unitario que describa el servicio. Un archivo unitario le "dice" al systemd cómo manejar el recurso. Crearemos un archivo de configuración llamado mongodb.service en el directorio /etc/systemd/system, con el editor de textos nano:

```
$ sudo nano /etc/systemd/system/mongodb.service
```

Con el siguiente contenido:

/etc/systemd/system/mongodb.service

```
[Unit]
Description=High-performance, schema-free document-oriented
database
After=network.target

[Service]
User=mongodb
ExecStart=/usr/bin/mongod --quiet --config /etc/mongod.conf

[Install]
WantedBy=multi-user.target
```

Este archivo tiene la siguiente estructura (Digitalocean.com, 2017):

- La sección Unit contiene un resumen (por ejemplo una descripción legible para el humano que describe el servicio MongoDB) así como las dependencias que deberán existir antes de que el servicio inicie. En nuestro caso, MongoDB depende de que la red esté disponible, por lo tanto agregamos network.target aquí.
- La sección Service indica cómo deberá iniciar el servicio. La directiva User especifica que el servicio deberá ejecutarse bajo el usuario

[6] Recordemos que en esta empresa teemos nuestro servidor virtual.

MongoDB, y la directiva ExecStart inicia el comando para arrancar el servidor MongoDB.

- La última sección, Install, le dice a systemd cuándo el servicio debe iniciar automáticamente. Por su parte, multi-user.target es un sistema de secuencias de arranque estándar, que significa que el servicio se ejecutará automáticamente al arrancar.

4.1.1.1.4 API REST: versión inicial

A esta altura del proyecto tenemos el servidor listo para desarrollar y probar la primera versión de nuestra API. Es interesante advertir que nuestro código está escrito en JavaScript y que será ejecutado en el lado del servidor, lo que de alguna manera evidencia la progresión de JavaScript, que en sus inicios era utilizado solo del lado del cliente. En nuestra opinión, esto ha sido posible gracias a entornos de ejecución como NodeJS.

En NodeJS tienen especial relevancia los denominados módulos, que se pueden definir como un conjunto de ficheros que incluyen desarrollos con funcionalidades comunes que podemos incluir y usar fácilmente desde nuestro código, con la consiguiente aceleración del desarrollo. Los módulos utilizados en esta primera versión son los siguientes:

- **express** (v4.13.3): **Express.js** es un framework para NodeJS que sirve para ayudarnos a crear aplicaciones web en menos tiempo ya que nos proporciona funcionalidades como el enrutamiento, opciones para gestionar sesiones y cookies, y un largo etcétera. **Express.js** está basado en **Connect**, que a su vez es un framework basado en http para NodeJS. Podemos decir que **Connect** tiene todas las opciones del módulo http que viene por defecto con Node y le suma funcionalidades. A su vez, **Express** hace lo mismo con **Connect**, con lo que tenemos un framework ligero y muy rápido (Eneko, 2017).

- **mongoose** (v4.7.2): **Mongoose** es una herramienta que nos ayuda con el modelado de datos de **MongoDB** para **NodeJS**. Incluye tipos,

validación, construcción de consultas y un largo etcétera (GitHub, 2017).

- **body-parser** (v1.14.1): Body-parser es un middleware para el parseo del "body" que nos llega por la url en la llamadas a la API REST (GitHub, 2017).
- **express-fileupload** (v0.0.7): middleware para la subida de ficheros en **NodeJS**. Cuando se sube un fichero a la API REST, el fichero estará accesible en **req.files** (GitHub, 2017)
- **moment** (v2.17.1): **Moment** es un módulo de NodeJS que nos ayuda a parsear, validar, manipular y mostrar fechas y horas en Javascript (Momentjs.com, 2017).
- **mkdirp** (v0.5.1): **Mkdirp** es un módulo de NodeJS que nos permite crear directorios desde nuestro NodeJS (GitHub, 2017).

Una vez definidos los módulos necesarios, procedemos a explicar lo más destacado de nuestro código para "levantar" nuestra API-REST en el puerto 3000. El resultado final se mantendrá como parte fundamental del sistema, ya que estará siempre en ejecución en el servidor.

En la primera parte de nuestro código necesitamos incluir los módulos necesarios para completar nuestra implementación. En esta zona añadiremos más módulos en el futuro si fuesen necesarios. Utilizamos la directiva **require ('nombre del módulo')** (Figura 13: Directiva require).

```
var express = require('express');
var bodyParser = require('body-parser');
var mongoose = require('mongoose');
var app = express();

var fileUpload = require('express-fileupload');
```

Figura 13: Directiva require

En la segunda parte procedemos con la iniciación y la configuración del **MongoDB.** Para ello utilizamos **mongoose** para conectarse a la base de datos del servidor (Figura 14: Iniciación y configuración de MongoDB).

```
mongoose.Promise = global.Promise;
// Connection to DB
mongoose.connect('mongodb://localhost/wiseView', function(err, res) {
    if(err) throw err;
    console.log('Connected to Database');
});
```

Figura 14: Iniciación y configuración de MongoDB

En la tercera parte configuramos **express** (Figura 15: Carga y configuración de middlewares para express). En primer lugar, definimos la variable **AllowCrossDomain** y permitimos los métodos fundamentales del protocolo HTTP, a saber: **GET**, **PUT**, **POST** y **DELETE** (conocidos como verbos principales de una API REST). Debemos permitir que los métodos estén disponibles para cualquier IP de origen, ya que nuestra aplicación móvil estará instalada, esperemos, en muchos móviles y con diferentes operadores. Esta configuración se consigue con la siguiente instrucción, donde el asterisco (*) significa cualquier IP:

```
res.header(“Access-Control-Allow-Origin”, “*”);
```

En segundo lugar procedemos a cargar y configurar los middlewares que necesita el módulo express (Figura 15: Carga y configuración de middlewares para express). Para ello usamos la variable **app** (donde teníamos cargado el módulo express) mediante la función .use (*NombreMiddleware*):

```
//CORS middleware
var allowCrossDomain = function(req, res, next) {
    res.header("Access-Control-Allow-Origin", "*");
    res.header('Access-Control-Allow-Methods', 'GET,PUT,POST,DELETE');
    res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

    next();
};

// Middlewares
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());
app.use(fileUpload());
app.use(allowCrossDomain);
```

Figura 15: Carga y configuración de middlewares para express

Podríamos decir que las tres secciones anteriores son estáticas, en el sentido de que variarán poco o nada desde esta primera fase de prototipo a la tercera. Sin embargo, la siguiente sí lo hará debido a que define el cuerpo de la API. Por su parte, Express nos permite enrutar las peticiones que nos llegan. Simplificando mucho el mecanismo de enrutado, podríamos decir que nos permite dirigir la query a un fichero.js (controlador) previamente creado y que contiene la implementación del comportamiento de cada una de las funciones de nuestra API-REST. (Figura 16: Cuerpo de la API).

```
var router = express.Router();

// Index - Route
router.get('/', function(req, res) {
    res.send("Bienvenido a la API de Wise's View");
});

app.use(router);

// API routes ionic
var api = express.Router();

api.route('/users')
    .get(UserCtrl.findAll);

app.use('/api', api);
```

Figura 16: Cuerpo de la API

Por último, solo nos queda levantar nuestra API REST en el puerto 3000 y mantenerla esperando asíncronamente peticiones 24 horas al día todos los días de la semana (Figura 17: Comando para "levantar" nuestra API REST).

```
// Start server
app.listen(3000, function() {
    console.log("Node server running on http://localhost:3000");
});
```

Figura 17: Comando para "levantar" nuestra API REST

Todo lo anterior nos permite levantar y comprobar que nuestra API-REST está bien configurada y operativa, pero ahora, sobre este esqueleto, debemos implementar las funcionalidades definidas para esta fase: la comprobación del login y la recepción del paquete de datos con la fotografía.

Para ello nos dirigimos a la sección que define el cuerpo de la API-REST y procedemos a crear dos rutas: '**/checkUser**' y '**/addPaquete**', donde respectivamente implementaremos ambas funcionalidades. Siguiendo la estructura y organización del código, hemos creado dos controladores con una función cada uno en dos ficheros distintos: el controlador **UserCtrl** en el fichero '**controllers/userCtrl.js**', y el controlador **PaqueteCtrl** en el fichero '**controllers/paqueteCtrl.js**', que serán los destinos de las dos rutas anteriores.

Describimos brevemente su contenido:

- El controlador **UserCtrl** implementa la función **checkUser(),** que comprueba si el usuario está registrado en el sistema. Para ello comprueba que el usuario y su credencial de acceso sean correctas. En caso negativo, devuelve un error.

- El controlador **PaqueteCtrl** implementa la función **addPaquete(),** que recibe y almacena el fichero de imagen. Usa el módulo express fileUpload() y la imagen se recibe en **req.files**, al mismo tiempo que es almacenada en una ruta segura de nuestro servidor. La imagen está asociada con un nombre de usuario y la fecha y hora de recepción que ejerce de identificador único. En la base de datos se añade este nuevo documento a la colección de dicho usuario. (Ver Sección **4.1.1.1.5 Esquema MongoDB**).

Nos gustaría señalar que el nombre de la función es más general que simplemente **uploadFoto()**, nombre más adecuado para lo que hace actualmente la función. Esto se debe a haber realizado un buen diseño y estudio del problema, que deriva en un desarrollo incremental y, por tanto, prevemos que en posteriores fases de la implementación la información a almacenar aumente.

4.1.1.1.5 Esquema MongoDB

El modelo de esta primera fase consiste en conseguir definir dentro de MongoDB los esquemas adecuados para soportar un usuario. Lo hemos denominado **userSchema** (ver parte inferior Figura 18: Esquemas definidos en MongoDB). Debe almacenar el nombre de usuario y su contraseña, así como un array que almacenará los datos proporcionados por la app móvil. Por tanto, necesitamos definir el tipo de datos de este array como un nuevo schema, al que hemos llamado ***paqueteSchema*** (ver zona central Figura 18: Esquemas definidos en MongoDB).

PaqueteSchema contiene la fecha y hora de creación de la imagen en el smartphone, y la fecha y hora en la que ha sido recibida por el servidor.

Pensando en el futuro, tomamos la decisión de definir el array ***data*** como un schema (**dataPaqueteSchema**) que almacena la URL de la imagen (ver zona superior de la Figura 18: Esquemas definidos en MongoDB). La razón de que sea un schema se debe a que tal vez en el futuro debamos almacenar otros campos además de la URL y la elección de un array se justifica porque quizás sería necesario subir más de una imagen en un mismo paquete.

```
var dataPaqueteSchema = new Schema(
    {type: { url: String, required: true }
}, {strict: false});

var paqueteSchema = new Schema({
    created_at: { type: Date, required: true },
    upload_at: { type: Date, default: Date.now  },
    data: [dataPaqueteSchema]
}, {strict: false});

var userSchema = new Schema({
    userName:    { type: String, required: true },
    password:    { type: String},
    paquetes:  [paqueteSchema]
}, {strict: false});
```

Figura 18: Esquemas definidos en MongoDB

Una de las características que nos gustaría señalar y que sostienen la decisión de usar MongoDB como BD es el uso de '**strict:false**' en la definición de un Schema. **Strict** nos permite dos opciones: si es igual a **true,** obliga a que el documento que se inserte tenga solo los campos definidos previamente en el modelo; por contra, si es igual a **false,** permite la libertad de añadir en caliente campos nuevos a los definidos previamente en el modelo, conforme se van necesitando con el consiguiente beneficio, flexibilidad y adaptabilidad a los nuevos requerimientos del sistema.

4.1.1.2 Aplicación web

La aplicación web o panel de administración se basa en AngularJS (sección 3.5.1 AngularJS). AngularJS permite crear una estructura propia de directorios y ficheros, es decir, deja en manos del programador la modulación del software y la organización de los ficheros. Esto es habitual en cualquier framework de desarrollo. No obstante, la experiencia adquirida tanto a nivel académico como profesional nos advierte de que el uso de generadores de código o el uso templates que definen un guion para la ordenación del software suelen reportar beneficios a la larga, máxime cuando se espera que el proyecto en cuestión alcance una envergadura considerable. Es más, a nivel personal, considero como uno de los aspectos más importantes usar una estructura de ficheros bien definida, y a ser posible, estandarizada de alguna manera para mejorar la legibilidad del código y su mantenimiento.

Finalmente, decidimos crear el "esqueleto" para nuestra aplicación web en AngularJS mediante Yeoman.io (Yeoman.io, 2017). Yeoman ayuda a iniciar nuevos proyectos en base a buenas prácticas y mediante herramientas que optimizan la productividad. En nuestro TFM utilizamos el generador de AngularJS de Yeoman (GitHub, 2017). Para ello, primero hay que instalar una serie de paquetes con **NPM** de la siguiente forma:

```
$ npm install -g grunt-cli bower yo generator-karma generator-angular
```

- **grunt-cli**: Grunt.js es una librería JavaScript que nos permite configurar tareas automáticas y así ahorrarnos tiempo en nuestro desarrollo y despliegue de aplicaciones webs. Con un simple fichero JS que llamaremos Gruntfile.js, indicamos las tareas que queremos automatizar con un simple comando y las escribimos en él en formato JSON (Carlos Azaustre | Formación en JavaScript, 2017).
- **bower**: Bower es un proyecto open-source salido de Twitter. Funciona de forma parecida a NPM en NodeJS. En un fichero JSON anotamos las librerías que necesitamos y con un simple comando se descargan en un directorio concreto de directorio (Carlos Azaustre | Formación en JavaScript, 2017).
- **karma**: Karma es una herramienta para crear y ejecutar tests unitarios en javascript (Karma-runner.github.io, 2017).

Una vez instalado Yeoman, con el comando **$ yo angular** procedemos a la construcción del primer "esqueleto" de AngularJS para nuestra aplicación web (GitHub, 2017).

Nos gustaría señalar aquí uno de los problemas que hemos encontrado en la documentación oficial (GitHub, 2017). Esta describe que hay que ejecutar el comando **$ grunt serve**, y este comando muestra uno a uno los paquetes necesarios para satisfacer las dependencias. Procedimos instalando de forma manual cada uno de los paquetes necesarios tanto para NodeJS como para Bower. El proceso es tremendamente tedioso. Tras sucesivas búsquedas en la web, en foros especializados, etc., encontramos una solución mucho más lógica. Simplemente, una vez generado el proyecto de AngularJS con Yeoman, hay que acceder al directorio raíz del proyecto y ejecutar la instrucción **$ npm install && bower install** (insistimos, no aparece en la documentación oficial) para que instale automáticamente los paquetes y sus dependencias que vienen por defecto descritos en package.json para NodeJS y bower.json para Bower. Esto, obviamente, facilita y simplifica el proceso, y evita un posible abandono prematuro de la instalación. La ejecución del comando anterior también construye la estructura de directorios del proyecto.

Los principales comandos de **Yeoman** que nos facilitan el trabajo a lo largo de todo el desarrollo son los siguientes:

- **yo angular:controller** nombre: Crea un controlador nuevo
- **yo angular:directive** nombre: Crea una nueva directiva
- **yo angular:service** nombre: Crea un nuevo servicio
- **yo angular:route** nombre: Crea una nueva ruta además del controlador y la vista
- **yo angular:view** nombre: Crea una nueva vista

4.1.1.2.1 Principales hitos de la implementación

Una vez creado todo el directorio de AngularJS ya tenemos todo configurado y listo para trabajar, y empezamos desarrollando la aplicación web propiamente dicha, que será el panel de administración que usarán los administradores para manejar las imágenes subidas por los usuarios de la aplicación móvil.

4.1.1.2.1.1 Login de usuario

El primero de los hitos marcados fue conseguir el proceso de login. Este proceso era crucial ya que debe tener identificado al administrador que va a organizar, manejar y catalogar las imágenes subidas por los usuarios de la aplicación móvil. Nuestra primera aproximación fue crear un login con usuario y contraseña básico en el que, previamente, hemos dado de alta al administrador manualmente en la base de datos. Este primer punto implica trabajar con el diseño y con el front-end. El resultado para la primera versión del proceso de login de la aplicación web se muestra en la (Figura 19: Login aplicación web).

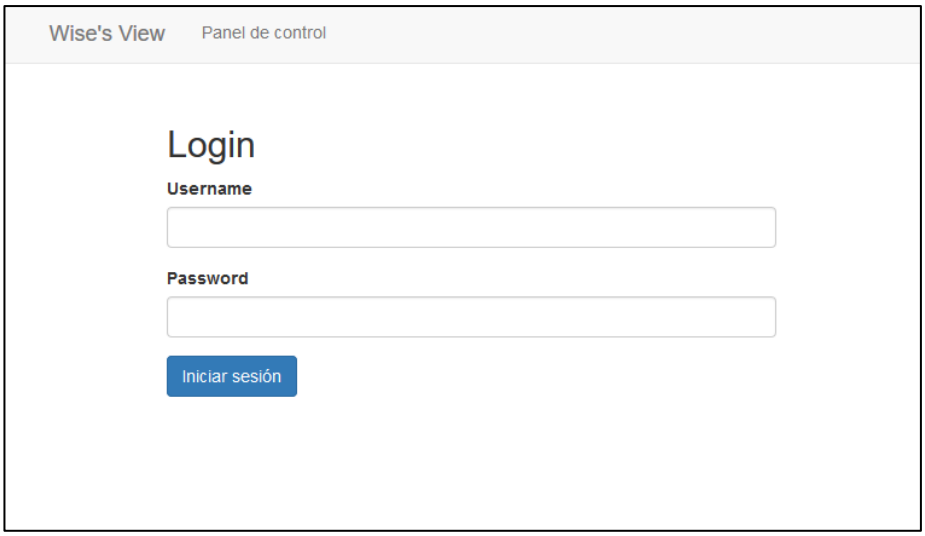

Figura 19: Login aplicación web

4.1.1.2.1.2 Listado de imágenes y datos subidas desde la aplicación móvil

Una vez logueado el administrador, pasamos a desarrollar el siguiente hito, una vista básica de todas las imágenes que se han almacenado en el servidor sin ningún tipo de filtro ni ordenación para completar todo el círculo de nuestro TFM. Para crear dicho listado, creamos un servicio en nuestra aplicación web que se llama **PaquetesService,** y que en una primera aproximación solo se encarga de hacer una petición a nuestra API REST solicitando todas las imágenes que han llegado a nuestro servidor y devolviendo dicha API REST un array de URLs de todas las imágenes para mostrarlas en la vista de nuestra aplicación web (Figura 20: Vista de las imágenes en la aplicación web)

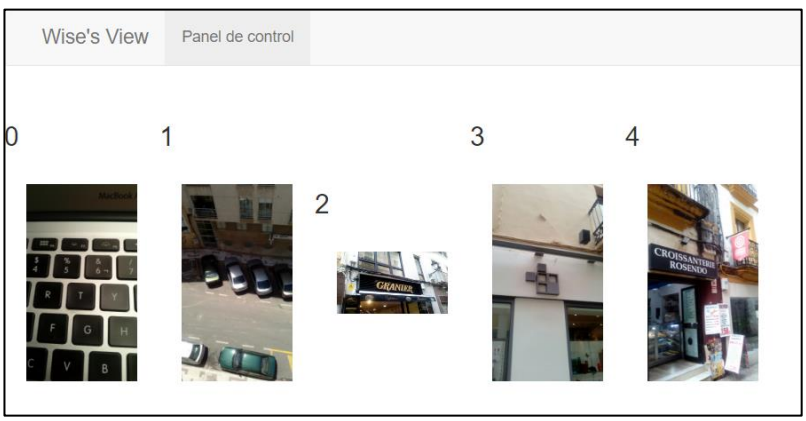

Figura 20: Vista de las imágenes en la aplicación web

4.1.1.3 Aplicación móvil

Como hemos indicado anteriormente, para crear la aplicación móvil vamos a usar el framework Ionic. Durante la realización del TFM, Ionic ha publicado la versión 2.0; no obstante, hemos decidido mantenernos en la versión 1.0 por razones operativas, por no asumir la carga de trabajo que supone la adaptación y porque seguirá existiendo soporte para la versión 1.0.

