

Diseño y desarrollo de una aplicación móvil para la monitorización de personas de la 3ª edad

**José Manuel Castellano Domínguez**

Master Universitario en Aplicaciones Multimedia – Área Profesionalizadora

## **Consultor: Sergio Schvarstein Liuboschetz**

Fecha de entrega: --/01/2020

  
Esta obra está sujeta a una licencia de Reconocimiento [3.0 España de Creative Commons](http://creativecommons.org/licenses/by/3.0/es/)

© (José Manuel Castellano Domínguez / 2020)

Reservados todos los derechos. Está prohibido la reproducción total o parcial de esta obra por cualquier medio o procedimiento, comprendidos la impresión, la reprografía, el microfilme, el tratamiento informático o cualquier otro sistema, así como la distribución de ejemplares mediante alquiler y préstamo, sin la autorización escrita del autor o de los límites que autorice la Ley de Propiedad Intelectual.

**FICHA DEL TRABAJO FINAL**

|  |  |
| --- | --- |
| **Título del trabajo:** | Diseño y desarrollo de una aplicación móvil para la monitorización de personas de la 3ª edad. |
| **Nombre del autor:** | José Manuel Castellano Domínguez |
| **Nombre del consultor:** | Sergio Schvarstein Liuboschetz |
| **Fecha de entrega (mm/aaaa):** | 01/2020 |
| **Titulación:** | *Máster Universitario en Aplicaciones Multimedia* – Área Profesionalizadora |
| **Resumen del Trabajo** | |
| Este proyecto de profesionalización consiste en el diseño y desarrollo de una aplicación móvil, implementada como una *Web App,* que permita la monitorización de una serie de constantes vitales de los ancianos que lleven puesta una *camiseta inteligente* que será la encargada de suministrar la información que podrá ser visualizada desde la *App.*  La idea del proyecto viene a raíz de una idea de negocio con el que he hablado extensamente con un excompañero de trabajo durante el último año. El negocio que se quiere llevar cabo tiene 2 líneas de trabajo, aunque relacionadas entre sí.  - La primera de las líneas de negocio consiste en el diseño y montaje de productos electrónicos usando microcontroladores y sensores basados en Arduino.  - La segunda de las líneas de negocio consiste en el diseño e implementación de aplicaciones móviles que permitan interactuar con el producto electrónico diseñado en la primera línea de negocio. Las acciones pueden ir desde la simple consulta de datos obtenidos del producto a realizar acciones con el mismo.  El primer proyecto que se va a llevar a cabo es diseñar una *camiseta inteligente* capaz de obtener una serie de constantes vitales y permitir que los datos obtenidos puedan ser consultados posteriormente por una aplicación móvil. El diseño y desarrollo de este producto se desarrolla bajo las dos líneas de negocio descritas anteriormente  La primera de las líneas de negocio desarrollará la *camiseta inteligente.* El diseño de la *camiseta inteligente* queda *excluido íntegramente* del ámbito de este TFM. Queda fuera del ámbito del TFM, tanto el diseño y montaje del circuito electrónico, como la implementación del código necesario para hacer funcionar dicho producto. Actualmente se está trabajando en un prototipo de la *camiseta inteligente,* que se espera que esté disponible a finales de diciembre del 2019.  La segunda de las líneas, que es la que *se realizará* en el ámbito del TFM. Consiste en el diseño y desarrollo de una aplicación móvil, como una *Web App*, que pueda ser instalada en teléfonos bajo sistemas operativos Android e IOS. A lo largo del TFM se diseñará y desarrollará la aplicación móvil que debe ajustarse a las funcionalidades que permita la *camiseta inteligente*.  La aplicación móvil tendrá las siguientes funcionalidades.   * Gestión de la cuenta * Gestionar camisetas. * Visualización de las constantes vitales actuales e históricos (Frecuencia cardiaca, sudoración y temperatura). * Definición de umbrales de notificación. * Notificar al usuario si se ha superado un cierto umbral definido. * Compartir los datos de la camiseta   Además de desarrollar la interfaz y la funcionalidad de la aplicación móvil se implementará un *backend* que atenderá tanto las peticiones que provengan tanto de la *camiseta inteligente* como de la aplicación móvil. | |
| **Abstract (in English):** | |
| This professionalization project consists of the design and development of a mobile application, implemented as a Web App. This app will allow the monitoring of a series of vital signs of the elderly who wear a smart shirt that will be responsible for providing the information that can be seen in the App.  The idea of ​​the project comes from a business idea with which I have spoken extensively with a former co-worker during the last year. The business to be carried out has 2 lines of work but they are related to each other.  - The first of the business lines is the design and assembly of electronic products using microcontrollers and sensors based on Arduino.  - The second of the business lines is the design and implementation of mobile applications that allow interacting with the electronic product designed in the first line of business. The actions can go from the simple consultation of data obtained from the product to perform actions with the same.  The first project to be carried out is to design a smart shirt capable of obtaining a series of vital signs and allowing the data obtained to be consulted later by a mobile application. The design and development of this product is developed under the two business lines described above.  The first of the business lines will develop the smart shirt. The smart shirt design is completely excluded from the scope of this Master’s Thesis. It is outside the scope of the Master’s Thesis, both the design and assembly of the electronic circuit, as well as the implementation of the code necessary to operate said product. Currently working on a prototype of the smart shirt, which is expected to be available at the end of December 2019.  The second of the lines, which is the one that will be carried out within the scope of the Master’s Thesis. It consists of the design and development of a mobile application, such as a Web App, that can be installed on phones under Android and IOS operating systems. Throughout the Master’s Thesis, the mobile application will be designed and developed that must conform to the functionalities that the smart shirt allows.  The mobile application will have the following functionalities.  - Account management  - Manage t-shirts.  - Visualization of current and historical vital signs (heart rate, sweating and temperature).  - Definition of notification thresholds.  - Notify the user if a certain defined threshold has been exceeded.  - Share t-shirt data.  In addition to developing the interface and functionality of the mobile application, a backend will be implemented that will address both requests that come from both the smart shirt and the mobile application. | |
| **Palabras clave (7):** | |
| Android, IOS, Web App, 3ª edad, salud, constantes vitales, camiseta inteligente | |

**Índice**

[1. Introducción 1](#_Toc368261772)

[1.1 Contexto y justificación del Trabajo 1](#_Toc368261773)

[1.2 Objetivos del Trabajo 1](#_Toc368261774)

[1.3 Enfoque y método seguido 1](#_Toc368261775)

[1.4 Planificación del Trabajo 1](#_Toc368261776)

[1.5 Breve sumario de productos obtenidos 1](#_Toc368261777)

[1.6 Breve descripción de los otros capítulos de la memoria 1](#_Toc368261778)

[2. Resto de capítulos 2](#_Toc368261779)

[3. Conclusiones 3](#_Toc368261780)

[4. Glosario 4](#_Toc368261781)

[5. Bibliografía 5](#_Toc368261782)

[6. Anexos 6](#_Toc368261783)

**Lista de figuras**

[Figura 1 - Estadística del Padrón Continuo a 1 de enero de 2018 (Fuente: INE) 1](#_Toc24225175)

[Figura 2 - Afectación de la vida diaria. (Fuente: [3]) 2](#_Toc24225176)

[Figura 3 - Algunos modelos de Smartwatch (Fuente: Computerhoy) 3](#_Toc24225177)

[Figura 4 - Esquema de conexión de camiseta inteligente (Fuente: [8]) 4](#_Toc24225178)

[Figura 5 - Capturas aplicación Salud Conectada (Fuente: [10]) 5](#_Toc24225179)

[Figura 6 - Capturas aplicación Heart Rate OS (Fuente: [11]) 5](#_Toc24225180)

[Figura 7 - Capturas aplicación Health Wearable. Fuente ([12]) 6](#_Toc24225181)

[Figura 8 - Interfaz aplicación proyecto LOBIN. Fuente ([13]) 7](#_Toc24225182)

# 1. Introducción

## **1.1 Contexto y justificación del Trabajo**

Actualmente la sociedad española va envejeciendo lentamente. Según el informe más reciente del CSIC [1] hay más de 8 millones de personas en la tercera edad y se espera que este valor se incremente a medida que pasen los años. Tal y como se muestra en la figura 1, el grupo de edad que abarca la mayor población es el grupo de gente que comprende entre los 40 y los 50 años. Eso significa que en unos escasos 25 años el grupo de población de la 3ª edad sea el más numeroso.

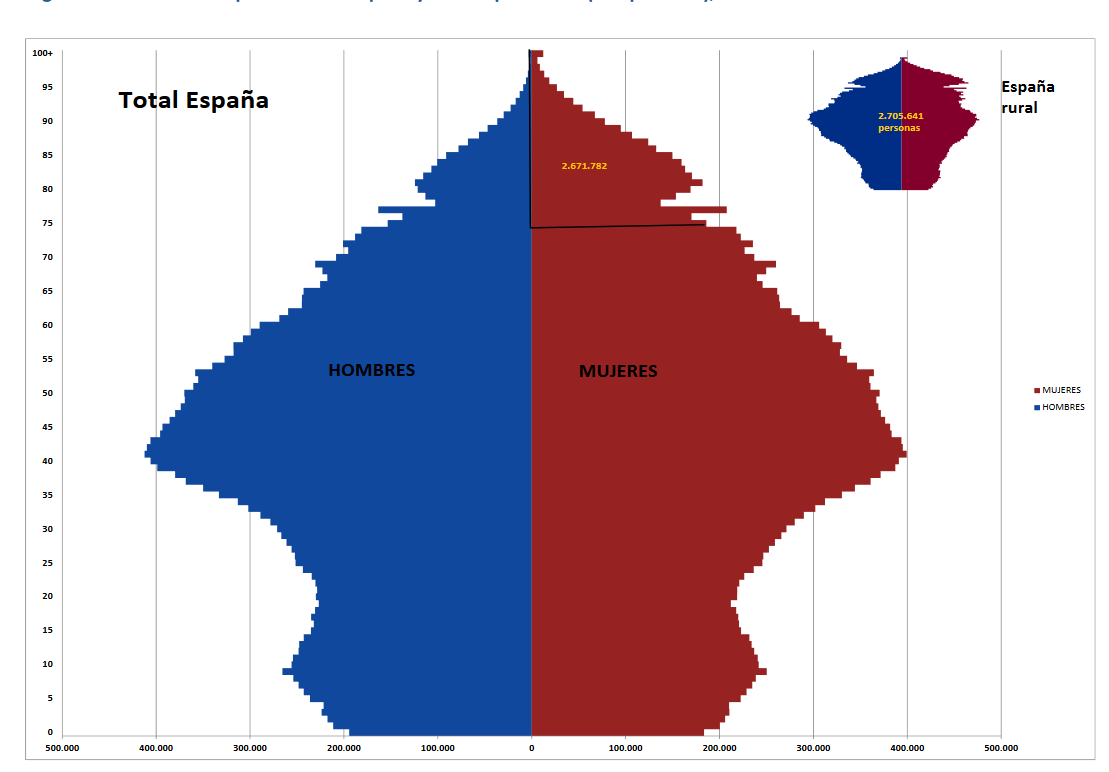


Figura 1 - Estadística del Padrón Continuo a 1 de enero de 2018 (Fuente: INE)

El cuidado de nuestros mayores implica un gran coste monetario que tal y como se indica en un artículo del periódico *El Mundo* en el año 2017 [2] se espera que se incremente a un ritmo de 580 millones de euros por año en los próximos 10 años debido al crecimiento de la población de la 3ª edad. Además del coste monetario también existe un gran coste en el tiempo invertido en el cuidado de la gente de la 3ª edad. Un estudio [3] realizado por el CSIC en colaboración con SEGG y LINDOR, indica que más del 88% de los cuidadores ven muy afectada su vida por el cuidado de gente mayor. En la figura 2 se puede observar una gráfica del citado informe, donde se desglosa en 9 niveles en cómo afecta a la vida de los cuidadores, el cuidado de la gente mayor (donde 1 es que apenas afecta a su vida y 9 que afecta completamente a su vida). Se observa que más del 88% de los cuidadores están en el nivel 5 o superior de cómo afecta esta actividad a sus vidas.

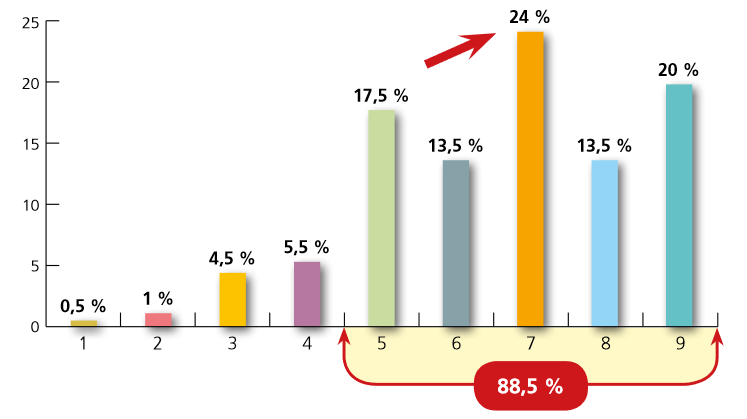


Figura 2 - Afectación de la vida diaria. (Fuente: [3])

La mayoría de la gente que tiene que cuidar de sus mayores, no posee ni dinero ni tiempo para encargarse de sus mayores. Algunas personas se limitan simplemente a enviar a sus padres/abuelos a una residencia, despreocupándose completamente del tema.

