Aplicación Móvil para Petición de Trabajadores en Demanda.

Javier Argüello Flores
Master en Desarrollo de Aplicaciones Móviles

Consultor: Roger Monserrat Ribes

Fecha de entrega: 15/03/2017


Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

Licencias alternativas (elegir alguna de las siguientes y sustituir la de la página anterior)

A) Creative Commons:

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-CompartirIgual 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-SinObraDerivada 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento-CompartirIgual 3.0 España de Creative Commons

Esta obra está sujeta a una licencia de Reconocimiento 3.0 España de Creative Commons

B) GNU Free Documentation License (GNU FDL)

Copyright © AÑO TU-NOMBRE.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Soft-

ware Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU Free Documentation License".

C) Copyright

© (el autor/a)
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:	Aplicación Móvil para Petición de Trabajadores en Demanda.
Nombre del autor:	Javier Argüello Flores
Nombre del consultor:	Roger Monserrat Ribes
Fecha de entrega (mm/aaaa):	MM/AAAA
Titulación:	*Máster Universitario en Desarrollo de Aplicaciones para Dispositivos Móviles*

Resumen del Trabajo (máximo 250 palabras):

Este Trabajo Final de Master consiste en el diseño e implementación de una Aplicación Móvil en demanda, que permita a los clientes y trabajadores estar en contacto siempre que un cliente potencial requiera un trabajo del día a día en cualquiera de las diferentes áreas de trabajo cotidianas existentes en nuestra sociedad actual.

El cliente puede requerir para reparar un dispositivo electrónico en su casa y existe un trabajador en su zona que tiene los conocimientos necesarios para solucionarlo. Esta aplicación les pondrá en contacto y les permitirá acordar el precio y otros detalles en la transacción.

Los trabajadores no necesitarán un amplio CV para trabajar con los clientes, sólo tendrán que demostrar tener las habilidades suficientes para hacer el trabajo. El cliente estará seguramente agradecido en caso de un buen trabajo, y una buena calificación será probablemente.

La aplicación estará disponible para dispositivos iOS (iPhone, iPad) y requerirá conexión a Internet y capacidad de localización. Este TFM incluye el desarrollo de la aplicación móvil, además del servidor utilizado por la aplicación.

Abstract (in English, 250 words or less):

This Master Thesis consists in the design and implementation of a Mobile Application on demand that allow customers and workers to be in touch whenever a potential customer requires a daily work to be done in any of the different quotidian working areas existing in our current society.

Customer may require to repair one electronic device at their home and a worker has the knowledges to fix it. This application will put in touch them and

will permit them to agree the Price and other details in the transaction.

Workers will not need a huge CV to work with customers, just will have to demonstrate having the enough skills to make the work done. Customer will be grateful in case of a good job, and a good rating will be likely.

The application will be available for iOS devices (iPhone, iPad) and will require internet connection and location capabilites. This thesis include the mobile application development and also the server used by the application.

Palabras clave (entre 4 y 8):

Aplicación, Móvil, En, Demanda, Trabajadores, Clientes.

Índice

Lista de figuras

No se encuentran elementos de tabla de ilustraciones.

1. Introducción

1.1 Contexto y justificación del Trabajo

A día de hoy, nos encontramos en una época donde prácticamente todos los sectores laborales de nuestro entorno están informatizados, pasando desde las Tiendas online, que desde finales de los 90 y principios de 2000 empezaron a desembarcar y hacerse cada vez más populares hasta aplicaciones de móviles para solicitar comida a domicilio, alquiler de apartamentos turísticos, o solicitar taxis y coches por demanda (como puede ser el caso de Uber).

En el contexto de las aplicaciones de petición por demanda de servicios [1] existe una necesidad no cubierta para trabajadores autónomos de diversos oficios, donde en muchas ocasiones, potenciales usuarios requieren de un servicio profesional en lugares o sectores desconocidos y, por otra parte, existen trabajadores con conocimientos para cubrir esas necesidades no pueden llegar a ponerse en contacto con su potencial cliente.

Existen varios sitios webs que proporcionan desde hace años un directorio de trabajadores profesionales y comercios categorizados por diferentes profesiones como pueden ser Páginas Amarillas [2] y QDQ [3]. Ambos han competido enviando a domicilio guías en papel, y hoy día han informatizado sus servicios, y disponen de sitio web y aplicación móvil. Además, el buscador de Google (haciendo uso de Google Maps), permite localizar negocios y situarlos en el mapa, al igual que Páginas Amarillas y QDQ.