Con los conocimientos adquiridos mediante los tutoriales y cursos realizados, el primer paso es la instalación y configuración de Ionic en nuestra máquina para comenzar el desarrollo de nuestra aplicación móvil propiamente dicha.

Para instalar Ionic primero debemos tener instalado NodeJS (**4.1.1.1.1 Instalación y configuración de NodeJS**). Junto a este entorno de ejecución se distribuye NPM (Node Package Manager), lo que quiere decir que ya lo tenemos instalado en la máquina. **NPM** es el gestor de paquetes de NodeJS, que nos permite instalar librerías y módulos a través de la línea de comandos y, por tanto, es de gran ayuda para administrar nuestros módulos, utilizar software de terceros, y agregar dependencias de una manera simple.

NPM es un gestor de paquetes para NodeJS y nos permite, entre otras cosas, instalar Ionic y Cordova de forma sencilla. Recordamos al lector que Cordova e Ionic están íntimamente relacionados pues Cordova se encarga de "traducir/compilar" la aplicación móvil creada en Ionic a la plataforma seleccionada (Ver Sección **3.4.1 Ionic**). Por tanto, Para instalar Ionic procedemos a instalar primero Cordova (Ionic Framework, 2017).

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

Con la opción **-g** o **--global** se le está indicando a npm que debe instalar el paquete de manera global, esto quiere decir que se va a poder utilizar desde cualquier directorio (El blog { del programador };, 2017).

Una vez instalado Cordova, pasamos a instalar de la misma forma Ionic como sigue:

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

Una vez instado **Cordova** e Ionic montamos el "esqueleto" de nuestra aplicación, creando para ello un nuevo proyecto de **Cordova**:

```
$ ionic start nombreDeNuestraAPP tabs
```

Hemos denominado internamente a nuestra aplicación *wiseview*. Por tanto, este comando nos crea una carpeta llamada wiseview en el directorio donde nos encontramos. La estructura de este proyecto Ionic será la siguiente (Ionicframework.com, 2017):

```
├── bower.json      // bower dependencies
├── config.xml      // cordova configuration
├── gulpfile.js     // gulp tasks
├── hooks           // custom cordova hooks to execute on specific commands
├── ionic.config.json  // ionic configuration
├── package.json    // node dependencies
├── platforms       // iOS/Android specific builds will reside here
├── plugins         // where your cordova/ionic plugins will be installed
├── sass            // scss code, which will output to www/css/
└── www             // application - JS code and libs, CSS, images, etc.
```

Del esquema anterior, describimos la función/contenido de aquellos ficheros no autodescriptivos para facilitar la comprensión del lector.

- **bower.json**: Fichero JSON que describe las dependencias bower que son utilizadas por nuestro proyecto.
- **config.xml**: Fichero de configuración de Cordova para nuestro proyecto Ionic. Este fichero permite a Ionic construir la aplicación para iOS y Android, configurando parámetros como la versión, el nombre, descripción, autor…

- **gulpfile.js**: **Gulp** (Gulpjs.com, 2017) es un conjunto de herramientas que permiten automatizar tareas que consumen mucho tiempo en el desarrollo de aplicaciones. **Ionic** implementa de manera automática **Gulp** para tareas como concatenar ficheros automáticamente.
- **hooks** (Cordova.apache.org, 2017) representa una serie de scripts que Ionic añade a la aplicación para personalizar los comandos de **Cordova**.
- **ionic.config.json** es un fichero JSON de configuración del proyecto Ionic. En este fichero se configura el proyecto con valores como nombre y versión de Ionic utilizada en este proyecto.
- **package.json** es un fichero JSON que especifica las dependencias de NodeJS que tiene nuestro proyecto Ionic.
- **platforms:** Directorio donde se guardarán las aplicaciones construidas tanto para iOS como para Android.
- **plugins:** Directorio donde se instalan los plugins de Cordova e Ionic utilizados en el proyecto.
- **sass: Sass** (Sass-lang.com, 2017) es un metalenguaje de hojas de estilos en cascada (**CSS**), es un lenguaje de script que es traducido a **CSS**. Ionic añade la traducción de dicho script a la carpeta **www/css**.
- **www**: Directorio principal donde trabaja el desarrollador editando los ficheros **JS,** vistas **HTML**, imágenes y estilos CSS.

De la misma forma, procedemos a describir los comandos más relevantes y más usados de Ionic para construir la Perspectiva del Sabio.

- **Ionic serve**: Su cometido es testear la aplicación desde nuestro navegador
- **Ionic cordova plugin list:** Lista los plugins de Cordova que tiene el proyecto
- **Ionic cordova plugin add/rm** *nombre_plugin*: Añade/Elimina un plugin de Cordova al proyecto
- **Ionic cordova platform add/rm** *iOS/Android*: Añade/Elimina las plataformas para las que queremos generar la aplicación

- **Ionic cordova build** *iOS/Android***:** Construye (prepara + compila) un proyecto Ionic para una plataforma determinada
- **Ionic cordova emulate** *iOS/Android***:** Emula un proyecto Ionic en el simulador

4.1.1.3.1 Principales hitos de la implementación

Una vez que tenemos todo instalado y configurado y hemos adquirido dominio en los diferentes elementos, vamos a continuar con el desarrollo propiamente dicho de la aplicación móvil. Para ello subrayemos los hitos o logros más importantes en el desarrollo, mostrando la línea incremental y la división por fases que ha sufrido este TFM.

4.1.1.3.1.1 Login de Usuario

El primero de los hitos marcados fue conseguir el proceso de login. Este proceso era crucial ya que debíamos tener identificada a la persona que nos enviaba las imágenes desde su dispositivo móvil. Nuestra primera aproximación fue crear un login con usuario y contraseña básico en el que, previamente, habíamos dado de alta al usuario manualmente en la base de datos. Este primer punto implicaba trabajar con el diseño y con el front-end. No debemos olvidar nuestro público objetivo: personas mayores, por lo que la vista de la aplicación tiene que ser limpia, sencilla, en colores claros que faciliten su lectura, con el tamaño de fuente adecuado, etc. El resultado para la primera versión del proceso de login de la aplicación móvil se muestra en la (Figura 21: Proceso login aplicación móvil).

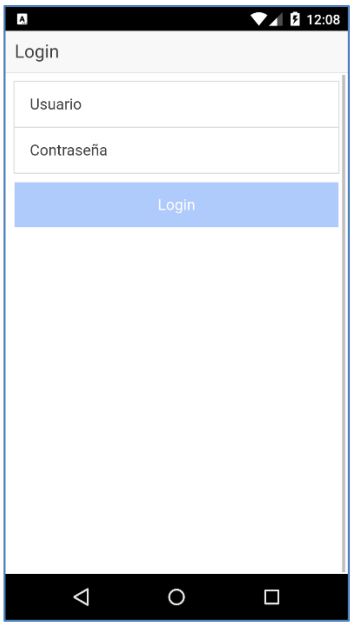

Figura 21: Proceso login aplicación móvil

La Figura 21: Proceso login aplicación móvil define la vista completa del login de usuario. El siguiente paso es crear el **controlador**. 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 vista 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**. El controlador envía los datos necesarios a la plantilla a través de la variable **$scope**, y de esta forma, desde la vista solo tenemos que mostrar los datos recibidos dándoles el aspecto adecuado.

Los **servicios** de Ionic son objetos que siempre tienen que devolver un objeto. Gracias a los servicios, por ejemplo, podemos crear objetos que consuman los datos de un API REST, como en nuestro caso, y que además sean reutilizables e independientes de las vistas que hemos creado.

En consecuencia, hemos implementado el servicio LoginService, y su funcionalidad es hacer login mediante una llamada a la API REST de nuestro servidor. La llamada es CheckUser. Desde el controlador, se llama al

servicio LoginService y asíncronamente, tras recibir la respuesta de la API, se valida el proceso login o no.

4.1.1.3.1.2 Captura y subida de una imagen

Una vez que el usuario se ha logueado, pasamos a implementar el siguiente hito: la captura y subida de una imagen desde el smartphone a nuestro servidor.

Para capturar la imagen desde nuestra app en Ionic tenemos que usar un plugin de Cordova, el cual instalaremos en nuestro proyecto mediante el siguiente comando (GitHub, 2017):

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

Una vez instalado el plugin ya podemos usar la cámara del smartphone llamando a la función getPicture(), la cual automáticamente le pide permisos al usuario de la aplicación para utilizar la cámara. Si el usuario los acepta, pasa a capturar la imagen. Si hace la captura correctamente, le devuelve la URL local donde se encuentra la imagen realizada por el usuario.

A continuación, teniendo la URL de dicha imagen, creamos un servicio como hicimos para el login de usuario llamado **PaqueteService** en el que tendremos una función que será la de **uploadPaquete**, que se conectará con la API REST realizando una llamada a **addPaquete** mediante un plugin especial para mandar ficheros a la API REST que se llama cordova-plugin-file-transfer (Cordova.apache.org, 2017).

Este plugin nos permite mandar un fichero a la API REST y que la API REST pueda recibir la imagen en **req.files** como hemos explicado anteriormente (**4.1.1.1.4 API REST: versión inicial**).

Para finalizar, creamos dos vistas de la aplicación en la que se creó un botón que llamaba a dicho servicio, y una vez finalizada y enviada la imagen mostraba un modal mostrando el resultado de dicho servicio (Figura 22: Vistas aplicación móvil).

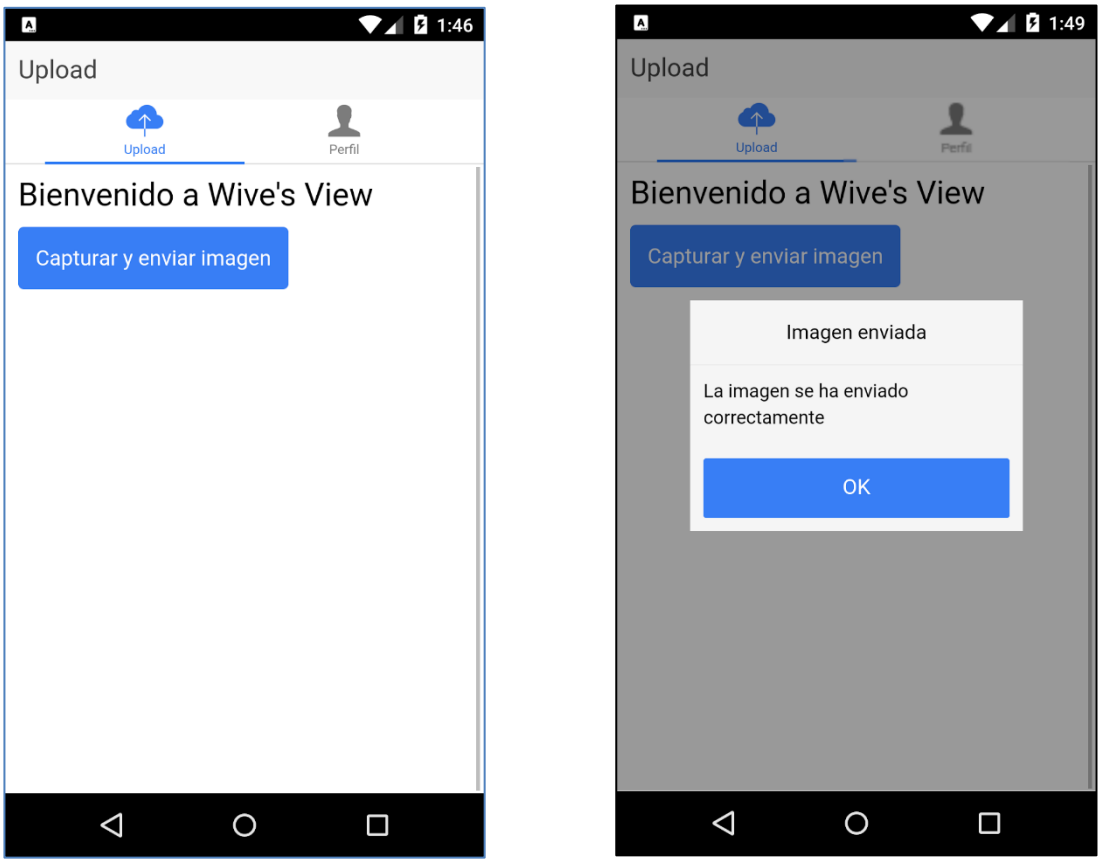

Figura 22: Vistas aplicación móvil

Uno de los mayores problemas que tuvimos con la aplicación móvil fue al intentar conectar con la API REST. El problema era con el CORS (Cross origin resource sharing) (The Official Ionic Blog, 2017).

Las primeras pruebas siempre se hacían utilizando **Ionic serve**, es decir, utilizando el navegador web para ver la funcionalidad y vistas de nuestra aplicación móvil. Este comando crea un servidor web local y arranca la aplicación en un navegador para ver la aplicación en acción. Nuestra API REST estaba funcionando ya en el servidor y hacíamos llamadas a dicho servidor desde la aplicación, pero el origen de este host era distinto al host en el que corría la aplicación (localhost) por lo que no nos permitía los accesos al exterior. Para solucionar este problema, utilizamos una extensión de Google Chrome que se llama Allow-Control-Allow-Origin

(Chrome.google.com, 2017) y que permite el acceso a cualquier host desde nuestro localhost para poder acceder a la API REST de nuestro servidor sin problemas.

4.1.1.4 Revisión con el cliente

Al terminar los objetivos de desarrollo de esta primera fase nos reunimos con Lidia para mostrarle el primer embrión de su sistema. Aún es pronto para que ella pueda hacerse una idea del resultado final. La aplicación móvil es muy básica y la aplicación web simplemente lista las imágenes capturadas (**4.1.1.2.1.2 Listado de imágenes y datos subidas desde la aplicación móvil**). No obstante, comprobar que ya es posible tener una prueba de concepto del resultado final que nos permite capturar una foto con un teléfono móvil y que automáticamente aparezca en orden en la página web, es de su agrado.

Las conclusiones más interesantes que sacamos de esta revisión tienen que ver con las especificaciones en sí mismas. Una de ellas es que nos damos cuenta de que la idea inicial de subir la foto es una buena idea, pero que a veces puede no quedar claro qué es lo que el usuario pretende capturar de su entorno. Sin duda, esto será el germen de la posterior implementación de un método que permite adjuntar una grabación de audio a la fotografía.

En segundo lugar, observamos que el proceso de login, aunque sencillo, (simplemente hay que introducir un nombre de usuario y una contraseña), conlleva que el usuario tenga que recordar un nombre y una contraseña que le hemos proporcionado nosotros. Recordamos al lector que para esta primera aproximación damos de alta manualmente a los usuarios en el sistema. Esto conlleva un esfuerzo de memoria por parte del usuario, y sinceramente, todos tenemos ya bastantes nombres de usuario y contraseñas que nos dan acceso a diferentes sitios web.

De otra parte, discutimos la posibilidad de plantear un proceso de login convencional, en el que usuario elige un nombre y una contraseña. Pero encontramos inconvenientes similares a los anteriores, y además, nos exige

un esfuerzo extra en el desarrollo, ya que no solo tenemos que controlar internamente el alta de un nuevo usuario, sino que en caso de que exista, debemos crear formas de avisar al usuario para pedir de nuevo la información, controlar que la contraseña sea segura, etc. Incluso, y pensando en la aplicación web, puede ser que nos encontrásemos con fotos de usuarios que pueden haber elegido un nombre basado en un alias, de forma que no nos permita identificar fácilmente quién es la persona responsable de esa captura de datos.

Por tanto, y aunque no se nos ocurre durante la revisión, también es el germen del nuevo proceso de login basado en el número de teléfono y que implementaremos en la siguiente fase.

4.1.2 Segunda fase beta

La segunda fase arranca sobre la validación de la primera pero con una diferencia fundamental: el entorno de trabajo está operativo y listo. Al terminar esta fase el proyecto tendrá un aspecto global más cercano a los deseos de nuestra investigadora. Con un entorno de trabajo estable nos podemos centrar e invertir nuestro tiempo en el desarrollo de las funcionalidades necesarias. En las siguientes secciones, describimos manteniendo el mismo esquema: servidor, aplicación web y aplicación móvil, los avances más notables en cada una de ellas.

4.1.2.1 Servidor

En el servidor necesitamos añadir las nuevas funcionalidades que debe soportar la API REST además de modificar el modelado de MongoDB para que se adapte a los nuevos requisitos de la aplicación.