Además, aquellos que no pueden o no quieren enviar a sus padres/abuelos o aquellos ancianos que no tienen ninguna familia, se encuentran en una situación de abandono, sin que haya un seguimiento médico. En ocasiones vemos noticias que indican que se ha encontrado un anciano en su vivienda y nadie se ha preocupado de él [4], o incluso situaciones en que los hijos han abandonado por falta de tiempo a sus padres y se han acabado enterando tiempo después de que ha muerto [5].

El motivo con el que se pretende diseñar y desarrollar una aplicación móvil es doble. En primer lugar, es impulsar un nuevo tipo de negocio basado en la venta de productos electrónicos que puedan ser consultados o manipulados mediante el uso de aplicaciones móviles. Este es el primero de los productos que se quiere realizar donde la idea es vender el producto electrónico a un precio, mientras que la aplicación móvil será completamente gratuita para el consumidor, aunque para utilizarla será necesario haber adquirido el producto electrónico.

En segundo lugar, se pretende mejorar el bienestar tanto de la gente de la 3ª edad como de los cuidadores o familiares más directos. En algún momento, todos tenemos algún familiar que envejece y que necesita de cuidados especiales o ser monitorizado. De hecho, el momento de escribir el presente documento, mis padres están rozando casi los 70 años y quiero que pasen una buena vejez y poder saber en todo momento que se encuentran bien y no depender de tener que llevarlos a una residencia de ancianos o de contratar a un cuidador que tenga que estar todo el día con ellos.

## **1.2 Estado del arte**

Con la llegada del nuevo milenio, surgió el boom y auge de los dispositivos móviles e inteligentes. Este boom se inició con la aparición del primer *Iphone* lanzado por la compañía Apple y la presentación del sistema operativo Android por parte de Google, ambos durante el año 2007. Actualmente los sistemas operativos de *Android* (*Google*) e *IOS* (*Apple*) se ejecutan en incontables dispositivos como son teléfonos móviles, tabletas, pulseras, relojes, automóviles y televisores. Por poner algún ejemplo de la variedad de dispositivos tenemos, por el lado de Android, televisores de marcas como *Philips y Sony* o *smartwatches* donde se pueden encontrar cientos de modelos (*Huawei, Xaomi, etc)*. Por el lado de *IOS,* se tiene la propia televisión diseñada por *Apple* y como *smartwatch* tiene su propio modelo de reloj denominado *Apple Watch.*



Figura 3 - Algunos modelos de Smartwatch (Fuente: Computerhoy)

El boom no se limita únicamente simplemente a los dispositivos descritos anteriormente, sino a cualquier dispositivo que pueda estar conectado a una red. En la última década, el número de dispositivos interconectados entre sí, ha crecido exponencialmente. Este término se ha acuñado con el nombre de *IoT (*Internet of Things) y fue acuñado porDave Evans perteneciente a Cisco. En [6] se indica en que el número de dispositivos interconectados superaba ya al de personas y que esta diferencia se iría incrementando a medida que fueran pasando los años. En el mencionado artículo, se indica que cada vez habrá más dispositivos con sensores que permitirán enviar transmitir dicha información para que pueda ser analizada o procesada por otros dispositivos.

Aquellos dispositivos que puedan ser sujetos a ser llevados o vestidos por una persona se les conoce como *wearables.* Este término fue acuñado [7] durante la década de los 90s, indicando que un *wearable* es una pieza de vestimenta que tiene capacidad de computación, es decir, es una pieza de ropa que actúa igual que un ordenador y que posee diferentes entradas para capturar eventos del entorno. Los *wearables* más conocidos son los *smartwatches*, aunque existen otros como camisetas, zapatillas, etc.

La mayoría de los *smartwatches* incorporan una serie de sensores que recogen información como el número de pasos, la frecuencia cardiaca o el número de horas de sueño que luego envían a otro dispositivo mediante una tecnología inalámbrica. La gran mayoría de los *smartwatches* actuales funcionan vinculándose a un *Smartphone* u teléfono inteligente mediante *Bluetooth* (lo que crea una red de área personal, o denominada PAN, formada por el teléfono inteligente y el reloj)*,* aunque en los *smartwatches* más recientes se permite realizar este vínculo gracias a una red *Wifi*. Este comportamiento se extrapola a la gran mayoría de los *wearables.* En el caso del presente proyecto, el *wearable* que se utiliza es una camiseta inteligente en vez de un reloj, aunque ambos tienen el mismo funcionamiento.

Por poner un ejemplo de lo indicado en el párrafo anterior, en el 2014 se presentó un estudio [8] que se pretendía construir una camiseta inteligente y una aplicación móvil, para el seguimiento de pacientes con reumatismo, que permite vincularse vía *Bluetooth* con la camiseta para así obtener y mostrar los datos en la pantalla del teléfono inteligente. En la figura 2 se muestra el esquema de conexión del mismo entre la camiseta y el teléfono inteligente.



Figura 4 - Esquema de conexión de camiseta inteligente (Fuente: [8])

Como se observa en la figura 4, la camiseta envía, vía *Bluetooth,* los datos obtenidos de los sensores instalados, la aplicación móvil los muestra y actúa de pasarela para publicarlos en la nube, desde donde el médico en cuestión puede consultarlos. Esta arquitectura funciona bien cuando se tiene el teléfono y la camiseta cerca, pero en el momento que se aleje, los datos dejan de ser obtenidos. Adicionalmente existe otro problema; el teléfono inteligente del usuario es el encargado de almacenar y subir los datos al servidor lo que hace que el consumo de batería sea elevado, más teniendo en cuenta de que se tiene que tener el *Bluetooth* activo para poderse vincular con la camiseta.

Si en vez del hardware, se centra el estudio sobre las aplicaciones para dispositivos móviles, basta con hacer una simple búsqueda en Google Play [9] para observar que existe gran variedad de aplicaciones que pueden ser utilizadas en los *wearable* o dependen de los datos del *wearable* para funcionar. En concreto, si se centra la investigación sobre aplicaciones relacionadas con la salud, se pueden encontrar cientos de las mismas. La gran mayoría de las aplicaciones que se encuentran en el market que implican el uso de un *wearable* están relacionadas con apps la detección de constantes relacionadas con la actividad deportiva, como son el número de pasos, la frecuencia cardiaca o el número de calorías quemadas. Una investigación de cada una de estas aplicaciones daría para escribir varios libros, por lo que simplemente se limitará a explicar las aplicaciones más relevantes encontradas respecto al ámbito que abarca el presente TFM, que consiste el uso de un *wearable* para obtener constantes vitales de un paciente.

### Salud Conectada

Esta aplicación [10], desarrollada por la compañía de seguros médicos Sanitas, permite conectarse a cualquier *wearable* via *Bluetooth* para obtener las constantes vitales del paciente y las muestra en el teléfono móvil. Adicionalmente permite apuntar las citas, recordatorios y un listado de medicamentos a tomar. El problema que tiene principalmente es la gran dependencia que tiene del *Bluetooth* para funcionar, de hecho, la principal queja de los usuarios es que se desvincula fácilmente el *wearable* del dispositivo móvil. Además, para obtener valores ambos dispositivos deben de estar cerca. Otro problema se ha detectado mientras se hizo una prueba de la aplicación para el presente estudio, debes de tener una cuenta en Sanitas para poderla utilizar y no se sugiere la creación de una desde la aplicación.



Figura 5 - Capturas aplicación Salud Conectada (Fuente: [10])

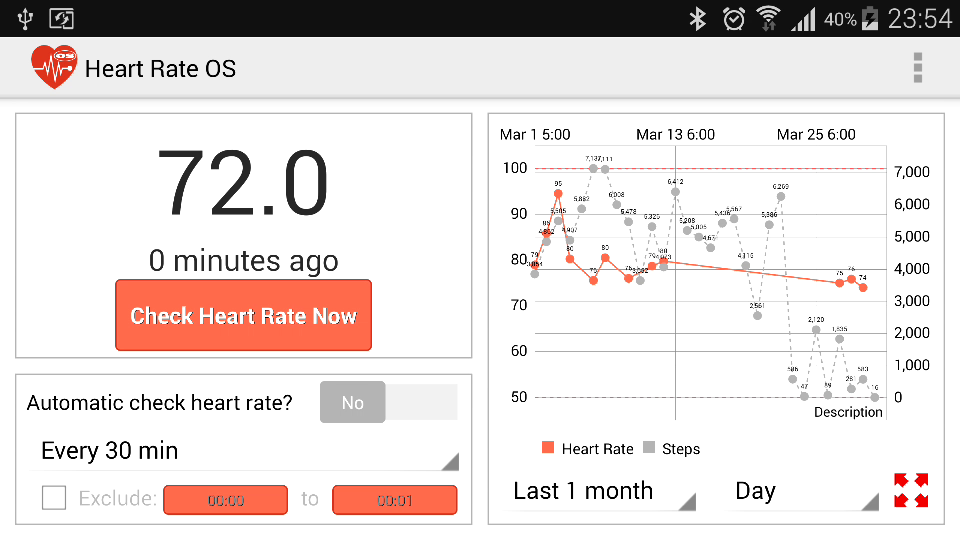


Figura 6 - Capturas aplicación Heart Rate OS (Fuente: [11])

### Heart Rate OS – Android Watch

Esta aplicación [11], desarrollada por la LFApp y cuya aplicación puede visualizarse en la figura 6, permite medir la frecuencia cardiaca de la persona que lleva el reloj inteligente que esté vinculado al teléfono inteligente. La aplicación permite dos modos de funcionamiento. La primera de ellas es solicitando una medición manualmente, mientras que la segunda es programar la aplicación para que haga mediciones cada media hora. Cada medición dura 1 minuto. Luego, los datos obtenidos se sincronizan con los de *Google Fit*.

El principal inconveniente que tiene esta aplicación es que la monitorización de la frecuencia cardiaca no es permanente y se puede programar únicamente cada 30 minutos. Además, las opciones de sincronización solo permiten sincronizar con *Google Fit.*

### Health Wearable

Esta aplicación [12], desarrollada por Analog Devices Inc es una aproximación parecida a lo que se pretende abordar en el presente proyecto. Esta aplicación permite conectarse a un *smartwatch* específicamente creado [13] para suministrar una serie de constantes vitales a la aplicación móvil. Esta aplicación permite especificar entre que constantes se quieren realizar mediciones (permite frecuencia cardiaca, conductividad de la piel, temperatura, etc.).

La aplicación permite visualizar en tiempo real y mediante una serie de gráficas, las mediciones obtenidas a lo largo del tiempo y visualizar un histórico del mismo. La única diferencia respecto a las anteriores es que no existe opción para compartir y publicar los datos en la nube.

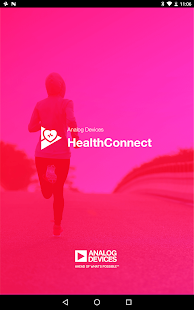
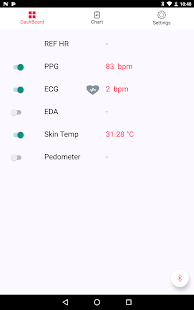
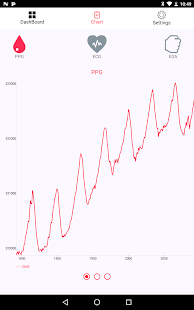
  

Figura 7 - Capturas aplicación Health Wearable. Fuente ([12])

## 

### Resumen aplicaciones vistas:

Las aplicaciones que se han visto presentan 3 características principales:

* Son apps nativas.
* Requieren de *Bluetooth* para vincularse con el *wearable*.
* En caso de detectar alguna anomalía en los datos, no se notifica al usuario.

Esto conlleva a una serie de características que tienen estas aplicaciones con el *wearable* asociado.

* Requieren estar cerca el dispositivo móvil del *wearable*.
* Se debe hacer un desarrollo por cada plataforma que se desee exportar la aplicación. Esto quiere decir que en caso de querer exportarlo a Android e IOS hay que realizar 2 desarrollos distintos.
* Si la app tiene opción de subir datos a la nube, utiliza la conexión a la red del dispositivo móvil, lo que utiliza los datos móviles del usuario e utiliza la batería del dispositivo.
* No existen mecanismos para avisar en caso de detectar un problema grave en las constantes vitales.
* No existen mecanismos que notifique al usuario que el wearable está a punto de quedarse sin batería.

Como se verá dentro del apartado de objetivos y el alcance, lo que se pretende con la nueva aplicación móvil es eliminar estas limitaciones que tienen las actuales aplicaciones.