4.1.2.1.1 Actualización de la API REST

Las nuevas funcionalidades de la API REST serán tanto para la conexión con la aplicación web como para la conexión con la aplicación móvil, ya que necesitamos añadir nuevas funciones y adaptar las que teníamos.

Una de las mejoras más relevantes de esta segunda fase ha sido conseguir un proceso de login aún más sencillo. Debido a nuestro público objetivo, toda simplificación es bienvenida. Se nos ocurrió la idea de tratar de hacer un login automático mediante el número de teléfono, es decir, al instalar la aplicación esta accedería automáticamente al número del teléfono y lo enviaría a nuestra API REST. En la sección **4.1.2.2 Aplicación móvil** se detalla el proceso. No obstante, para soportarlo necesitamos modificar nuestra API REST en consecuencia.

Para soportar este nuevo proceso de login mediante el número de teléfono, debemos actualizar la función **checkUser() (**del controlador **UserCtrl).** Ahora, en vez de comprobar el nombre de usuario y contraseña, debe guardar en MongoDB el número de teléfono, por lo que esta función comprobará si el número de teléfono está registrado. Si no está registrado, lo dará de alta automáticamente.

Otra actualización significativa tiene lugar en la función **addPaquete() (**del controlador **PaqueteCtrl),** adaptándose para añadir los nuevos parámetros acordados para esta fase: las coordenadas **GPS de latitud y longitud.** Por tanto, el paquete enviado desde la app debe incluirlas, de esta forma asociamos la fotografía al lugar donde se tomó.

Esta funcionalidad implica también que la API REST ofrezca dicha información a la aplicación web, razón por la que tenemos que actualizar el controlador **adminCtrl,** qie es el encargado de implementar todas las funciones necesarias para interactuar con la aplicación web.

En este parte, Lidia nos pidió una forma más humana de reconocer a los usuarios. Es decir, si el nombre de nuestro usuario es el número de teléfono es bastante difícil recordar quién es. Para resolverlo, implementamos un método (setDataPaquete) que permite desde la aplicación web asociar un nombre al número de teléfono. (Ver sección **4.1.1.2 Aplicación web**)

Por tanto, las dos funciones más importantes que se añaden son **getAllPaquetes** y **setDataPaquete.**

- **getAllPaquetes** devuelve todos los paquetes almacenados en MongoDB, organizados por usuarios y ordenados por fecha de llegada para mostrarlos en la aplicación web.
- **setDataPaquete** nos permite almacenar un nombre asociado al número de teléfono para una mejor organización de los usuarios que envían paquetes desde la aplicación móvil. Para lograrlo, añade los datos de un usuario al paquete mediante el verbo **PUT** de la API REST.

4.1.2.1.2 Actualización esquema en MongoDB

El esquema de **MongoDB** se modifica para adaptarlo a las nuevas funcionalidades, tanto de la aplicación móvil como de la aplicación web.

```
var userSchema = new Schema({
    phoneNumber:    { type: String, required: true },
    profile:      profileSchema,
    paquetes:  [paqueteSchema]
}, {strict: false});
```

Figura 23: userSchem del modelo de nuestra API REST

En este nuevo esquema (Figura 23: userSchem del modelo de nuestra API REST) podemos observar cómo en **userSchema** se han eliminado los campos **userName** y **contraseña** para sustituirlo por un nuevo campo, **phoneNumber**. En nuestro caso, y como en esta fase no tenemos datos capturados por los usuarios, procedemos a eliminar dichos campos. Pero imaginemos, y aquí es donde reside una de las ventajas de MongoDB, la situación hipotética de que la aplicación ya hubiera estado en producción y tuviéramos miles de registros con usuario y contraseña. En MongoDB simplemente debemos añadir el campo phoneNumber, actualizar nuestra aplicación móvil en el store, y cuando el usuario se la descargue automáticamente nos enviaría su número de teléfono. El proceso para el usuario sería transparente y podíamos seguir almacenando datos para el mismo usuario sin prácticamente cambiar nada de nuestra implementación.

Además, los nuevos usuarios se registran proporcionando únicamente el número de teléfono y no se almacena nada relativo al usuario y contraseña. Con una base de datos relacional deberíamos haber ejecutado una orden interna que añadiera a todos los registros de la tabla original una nueva columna con el campo phoneNumber a Null; esta sería rellenada por los nuevos registros, pero lastraría los campos user y contraseña a Null hasta el fin de sus días, para mantener la concordancia con los usuarios registrados anteriormente. Por tanto, phoneNumber se convierte en el nuevo identificador único de nuestros documentos.

```
var profileSchema = new Schema({
    name: { type: String}
}, {strict: false});
```

Figura 24: profileSchem del model de nuestra API REST

También hemos añadido un nuevo campo, **profile,** que definimos como un nuevo esquema, **profileSchema** (Figura 24: profileSchem del model de nuestra API REST), para poder añadir nuevos campos en un futuro, pensando en que tal vez desde la aplicación web se quiera añadir alguna información adicional, aunque por ahora solo contiene el campo **name**.

```
var paqueteSchema = new Schema({
    created_at: { type: Date, required: true },
    upload_at: { type: Date, default: Date.now  },
    location: {
        long: {type: String},
        lat: {type: String}
    },
    data: [dataPaqueteSchema]
}, {strict: false});
```

Figura 25: paqueteSchema del modelo de nuestra API REST

Dentro de **paqueteSchema** (Figura 25: paqueteSchema del modelo de nuestra API REST) hemos añadido un nuevo campo, **location,** en el que almacenamos la **latitud** y **longitud** asociados a un paquete determinado.

4.1.2.1.3 Gestor de procesos NodeJS PM2

En nuestra profesión es frecuente que mientras se desarrolla software, se siga aprendiendo y estudiando, lo que permite descubrir nuevas formas de realizar las tareas o técnicas más óptimas. Esto me ocurrió a mí durante este TFM: mientras profundizaba en el aprendizaje de NodeJS descubrí una herramienta que incluso ignoraban mis tutores: PM2

PM2 es un gestor de procesos de **NodeJS** que dispone de las siguientes características principales:

- Capacidad de manejar múltiples procesos **NodeJS**.
- Capacidad de monitoreo de **memoria** y **CPU** de nuestros procesos.
- Manejos de **logs**, tanto de salida de los procesos como de los errores que puedan suceder.
- **Equilibrado de carga** automáticamente de los distintos procesos **NodeJS**.
- **Iniciar** los procesos **NodeJS** automáticamente, si el servidor es reiniciado o se produce algún error en cualquiera de los procesos mencionados.

Los comandos principales de **PM2** son los siguientes:

- **pm2 start** *nombre_proceso.js* : Inicia el proceso **NodeJS** otorgándole un identificador y aplicando todas las características anteriormente mencionadas.
- **pm2 monit**: Monitoriza todos los procesos lanzados mediante **PM2**, mostrando información relevante como el log de salida, el consumo de memoria/CPU, fecha de lanzamiento, etc.
- **pm2 stop/start/delete** *id_proceso:* Para, inicia o elimina respectivamente un proceso **NodeJS** utilizando el identificador único que le otorga **PM2**.

- **pm2 list**: Lista los procesos lanzados por **PM2**, mostrando el estado de dicho proceso, su identificador único, etc.

Por todas estas características, decidimos utilizar **PM2** como gestor de nuestros procesos **NodeJS**. Su instalación y configuración es sencilla gracias, una vez más, a **NPM** (Pm2.keymetrics.io, 2017):

```
$ npm install -g pm2
```

Una vez instalado arrancamos nuestra API REST con **PM2** utilizando el siguiente comando:

```
$ pm2 start api-wiseView.js
```

La Figura 26: Salida del comando pm2 monit muestra la salida del comando **pm2 monit**, que nos permite monitorizar el estado de nuestra API REST.

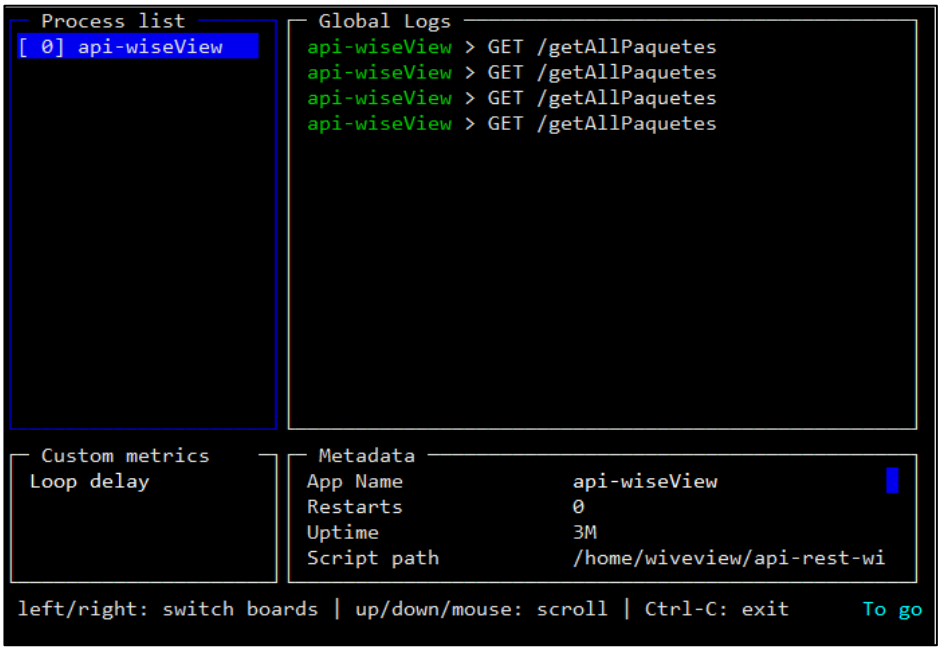

Figura 26: Salida del comando pm2 monit

4.1.2.2 Aplicación móvil

En esta fase, se nos ocurre una idea genial y clave para la aplicación móvil de nuestro proyecto. Decidimos junto con Lidia que, ya que nuestro público

objetivo son personas mayores, debíamos tratar de simplificar al máximo todos los procesos, y uno de ellos era el proceso de login, de forma que no tuvieran que recordar ni el usuario, ni la contraseña. Por tanto, la mejor forma de registrar a los usuarios con un identificador único sería su propio número de teléfono.

En esta fase, también incorporamos la ubicación en coordenadas GPS (latitud y longitud) del lugar donde se realizó la fotografía.

4.1.2.2.1 Login con número de teléfono

Para realizar el login mediante el número de teléfono empezamos creando un formulario muy sencillo para introducir el número de teléfono manualmente. Después de investigar, encontramos que existía una opción para obtener los datos del teléfono móvil automáticamente, aunque, desafortunadamente, solo funciona en ciertos terminales. Esto nos llevó a implementar en la aplicación un proceso híbrido en el que primero se intenta obtener el número de teléfono automáticamente del Smartphone, y si el modelo de terminal o versión del S.O. no permite esta operación, se requiere al usuario la introducción manual de su número de teléfono (Figura 27: Login manual con número de teléfono).

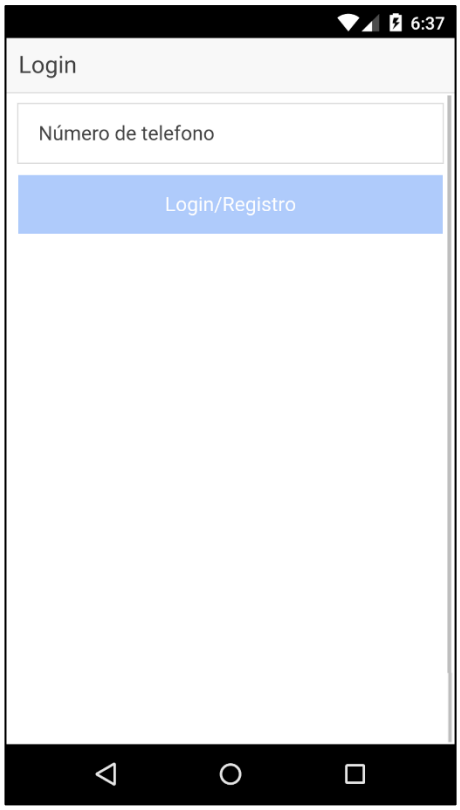

Figura 27: Login manual con número de teléfono

Desde el punto de vista técnico, el plugin para realizar dicha operación automática de obtención de datos del smartphone es **cordova-plugin-sim** (GitHub, 2017)**.** Este plugin de acceso al hardware se añade a Ionic, nuestro framework de desarrollo de aplicaciones móviles, y cuando construyamos la aplicación móvil formará parte de ella. Nuestra tarea consiste en implementarlo y configurarlo adecuadamente dentro del código. Básicamente, lo invocamos, y si el plugin es capaz de obtener los datos del dispositivo, automáticamente se realiza la llamada al servicio **LoginService** y este realiza el login/registro a través de nuestra API REST. En caso contrario, lanzaremos una nueva vista (Figura 27: Login manual con número de teléfono) para que el usuario introduzca el número de teléfono manualmente, y con ese número se realiza la llamada a **LoginService**.

4.1.2.2.2 Obtención de ubicación

La obtención de la ubicación es otro de los puntos importantes que debe realizar nuestra aplicación móvil. Es clave que dicha localización se realice inmediatamente después de tomar la fotografía para tenerla correctamente

localizada, por lo que creamos una función en nuestro controlador (**MainCtrl**) que se lanza automáticamente tras la captura de la imagen utilizando el plugin **cordova-plugin-geolocation** (Cordova.apache.org, 2017).

Este plugin permite obtener la localización mediante la función **getCurrentPosition.** La función solicita automáticamente el permiso de acceso al hardware del GPS y puede ser parametrizada mediante un conjunto de opciones. Destacamos las siguientes:

- **enableHighAccuracy**: Si es true permitirá una localización más precisa utilizando la conexión a un satélite. Si es false obtendrá una localización menos precisa basada en red. Uno de los problemas más importantes que tuvimos al configurar este plugin es que algunas veces, por falta de cobertura con los satélites, nos devolvía error si activábamos esta opción, mientras que otras veces, si no activábamos esta opción, nos devolvía error por no tener cobertura de red, aunque sí conexión a los satélites. Llegamos a este problema después de realizar muchas pruebas en distintos sitios y con distintos smartphones. Nuestra solución fue la de solicitar primero la localización con esta opción activada, y si no nos devolvía los valores de localización, intentarlo con la opción desactivada. De esta forma, en todas las pruebas que hemos realizado siempre tenemos la localización, ya sea más o menos precisa.

- **timeout**: El intervalo de tiempo en milisegundos que permitimos a la función **getCurrentPosition** intentar obtener la localización. Si al final de este tiempo no ha conseguido una localización correcta, nos devolverá el error con el código **PositionError.TIMEOUT**. En nuestro caso, y de forma experimental durante las pruebas, hemos definido a 4000 milisegundos, ya que era el intervalo que mejores resultados arrojaba.

Una vez que tenemos la fotografía y la localización de dicha fotografía, llamamos al servicio **paqueteService** para que realice la subida de la imagen como lo hacíamos en la primera fase pero añadiendo, además, las coordenadas GPS de latitud y longitud así como los parámetros de dicha

fotografía. La API REST se encarga de guardar la imagen y de almacenar los datos en la base de datos.

4.1.2.3 Aplicación web

Ahora que ya disponemos de la localización de la fotografía gracias a la aplicación móvil, tenemos que añadir un mapa para visualizar los datos de cada paquete de forma compacta e intuitiva. Además, es el momento de añadir la tercera columna, la de la derecha, (Figura 2: GUI en tres columnas) para mostrar los datos asociados a un paquete.

Por otro lado, después de hablar con Lidia, concluimos que sería útil tener una búsqueda por usuarios para filtrar los paquetes de un usuario concreto permitiendo así administrar mejor la información.

4.1.2.3.1 Google maps en la aplicación web

Después de buscar, analizar y comparar varias opciones de mapas para AngularJS como por ejemplo Leaflet (Tombatossals.github.io, 2017) y OpenLayers (Openlayers.org, 2017), llegamos a la conclusión de que tanto por funcionalidades como por comunidad, actualizaciones y revisiones, nuestra mejor opción era usar GoogleMaps. En AngularJS este tipo de integraciones se conocen como directivas ya que interactúan directamente con la vista de la aplicación web. Por tanto, utilizamos **GoogleMaps AngularJS Directive** (GitHub, 2017). Esta directiva tiene todas las funcionalidades que nosotros necesitamos como, por ejemplo, añadir marcas personalizadas en el mapa, crear figuras por encima del mapa para resaltar zonas, e incluso, incluir vistas de **street view** directamente desde nuestra directiva (Figura 28: Ejemplo ilustrativo de la directiva gogole maps para Angular JS).

Figura 28: Ejemplo ilustrativo de la directiva gogole maps para Angular JS

Para instalar esta directiva en nuestro proyecto usaremos el siguiente comando:

```
$ bower install ngmap
```

Una vez instalado tenemos que añadir a nuestro módulo de angular la directiva **ngMap** como sigue:

```
angular.module(wiseViewApp, ['ngMap']);
```

Para utilizarlo en nuestra vista o template tenemos que escribir el código HTML que lo configura, añadiendo una serie de opciones, así como añadiendo las marcas que vienen especificadas en el controlador y usando las variables **paquetes.marker.lat** y **paquetes.marker.long** (latitud y longitud, respectivamente). El resultado de este proceso consigue una impactante y útil vista en la aplicación web. La Figura 29: Mapa y "chincheta" de Google, nos muestra el mapa y la conocida chincheta de Google asociada al lugar desde donde ser realizó la foto.