Por último y para acabar el apartado, se abandona el terreno de las aplicaciones móviles para centrarnos en aplicaciones sanitarias relacionadas con el cuidado de pacientes mediante el uso de *wearables.*

En el año 2010, la universidad Rey Juan Carlos impulsó un proyecto para controlar las constantes vitales y tener localizados en todo momento a todos los pacientes del hospital. La idea principal en la que se basa el proyecto LOBIN [14], es tener una camiseta con un conjunto de sensores y que mediante Wifi comunique esta información al sistema central del Hospital La Paz, situado en Madrid. Principalmente esta camiseta está dirigida a aquellos pacientes con problemas de corazón, aunque la misma es capaz de medir otros factores como la temperatura del paciente o detección y alarma en caso de detectar que el paciente abandona la zona de la que le fue asignada.

El proyecto LOBIN se centra principalmente en la arquitectura del sistema (desde la implementación de la camiseta a la comunicación con el sistema central) y no en la aplicación en sí, pero si hay indicaciones de que como se tenía que desarrollar la aplicación y de lo que tenía que contener.



Figura 8 - Interfaz aplicación proyecto LOBIN. Fuente ([13])

La aplicación consiste en una aplicación Java con varias ventanas, las cuales cada una tiene una función muy específica. En la ventana principal se ve el árbol de pacientes y un mapa del hospital donde se localizan los pacientes. Cuando se pulsa sobre el paciente, se abre una nueva ventana con las constantes del paciente.

## **1.3 Objetivos del Trabajo**

Como se ha visto en el apartado anterior, la mayoría de las aplicaciones para dispositivos móviles que se han desarrollado se han diseñado de forma nativa e utilizan *Bluetooth* para vincularse con el proyecto.

La idea del presente TFM es enfocar el desarrollo de la aplicación bajo otro punto de vista. La idea principal es desarrollar una aplicación para dispositivos móviles que pueda ser ejecutado bajo distintas plataformas, es decir, el desarrollo no va a ser nativo, ya que de lo contrario requeriría de un desarrollo por cada una de las plataformas. Por ello en primer lugar, la nueva aplicación móvil será una Web App que pueda ser fácilmente producida para distintas plataformas. Con ello se consigue que con un solo desarrollo conseguir tenerla para distintas plataformas. En principio, se desarrollará y se testeará para la plataforma *Android*, aunque con el mismo código implementado se podrá producir una versión para la plataforma *IOS*.

La Web App será implementada bajo el framework *IONIC*. La versión de *IONIC* sobre la que se va a trabajar es la 4.0 que trabaja bajo Angular 5.0. Como entorno de ejecución se utilizará *NODE* y gestión de paquetes *NPM.* Además, en la medida de lo posible, se van a utilizar componentes que permitan gestionar de manera eficiente y que permita la reutilización de código, como ejemplo de ello se puede mencionar el uso de los preprocesadores *SCSS* o el uso de *minify* para las entregas finales. Además, para el control de versiones y el reporte de los bugs detectados se utilizará como repositorio de versiones *Git.*

El requisito fundamental de esta app será obtener los datos que se van obteniendo de la camiseta, pero la principal diferencia es que las mismas serán obtenidas a través de un servidor en vez de ser obtenidas a través del *Bluetooth.*

Las principales funcionalidades que debe cumplir la aplicación son las siguientes:

* Debe permitir al usuario autentificarse en el sistema. Una vez autenticado en el sistema, los datos del resto de apartados deben mostrarse únicamente información de dicho usuario.
* Debe permitir al usuario registrar una cuenta.
* Debe listar el conjunto de camisetas que tiene el usuario registradas en la aplicación.
* Debe permitir añadir, modificar y eliminar camisetas.
* Debe mostrar el último valor obtenido por la camiseta para las siguientes constantes vitales.
  + Frecuencia cardiaca.
  + Sudoración.
  + Temperatura.
* Debe mostrar los datos de las constantes vitales en un periodo de tiempo mediante gráficas de las siguientes constantes vitales.
  + Frecuencia cardiaca.
  + Sudoración.
  + Temperatura.
* Debe permitir definir un umbral de notificación para una o más de las constantes vitales o un tiempo en el que no se reciban datos.
* Debe notificar al usuario si alguno de los umbrales definidos se supera.
* Debe notificar al usuario si la persona que lleva la camiseta ha sufrido una caída.
* Debe notificar al usuario si la batería de la camiseta es baja.
* Debe permitir al usuario compartir los datos en tiempo real o los datos del conjunto histórico de datos.

Todas estas funcionalidades deberán ser testeadas al finalizar sus respectivos desarrollos, así como las entregas que se hagan al cliente final. En este proyecto es imposible reunir a un conjunto de testers bajo un mismo sitio para la realización de las pruebas, por lo que se va aplicar otro enfoque. Para conocer en todo momento los errores y excepciones que se puedan producir mientras los usuarios prueban la aplicación, se utilizara el servicio *Crashlytics* de *Firebase* [15]. Este servicio permite al desarrollador consultar los errores y excepciones producidos por la aplicación móvil y da información del dispositivo. Cuando un desarrollo este completo, simplemente se subirá la versión a *Google Play* como versión *Alpha* para los testers.

Por último y para la realización de las pruebas finales y la entrega del proyecto al cliente; dado que no se va a disponer del prototipo de la camiseta hasta principios del 2020, se generará un algoritmo en el servidor que genere la información como si se hubiera obtenido de la camiseta y se transfiera a la aplicación móvil. Estas camisetas serán registradas como camisetas virtuales y el origen de los datos será el propio servicio *backend* al que se conectaran los dispositivos móviles.

## **1.4 Enfoque y método seguido**

Para desarrollar la aplicación, se va a utilizar una aproximación PMBOK, enfocándonos en una serie de áreas que tal y como se indica en [16] y manteniendo un equilibrio entre todas ellas. Sobre todo, se va enfocar en los aspectos de *calidad, tiempo y riesgos*, que son las áreas más importantes en el proyecto, pero sin dejar de lado el resto de los aspectos. Para conseguir dicho equilibrio se seguirá una filosofía basada en *SCRUM* [17] en la cual los *sprint* tendrán 2 semanas de duración. En cada uno de los *sprint* se abarcará el desarrollo una o 2 pantallas, según el coste de implementar dichas pantallas. Si durante un *sprint* se desarrolla más de 2 pantallas, aquellas tareas que requieran interactuar con los *early adopters,* se realizarán en el mismo periodo de tiempo. Estas tareas se indican con un asterisco. El desarrollo de cada una de las pantallas consistirá en los siguientes pasos:

* Entrevista con los *early adopters* para determinar cuáles son las necesidades para dicha pantalla. (\*)
* Realización de un primer boceto en *Balsamiq*.
* Los *early adopters* testean y validan el boceto en *Balsamiq*. (\*)
* Se implementa la pantalla junto con la funcionalidad definida.
* Se realiza un proceso de pruebas, corrigiendo los errores detectados. (\*)
* Documentación

Existirán un total de 6 *sprint* distintos y habrá una entrega de código y documentación cada 2 *sprint.* La única excepción será el 6º *sprint* que no durará exactamente 2 semanas, sino una semana y media para adecuarlo al calendario de la entrega final del proyecto. Además de los 6 *sprint,* habrá un *sprint 0* que se iniciará con anterioridad a los 6 *sprint* y que tan sólo durará una semana. En este *sprint* inicial se hará una colección inicial de todas las tareas y requisitos que debe cumplir el proyecto en general, que permita generar un *product backlog* inicial.

## **1.5 Planificación del Trabajo**

Al inicio del proyecto, no se conocen todas las características que va a tener la aplicación móvil ya que se desconoce todas las constantes vitales que van a medir la camiseta, por lo que se dejará margen de maniobra para poder definir requisitos adicionales, eliminar los que ya no sean necesarios o correcciones del proyecto a medida que se vayan avanzado. Durante los primeros *sprint* se irán especificando implementando las funcionalidades más críticas del programa (como por ejemplo la ventana en la que se visualizan las constantes vitales) y en los siguientes *sprint* se implementarán los requisitos menos importantes de la aplicación (notificaciones, dar de alta la camiseta, etc.). Se ha dejado hueco en los últimos *sprint* para cubrir posibles desviaciones en el proceso, cubrir nuevos requisitos a cubrir o corregir el *feedback* que provenga del cliente.

Cada 2 *sprint* habrá un hito que coincidirá con una entrega al cliente para que pueda evaluar una aplicación funcional y pueda ir reconduciendo el contenido de las siguientes entregas según su criterio. El 2º sprint después de cada hito (el *sprint* previo al siguiente hito), se definirá una tarea que consistirá en aplicar aquellas correcciones o modificaciones que vengan impulsadas por el cliente. El 3º hito (tras *6 sprint)* se realizará la entrega final del proyecto y por tanto se procederá a pasar al mantenimiento del mismo.

En la figura 9, se expone una pequeña parte del diagrama de Gantt inicial con los *sprint* planificados. El mismo se puede encontrar completo para todo el ámbito del proyecto en el anexo A. Los *sprint* están representados en rojo y las tareas principales del proyecto en morado. Las subtareas que conforman las tareas principales se encuentran de color azul. Los hitos del proyecto corresponden con las entregas del proyecto (PEC 2, 3, 4 y 5) se encuentran de color amarillo. Por último, la defensa del TFM se encuentra de color verde, incluyéndose dentro de la planificación del proyecto.

Para realizar el trabajo se necesitará, tanto de una serie de herramientas como de una buena planificación para llevar a cabo el mismo.

Las herramientas que se utilizarán durante el presente proyecto son las siguientes:

* Un PC con un SO Windows 10 o Linux.
* Un Mac Mini con el XCode instalado.
* El SDK de Android.
* El SDK de IOS.
* Varios dispositivos móviles Android (1 teléfono y 1 tablet) para probar la aplicación en distintas pantallas.
* IONIC como framework para desarrollar la Web App.
* NPM como gestor de paquetes.
* Visual Studio Code como IDE de desarrollo.
* Hosting para poder subir el Backend de la aplicación móvil.
* Github como servidor remoto de repositorios y Git como herramienta de control de versiones.
* MySQL como servidor de Base de Datos.

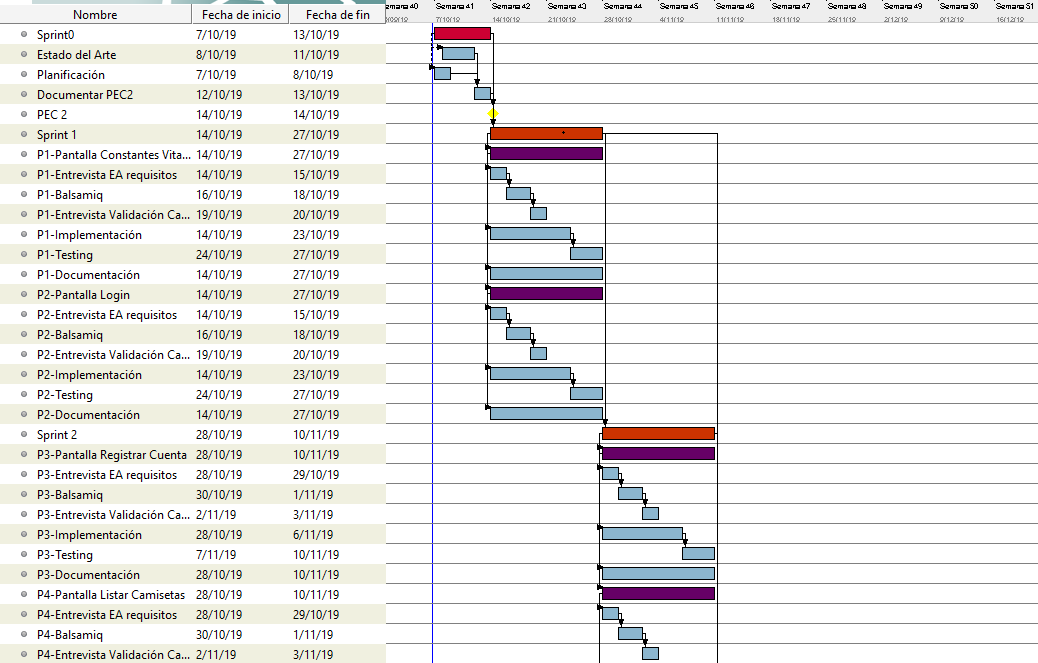


Figura 9 - Previsualización planificación del proyecto

## **1.5 Productos Finales**

Los productos obtenidos al final de este trabajo serán:

* Aplicación Móvil en Android e IOS.
* Manual de Instalación
* Manual de Puesta en Producción.
* Manual de Usuario.
* Memoria del Trabajo Final de Master.

## **1.6 Breve descripción de los otros capítulos de la memoria**

En los siguientes capítulos de la memoria tendremos:

* Early Adopters
* Desarrollo
* Conclusiones

En el apartado de *Early Adopters* se introducirá brevemente a las personas que se han entrevistado, explicando sus inquietudes y preocupaciones. Estas personas irán transmitiendo bajo su punto de visto los aspectos que debería solucionar la aplicación e irán validando las diversas pantallas de las que se va a componer nuestra aplicación.

En el apartado de *Desarrollo* se explicará desde el proceso de indicar el porqué del uso de las tecnologías que se van a usar a lo largo del proyecto, así como la estructura que va a tener el proyecto en sí. Incorporando las pantallas que va a tener la aplicación en sí, la navegación entre dichas ventanas y como se ha enfocado la solución para resolver los problemas que los *Early Adopters* nos van comunicando, así como las pruebas que se van realizando.

En el apartado de

# 2. Diseño

2.1 Usuarios y Contexto de Uso:

Habrá dos tipos de usuario diferenciados en la aplicación que ejecutarán los roles de consumidor y proveedor:

* Cliente
* Profesional

El Cliente será el usuario que demandará servicios en la aplicación, por lo tanto el consumidor, mientras el Profesional será el usuario que proporcionará esos servicios en demanda, es decir, el proveedor.

La aplicación permitirá ser utilizada por ambos tipos de usuario, donde cada uno tendrá disponibles una serie de funcionalidades y podrá interactuar en diferentes casos de uso.

Todo usuario en esta aplicación será Cliente, y se le permitirá demandar servicios profesionales. Por otra parte, para pertenecer a la clase de usuario Profesional, se ha de suscribir a alertas de los oficios que se sabe desempeñar.

En una primera fase, el registro del usuario será gratuito e indistinto para ambos tipos de usuario, pero en fases posteriores se planteará la posibilidad de requerir un pago periódico (mensual o anual) a los usuarios que quieran utilizar la aplicación como usuario profesional, de manera que sirva de método para la monetización del producto.

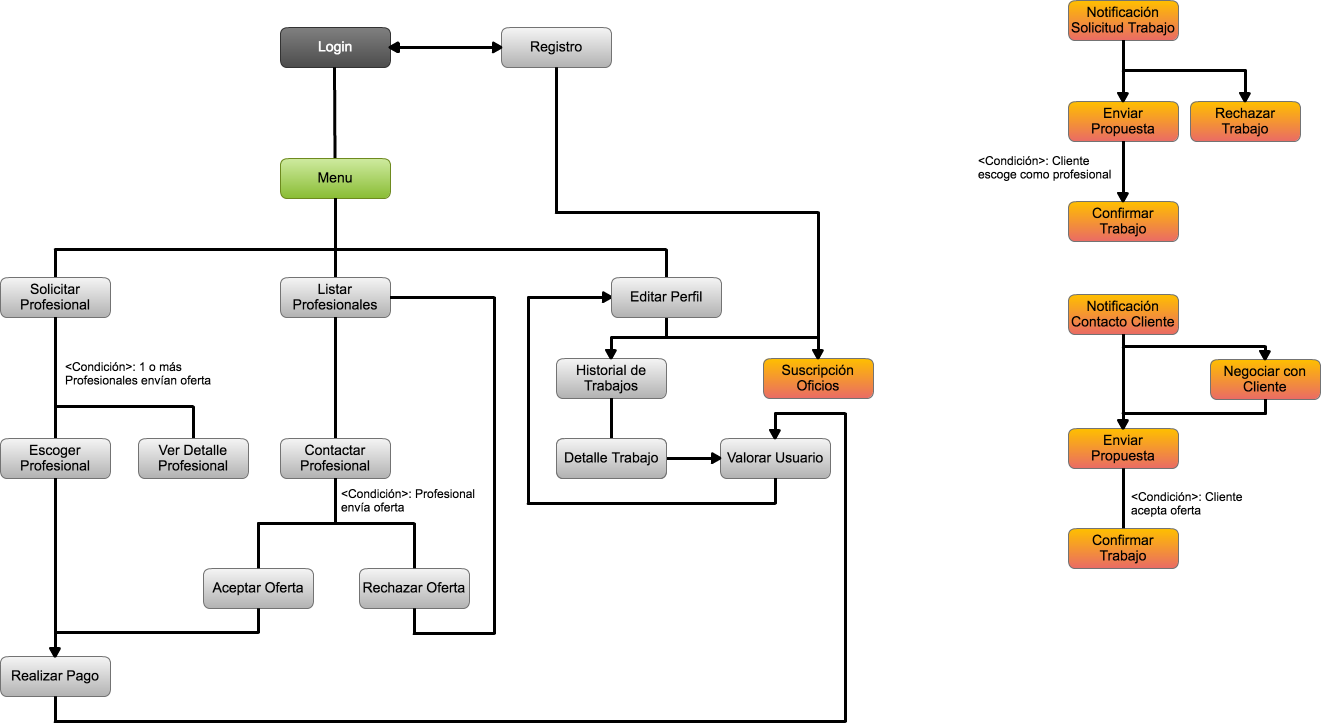
Por otra parte, el usuario también debe disponer de la posibilidad de ver un listado de profesionales cercanos para poder contactar y plantear dudas sin ningún tipo de compromiso, y desde ahí tener la posibilidad de llegar a acuerdos.

2.2 Diseño Conceptual

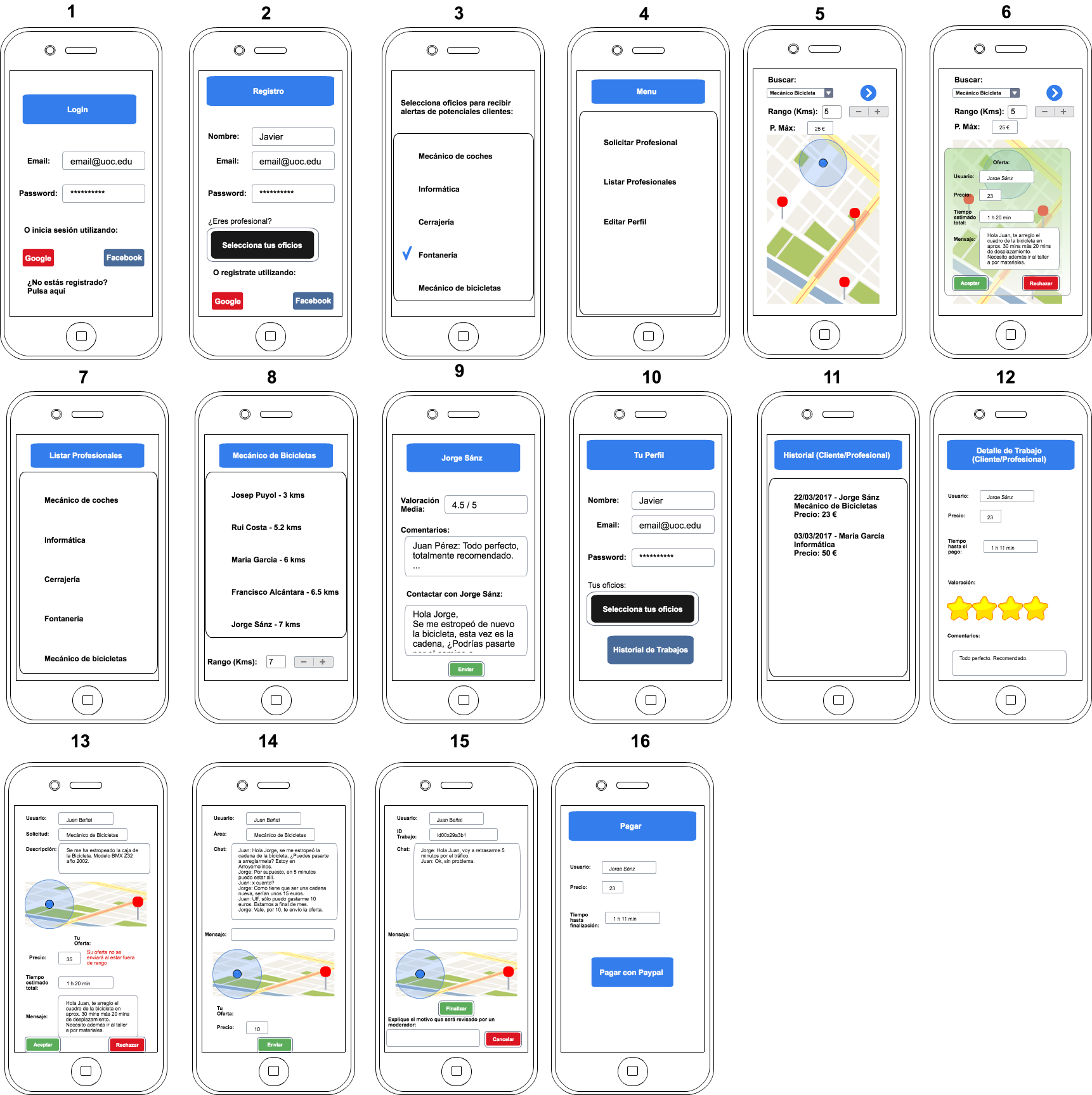
Para el diseño conceptual se ha realizado un árbol de navegación donde se transita entre diferentes estados, que están representados en diferentes colores según su tipo:

* Negro: Estado inicial. La aplicación requiere iniciar sesión para acceder al resto de funcionalidad.
* Verde: Menú. Estado inicial mostrado tras iniciar sesión en la aplicación. Desde aquí se puede acceder al resto de la aplicación.
* Naranja: Estados a los que sólo se puede acceder si se va a utilizar un rol de profesional en la aplicación.
* Gris: Resto de estados.

Además, algunos estados necesitan que se produzca una precondición para poder llegar a los mismos. Por ejemplo, “Escoger profesional” y “Ver detalle Profesional” sólo se mostrarán si tenemos al menos 1 oferta.



2.3 Prototipo de la aplicación



Para el prototipo de la aplicación se ha realizado un conjunto de Wireframes que representan el diseño de las diferentes pantallas de la aplicación para un dispositivo iOS (iPhone en concreto). Se han numerado del 1 al 16, y a continuación se describe cada pantalla:

1. Login: Pantalla que permite logar al usuario registrado mediante email y contraseña, cuenta de Google o cuenta de Facebook. En caso de no estar registrados, nos permite acceder a la pantalla de registro (2).
2. Registro: Pantalla que permite registrar al usuario rellenando datos personales junto al email y contraseña, o usando cuenta de Google o Facebook. Además nos permite seleccionar los oficios en los que somos expertos en caso de querer utilizar la aplicación a nivel profesional.
3. Selección Oficios: Nos permite seleccionar los oficios en los que somos especialistas para recibir notificaciones de clientes pidiendo nuestros servicios.
4. Menú de la aplicación: Desde aquí podemos acceder a nuestro perfil para editarlo o consultar el histórico, acceder a la pantalla de petición de profesionales en demanda (5) y a la pantalla con el listado de profesionales (7).
5. Petición de Profesionales: Pantalla desde la que podemos solicitar profesionales en un rango de kilómetros y precio. Muestra un mapa con profesionales que enviaron su oferta.
6. Oferta del Profesional: Desde la pantalla anterior (5. Petición de Profesionales) podemos observar las ofertas recibidas, donde se nos mostrará el precio, tiempo aproximado que tardará en realizarse el trabajo y otros detalles que nos deje el profesional en su mensaje. Se nos permitirá aceptarla o rechazarla.
7. Listado de profesionales (categorías): En esta pantalla se nos muestran las distintas categorías profesionales.
8. Listado de profesionales: En esta pantalla, tras haber seleccionado la categoría, se muestran los profesionales disponibles en un determinado rango de distancia (Kms).
9. Detalle del Profesional: Se le muestra al usuario el perfil de un profesional, con comentarios y nota media del mismo. En esta pantalla también es posible contactar con el profesional vía chat.
10. Perfil de usuario: En esta pantalla se permite al usuario consultar su perfil junto a su historial (tanto de cliente como de trabajador). También se permite acceder a la pantalla de Selección de Oficios (3).
11. Historial: Se nos permite ver el historial de en la aplicación con todos los trabajos realizados y nos permite acceder al detalle de los mismos.
12. Detalle del Trabajo: Se puede ver detalles del trabajo, y si no se hizo antes, puntuar al otro usuario y dejar un comentario en su perfil.
13. Petición de Trabajo: Esta es la pantalla que se le muestra al trabajador cuando recibe una notificación de un cliente potencial pidiendo un servicio. El trabajador podrá enviar su oferta, donde indicará el precio y tiempo estimado. Si el precio está encima del rango del usuario, se le avisará, ya que en ese caso esa oferta no podrá ser enviada (por lo cual debería rebajar el precio o rechazar el trabajo).
14. Contacto con cliente: El profesional recibe una consulta y puede hablar a través del chat con el cliente. Desde esta pantalla, el profesional puede hacer una oferta al cliente.
15. Trabajo en proceso: Pantalla que se mostrará después de haberse realizado el trato entre el cliente y el trabajador. Se les mostrará un chat y un número de teléfono para que puedan contactar. Cuando ambos pulsen en finalizar se procederá a la pantalla de pago. Por último, habrá un botón de cancelar que permitirá cancelar en caso de que ocurran imprevistos de última hora, los cuales podrán ser revisados por moderadores para evitar fraudes a la hora de pagar.
16. Pantalla de Pago: A través de esta pantalla se redirigirá al cliente a un servicio de pago (Paypal) para realizar el pago del servicio.

2.4 Evaluación del prototipo

Para la realización de la evaluación del prototipo se ha escogido a varios posibles usuarios en mi entorno familiar y de amistades entre los cuales hay usuarios potenciales de la aplicación para los distintos roles de la aplicación, cliente y profesional.