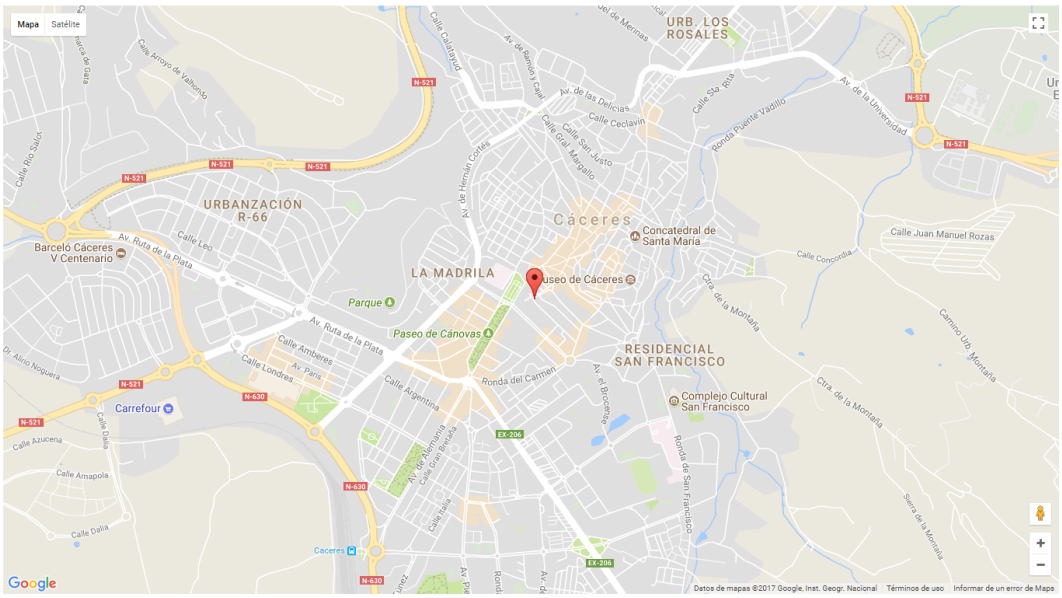

Figura 29: Mapa y "chincheta" de Google

4.1.2.3.2 Revisión y edición de la información proporcionada por el usuario

Como hemos explicado anteriormente, además del número de teléfono que envía el paquete vamos a introducir el nombre del usuario que realiza la fotografía para tener una mayor organización en nuestro esquema. Para ello, en la columna derecha de nuestra aplicación web (Figura 2: GUI en tres columnas) mostramos los campos asociados a cada uno de los paquetes. La Figura 30: Información el la aplicación web del paquete seleccionado muestra la información relativa al paquete seleccionado. Observamos que existe un botón guardar. El comportamiento de este botón es autodescriptivo y nos permite incluir en los campos editables de esta fase el campo "Nombre" para almacenar la nueva información. Por tanto, el accionamiento de este botón realiza una llamada al método correspondiente de nuestra API REST (setDataPaquete) enviando un nuevo paquete de datos que reemplazará al anterior en MongoDB.

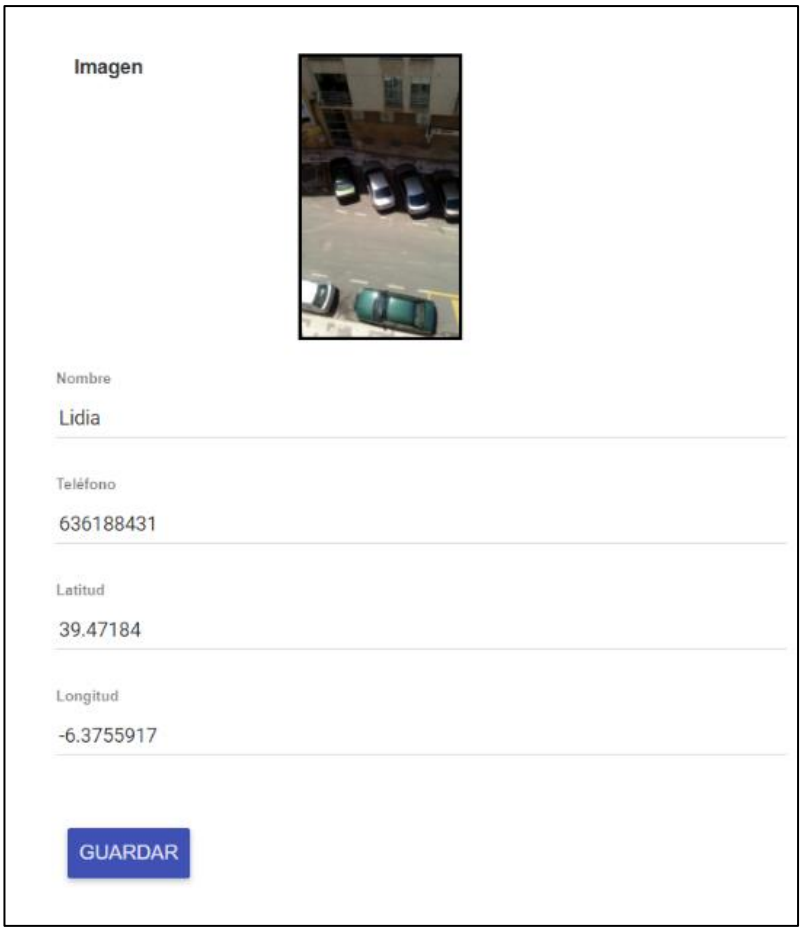

Figura 30: Información el la aplicación web del paquete seleccionado

4.1.2.3.3 Búsqueda por usuario

Para mejorar la organización y para cubrir las necesidades de Lidia, implementamos una búsqueda por usuario que le permitiese analizar las fotografías realizadas por un usuario en concreto. En el diseño de la vista de la aplicación web decidimos utilizar la zona superior e inmediatamente inferior a la zona de menús para este tipo de utilidades y la llamamos, consecuentemente, “zona de utilidades” (Figura 2: GUI en tres columnas).

La Figura 31: Buscador por usuarios en la aplicación web. muestra una vista detalle del buscador, donde se aprecia la sección del menú. La búsqueda se

realiza mediante el identificador único: el número de teléfono, y la función devuelve todos los paquetes de datos asociados a ese usuario. [7]

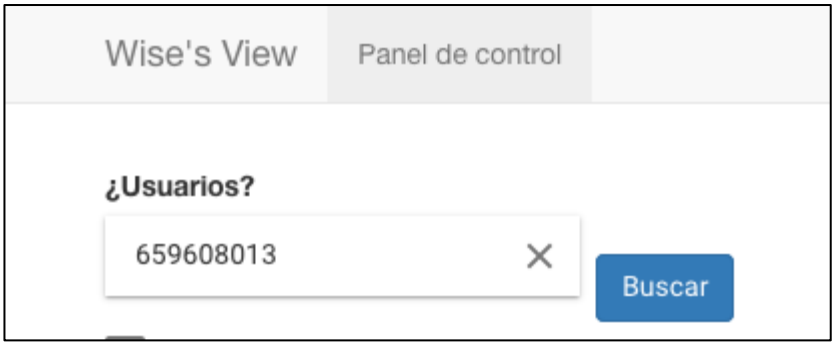

Figura 31: Buscador por usuarios en la aplicación web.

4.1.2.4 Revisión con el cliente

Al terminar esta fase tenemos una visión bastante completa de lo esperado por nuestra investigadora. El sistema está operativo y necesitará algunos detalles más para alcanzar su versión final. Siguiendo el mismo orden en la explicación de las partes del TFM, procedemos a señalar lo más relevante de la aplicación móvil para posteriormente seguir con la aplicación web. Obviamos el servidor, puesto que es indiferente para nuestro cliente.

En esta fase, la aplicación móvil tiene un aspecto similar al de la fase final, aunque aún no dispone de todas las pantallas. Aún no está en el store, pero tanto nosotros como Lidia hemos probado diferentes versiones mediante la instalación directa de una aplicación para Android y otra para IOS. El salto de calidad en el diseño es evidente, y eso nos agrada a todos (en la Figura 32: Pantalla inicial aplicación móvil se muestra una captura de la pantalla inicial). Para proceder a la toma de una fotografía, simplemente hay que presionar la zona “capturar imagen”. Todo el ancho responde a la acción y no solo el botón para simplificar el acceso.

[7] Entre nuestros hitos futuros, procederemos a incluir la búsqueda por el nombre de usuario (Ver sección **7 Trabajos Futuros**).

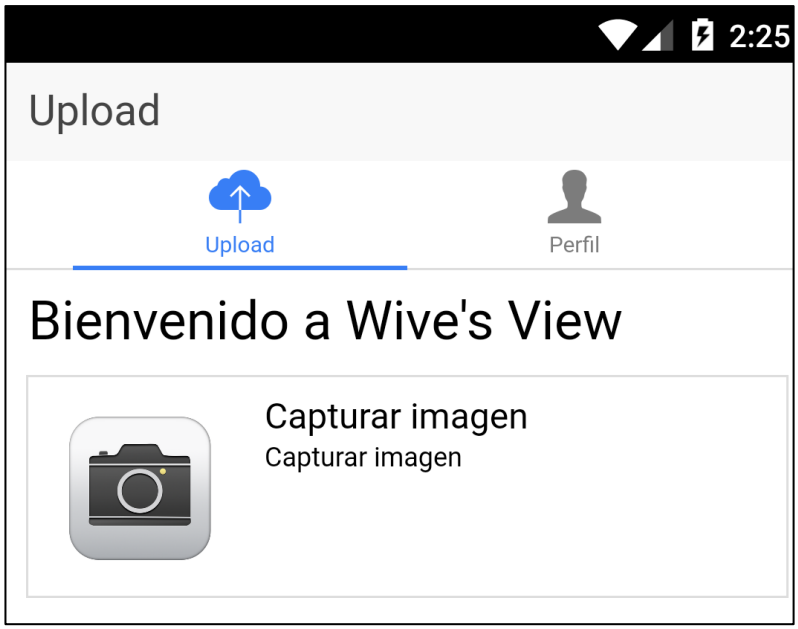

Figura 32: Pantalla inicial aplicación móvil

La acción anterior da acceso a la cámara del smartphone y permite la toma de la fotografía en cuestión. Al finalizar, tenemos nuestro paquete listo: automáticamente la aplicación ha obtenido las coordenadas GPS y ha adjuntado la fecha de creación. Ahora está en manos del cliente decidir si envía o no envía este paquete de datos. Decidimos colocar dos botones en colores llamativos, aunque habituales en las aplicaciones (Figura 33: Pantalla de envío de paquete en la aplicación móvil). Normalmente, el color rojo se utiliza para indicar una advertencia o para cancelar una acción y de la misma forma procedemos aquí, rellenado el fondo del botón de ese color. Incorporamos también un texto claro e identificativo de la acción “cancelar” así como en el icono de “papelera”, muy habitual cuando se quiere descartar o eliminar fichero. A su derecha, y en color azul, por razones similares a las anteriores, colocamos el botón de envío, con un texto auto-descriptivo de su acción: “¡Enviar!”, y un icono que representa que algo es enviado a la nube.

En general, la satisfacción de la investigadora con el resultado de esta fase fue muy elevada. En sus propias palabras: “tengo la sensación de estar usando una aplicación profesional como las otras que tengo el teléfono, ¡que pasada!”. Personalmente, este reconocimiento me infundió motivación y

ganas de enfrentarme a los nuevos desafíos y de alguna forma me recompensó las horas de trabajo y esfuerzo.

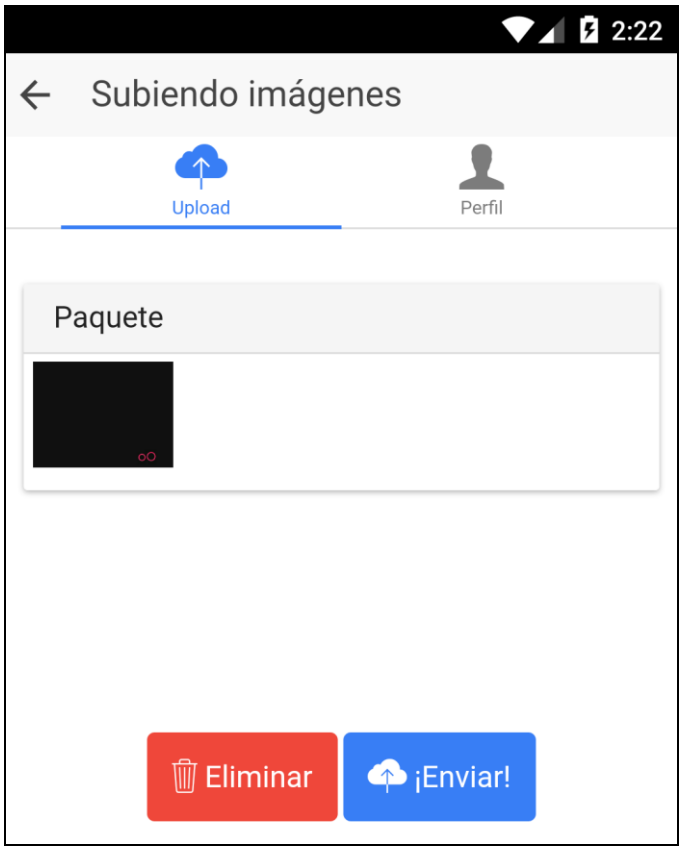

Figura 33: Pantalla de envío de paquete en la aplicación móvil

Uno de los aspectos más útiles para ella es el panel de administración web. En la Figura 34: Pantalla principal aplicación web al finalizar la segunda fase. se observa cómo han ido tomando forma las especificaciones iniciales y cómo hemos mantenido la coherencia usando las diferentes zonas definidas al principio. De la misma forma, hemos tratado de mantener un diseño limpio y elegante muy propio del gusto personal de Lidia.

La parte superior contiene el menú que, para este TFM, solo contendrá un nivel: el propio panel de control. En la zona inmediatamente inferior se ubica la primera de las utilidades demandada: el buscador por usuario.

A continuación, y ocupando todo el ancho de la pantalla, aparece la división principal en tres columnas: la primera se usa para listar los datos almacenados en el sistema, la segunda para su visualización en el mapa del

lugar desde donde fue recogido, y la tercera para revisar y editar el paquete de datos.

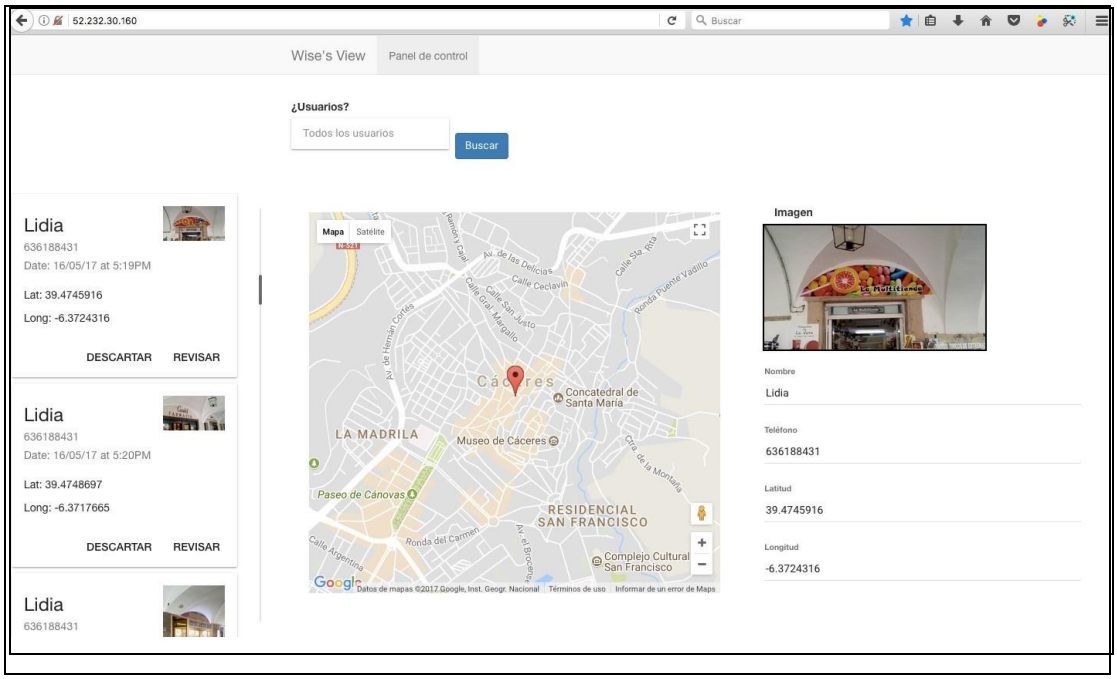

Figura 34: Pantalla principal aplicación web al finalizar la segunda fase.

Debemos decir que Lidia quedó impresionada por el resultado tan elegante a la vez de sencillo y útil. Se observa también en la Figura 34: Pantalla principal aplicación web al finalizar la segunda fase. que son imágenes reales capturadas por y desde el teléfono móvil de nuestra investigadora.

4.1.3 Tercera fase beta abierta

En esta fase del TFM describimos los últimos cambios en el servidor, aplicación móvil y aplicación web. Al finalizar, es el momento de subir la aplicación a la Apple Store y a la Play Store como beta abierta.

4.1.3.1 Servidor

Los cambios y actualizaciones que tienen lugar en el servidor comprenden la adición de las últimas funcionalidades. Estas conllevan modificaciones en el modelado de datos, así como la creación de nuevas rutas y funciones en la API REST para completar el sistema.

4.1.3.1.1 Actualización de la API REST

Del conjunto de actualizaciones realizadas en la API REST en esta fase destaca la adición de una nueva funcionalidad que permite la recepción de un fichero de audio en el servidor desde la aplicación móvil. Este comportamiento, a su vez, necesita que modifiquemos la función **getAllPaquetes** (**4.1.2.1.1 Actualización de la API REST**) del controlador adminCrtl para que devuelva también la URL de dicho fichero de audio a la aplicación web.

Para la recepción de un fichero de audio, creamos la ruta **uploadRecorder** que llama a la función **addSound()** en el controlador **PaqueteCtrl**. Dicha función, al igual que **addPaquete** (**4.1.2.1.1 Actualización de la API REST**) para la subida de una imagen, utiliza el módulo **express-fileUpload,** que permite que el fichero de audio sea recibido por la API REST en **req.files** desde la aplicación móvil. Una vez que recibimos el audio el fichero, lo almacenamos en nuestro servidor, en una ruta segura y con un nombre único y, al igual que con la imagen, actualizamos la base de datos con la URL de este fichero. Cuando todo se ha completado y si no ha ocurrido ningún error, devolvemos por la API REST a la aplicación móvil que todo se ha completado correctamente; si ha ocurrido algún error, devolvemos a la aplicación el error para que muestre un mensaje.

Por otra parte, modificamos la función **getAllPaquetes** del controlador **adminCtrl** (**4.1.2.1.1 Actualización de la API REST**) para que devuelva entre sus campos la URL del sonido asociado a cada paquete.

En este punto modificamos el schema de la segunda fase para añadir un par de nuevos atributos requeridos por nuestra investigadora y propagamos los cambios a MongoDB para soportar estos nuevos campos: transcripción y notas.

El primero de ellos contendrá una transcripción literal del mensaje de audio. Esta se realiza de forma manual desde el panel de administración (Figura 46: Resultado final aplicación web). [8]

El segundo de ellos contiene una nota de texto que se hace de forma manual desde la aplicación web y es llevada a cabo por la investigadora. Su objetivo es que pueda tomar algún apunte relevante acerca del paquete en cuestión.

En consecuencia, debemos actualizar en el controlador **adminCtrl** la función **setDataPaquete** (**4.1.2.1.1 Actualización de la API REST**), ya que es la encargada de recibir el paquete de datos desde la web. Se procede, igual que para el campo nombre en la fase anterior, a la actualización de los nuevos campos “transición” y “notas”.

4.1.3.2 Aplicación móvil

En esta última fase de la aplicación móvil el hito de implementación más importante es el acceso al micrófono para la grabación de audio.

4.1.3.2.1 Grabación de audio e incorporación al paquete de datos

Para conseguir la grabación de audio, al contrario de lo que ocurre con los dos plugins usados anteriormente que solicitan automáticamente el permiso de acceso al hardware (**4.1.1.3.1.2 Captura y subida de una imagen** y **4.1.2.2.2 Obtención de ubicación**), debemos implementar la solicitud de

[8] La transcripción automática queda propuesta como Trabajo Futuro (Sección **7 Trabajos Futuros**).

permisos al usuario. Esta petición la efectuamos utilizando el plugin **Cordova diagnostic plugin** (GitHub, 2017), que nos permite llamar a la función **requestMicrophoneAuthorization** encargada de solicitar permisos al usuario de la aplicación para utilizar el micrófono de su smartphone.

El siguiente paso es la implementación. Para ello, por una parte debemos implementar la creación y almacenaje del fichero de audio en el terminal, y por otra, su envío a nuestro servidor, que está asociado a la imagen y las coordenadas dentro del mismo paquete de datos.

Conseguimos esta funcionalidad una vez más gracias a un plugin de Cordova, concretamente al plugin **cordova-plugin-audio-recorder-api** (GitHub, 2017), que nos permite capturar el audio del micrófono.

Su uso es sencillo y simplemente debemos invocar dos funciones: **audioRecorderAPI.record** y **audioRecorderAPI.stop** para inicializar la grabación y detenerla respectivamente. Este comportamiento es accionado por el usuario de la aplicación que decide cuando comenzar y detener la grabación. La Figura 35: Creación de audio desde la aplicación móvil muestra la vista creada en la aplicación móvil para las acciones descritas.

Una vez el audio se ha creado correctamente, se almacena en local hasta que el paquete sea enviado. Una vez ha sido enviado, lo eliminamos, al igual que hacemos con la imagen y las coordenadas, para que no ocupen memoria innecesaria en el teléfono del usuario.

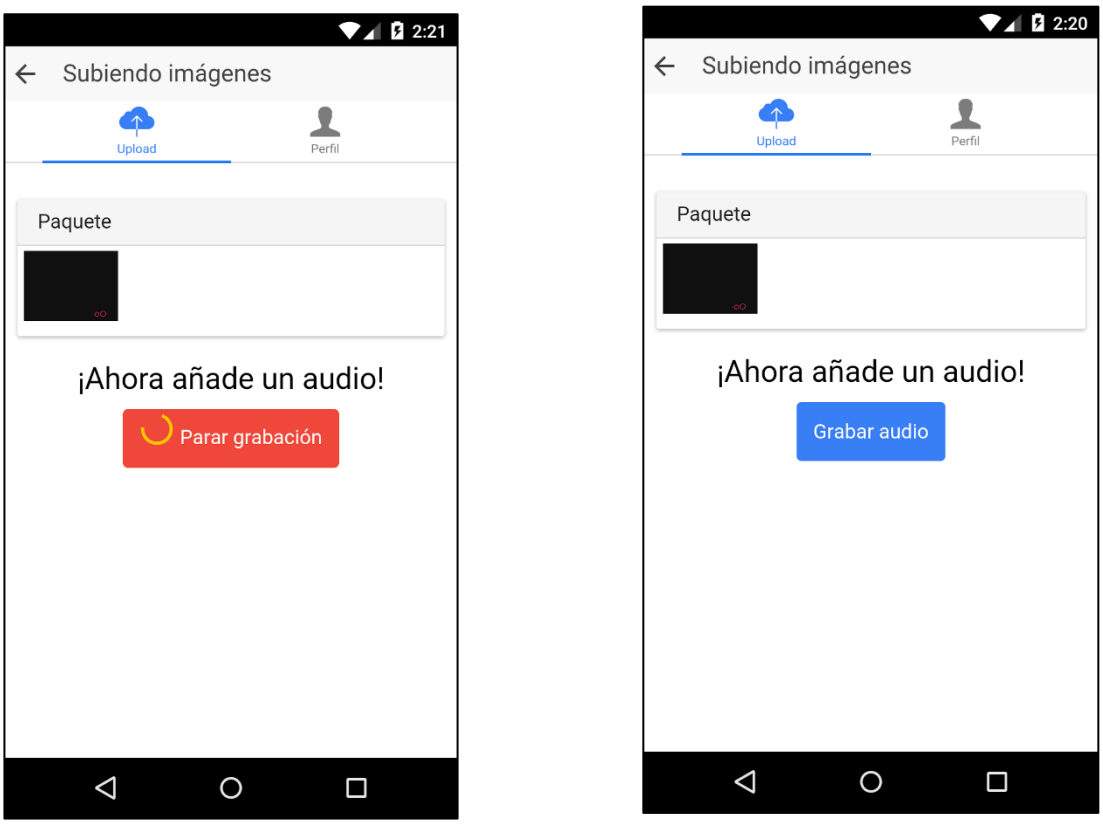

Figura 35: Creación de audio desde la aplicación móvil

4.1.3.3 Aplicación web

Una vez que tenemos completada la segunda fase de la aplicación web con el listado de paquetes, un mapa donde se localiza el paquete y una vista de los campos de los que dispone dicho paquete, realizamos una serie de mejoras sobre ella.

4.1.3.3.1 Nuevos campos en la sección de revisión de la información

Procedemos a añadir dos nuevas variables solicitadas por nuestra investigadora en el listado de campos de dichos paquetes: transcripción y notas. Para ello añadimos en la vista dichas variables y también en el controlador que modifica la información del paquete mediante la API REST (**4.1.3.1.1 Actualización de la API REST**).

La Figura 36: Columna dedicada a la revisión de datos muestra un detalle del panel de administración. Es la columna dedicada a la revisión de los datos por parte de Lidia. Se muestran los campos mencionados anteriormente y destaca la inclusión de un nuevo botón, “Play Audio”, que permite desde el panel de control oír la grabación. De esta forma, se puede introducir manualmente su transcripción. Como en la fase anterior, una vez terminada la edición guardamos los cambios mediante el botón guardar.

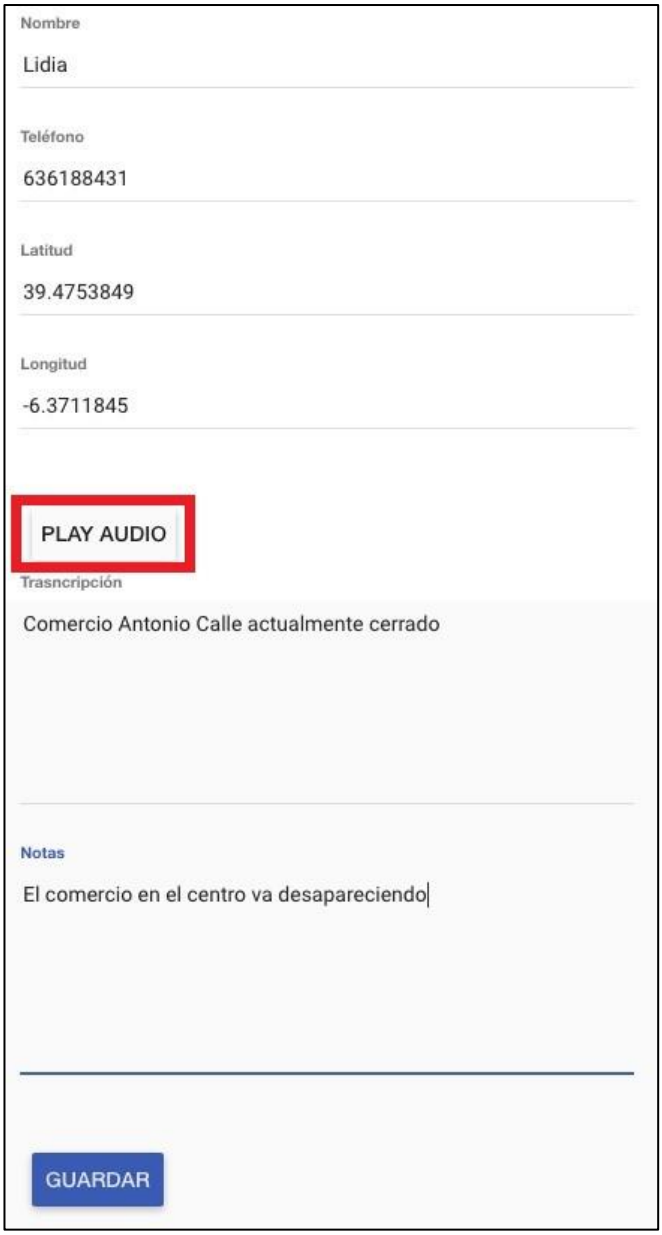

Figura 36: Columna dedicada a la revisión de datos

4.1.3.3.2 Exportación de datos a ArcGIS

Una de las máximas que hemos seguido en el desarrollo de este TFM es la de adecuarnos a los requisitos de nuestra investigadora. Tras las sucesivas reuniones descubrimos que ella manejaba con soltura un software para el tratamiento de datos geográficos: ArcGIS. Este software le permite realizar una cantidad enorme de operaciones, dispone de mapas que puede personalizar, puede obtener gráficas y, en definitiva, le es muy útil para el análisis de los datos. El escenario sería la integración de este software en la aplicación web, pero ello conllevaría un esfuerzo que supera el ámbito de este TFM[9].

No obstante, instalamos y estudiamos este software, y encontramos una solución que podía servir y que además es lógica en un desarrollo incremental. Descubrimos que ArcGIS tiene la capacidad de exportar e importar sus datos a diferentes formatos. En el conjunto de formatos aceptados se encontraba el clásico CSV (Es.wikipedia.org, 2017). Por tanto, si podíamos exportar nuestros datos a este formato, podríamos importarlos con ArcGIS. Necesitábamos construir un proceso interno en la web app que fuera activado mediante el accionamiento de un botón en la interfaz se encargue encargue de realizar la operación. El proceso recorre todos los datos mostrados en la columna izquierda y los exporta al formato csv. Para realizar esta tarea usamos un plugin de AngularJS llamado ngCsv (GitHub, 2017). La Figura 37: Botón de exportar de la aplicación web muestra el detalle del botón exportar en la aplicación web.

[9] No obstante, ha sido propuesto como uno de los **7 Trabajos Futuros**.

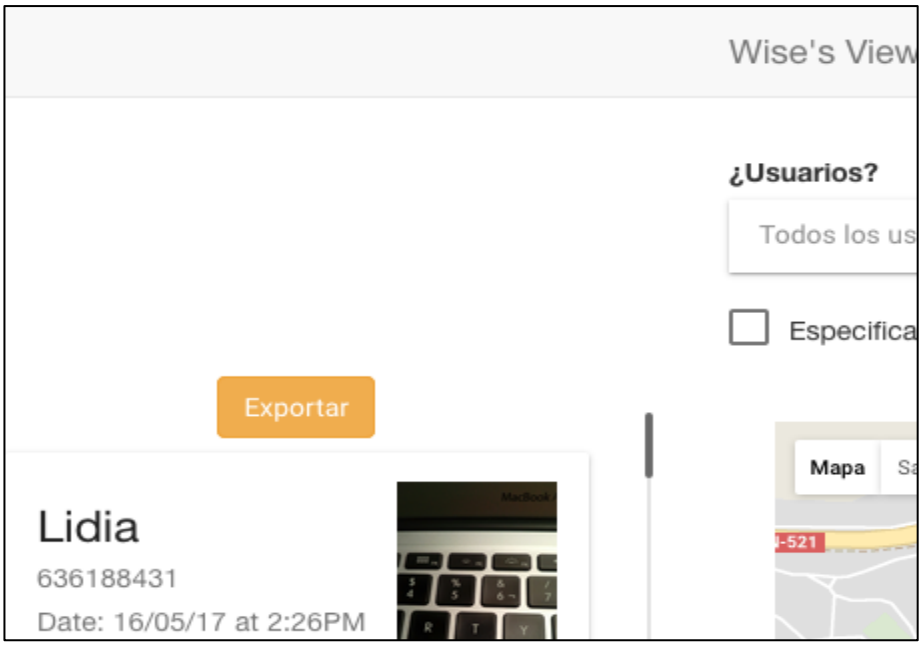

Figura 37: Botón de exportar de la aplicación web

4.1.3.3.3 Expansión de funcionalidad en el buscador

Esta mejora consiste en añadir a la búsqueda una información y la posibilidad de acotarla a un intervalo de fechas (Figura 38: Buscador tanto por usuarios como por fechas en la aplicación web).

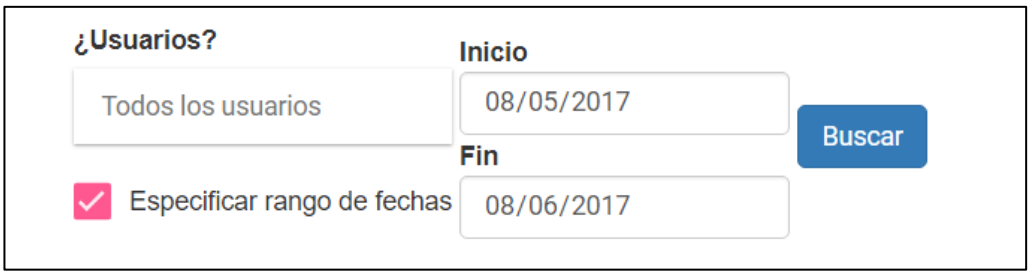

Figura 38: Buscador tanto por usuarios como por fechas en la aplicación web

Una característica interesante de este proceso es que nos permite exportar solamente los datos filtrados. Por tanto, es posible realizar la búsqueda de los datos recibidos en la última semana y proceder, con la funcionalidad descrita anteriormente, a exportar ese subconjunto de datos al formato compatible con ArcGIS.

4.1.3.4 Uso y manejo de los Stores: Subir una aplicación

Para finalizar nuestra beta abierta, procedemos a subir tanto al **Play Store** de Google como al **Apple Store** de Apple la aplicación **La Perspectiva del Sabio**. Como hemos señalado anteriormente, nuestra aplicación móvil es híbrida. Ionic nos permite crear tanto la versión de lanzamiento de Android (.**apk**) como la versión de lanzamiento de iOS.

4.1.3.4.1 Creación de un icono y un splash para la aplicación

Uno de los requisitos obligatorios de ambos stores es la inclusión de un icono para mostrarlo en el escritorio del dispositivo y un splash, que es una pantalla de carga que aparece hasta que la aplicación está en ejecución. Es interesante señalar que ambos deben tener un tamaño concreto para cada tipo de dispositivo (smartphone, tablet…). Afortunadamente, Ionic nos proporciona un template para PhotoShop, y nos recomienda usar una sección del lienzo para conseguir todos los tamaños a partir del mismo diseño. Afortunadamente también, nuestra investigadora, además de socióloga, es una entusiasta del retoque fotográfico, al mismo tiempo que tiene ideas y buen gusto diseñando logotipos. La Figura 39: Icono de la Perspectiva del sabio muestra el diseño realizado para **La Perspectiva del Sabio**. Es usado tanto para el splash como para el icono.

Figura 39: Icono de la Perspectiva del sabio

4.1.3.4.2 Construcción del APK de Android y subida al Play Store

Los requisitos para la creación de la versión de lanzamiento de Android son disponer de un ordenador con **Windows** o **Mac OS**, una versión igual o superior a **Java 1.6**, tener instalado **Java Development Kit (JDK),** instalar una versión igual o superior de **Android SDK 2.0,** y tener una cuenta de desarrolladores de google **(Ionicframework.com, 2017).**

En primer lugar procedemos a construir el archivo .**apk** que subiremos al Play Store. Para ello ejecutamos el siguiente comando desde el directorio de nuestro proyecto Ionic:

```
$ ionic cordova build --release android
```

Este comando nos genera un .**apk** inicial, el cual tenemos que firmar con un certificado antes de poder subirlo a Play Store. Para firmarlo, generamos una clave privada utilizando el comando **keytool** de java, el cual genera el fichero **my-release-key.keystore**. Es muy importante guardar copias de dicho fichero ya que, para posteriores modificaciones de la aplicación móvil, necesitamos firmarla con el mismo fichero. Si no, no podremos realizar modificaciones y no existe un proceso de recuperación de firma. A continuación procedemos a firmar el .**apk** con **jarsigner.** Previamente a la subida a Play Store, debemos utilizar la herramienta **Zipalign** que nos devuelve el .**apk** final que, al fin, podemos subir al Play Store[10].

Para subir la aplicación al Play Store accedemos a la dirección de Google Play Store Developer Console (Play.google.com, 2017). Una vez logueados iniciamos la creación de una aplicación con un nombre original que no esté siendo utilizado por otra aplicación. Hay que completar una ficha de la

[10] Obviamente, para poder subir una aplicación al Play Store debemos tener una cuenta de desarrolladores de Google con un precio de 25€ al año. Durante ese periodo, el desarrollador puede subir todas las aplicaciones que desee.

aplicación con distintos campos informativos como una descripción breve y otra detallada del cometido de la aplicación, así como añadir capturas de pantalla de la aplicación y realizar una clasificación de contenido contestando una serie de preguntas.