Los usuarios han visto el prototipo, en término general, de una manera positiva, donde se han comparado ciertos aspectos del diseño con aplicaciones actuales que tienen gran aceptación entre el público de las aplicaciones móviles.

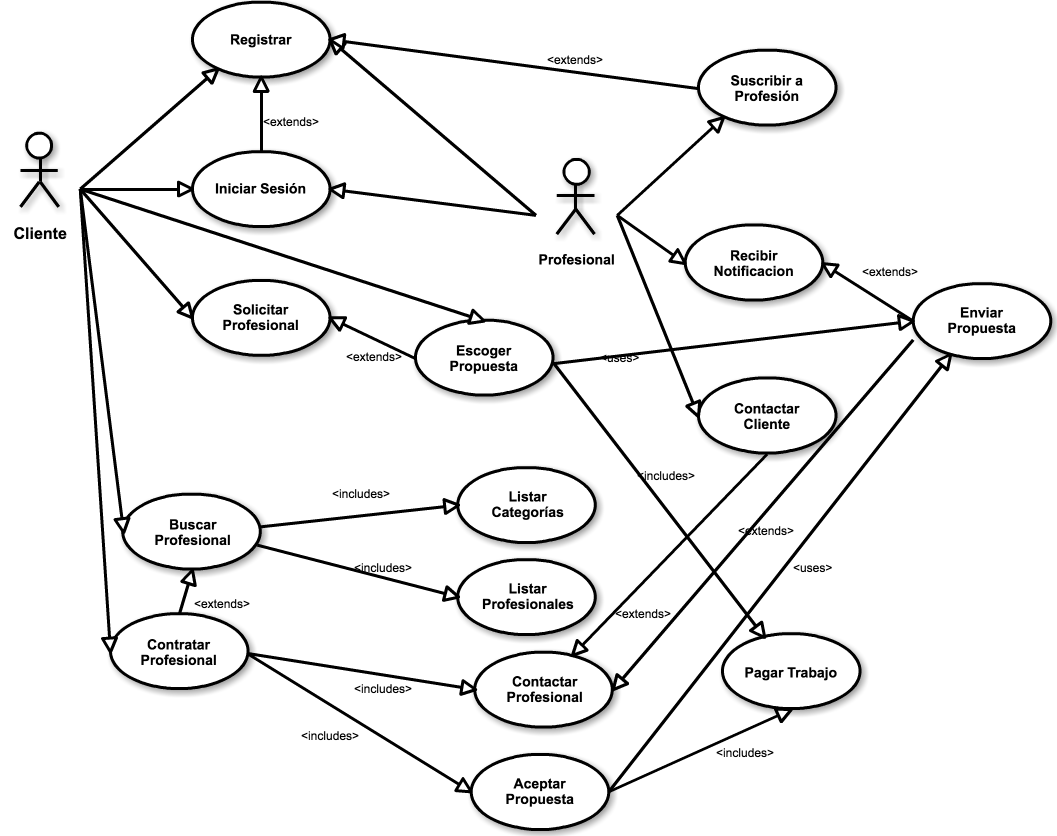
Esto ha ayudado a corregir en primer lugar, la falta de botones y otros elementos de la Interfaz necesarios en ciertas pantallas del prototipo. Citando algunos ejemplos:

* Botón para acceder a la pantalla de Registro desde la pantalla del Login, lo cual hacía imposible su acceso.
* Elementos de la Interfaz que permiten al usuario establecer un precio máximo en las pantallas 5 y 6 de la interfaz (Solicitar Profesional).

Por otra parte, se ha añadido una pantalla al prototipo (15. Trabajo aceptado) en la cual el cliente y el trabajador tendrán un contacto directo vía Chat y se les proporcionan a ambos usuarios sus números de teléfono.

2.5 Diseño Técnico

a. Definición de los casos de uso



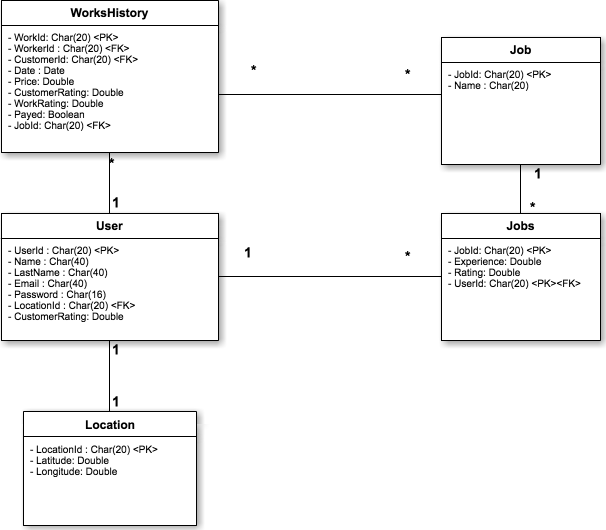
Los actores en la aplicación serán los usuarios Cliente y Profesional.

Los casos de uso del Cliente serán los siguientes:

* CU001 – Registrar
  + Actores: Cliente, Profesional.
  + Precondiciones: Ninguna.
  + Postcondiciones: Ninguna.
  + Descripción: Permite al Usuario registrarse en la aplicación mediante el uso de email, contraseña y datos personales, o una cuenta de Google o Facebook.
* CU002 - Iniciar Sesión
  + Actores: Cliente, Profesional
  + Precondiciones: CU001. Ya que es necesario estar registrado previamente.
  + Postcondiciones: Ninguna.
  + Descripción: Permite iniciar sesión al Usuario en la aplicación para dar acceso al resto de funcionalidades.
* CU003 - Solicitar Profesional
  + Actores: Cliente.
  + Precondiciones: CU001 y CU002. El usuario debe estar registrado y logado previamente.
  + Postcondiciones: Ninguna.
  + Descripción: Permite al usuario solicitar un profesional para realizar un servicio en demanda.
* CU004 - Escoger Propuesta
  + Actores: Cliente.
  + Precondiciones: CU001, CU002, CU003. Además de estar registrado y logado, requiere que se haya solicitado un profesional anteriormente, y que al menos un profesional haya enviado una propuesta.
  + Postcondiciones: CU0XX. Requiere que al finalizar, se pague el trabajo.
  + Descripción: Permite escoger una propuesta dentro de las recibidas por parte de distintos profesionales.
* CU005 – Buscar Profesional
  + Actores: Cliente
  + Precondiciones: CU001, CU002. Requiere estar logado y registrado.
  + Postcondiciones: CU006, CU007.
  + Descripción: Permite al cliente buscar un profesional por categoría.
* CU006 – Listar Categorías
  + Actores: Cliente
  + Precondiciones: CU001, CU002. Requiere estar logado y registrado.
  + Postcondiciones: Ninguna.
  + Descripción: Permite al cliente acceder al listado de categorías profesionales de la aplicación
* CU007 – Listar Profesionales
  + Actores: Cliente
  + Precondiciones: CU001, CU002, CU006. Requiere estar logado y registrado. Además, requiere haber listado anteriormente las categorías
  + Postcondiciones: Ninguna.
  + Descripción: Permite al cliente listar los profesionales en una categoría determinada.
* CU008 – Contratar Profesional
  + Actores: Cliente
  + Precondiciones: CU001, CU002, CU005. Requiere estar logado y registrado. Además, requiere haber buscado anteriormente al profesional.
  + Postcondiciones: CU009, CU010.
  + Descripción: Permite al cliente contratar un profesional.
* CU009 – Contactar profesional
  + Actores: Cliente
  + Precondiciones: CU001, CU002, CU005. Requiere estar logado y registrado. Además, requiere haber buscado anteriormente al profesional.
  + Postcondiciones: Ninguna.
  + Descripción: Permite al cliente contactar vía chat con el profesional.
* CU010 – Aceptar propuesta
  + Actores: Cliente
  + Precondiciones: CU001, CU002, CU005, CU009, CU012. Requiere estar logado y registrado. Además, requiere haber contactado previamente con el profesional y haber recibido una propuesta.
  + Postcondiciones: CU011. Requiere pagar el trabajo.
  + Descripción: Permite aceptar una propuesta realizada por el trabajador, aceptando el precio pactado.
* CU011 – Pagar Trabajo
  + Actores: Cliente
  + Precondiciones: CU001, CU002, CU010 ó CU013. Requiere estar logado y registrado. Además, requiere haber aceptado la propuesta del trabajador, o haber escogido una entre las distintas propuestas.
  + Postcondiciones: Ninguna.
  + Descripción: El cliente tras realizarse un trabajo debe pagar al trabajador.
* CU012 – Enviar Propuesta
  + Actores: Profesional
  + Precondiciones: CU001, CU002, CU015 ó CU009. Requiere estar logado y registrado. Además, requiere que el cliente haya contactado con él o recibir una notificación.
  + Postcondiciones: Ninguna.
  + Descripción: El profesional, envía una propuesta al cliente para la realización de un trabajo donde se indica precio y el tiempo estimado.
* CU013 – Escoger propuesta
  + Actores: Cliente
  + Precondiciones: CU001, CU002, CU003, CU012. Requiere estar logado y registrado. Además, requiere haber solicitado un profesional y haber recibido una propuesta.
  + Postcondiciones: CU011. El cliente debe pagar al trabajador tras haber finalizado el trabajo.
  + Descripción: El cliente acepta la propuesta enviada por el trabajador.
* CU014 – Suscribir a Profesión.
  + Actores: Profesional
  + Precondiciones: CU001. Requiere que el usuario esté registrado.
  + Postcondiciones: Ninguna.
  + Descripción: Permite al trabajador suscribirse a alertas de trabajos en distintos oficios en los que es experto.
* CU015 – Recibir notificación.
  + Actores: Profesional
  + Precondiciones: CU001, CU014, CU003. Requiere que el usuario esté registrado, logado, suscrito a profesiones, y que un cliente haya pedido un profesional
  + Postcondiciones: Ninguna.
  + Descripción: El trabajador recibirá notificaciones de clientes en su zona pidiendo servicios de oficios en los que esté suscrito.

b. Diseño de la arquitectura de la aplicación

Diseño de la Base de Datos:



Diseño de Entidades y Clases:

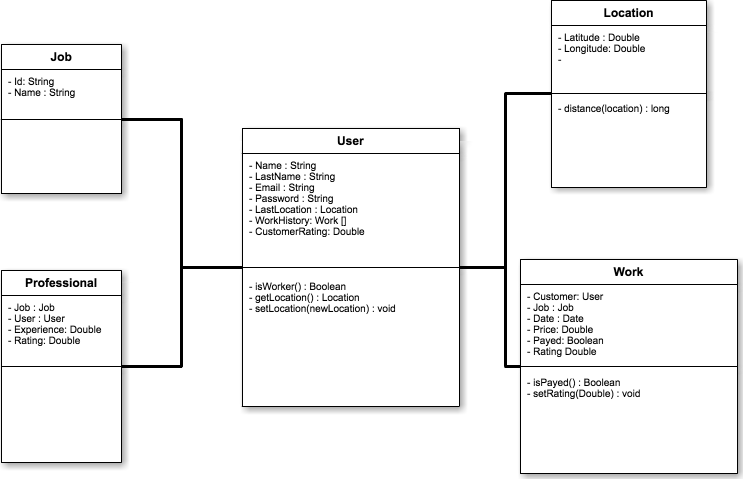
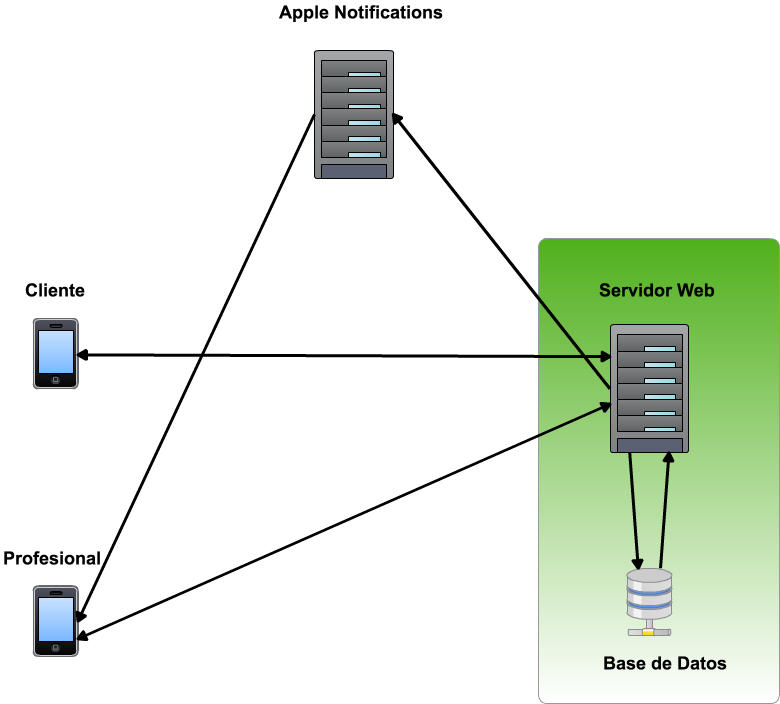
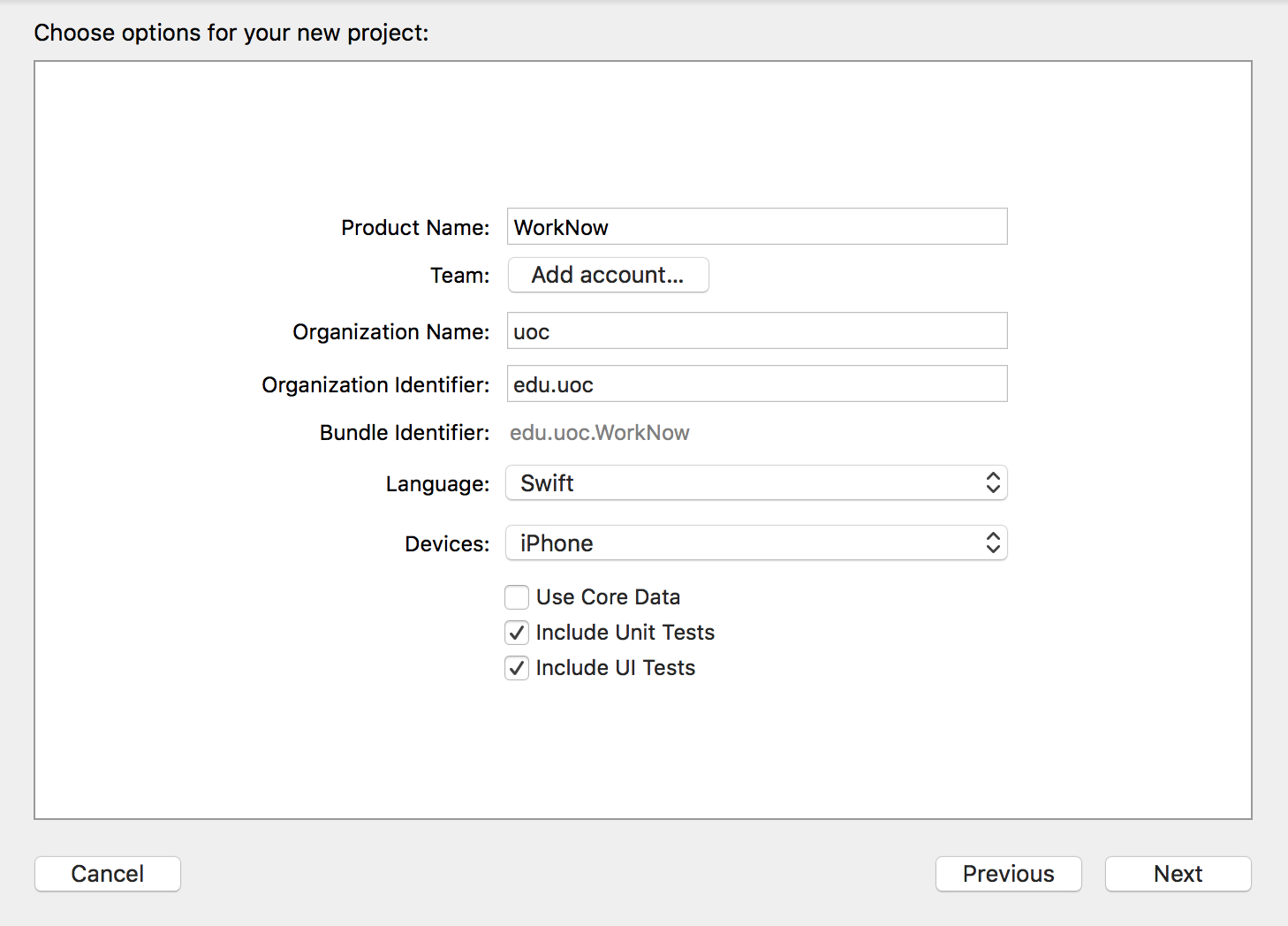


Diagrama de Arquitectura del Sistema:



# 3. Implementación

Para la fase de implementación se ha realizado una aplicación iOS en lenguaje Swift para dispositivos iPhone, descartando la utilización en iPad. El estilo de aplicación escogido es una Single View Application.



**3.1 Uso de Firebase**

Con el propósito de acelerar el desarrollo de la aplicación se ha hecho uso del servidor backend de Google Firebase [8].

Este servidor permite desarrollar aplicaciones usando su sistema de autenticación y base de datos alojada en sus servidores.

Para usar Firebase es necesario descargar mediante Cocoapods las librerías incluyéndolas en el Pod file:

pod 'Firebase'

pod 'Firebase/Core'

pod 'Firebase/Auth'

pod 'Firebase/Database'

El formato de los datos almacenados en la base de datos es de objetos JSON [9], aunque no es necesario que sean convertidos explícitamente en la aplicación iOS, el servidor es capaz de traducirlos desde tipos de datos simples como String, Int, Boolean, Array o Dictionary.

A continuación se muestra un ejemplo de un usuario almacenado en la base de datos Firebase:

"f8W3YGy0AqVoLs86lHpL1vk6kG92" : {

"apellidos" : "Arguello Flores",

"email" : "javiaf@outlook.com",

"latitude" : "41.385748",

"longitude" : "2.166745",

"nombre" : "Javier",

"oficios" : [ {

"nombre" : "Informática",

"selected" : true

}, {

"nombre" : "Mecánica",

"selected" : false

}, {

"nombre" : "Mecanica bicicletas",

"selected" : true

}, {

"nombre" : "Electricista",

"selected" : false

}, {

"nombre" : "Cerrajería",

"selected" : true

}, {

"nombre" : "Fontanería",

"selected" : false

}, {

"nombre" : "Construcción",

"selected" : false

}, {

"nombre" : "Limpieza",

"selected" : false

} ],

"password" : "12345678"

}

3.1.1 Conexión con el servidor:

Registro:

Para registrar un usuario, el código Swift para realizar esta operación se muestra a continuación. En esta fase, la aplicación aún no permite realizar registro y logados con Google o Facebook, por lo que debemos registrarnos con email y password.

FIRAuth.auth()?.createUser(withEmail: email, password: pass, completion: {(user, error) in

if error != nil {

self.showAlert(title: "Registro fallido",message: error!.localizedDescription)

return

}else{

//REGISTRO CORRECTO

return

}

})

Login:

Para realizar la operación de logado en la aplicación, necesitaremos habernos registrado previamente y como se anotó anteriormente, en esta fase solo podemos entrar utilizando email y contraseña, el código Swift a utilizar será el siguiente:

FIRAuth.auth()?.signIn(withEmail: email, password: pass){ (user, error) in

if error != nil{

self.showAlert(title: "Error al iniciar sesión",message: (error?.localizedDescription)!)

}else{

self.performSegue(withIdentifier: "LoginToMenu", sender: self)

}

Guardar Datos:

Para guardar datos lo primero que se necesita es tener una referencia al path donde queremos guardar nuestro objeto. Si queremos añadir un objeto JSON bajo la clave “users”, lo primero será crear una referencia al path:

let ref:FIRDatabaseReference = FIRDatabase.database().reference(withPath: "users")

A continuación creamos un Usuario de la clase User con los datos recogidos de los Campos de Texto de la Vista. Con el método toAnyObject convertimos el objeto Usuario a un objeto genérico para facilitar el guardado en Firebase.

var usuario:User = User(objectId: "1",nombre: self.nombreTF.text!,apellidos: self.apellidosTF.text!,email: email,password: pass)

let userRef = self.ref.child(user!.uid)

usuario.oficios = self.oficiosList

userRef.setValue(usuario.toAnyObject())

userRef.child("oficios").setValue(usuario.oficiosToAnyObjectList())

Finalmente con el método setValue podemos guardar o editar los datos en Firebase. Esto último se hace en la última línea, donde un array de datos es añadido al usuario anteriormente creado bajo la clave “oficios”.

**3.1.2 Implementación del Chat usando Firebase**

Una parte importante de la aplicación es la comunicación entre cliente y profesional, por lo que es requerido al menos un chat para que ambos puedan intercambiar opiniones, números de teléfono y negociar precios.

Para realizar el chat se ha hecho uso de una librería externa, llamada “JSQMessagesViewController” [10] que nos proporciona un ViewController con la interfaz gráfica y elementos de control necesarios para realizar un Chat entre al menos dos dispositivos.

Para añadir esta librería al proyecto es necesario utilizar Cocoapods e incorporar el pod de JSQMessagesViewController añadiendo la siguiente línea a nuesto PodFile:

pod ‘JSQMessagesViewController’

Posteriormente se requiere instalar la librería ejecutando desde el terminal de Mac:

pod install

Un elemento importante a destacar de la implementación del ViewController que hereda del incluído en la librería es la estructura del chat en Firebase. Este se guarda en la base de datos como un objeto JSON al igual que los Usuarios, Oficios y otros objetos. Todos los chats derivan se guardan bajo la clave “chats”.

Cada chat necesita un identificador único, el cual se crea mediante la unión de los identificadores de usuario acortados, y usando las iniciales del usuario. Para que siempre se utilice el mismo identificador (sin importar quien inicie la sesión), se comprueba que identificador es mayor alfabéticamente hablando.

El método para crear el identificador es el siguiente:

    func createChatIdentifier(id1: String, id2: String, name1: String, name2: String, surname1: String, surname2: String) -> String {

        var identifier: String = ""

        if(correctIdLen(id1,id2) && correctNameLen(name1, name2) && correctNameLen(surname1, surname2)){

            let id1index = id1.index(id1.startIndex, offsetBy: 5)

            let id2index = id2.index(id2.startIndex, offsetBy: 5)

            let n1index = name1.index(name1.startIndex, offsetBy: 1)

            let n2index = name2.index(name2.startIndex, offsetBy: 1)

            let sn1index = surname1.index(surname1.startIndex, offsetBy: 1)

            let sn2index = surname2.index(surname2.startIndex, offsetBy: 1)

            identifier = identifier + id1.substring(to: id1index) + name1.substring(to: n1index) + surname1.substring(to: sn1index)

            identifier = identifier + id2.substring(to: id2index) + name2.substring(to: n2index) + surname2.substring(to: sn2index)

        }

        return identifier

    }



En el caso de la captura, tendríamos como identificador ‘Gvu9aAGNaNNiJA’, que serían las primeras 5 letras del idenficador de el primer usuario, AG sus iniciales, las siguientes 5 letras el identificador del segundo usuario y JA sus respectivas iniciales.

Por último, para que el chat trabaje en tiempo real, necesitamos que el ViewController esté ejecutando un manejador para que observe los mensajes como se muestra en el método siguiente.

    private func observeMessages() {

        messageRef = chatRef!.child("messages")

        // 1.

        let messageQuery = messageRef.queryLimited(toLast:25)

        // 2. We can use the observe method to listen for new

        // messages being written to the Firebase DB

        newMessageRefHandle = messageQuery.observe(.childAdded, with: { (snapshot) -> Void in

            // 3

            let messageData = snapshot.value as! Dictionary<String, String>

            if let id = messageData["senderId"] as String!, let name = messageData["senderName"] as String!, let text = messageData["text"] as String!, text.characters.count > 0 {

                // 4

                self.addMessage(withId: id, name: name, text: text)

                // 5

                self.finishReceivingMessage()

            } else {

                print("Error! Could not decode message data")

            }

        })

    }

La clave está en messageQuery.observe(.childAdded…), que se ejecutará siempre que un elemento hijo se añada a la clave “messages” de nuestro chat en la base de datos de Firebase.

Por último, comentar que al ser un ViewController de un tercero, la interfaz gráfica requiere ser adaptada a la nuestra desde código, donde se ha cambiado el fondo, se han eliminado botones no necesarios en esta aplicación, y otros cambios para adecuar el diseño al resto de la aplicación.

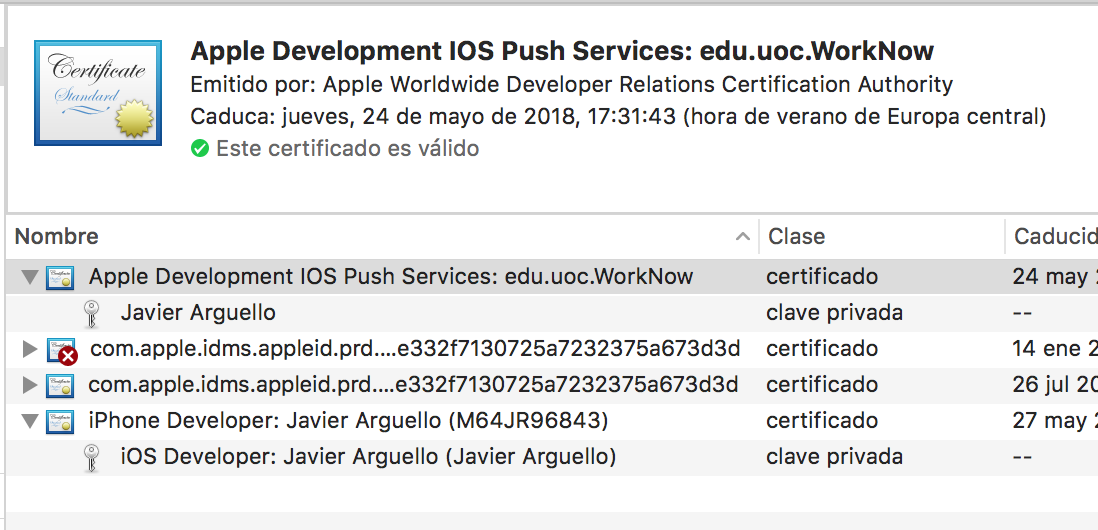
**3.1.3 Notificaciones Push iOS mediante Firebase**

La parte más costosa a nivel de implementación ha sido sin duda el uso de las notificaciones push.

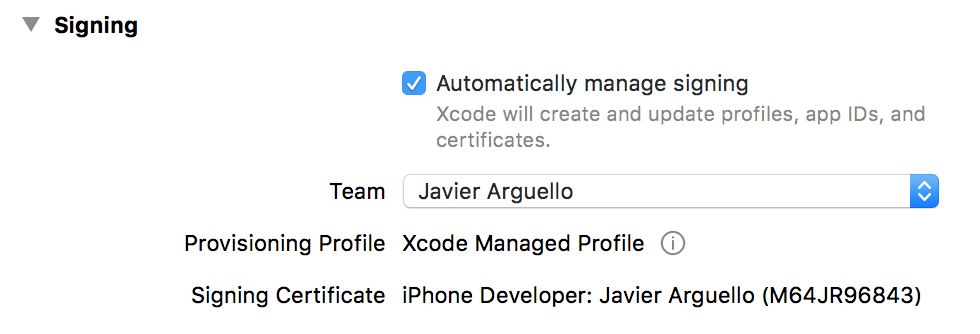
En primer lugar, es requerido registrarse como desarrollador iOS en la web de Apple Developers [11] mediante el previo pago de aproximadamente 100 euros.



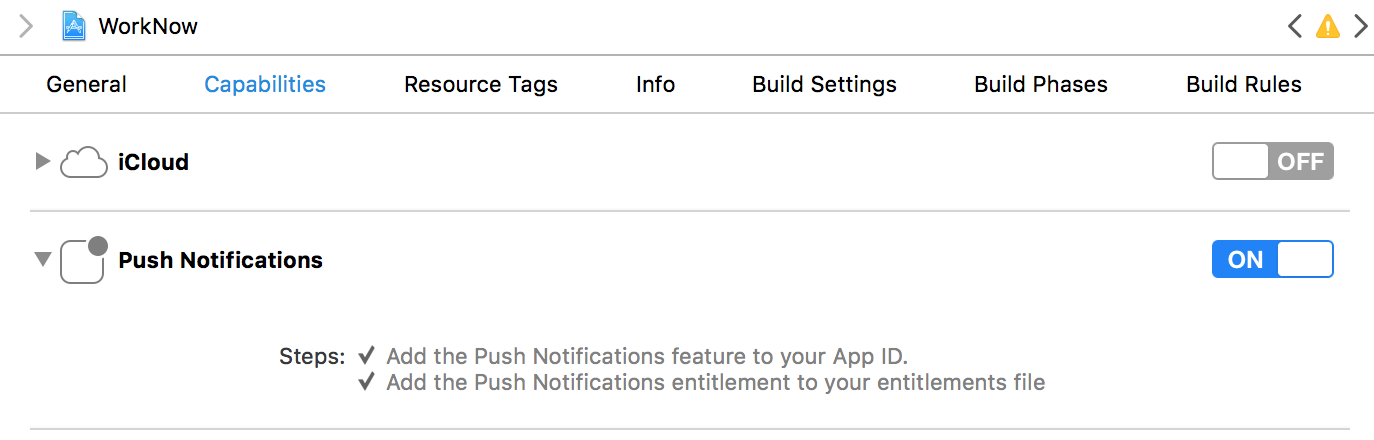
Después de completar el registro como desarrollador de Apple, se requiere crear un certificado de desarrollador iOS (para poder utilizarlo en dispositivos) y otro para notificaciones push de iOS.



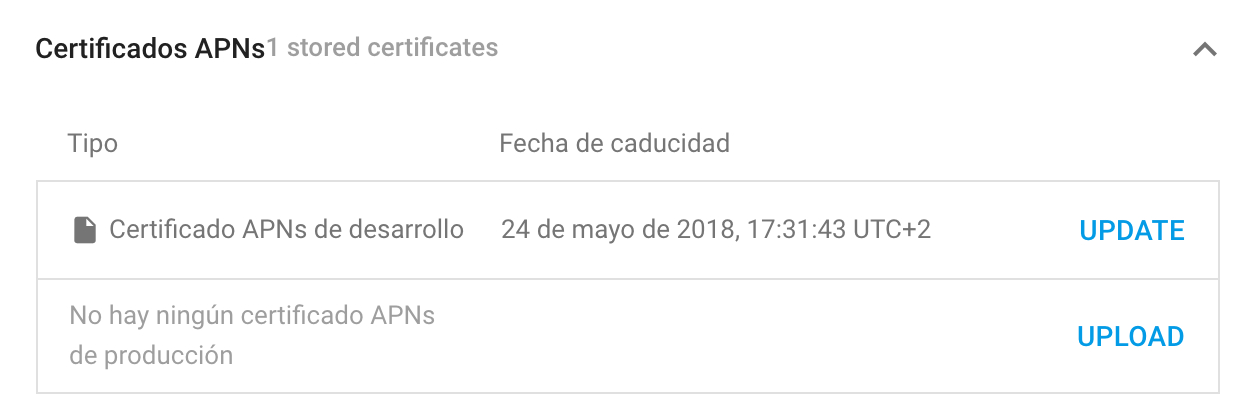
Ambos certificados han de estar instalados en el Administrador de Llaveros de nuestro Mac para habilitar las notificaciones. Además, el certificado de desarrollador de iPhone debe estar incluído en el proyecto de Xcode en **Proyecto > General > Signing.**



En la sección **Capabilities** del Proyecto, es necesario habilitar las notificaciones Push.



Por último, es necesario subir el certificado para uso de notificaciones Push a Firebase desde la Configuración del proyecto, en la sección “Mensajería en la Nube”



Para hacer uso de las notificaciones Push es necesario añadir al AppDelegate del proyecto funciones y líneas de código a funciones ya implementadas para habilitar las mismas.

En primer lugar, en la función didFinishLaunchingWithOptions es necesario añadir el siguiente código para habilitar las notificaciones, diferenciando si nos encontramos en iOS 10 o una versión anterior, donde las notificaciones funcionaban de otra manera.

 func application(\_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {

        // Override point for customization after application launch.

        if #available(iOS 10.0, \*) {

            // For iOS 10 display notification (sent via APNS)

            UNUserNotificationCenter.current().delegate = self

            let authOptions: UNAuthorizationOptions = [.alert, .badge, .sound]

            UNUserNotificationCenter.current().requestAuthorization(

                options: authOptions,

                completionHandler: {\_, \_ in })

            // For iOS 10 data message (sent via FCM

            FIRMessaging.messaging().remoteMessageDelegate = self

        } else {

            let settings: UIUserNotificationSettings =

                UIUserNotificationSettings(types: [.alert, .badge, .sound], categories: nil)

            application.registerUserNotificationSettings(settings)

        }

        application.registerForRemoteNotifications()

        registerForPushNotifications()

}

También es requerido añadir una función para recibir los mensajes llamada applicationReceivedRemoteMessage, la cual incluye un diccionario (appData) con claves incluidas en la notificación.

func applicationReceivedRemoteMessage(\_ remoteMessage: FIRMessagingRemoteMessage) {

        print(remoteMessage.appData)

        let dict = remoteMessage.appData["notification"] as! NSDictionary

...

}

La siguiente llamada desde AppDelegate, nos permite enviar a un ViewController en concreto la notificación, para que sólo se reciban las notificaciones en caso de que estemos dentro de ese ViewController, el cual tiene que tener un manejador.

NotificationCenter.default.post(name: NSNotification.Name(rawValue: Constants.NotificationKeys.WorkerOffered), object: nil, userInfo: remoteMessage.appData)

También en esta aplicación es necesario recibir notificaciones basadas en los oficios a los que está suscrito el usuario, lo cual se hace con una llamada al siguiente método de FIRMessaging.

FIRMessaging.messaging().subscribe(toTopic: Global.convertString(str: oficio.nombre))

Por último, cabe mencionar que necesitamos llamar a Firebase para realizar notificaciones Push desde la propia aplicación, pero no existe un API para iOS que nos permita realizar esto, por lo que tenemos que realizar peticiones HTTP POST en formato JSON a la URL https://fcm.googleapis.com/fcm/send.

        let json = [

            "to": "/topics/\(Global.convertString(str: categoria))",

         "notification": [

            "body": "Has recibido una nueva oferta de trabajo",

            "title": "Trabajo"

         ],

         "data": [

                "sender\_token": Global.deviceToken,

                "categoria": categoria,

                "precio\_max": precio,

                "user\_id": userid,

                "user\_name": username,

                "user\_surname": surname,

                "latitude": "\((location?.coordinate.latitude)!)",

                "longitude": "\((location?.coordinate.longitude)!)",

                "descripcion": description

            ]

         ] as [String: Any]

        var request = URLRequest(url: URL(string: "https://fcm.googleapis.com/fcm/send")!)

        request.httpMethod = "POST"

        request.addValue("application/json", forHTTPHeaderField: "Content-Type")

        request.addValue("key=\(Global.apiKey)", forHTTPHeaderField: "Authorization")

        request.httpBody = try! JSONSerialization.data(withJSONObject: json, options: [])

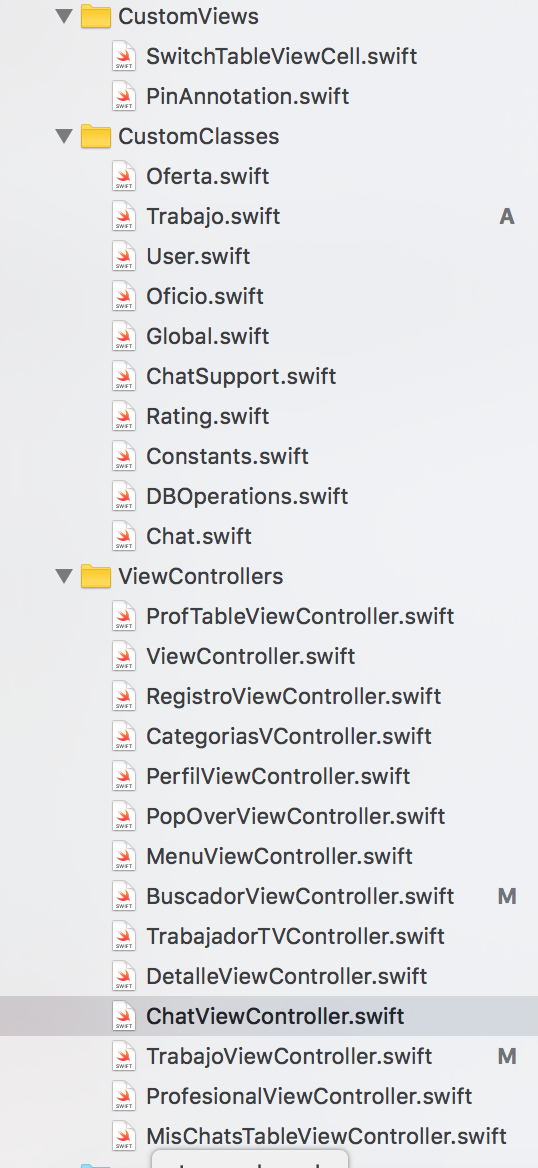
Dos campos importantes de la petición son los siguientes:

* to: En este campo indicamos a quien le queremos enviar la petición. Podemos enviar a todos los usuarios de la aplicación WorkNow, a los suscritos a un tema o a un dispositivo concreto. Para enviar a un dispositivo concreto necesitamos el token ID del dispositivo y para un tema en concreto (un oficio) enviaríamos a “/topics/<oficio>”.
* Authorization: En este campo es necesario utilizar la clave del servidor que se obtiene en la consola de Firebase. Sin este token no nos dejaría realizar peticiones http, además de identificar nuestro proyecto en Firebase.