Una vez completada la ficha de la aplicación seleccionamos el tipo de versión que queremos subir y que, en nuestro caso, será **beta**.

A continuación subimos el .**apk**, el cual entra en una fase de revisión por parte de Play Store. Normalmente, el revisor tarda alrededor de 36 horas en ofrecer una respuesta sobre la petición de lanzamiento de la aplicación. En el caso de **la Perspectiva del Sabio**, nos abrieron una incidencia solicitándonos especificar parte de la información proporcionada; en nuestro caso, se refería a quién era el público objetivo de nuestra aplicación. Respondimos explicando que nuestro público objetivo serían personas mayores de 65 años. Fue suficiente y en menos de 12 horas la aplicación estaba disponible en Play Store como beta.

1.1.1.1.1 Construcción de la versión de lanzamiento de iOS y subida al Apple Store (Ionicframework.com, 2017)

Los requisitos para la creación de la versión de lanzamiento en iOS son disponer de un ordenador con **MAC OS**, así como tener una versión igual o superior de **Xcode 8** (programa de desarrollo de aplicaciones iOS) (Developer.apple.com, 2017), y tener una cuenta de desarrolladores de Apple.

A diferencia de la construcción de la versión de lanzamiento de Android, donde debíamos firmar, guardar el certificado y optimizar el .apk, para construir la versión de iOS solamente tenemos que ejecutar el siguiente comando desde el directorio de nuestro proyecto Ionic:

```
$ ionic cordova build --release ios
```

Lo que no cambia para subir la aplicación es la necesidad de tener una cuenta de desarrollador de Apple. Para ello accedemos a la dirección web de Apple Developer Program (Developer.apple.com, 2017) y creamos una cuenta de desarrolladores, que tiene un precio de 99€ anuales, otorgando un certificado válido por un año para subir aplicaciones (Figura 40: Membership details Apple Developer Program).

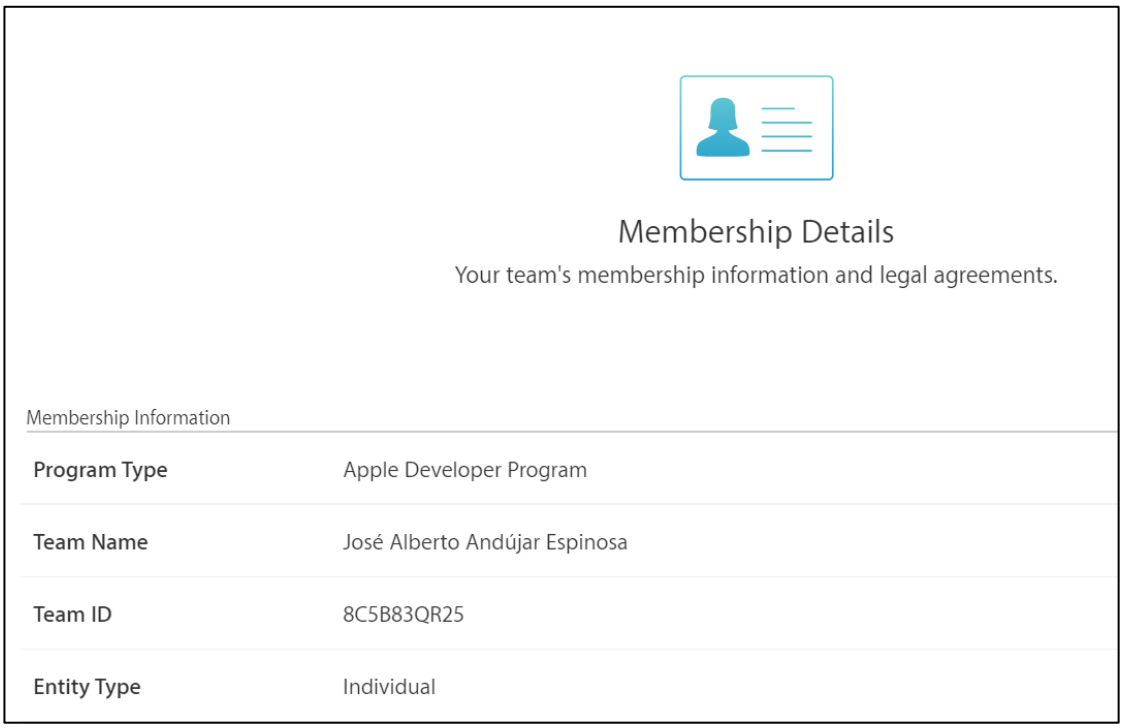

Figura 40: Membership details Apple Developer Program

Dentro de la página web de Apple Developer Program (Developer.apple.com, 2017) creamos un certificado para firmar nuestras aplicaciones (Figura 41: Certificados en producción para iOS).

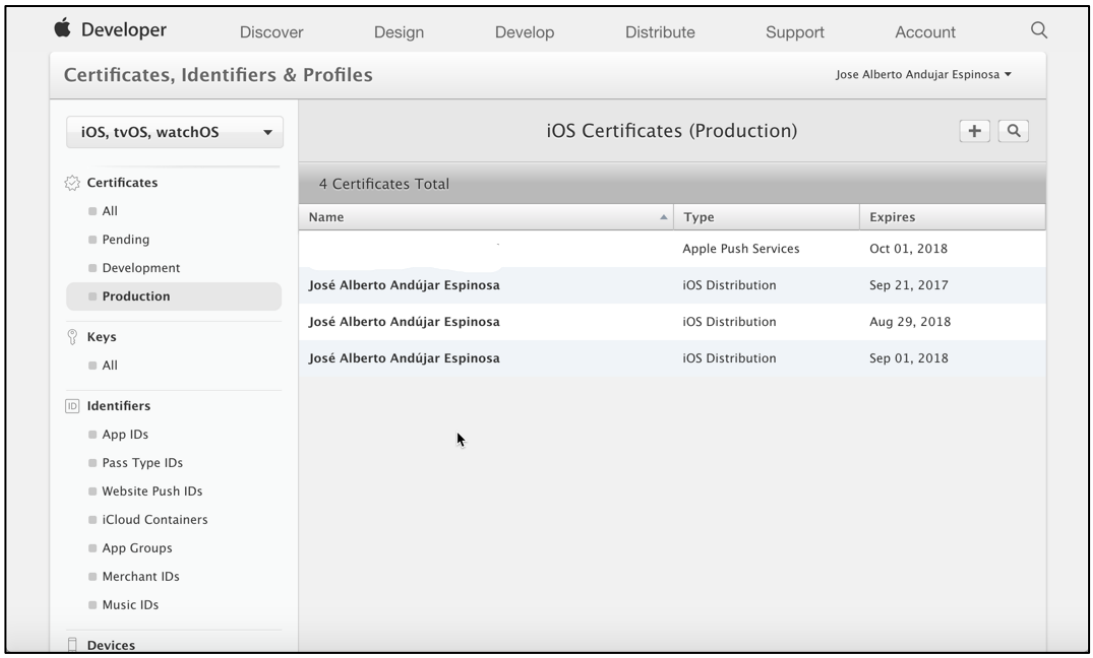

Figura 41: Certificados en producción para iOS

A continuación, abrimos el programa de desarrollo de Mac OS, **Xcode,** introducimos nuestras credenciales de desarrollador, y automáticamente se accede a los certificados creados en la web. Posteriormente, sin salir de **Xcode**, cargamos nuestra aplicación construida en Ionic y la firmamos con nuestro certificado. De esta forma queda configurada para la subida a Apple Store (Figura 42: Firma de las aplicaciones en Xcode).

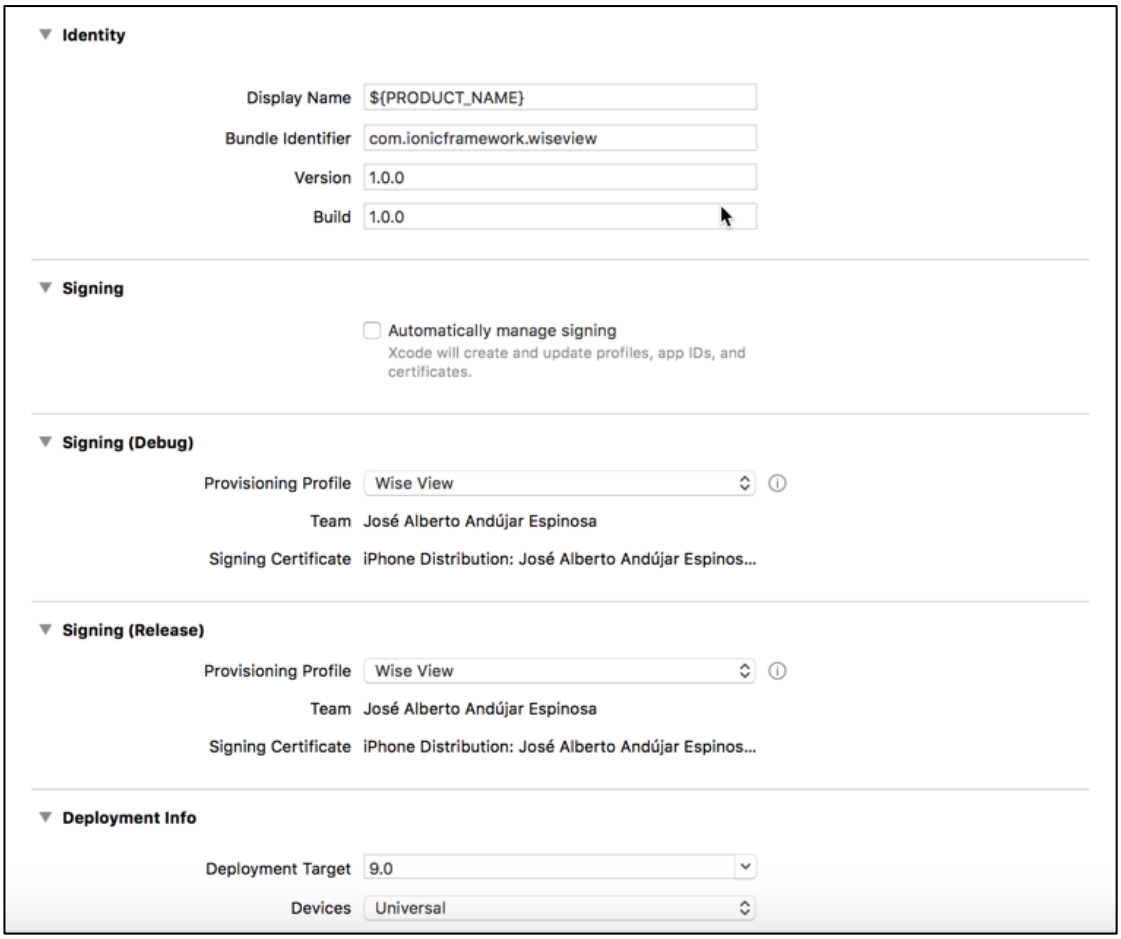

Figura 42: Firma de las aplicaciones en Xcode

En la página web de Apple Developer Program (Developer.apple.com, 2017) configuramos nuestra ficha de la aplicación introduciendo los datos solicitados: nombre, URL de la política de privacidad, precio y disponibilidad (Figura 43: Ficha de aplicaciones en Apple Developer Program).

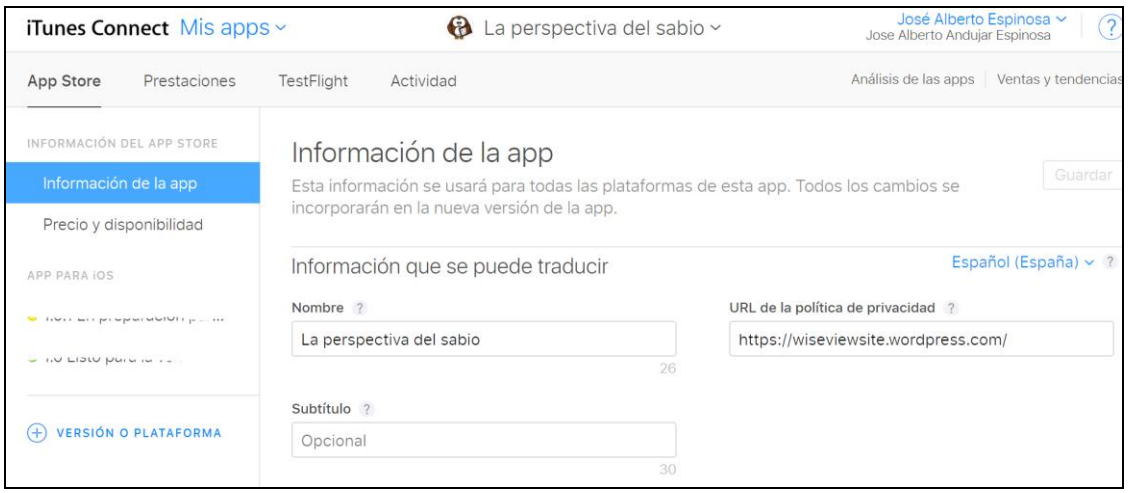

Figura 43: Ficha de aplicaciones en Apple Developer Program

Creada ya la ficha en el Apple Store, volvemos a **Xcode** para subir nuestra aplicación que está asociada a dicha ficha (Figura 43: Ficha de aplicaciones en Apple Developer Program). A diferencia del Play Store, el equipo de Apple encargado de la revisión emplea, normalmente, más de 72 horas en validarla. Tras su validación estará disponible en Apple Store, también, en beta abierta.

4.1.3.5 Revisión con el cliente y testeo final

Alcanzado este punto, el producto está listo y es el momento de realizar una prueba final más extensa. En nuestro caso, la prueba estaba clara: Lidia pretende utilizar la aplicación como si fuera un usuario final como si fuera un usuario demo. Pero no solo se quiere limitar a hacer fotografías al azar, subir audios, etc., es decir, no solo quiere validar tecnológicamente el sistema, sino que quiere categorizar los dos barrios de la ciudad de Cáceres que forman parte de su estudio utilizando para ello la aplicación móvil que hemos desarrollado. La prueba fue altamente satisfactoria y, como se podrá observar en la sección **5 Resultados y discusión**, le sirvió para publicar un póster en un congreso científico.

En cualquier caso, a continuación describiremos el resultado final de la aplicación móvil y web así como los aspectos más relevantes de la misma.

Tal vez la parte más impactante de esta última fase sea el resultado visual final de aplicación de móvil. El prestigio de encontrarla en los stores consigue un grado de profesionalidad del gusto de nuestra investigadora. Muestra de ello es la visión de la aplicación en Play Store antes de su instalación y el icono de la app en el escritorio de un Smartphone junto a otras famosas aplicaciones (Figura 44: Vista de la aplicación en el Play Store y vista de la aplicación en nuestro terminal). Sin duda, un acabado bonito y elegante a la altura del proyecto.

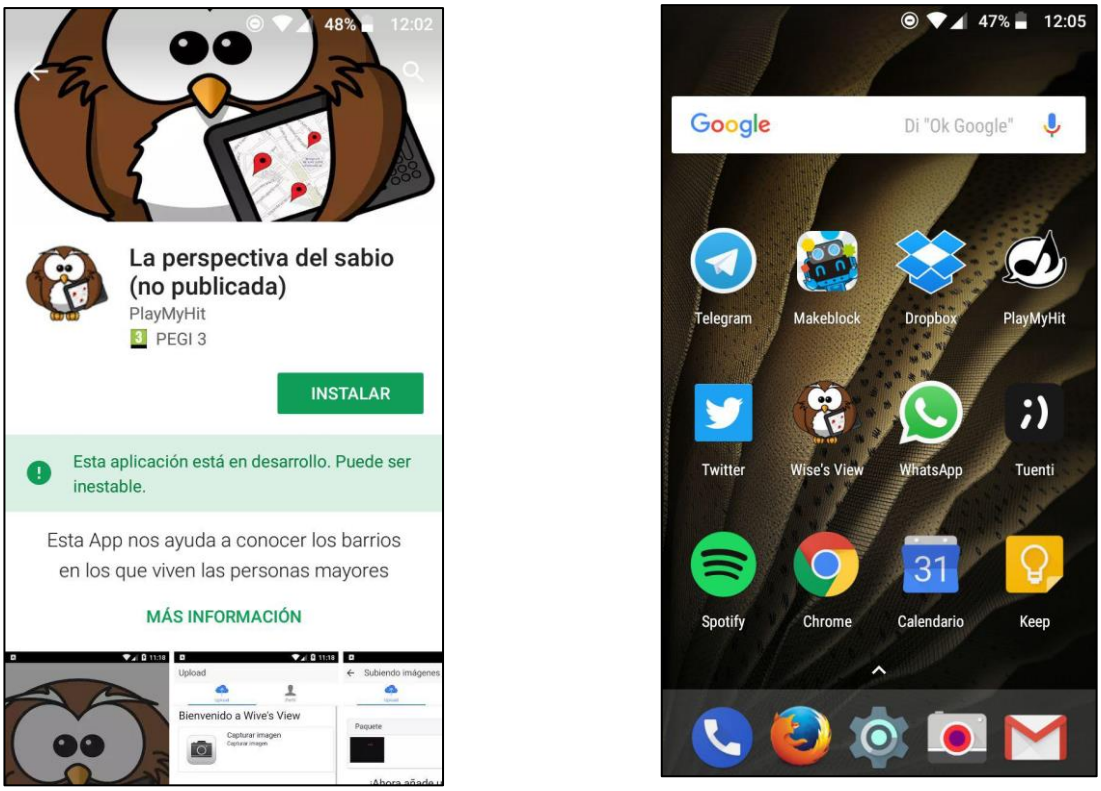

Figura 44: Vista de la aplicación en el Play Store y vista de la aplicación en nuestro terminal

El diseño visual de la aplicación, como dijimos en el planteamiento del problema, se basa en vistas sencillas y útiles, en colores claros y con un tamaño de fuente suficiente para nuestros usuarios. Se evitan opciones innecesarias tratando de no distraer al usuario del objetivo principal de recabar la información. Una vez registrados, cada pantalla recaba una parte de la información, y el proceso no se inicia hasta que el usuario lo decide: primero la imagen, luego el audio y finalmente el usuario envía el paquete completo, al que hemos añadido automáticamente la ubicación y las fechas de creación de los ficheros. La Figura 45: Secuencia completa de vistas implementadas muestra la secuencia completa de vistas implementadas y su aspecto final en un dispositivo Android, así como las solicitudes de permiso de acceso al hardware para el GPS y para el micrófono. Se omiten, por razones de espacio en la página, la solicitud de permisos para el acceso a la cámara y sistema de ficheros, y las vistas dedicadas a informar de errores.

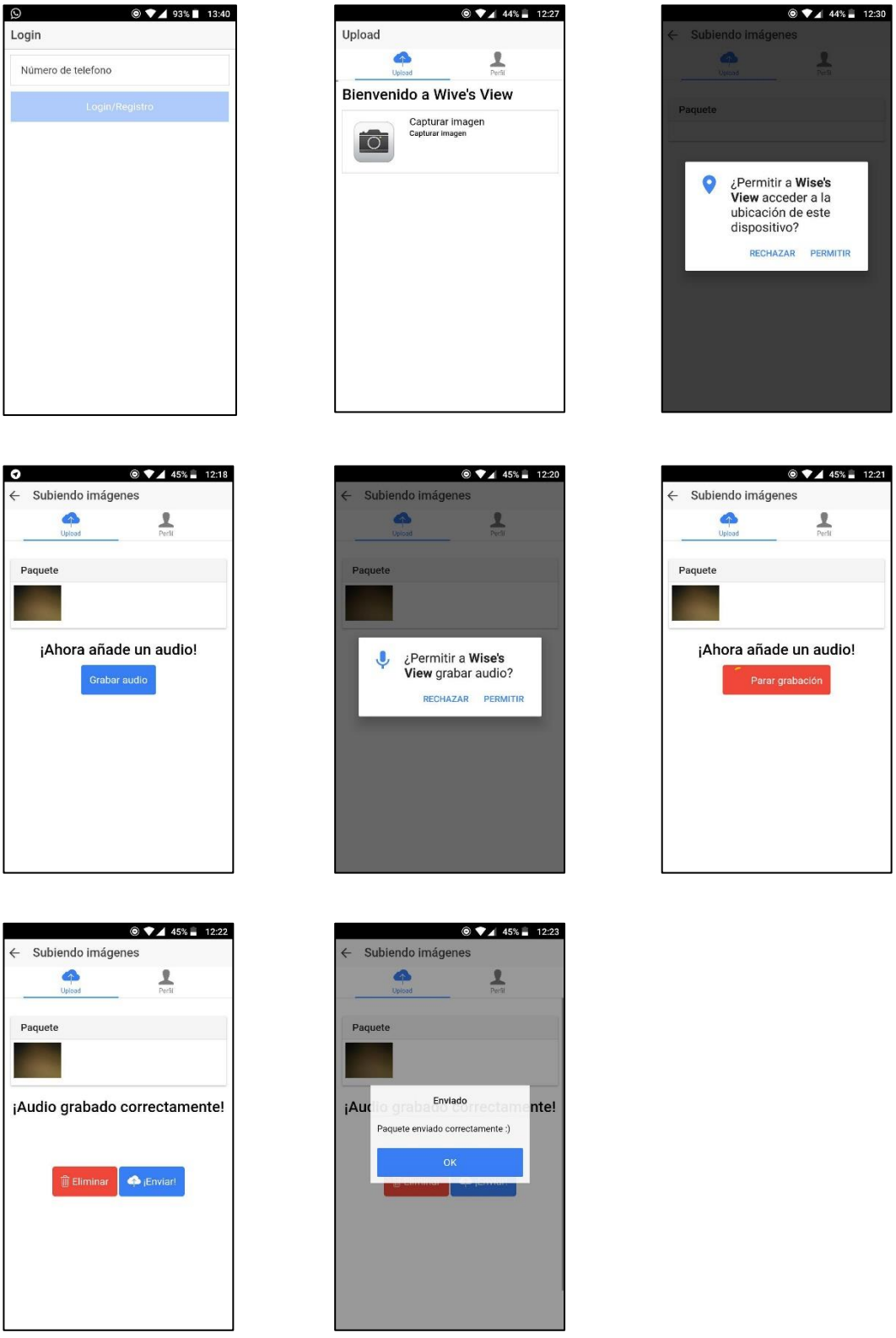

Figura 45: Secuencia completa de vistas implementadas

Consideramos que el resultado final supera el esperado. No obstante, y como no hemos podido efectuar tantos testeos y con tantos usuarios como nos hubiera gustado, hemos decido subirla a los stores como beta abierta para que podamos obtener un feedback productivo. Si algo hemos aprendido sobre el desarrollo del software durante la carrera y el máster es que siempre existirán errores, y por ello las actualizaciones son fundamentales para corregirlos y mejorar la experiencia de usuario.

Respecto a la aplicación web, el resultado final cumple con las expectativas y demandas de Lidia, y este era nuestro objetivo principal, por lo que estamos también satisfechos en ese sentido. El aspecto final es más funcional que estético (recordemos que buscábamos una vista compacta que permitiese percibir la información de un vistazo), aunque hemos mantenido las pautas de diseño de claridad y sencillez, así como la división por zonas de forma que sean intuitivas, tanto su navegación como utilización. La Figura 46: Resultado final aplicación web muestra el aspecto final de la aplicación web[11].

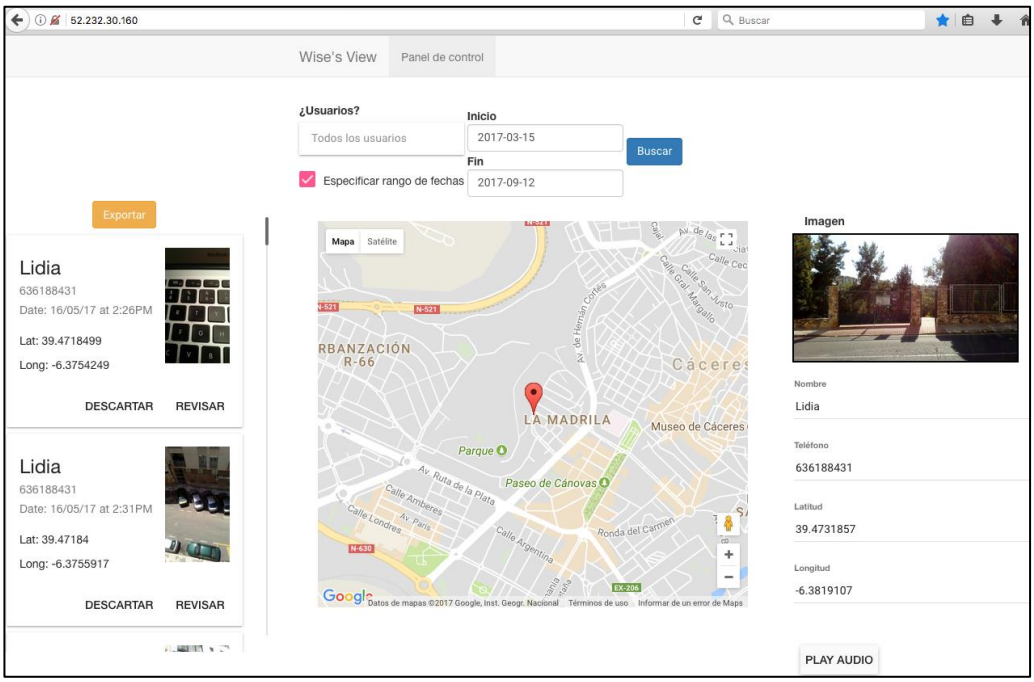

Figura 46: Resultado final aplicación web

[11] Recordamos que el acceso se realiza a través de una IP pública (52.232.30.160) y mediante login. Aunque no se comenta en los trabajos futuros, es obvio que si hubiera financiación se compraría un nombre de dominio.

La zona izquierda de la Figura 46: Resultado final aplicación web nos permite la navegación por los datos, el número de ellos corresponde al filtrado que se realiza en la zona de búsqueda (parte superior de la imagen), y que puede ser para un usuario en particular o para todos. Para cualquiera de las búsquedas permite acotarla a un intervalo de fechas. Los datos mostrados en la columna izquierda, derivados del filtrado anterior, son los que se exportarán al formato compatible con ArcGIS. (**4.1.3.3.2 Exportación de datos a ArcGIS**)

Para cada elemento de esta columna izquierda tenemos dos comportamientos: descartar y revisar. El primero elimina el dato de la BD y el segundo lo muestra en la columna derecha de la imagen y nos detalla la información contenida. Los campos editables para el registro son: Nombre, Transcripción y Notas. Una vez realizados los cambios pulsamos Guardar (Figura 36: Columna dedicada a la revisión de datos). En esta zona también destaca la posibilidad de reproducir el audio almacenado mediante el botón PlayAudio.

En la zona central, ocupando un lugar destacado en la interfaz, se encuentra el mapa embebido de GoogeMaps. Si revisamos un paquete, el mapa se desplaza automáticamente centrándose sobre la “chincheta”, es decir, sobre sus coordenadas GPS. Sobre el mapa se pueden realizar zoom y desplazamientos verticales y horizontales. Es posible también ampliarlo al modo pantalla completa y utilizar la vista satélite o usar directamente Google Street View para completar algún detalle acerca de la ubicación. (Figura 47: Detalle de la figura Street View en la aplicación web)

Figura 47: Detalle de la figura Street View en la aplicación web

Todo este proceso de creación y adecuación a las especificaciones de Lidia implicó muchas horas de bocetos, errores, descartes y conversaciones con ella. No obstante, estamos orgullosos del resultado, especialmente por su utilidad y sencillez a la hora de mostrar la información, así como de las posibilidades de expansión que nos ofrece.

5. Resultados y discusión

Uno de nuestros objetivos principales era el de ser capaz de conseguir crear la Perspectiva del Sabio. El desafío era notable. No solo consistía en una aplicación móvil, había que construir un sistema entero que permitiese almacenar los datos proporcionados por dicha aplicación y que además los pudiese ofrecer de una determinada manera a nuestra investigadora. A la luz de lo expuesto en esta memoria, consideramos conseguido este objetivo ya que hemos logrado concebir y hacer realidad un sistema acorde con las especificaciones de nuestra socióloga Lidia Domínguez.

Los resultados son prometedores, ya que han permitido a la investigadora categorizar los barrios desde su perspectiva fotografiando tiendas, parques y sus ubicaciones y construyendo, con todo ello, un mapa personalizado para ArcGIS.

Fruto de lo anterior, y junto al análisis previo de las entrevistas individuales, nuestra investigadora ha conseguido un resultado científico para su carrera: la publicación y defensa del póster *Las personas mayores sactivas en entornos urbanos* en el V Congreso Internacional de Ciencias Sociales celebrado el pasado julio en la Universidad Rey Juan Carlos de Madrid (Domínguez-Párraga, 2017). Actualmente se encuentra trabajando en la publicación de un artículo científico que extiende el contenido del póster.

De la misma forma hemos logrado los objetivos relacionados con el aprendizaje de los dos stores más importantes. La aplicación móvil ha sido validada por los responsables de Play Store y Apple Store, y está disponible en fase beta abierta. Puede ser descargada e instalada por cualquier usuario.

En cambio, es una lástima que antes de terminar este TFM no haya sido probada masivamente por el público objetivo de nuestra investigadora, dado que no ha podido contactar y organizar su grupo de trabajo. No obstante, previsiblmente podrá hacerlo durante último trimestre del 2017 y el primer trimestre de 2018.

Aun así, estamos muy orgullos de las valoraciones positivas que hemos recibido sobre la aplicación móvil por parte de algunos de sus mayores que la han considerado muy sencilla, bonita e intuitiva.

Desde el punto de vista técnico nos hubiera gustado haber conseguido que la aplicación hubiera sido utilizada por una comunidad de mayores más amplia, debido a que su feedback hubiera sido muy interesante; además, se habrían corregido errores, se habría mejorado la versión actual en los stores, y tal vez, podríamos haber alcanzado la versión de producción. No obstante, esperamos alcanzarlo en un futuro cercano.

Al nivel personal, estoy muy contento con el resultado de este TFM. He contribuido con mi esfuerzo a crear una herramienta que puede mejorar la sociedad y, al mismo tiempo, he descubierto un mundo alejado de la Informática: la Sociología, que me ha permitido reflexionar sobre nuestros hábitos y sobre nuestro entorno descubriendo, con asombro, lo predecible que puede ser el ser humano en su conjunto.

6. Conclusions

The following items describe the main conclusions of this Master thesis:

- This project has allowed me to acquire a set of competencies very important to my profession. The knowledge required to develop this work situate me at an optimal level in a set of cutting-edge technologies that are very demanding today.
- Sociological objective of this work has allowed me to reflect on how important can be the technology to obtain data and relevant information that allows us to understand what are the need of the different communities that compose our society.
- The smartphone is almost in everybody’s pockets and can serve as a source of data for the study of the requirements that society has. Our approach can be extended to cover other extracts of society and achieve a city, a neighborhood, more appropriate and more thought for all citizens. This information can help corporations, public services and private companies come up with solutions to these problems and generate business opportunities. I think this trend is also manifested in the opendata that is allowing the appearance of new applications that try to make our life easier.
- It has been an exciting challenge and a pleasure the whole process of developing this project. After studying computer science and the subsequent master's degree, I have always had in mind to build something real with the knowledge acquired. This work is an example of how to build a complete system adapted to the real requirements of a fictitious client. The developed elements could be requested by an enterprise both individually or as a whole.
- Personally, and to achieve this, I have had to face and solve new challenges and problems, at the same time that I have learned new technologies and assimilated new concepts.

7. Trabajos Futuros

Entre los trabajos futuros, el principal es su mantenimiento. Me he comprometido a actualizar, mantener y mejorar el sistema durante el tiempo que necesite usar la aplicación nuestra investigadora Lidia Domínguez.

No obstante, y como suele ocurrir al término de un proyecto, es inevitable pensar en las mejoras posibles. Los siguientes puntos resumen las ideas y trabajos futuros más relevantes para convertir a **La Perspectiva del Sabio** en una herramienta más potente y versátil.

7.1 Trabajos futuros en la aplicación web

- Añadir un chat. Estamos convencidos de que una comunicación directa con las aplicaciones móviles desde el panel de administración sería muy útil, ya que nos permitiría formular la pregunta inicial mediante este canal, además de permitir a Lidia resolver dudas en tiempo real u orientar a los usuarios en el proceso de recogida de la información.
- Paginación de datos en la aplicación web. En la columna izquierda del panel de administración prevemos una cantidad enorme de datos, y sería interesante poder paginar los datos para un acceso más cómodo.
- Flag exportado. Necesitamos la inclusión de un atributo en la colección que nos permita saber si un dato ya ha sido exportado a ArcGIS. Esta opción estaría disponible en la aplicación web y simplifica el trabajo de Lidia.
- Búsqueda por el campo nombre. Incluir la búsqueda por el nombre de usuario que Lidia ha asociado al número de teléfono.
- Volcado de datos al mapa. Tanto si se muestran todos los datos o si se hace una búsqueda parcial, deberían aparecer sus ubicaciones en

el mapa, representadas con las famosas “chinchetas” de GoogleMaps. Esto permitiría tener una visión global de las zonas con mayor densidad de datos.

- Interacción con el mapa. Al hacer click en las “chinchetas” se debe poder revisar ese dato, al igual que ocurre cuando se selecciona en la columna izquierda.
- Reemplazar GoogleMaps por ArcGIS. Sin duda, la más ambiciosa de las expuestas. Esta mejora supondría tener directamente en el panel de administración todas las capacidades de ArcGIS, y sin duda, simplificaría y optimizaría el trabajo diario de la investigadora.

7.2 Trabajos futuros en la aplicación móvil

- Incluir la grabación de vídeo. No estamos seguros de si puede ser una buena opción o no, y aún está siendo debatida. A priori, parece interesante usar el vídeo para explicar una situación dinámica, difícil de explicar solo con el audio y la fotografía estática.
- Coordenada de altitud. Incluir la coordenada de altura en la información asociada a la fotografía puede ser útil para caracterizar mejor el terreno. Este cambio gracias, a la elección de las tecnologías usadas (JSON, API REST, NodeJS, MongoDB…), requiere un esfuerzo mínimo.

7.3 Servidor

- Transcripción automática del mensaje de texto. Una vez recibido el audio por parte de la aplicación, lanzamos un proceso interno que utiliza las APIs de sistemas de reconocimiento de voz como Cloud Speech API de Google, y obtiene una transcripción automática del mensaje de texto. De esta forma en el panel de administración se tendría una versión a validar por la investigadora.

Referencias bibliográficas

Almeida, M. (2014). A identidade da velhice. En M. Almeida y J. Apóstol (Eds.), Envelhecimento, Saúde e Cidadania (pp. 11-37). Coimbra: Unidade de Investigação em Ciências da Saúde: Enfermagem Escola Superior de Enfermagem de Coimbra. Doi. 10.1017/CBO9781107415324.004

Alves Apóstolo, J.L.,(2013). Envelhecimento e saúde e cidadania. *Revista de* enfermagem III serie, 9, 205-208.

Azure.microsoft.com. (2017). Microsoft Azure: plataforma y servicios de informática en la nube. [online] Available at: https://azure.microsoft.com/es-es/ [Accessed Jul. 2017].

BAEZ-IBARRA, A, ARELLANES-CANCINO, N y SOSA-PERDOMO, A. (2015). Efectividad de la aplicación de metodologías ágiles para el desarrollo de apps móviles. Un caso de estudio. *Revista de sistemas computacionales y TIC's (2)*

Bazo, M. T. y Maiztegui, C. (1999). Sociología de la vejez. En M. T. Bazo (coord.). Envejecimiento y sociedad: Una perspectiva internacional. (pp. 42-102). Madrid: Médica Panamericana.

— (2000). Sociedad y vejez: La familia y el trabajo. En R. Fernández-Ballesteros (Dir.). Gerontología social (pp. 241-250). Madrid: Editorial Pirámide.

BBVAOpen4U. (2017). *API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos*. [online] Available at: https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos [Accessed Jul. 2017].

Beard, J. R. & Bloom, D. E. (2015). Towards a comprehensive public health response to population aging. Lancet 385(9968), 658-661. Doi. 10.1016/S0140-6736(14)61461-6

Bermejo Corrales, M. (2016). *Desarrollo de un sistema de autoescalado dinámico de base de datos distribuida MongoDB sobre una plataforma cloud OpenStack* (Master's thesis).