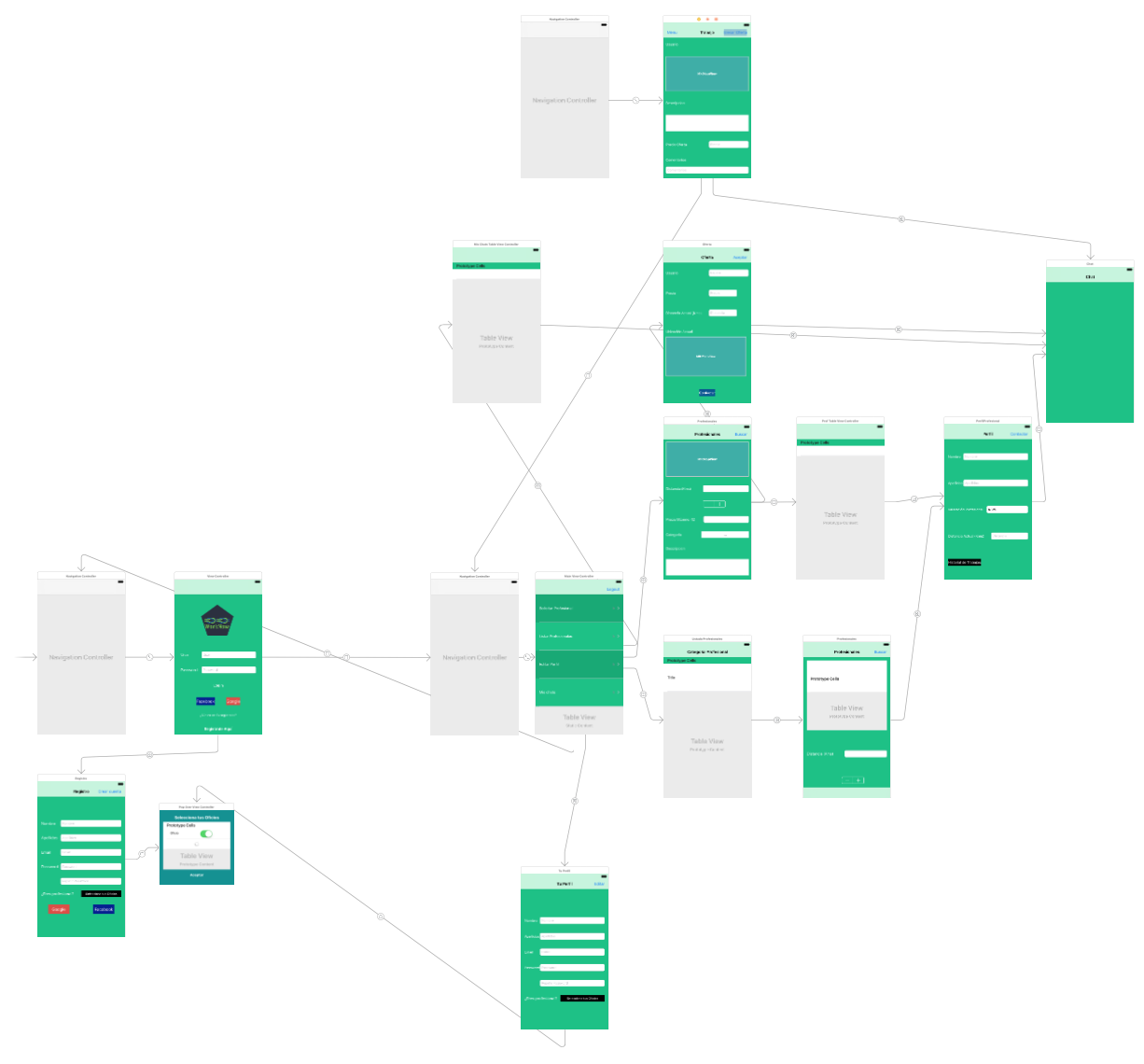
**3.2 Organización del proyecto XCode**

El proyecto se ha dividido en 3 tipos de clases dependiendo de la utildad que va a realizar en la aplicación, ya sean Vistas, Controladores de Vista o Clases que incluyen Modelos de datos:

* CustomViews: Son controladores de vistas simples personalizadas para una funcionalidad muy concreta. En esta etapa del proyecto sólo se ha realizado una Celda de una Tabla personalizada para incluir un interruptor (switch) además del nombre del oficio. Además se realizó un PinAnnotation personalizado para las notificaciones en el mapa.
* ViewControllers: Controladores para vistas completas que representan cada una, pantallas individuales de la aplicación, todas definidas en el Storyboard, que viene a ser el espacio de diseño de aplicaciones en iOS. La vinculación entre el Storyboard y los Controladores se realiza mediante el uso de identificadores, outlets y manejadores de acciones (para botones, interruptores y otros elementos de la interfaz).
* CustomClasses: En esta sección se incluye el modelo de datos de la aplicación. Con estas clases se podrán almacenar y manejar Usuarios, Oficios y Ofertas. Por otra parte, se tendrá una clase Global que almacenará datos utilizados por distintos ViewControllers a modo de Singleton.



El Storyboard del proyecto ha utilizado diferentes ViewControllers de distintos tipos como se explicó anteriormente. El resultado se puede ver en la captura incluída a continuación.

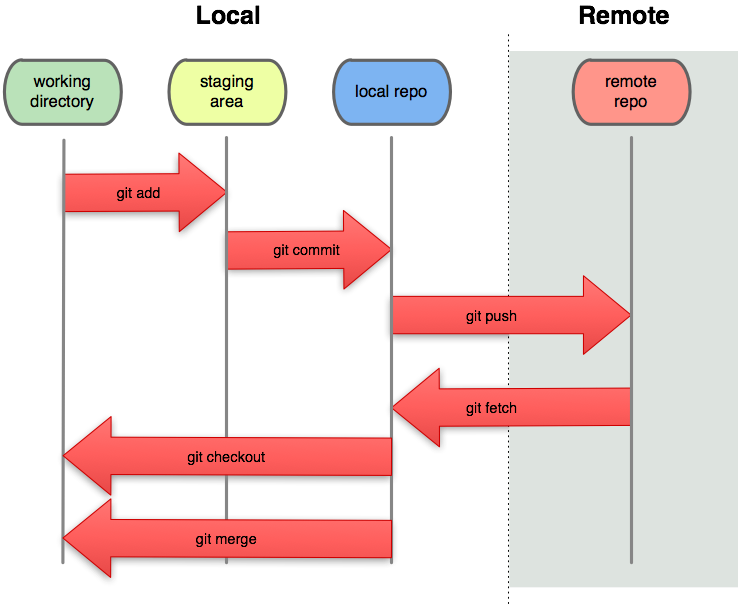


**3.3 Git**

Git es un software de control de versiones diseñado por Linus Torvalds, uno de los creadores de Linux, y está pensado en la eficiencia y confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de ficheros de código fuente. Una de las características que tiene git es la posibilidad de trabajar sobre ramas, permitiendo tener por ejemplo, una rama para actualizar los cambios sobre versiones estables, y otra rama para realizar cambios en versiones en desarrollo, con código todavía aún no estable. Se puede trabajar tanto en local como volcar lo cambios de las ramas en repositorios remotos, con la posibilidad de clonar el repositorio remoto en otro ordenador y seguir el trabajo tal como se dejó.

El manejo de git se realiza a través de línea de comandos, aunque pueden utilizarse herramientas que proporcionen al usuario una interfaz de usuario, facilitando su uso al usuario. Git dispone de una serie de comandos que cambian el estado de un fichero dentro del repositorio como muestra la figura a continuación:

* **config:** cambia los valores de configuración como nombre de usuario, email, etc.
* **init:** inicializa un repositorio git creando el directorio ‘.git’.
* **clone:** copia un repositorio git desde una fuente remota.
* **add:** añade ficheros que han cambiado en el directorio de trabajo en el índice para ser posteriormente actualizados en el repositorio git mediante un commit.
* **commit:** toma todos los cambios escritos en el índice y hace que la rama apunte a este nuevo commit con los cambios realizados en él.
* **push:** vuelca los cambios con todos los nuevos commits en un repositorio remoto, como puede ser GitHub.
* **status:** muestra las diferencias que existen entre el directorio de trabajo y el repositorio, avisando de si ha habido cambios en algún fichero para su posible actualización.
* **checkout <rama>:** cambia de una rama de trabajo a otra para que los cambios **fetch:** toma todos los objetos desde el repositorio remoto que no se encuentran en el repositorio local.



**3.4 Pruebas**

# 4. Conclusiones

**4.1 Conclusiones del trabajo**

**4.2 Resultados de los Objetivos del trabajo**

**4.3 Seguimiento de la planificación**

**4.4 Líneas de Trabajo futuro**

# 5. Glosario

# 6. Bibliografía

[1] - Bellán García, Antonio; Aceituno Nieto, Pilar; Pérez Díaz, Julio; Ramiro Fariñas, Diego; Ayala García, Alba; Pujol Rodríguez, Rogelio. Un perfil de las personas mayores en España, 2019. Indicadores estadísticos básicos. Madrid, Informes Envejecimiento en red nº 22, 38p. [Fecha de publicación: 06/03/2019]. Disponible en <http://envejecimiento.csic.es/documentos/documentos/enred-indicadoresbasicos2019.pdf>.

[2] – Viaña, Daniel. El gasto sanitario se incrementará más de 580 millones al año por el envejecimiento de la población. El Mundo [Internet], 2017 [Fecha de publicación: 04/05/2017]. Disponible en <https://www.elmundo.es/economia/macroeconomia/2017/05/04/590a3281e2704e9e178b46b1.html>

[3] - Bellán García, Antonio; Aceituno Nieto, Pilar; Pérez Díaz, Julio; Ramiro Fariñas, Diego; Ayala García, Alba; Pujol Rodríguez, Rogelio. Un perfil de las personas mayores en España, 2019. Indicadores estadísticos básicos. Madrid, Informes Envejecimiento en red nº 22, 38p. [Fecha de publicación: 03/11/2016]. Disponible en <http://envejecimiento.csic.es/documentos/documentos/Estudio-Cuidadores-segg.pdf>.

[4] – Agencia Atlas. Encuentran el cadáver de una anciana que llevaba cinco años muerta. La Provincia, Diario de Las Palmas [Internet], 2019 [Fecha de publicación: 24/04/2019]. Disponible en <https://www.laprovincia.es/multimedia/videos/sucesos/2019-04-24-172706-encuentran-cadaver-anciana-llevaba-cinco-muerta.html>

[5] – Sanz, J. Hallan un anciano que llevaba más de una semana muerto y rodeado de basura en Valladolid. El Norte de Castilla [Internet], 2019 [Fecha de publicación: 09/07/2019]. Disponible en <https://www.elnortedecastilla.es/valladolid/hallan-anciano-llevaba-20190709201004-nt.html>

[6] - Dave Evans. The Internet of Things: How the Next Evolution of the Internet Is Changing Everything. CISCO White Paper [Internet], 2011 [Fecha de publicación 01/04/2011]. Disponible en <https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf>

[7] - Man,S. Wereable computing: a first step toward personal imaging. Computer, vol. 30, no. 2, pp. 25-32, Feb. 1997.

[8] - Conor O'Quigley, C. ; Sabourin, M. ; Coyle, S. ; Connolly, J. ; Condall, J. ; Curran, K. ; Corcoran, B. ; Diamond, D. Characteristics of a Piezo-Resistive Fabric Stretch Sensor Glove for Home-Monitoring of Rheumatoid Arthritis. 2014 11th International Conference on Wearable and Implantable Body Sensor Networks Workshops, pp. 23-26, June 2014

[9] - Google LLC [Internet]. 2019. Google Play. [Consultado el: 07/10/2019]. Disponible en <https://play.google.com/>

[10] - Google LLC [Internet]. 2019. Salud Conectada. [Consultado el: 07/10/2019]. Disponible en <https://play.google.com/store/apps/details?id=es.sanitas.hosp.saludconectada&gl=ES>

[11] - Google LLC [Internet]. 2019. Heart Rate OS. Android Watch. [Consultado el: 07/10/2019]. Disponible en <https://play.google.com/store/apps/details?id=com.jwork.wearable.heartratesync2&gl=ES>

[12] - Google LLC [Internet]. 2019. Health Wereable. [Consultado el: 07/10/2019]. Disponible en <https://play.google.com/store/apps/details?id=com.analogdevices.healthwearable&gl=ES>

[13] – Broeders J. Transition from Wearable to Medical Devices. Analog Devices [Internet]. [Consultado el: 08/10/2019]. Disponible en <https://www.analog.com/en/technical-articles/transition-from-wearable-to-medical-device.html>

[14] – López, Gregorio; Custodia, Victor; Moreno, José Ignacio. LOBIN: E-Textile and Wireless-Sensor-Network-Based Platform for Healthcare Monitoring in Future Hospital Environments. IEEE Transactions on Information Technology in Biomedicine, vol.14, issue 6, Nov. 2010

[15] - Google LLC [Internet]. 2019. Crashlytics Firebase. [Consultado el: 10/10/2019]. Disponible en <https://firebase.google.com/docs/crashlytics/?hl=es-419>

[16] - Rodríguez, José Ramón. El trabajo final como proyecto. FUOC. Fundació para la Universitat Oberta de Catalunya. [Consultado el: 02/10/2019]

[17] – Scrum.org [Internet]. 2019. The home of the Scrum. [Consultado el: 09/10/2019]. Disponible en [*https://www.scrum.org/*](https://www.scrum.org/)

[18] – Clarisó, Robert. Introducción al trabajo final. FUOC. Fundació para la Universitat Oberta de Catalunya [Consultado el 03/10/2019].

[19] – Rodriguez, José Ramón. La gestión del proyecto a lo largo del trabajo final. FUOC. Fundació para la Universitat Oberta de Catalunya [Consultado el 04/10/2019]

# 6. Anexos