Bizspark.microsoft.com. (2017). Home | Microsoft BizSpark. [online] Available at: https://bizspark.microsoft.com/ [Accessed Jul. 2017].

Bsonspec.org. (2017). *BSON - Binary JSON*. [online] Available at: http://bsonspec.org/ [Accessed Jul. 2017].

Buffel, T.; Phillipson, C. & Scharf, T. (2012). Ageing in urban environments: Developing “age-friendly” cities. Critical Social Policy, 32(4), 597-617. Doi. 10.1177/0261018311430457

Caprara, M.G y López Bravo, M.D. (2014). Psicogerentología. Madrid: Editorial UDIMA.

Carlos Azaustre | Formación en JavaScript. (2017). *Automatizar tareas en JavaScript con Grunt.js*. [online] Available at: https://carlosazaustre.es/automatizar-tareas-en-javascript-con-grunt-js/ [Accessed Jul. 2017].

Carlos Azaustre | Formación en JavaScript. (2017). *Manejando librerías JavaScript con Bower.io*. [online] Available at: https://carlosazaustre.es/manejando-librerias-javascript-con-bower/ [Accessed Jul. 2017].

Chao, A. A. (2005). Desigualdades mundiales ante el proceso de envejecimiento demográfico. Recerca: revista de pensament i anàlisi, (5), 41-62.

Chrome.google.com. (2017). *Allow-Control-Allow-Origin: **. [online] Available at: https://chrome.google.com/webstore/detail/allow-control-allow-origi/nlfbmbojpeacfghkpbjhddihlkkiljbi [Accessed Jul. 2017].

Codecademy. (2017). *Codecademy - learn to code, interactively, for free*. [online] Available at: https://www.codecademy.com/ [Accessed Jul. 2017].

Cohen, D. A.; Mason, K.; Bedimo, A.; Scribner, R., Basolo, V. & Farley, T. A. (2003). Neighborhood physical conditions and health. American Journal of Public Health, 93(3), 467-471

Cordero del Castillo, P. (2006). Situación social de las personas mayores en España. Humanismo y trabajo social,(5),161-195. http://buleria.unileon.es/xmlui/handle/10612/1493

Cordova.apache.org. (2017). *Architectural overview of Cordova platform - Apache Cordova*. [online] Available at: https://cordova.apache.org/docs/en/latest/guide/overview/index.html [Accessed Jul. 2017].

Cordova.apache.org. (2017). *File Transfer - Apache Cordova*. [online] Available at: https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-file-transfer/ [Accessed Jul. 2017].

Cordova.apache.org. (2017). *Geolocation - Apache Cordova*. [online] Available at: https://cordova.apache.org/docs/en/latest/reference/cordova-plugin-geolocation/ [Accessed Jul. 2017].

Cordova.apache.org. (2017). *Hooks Guide - Apache Cordova*. [online] Available at: https://cordova.apache.org/docs/en/latest/guide/appdev/hooks/ [Accessed Jul. 2017].

Cuello J., Vittone J. (2015). *Diseñando apps para móviles*. XcUiDi. Argentina.

Delgado, M. (2003). La construcción social de la vejez. Jano, 64(1.474), 12-17.

Developer.apple.com. (2017). *Apple Developer Program - Apple Developer*. [online] Available at: https://developer.apple.com/programs/ [Accessed Jul. 2017].

Developer.apple.com. (2017). *Xcode - Apple Developer*. [online] Available at: https://developer.apple.com/xcode/ [Accessed Jul. 2017].

Dias, I. (2012). O uso das tecnologias digitais entre os seniores: motivações e interesses. Sociologia, Problemas e Práticas, (68), 51-77. Doi. 10.7458/SPP201268693

Digitalocean.com. (2017). *¿Cómo instalar MongoDB en Ubuntu 16.04?* | *DigitalOcean*. [online] Available at: https://www.digitalocean.com/community/tutorials/como-instalar-mongodb-en-ubuntu-16-04-es [Accessed Jul. 2017].

Dominguez Párraga, Lidia (2017) Las personas mayores activas en entornos urbanos. V congreso internacional de ciencias sociales. Universidad Rey Juan Carlos Madrid. 12-14 Julio 2017.

Eitler, T. W.; McMahon, E. y Thoerig, T. (2013). Ten principles for building healthy places. Washington, D.C.: Urban Land Institute.

El blog { del programador };. (2017). *NPM ¿Qué es y para qué sirve?*. [online] Available at: https://programadamente.wordpress.com/2015/11/28/npm-que-es-y-para-que-sirve/ [Accessed Jul. 2017].

En.wikipedia.org. (2017). AngularJS. [online] Available at: https://en.wikipedia.org/wiki/AngularJS [Accessed Jul. 2017].

En.wikipedia.org. (2017). *Apache Cordova*. [online] Available at: https://en.wikipedia.org/wiki/Apache_Cordova#Design_and_rationale [Accessed Jul. 2017].

En.wikipedia.org. (2017). Apache Cordova. [online] Available at: https://en.wikipedia.org/wiki/Apache_Cordova [Accessed Jul. 2017].

En.wikipedia.org. (2017). Cydia. [online] Available at: https://en.wikipedia.org/wiki/Cydia [Accessed Jul. 2017].

En.wikipedia.org. (2017). *Curl*. [online] Available at: https://en.wikipedia.org/wiki/Curl [Accessed Jul. 2017].

En.wikipedia.org. (2017). F-Droid. [online] Available at: https://en.wikipedia.org/wiki/F-Droid [Accessed Jul. 2017].

En.wikipedia.org. (2017). *Grid file system*. [online] Available at: https://en.wikipedia.org/wiki/Grid_file_system [Accessed Jul. 2017].

En.wikipedia.org. (2017). Ionic (mobile app framework). [online] Available at: https://en.wikipedia.org/wiki/Ionic_(mobile_app_framework) [Accessed Jul. 2017].

En.wikipedia.org. (2017). JavaScript. [online] Available at: https://en.wikipedia.org/wiki/JavaScript [Accessed Jul. 2017].

En.wikipedia.org. (2017). *Mean*. [online] Available at: https://en.wikipedia.org/wiki/Mean [Accessed Jul. 2017].

Eneko. (2017). *¿Qué es Express.js? Instalación y primeros pasos - Eneko*. [online] Available at: http://enekodelatorre.com/expressjs-instalacion-primeros-pasos/ [Accessed Jul. 2017].

Es.wikipedia.org. (2017). *Demonio (informática)*. [online] Available at: https://es.wikipedia.org/wiki/Demonio_%28inform%C3%A1tica%29 [Accessed Jul. 2017].

Es.wikipedia.org. (2017). *Servidor HTTP Apache*. [online] Available at: https://es.wikipedia.org/wiki/Servidor_HTTP_Apache [Accessed Jul. 2017].

Es.wikipedia.org. (2017). *Simple Object Access Protocol*. [online] Available at: https://es.wikipedia.org/wiki/Simple_Object_Access_Protocol [Accessed Jul. 2017].

Es.wikipedia.org. (2017). *Valores separados por comas*. [online] Available at: https://es.wikipedia.org/wiki/Valores_separados_por_comas [Accessed Jul. 2017].

esri España. (2017). ArcGIS - esri España. [online] Available at: http://www.esri.es/arcgis/ [Accessed Jul. 2017].

Falace (2017). *AngularJS application architecture*. [online] Slideshare.net. Available at: https://www.slideshare.net/GabrieleFalace1/angularjs-application-architecture [Accessed Jul. 2017].

Fernández-Ballesteros, R y Corraliza Rodríguez, J.A. (2004). Ambiente y vejez. En R., Fernández-Ballesteros. Gerontología social. (pp251-274). Madrid: Editorial Pirámide

Fernández-Ballesteros, R. (2011). Envejecimiento saludable. Comunicación presentada en el Congreso sobre envejecimiento: La investigación en España, Madrid, España

Fernández, J. M. y Kelh, S. (2001). La construcción social de la vejez. Cuadernos de trabajo social, 14, 125-161. http://revistas.ucm.es/index.php/CUTS/article/viewFile/CUTS0101110125A/7995

Fielding, R. T., & Taylor, R. N. (2000). Architectural styles and the design of network-based software architectures (p. 151). Doctoral dissertation: University of California, Irvine.

Flanagan, D. (2006). JavaScript: the definitive guide. " O'Reilly Media, Inc."

Flanagan, D. (2011). JavaScript: The definitive guide: Activate your web pages. " O'Reilly Media, Inc.".

Franco, M.; Diez-Roux A.V.; Glass T.; Lazo M., Caballero B. y Brancati F. (2008). Abailability of healthy foods and neighborhoods characteristics. American Jorunal of preventive medicine, Dec. 35(6):561-7 Doi. 10.3945/ajcn.2008.26434

GitHub. (2017). *allenhwkim/angularjs-google-maps*. [online] Available at: https://github.com/allenhwkim/angularjs-google-maps [Accessed Jul. 2017].

GitHub. (2017). *apache/cordova-plugin-camera*. [online] Available at: https://github.com/apache/cordova-plugin-camera [Accessed Jul. 2017].

GitHub. (2017). *asafdav/ng-csv*. [online] Available at: https://github.com/asafdav/ng-csv [Accessed Jul. 2017].

GitHub. (2017). *Automattic/mongoose*. [online] Available at: https://github.com/Automattic/mongoose [Accessed Jul. 2017].

GitHub. (2017). *creationix/nvm*. [online] Available at: https://github.com/creationix/nvm [Accessed Jul. 2017].

GitHub. (2017). *dpa99c/cordova-diagnostic-plugin*. [online] Available at: https://github.com/dpa99c/cordova-diagnostic-plugin [Accessed Jul. 2017].

GitHub. (2017). *emj365/cordova-plugin-audio-recorder-api*. [online] Available at: https://github.com/emj365/cordova-plugin-audio-recorder-api [Accessed Jul. 2017].

GitHub. (2017). *expressjs/body-parser*. [online] Available at: https://github.com/expressjs/body-parser [Accessed Jul. 2017].

GitHub. (2017). *pbakondy/cordova-plugin-sim*. [online] Available at: https://github.com/pbakondy/cordova-plugin-sim [Accessed Jul. 2017].

GitHub. (2017). *richardgirges/express-fileupload*. [online] Available at: https://github.com/richardgirges/express-fileupload [Accessed Jul. 2017].

GitHub. (2017). *substack/node-mkdirp*. [online] Available at: https://github.com/substack/node-mkdirp [Accessed Jul. 2017].

GitHub. (2017). *yeoman/generator-angular*. [online] Available at: https://github.com/yeoman/generator-angular [Accessed Jul. 2017].

Glass, T.A. y Bilal, U. (2016). Are neighborhoods causal? Complications arising from the 'stickiness' of ZNA. Social Science & Medic. doi: 10.1016/j.socscimed.2016.01.001.

Gulpjs.com. (2017). *gulp.js*. [online] Available at: https://gulpjs.com/ [Accessed Jul. 2017].

Gutierrez, A., Gutierrez, A. and Gutierrez, A. (2017). Aplicaciones web vs. aplicaciones nativas vs. aplicaciones híbridas. [online] Blogthinkbig.com. Available at: https://blogthinkbig.com/aplicaciones-web-nativas-hibridas [Accessed Jul. 2017].

Havighurst, R. J. (1963). Successful aging. Processes of aging: Social and psychological perspectives, 1, 299-320.

Heart Healthy Hood Project (2015). Photovoice Villaverde: Un estudio participativo sobre la alimentación a través de la fotografía (pp.31-40). Madrid: Heart Healthy Hood.

Hostingdiario.com. (2017). *Alternativas a Apache de alto rendimiento*. [online] Available at: https://hostingdiario.com/alternativas-a-apache-de-alto-rendimiento/ [Accessed Jul. 2017].

Intenational, E. The JSON Data Interchange Format, ECMA-404, Octubre de 2013.

Ionic Framework. (2017). Build Amazing Native Apps and Progressive Web Apps with Ionic Framework and Angular. [online] Available at: http://ionicframework.com/ [Accessed Jul. 2017].

Ionic Framework. (2017). Ionic Framework. [online] Available at: https://ionicframework.com/docs/intro/concepts/ [Accessed Jul. 2017].

Ionic Framework. (2017). *Getting Started with Ionic*. [online] Available at: http://ionicframework.com/getting-started/ [Accessed Jul. 2017].

Ionicframework.com. (2017). *Installing Ionic and its Dependencies - Ionic Framework*. [online] Available at: http://ionicframework.com/docs/v1/guide/installation.html [Accessed Jul. 2017].

Ionicframework.com. (2017). *Publishing your app - Ionic Framework*. [online] Available at: http://ionicframework.com/docs/v1/guide/publishing.html [Accessed Jul. 2017].

Ipfs.io. (2017). *AngularJS*. [online] Available at: https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/AngularJS.html [Accessed Jul. 2017].

Karma-runner.github.io. (2017). *Karma - Spectacular Test Runner for Javascript*. [online] Available at: https://karma-runner.github.io/1.0/index.html [Accessed Jul. 2017].

Medium. (2017). *Servicios y demonios en Linux – Jesús Torres – Medium*. [online] Available at: https://medium.com/jmtorres/servicios-y-demonios-en-linux-a424366336ac [Accessed Jul. 2017].

Momentjs.com. (2017). *Moment.js | Home*. [online] Available at: https://momentjs.com/ [Accessed Jul. 2017].

Nngroup.com. (2017). Mobile: Native Apps, Web Apps, and Hybrid Apps. [online] Available at: https://www.nngroup.com/articles/mobile-native-apps/ [Accessed Jul. 2017].

Openlayers.org. (2017). *OpenLayers - Welcome*. [online] Available at: https://openlayers.org/ [Accessed Jul. 2017].

Organización Mundial de la Salud (2002). Active Aging. A Policy Framework. Geneva: World Health Organization

Palomino, P. A.; Grande, M. L. & Linares, M. (2014). La salud y sus determinantes sociales. Desigualdades y exclusión en la sociedad del siglo XXI. Revista Internacional de Sociología, 72(Extra_1), 45-70 Doi. 10.3989/ris.2013.02.16

Phillipson, C. (2011). Growing older in urban environments: perspectives from Japan and the UK. London: The Internacional Longevity Centre. http://www.ilcuk.org.uk/index.php/publications/publication_details/growing_older_in_urban_environments_perspectives_from_japan_and_the_uk

Play.google.com. (2017). *Redirecting...*. [online] Available at: https://play.google.com/apps/publish [Accessed Jul. 2017].

Pm2.keymetrics.io. (2017). *PM2 ·*. [online] Available at: http://pm2.keymetrics.io/ [Accessed Jul. 2017].

Ramírez Vique, R. (2011). Métodos para el desarrollo de aplicaciones móviles. Universitat Oberta de Catalunya.: Barcelona, España.

Requena, A. T. (2006). El nuevo discurso de los mayores: la construcción de una nueva identidad social. RES. Revista Española de Sociología, (6), 65-90. http://www.fes-sociologia.com/files/res/6/04.pdf

Remy, J. & Voyé, L. (1976). La ciudad y la urbanización. (Trad. J. Hernández) Madrid: Instituto de estudios de la administración local.

Rodríguez, G.C.; Rodríguez, P.R.; Castejón, P.V. y Morán, E.A. (2013). Las personas mayores que vienen. Autonomía, solidaridad y participación social. Madrid: Fundación Pilares de la Tierra.

Ronzi, S.; Pope, D.; Orton, L. & Bruce, N. (2016). Using photovoice methods to explore older people's perceptions of respect and social inclusion in cities: Opportunities, challenges and solutions. SSM - Population Health, (2), 732-744. doi: 10.1016/j.ssmph.2016.09.004

Rose, G. (1985). Sick individuals and sick populations. J Epidemiol, 14(1):32-8. Doi. 10.1093/ije/30.3.427

Rowles, G.D. (1983). Geographical dimensions of social supportive in rural Appalachia. En G.D. Rowles & R. J. Ohta (Eds.) (1983). Ageing and Milieu. Environmental perspectives on growing old.(pp.111-128). New York : Academic Press Inc

Sancho, D. P., Andújar, D. H., y Rodríguez, P. R. (2015). Envejecer sin ser mayor: nuevos roles en la participación social en la edad de jubilación. Madrid: Fundación Pilares para la Autonomía Personal.

Sass-lang.com. (2017). *Sass: Syntactically Awesome Style Sheets*. [online] Available at: http://sass-lang.com/ [Accessed Jul. 2017].

The Official Ionic Blog. (2017). *Handling CORS issues in Ionic*. [online] Available at: http://blog.ionic.io/handling-cors-issues-in-ionic/ [Accessed Jul. 2017].

The Official Ionic Blog. (2017). Where does the Ionic Framework fit in?. [online] Available at: http://blog.ionic.io/where-does-the-ionic-framework-fit-in/?_ga=2.177832742.2047147977.1504799330-520415469.1500563514 [Accessed Jul. 2017].

Tombatossals.github.io. (2017). *Leaflet directive for AngularJS*. [online] Available at: http://tombatossals.github.io/angular-leaflet-directive/#!/ [Accessed Jul. 2017].

Udemy. (2017). *Online Courses - Learn Anything, On Your Schedule | Udemy*. [online] Available at: https://www.udemy.com/ [Accessed Jul. 2017].

W3schools.com. (2017). *Angular Routing*. [online] Available at: https://www.w3schools.com/angular/angular_routing.asp [Accessed Jul. 2017].

Weblogs.asp.net. (2017). *Dan Wahlin - Video Tutorial: AngularJS Fundamentals in 60-ish Minutes*. [online] Available at: https://weblogs.asp.net/dwahlin/video-tutorial-angularjs-fundamentals-in-60-ish-minutes [Accessed Jul. 2017].

Wiles, J. L., Leibing, A., Guberman, N., Reeve, J., & Allen, R. E. (2012). The meaning of “ageing in place” to older people. The gerontologist, 52(3),57-366. Doi. 10.1093/geront/gnr098

Yeoman.io. (2017). *The web's scaffolding tool for modern webapps | Yeoman*. [online] Available at: http://yeoman.io/ [Accessed Jul. 2017].