El punto fuerte que tienen todas ellas, es que tienen un negocio establecido desde hace años, y una base de datos de profesionales y negocios muy grande. En el caso de Google, además, su base de datos no es sólo de ámbito nacional, por lo que podemos hacer uso de ello en todo el mundo.

Como puntos débiles, todas ellas están hechas para profesionales y comercios con una localización fija y con unos horarios fijos, aunque Páginas Amarillas y QDQ también proporcionan teléfonos de profesionales 24 horas, donde en la mayoría de las ocasiones, las demandas son atendidas desde una centralita que envían profesionales a domicilio, que en la mayoría de las ocasiones no informan correctamente del rango de precio que se cobrará y la comunicación se produce entre el tele operador y el cliente, nunca entre cliente y profesional.
Por último, estas aplicaciones siempre van a proporcionar contactos de profesionales autónomos, que, en ocasiones, van a realizar tareas muy básicas y van a cobrar el mínimo por desplazamiento de un profesional de ese sector, que en ocasiones es muy elevado. En nuestro entorno existe mucha gente que es capaz de realizar trabajos (formatear un ordenador, cambiar la pieza rota de una bicicleta, desatascar un desagüe…) sin dedicarse plenamente a ello, y ofrecer unos precios mucho más competitivos, pero que, por otra parte, no es posible dar con este tipo de personas a no ser que sea por recomendación de conocidos.

1.2 Objetivos del Trabajo

El objetivo principal del trabajo será la creación de una aplicación móvil que sea capaz de poner en contacto a potenciales clientes con expertos en diversas profesiones para poder solucionar el problema del potencial cliente a cambio de una transacción económica.

La Lista de Requisitos Funcionales es la siguiente:

- RF001: La aplicación requiere registro del usuario, tanto para clientes como para profesionales. El registro permitirá hacerse mediante un formulario con correo electrónico, contraseña y datos personales, o utilizando Google o Facebook para acelerar el proceso.

- RF002: La aplicación permitirá iniciar sesión al usuario registrado utilizando correo electrónico y contraseña, o mediante el uso de Google o Facebook.

- RF003: La aplicación permitirá al usuario suscribirse a alertas de diferentes profesiones como trabajador.

- RF004: La aplicación permitirá al usuario pedir un servicio profesional en cualquier lugar, enviando su ubicación y la descripción del servicio requerido como campos obligatorios, y permitiendo indicar de manera opcional el precio máximo a pagar.

- RF005: La aplicación permitirá al trabajador suscrito a una alerta, realizar una propuesta al cliente potencial con un rango de precio, duración del mismo y otros datos de interés para el cliente.

- RF006: La aplicación mostrará en un mapa puntos con diferentes ofertas de los trabajadores que recibieron la petición. El usuario podrá acceder a cada una de estas ofertas y aceptar únicamente una de ellas.

- RF007: La aplicación debe mostrar listados por categorías de diferentes profesionales en un rango de kilómetros respecto a la ubicación actual del usuario. Los profesionales estarán ordenados por puntuación media.

- RF008: La aplicación permitirá al usuario puntuar y realizar un comentario al trabajador tras realizarse el trabajo y haberse pagado.

- RF009: La aplicación permitirá realizar pagos a través de la app de los servicios contratados.

La Lista de Requisitos No Funcionales es la siguiente:

- RNF001: La plataforma destino de la aplicación será iOS (iPad, iPhone) con posibilidad de ser ampliado en un futuro a Android y Aplicación Web.

- RNF002: El color base de la aplicación será el verde en una tonalidad oscura.

- RNF003: Se mostrará una pantalla de presentación con el logo de la aplicación en medio con fondo verde oscuro durante 2 segundos.

- RNF004: La aplicación mostrará un listado de oficios que incluya las profesiones más comunes desarrolladas por trabajadores autónomos (electricista, mecánico, cerrajero, fontanero, etc.), aunque esta lista podrá ser ampliada por demanda de los usuarios.

- RNF005: Los resultados obtenidos aparecerán o bien en un listado en forma de tabla o un mapa donde aparecerán puntos con los distintos profesionales encontrados.

- RNF005: Se hará el uso de la vibración del dispositivo en caso de encontrarse profesionales que acepten realizar el encargo.

- RNF006: La valoración del usuario se mostrará en un rango de 1 a 5 estrellas, permitiendo valores intermedios.

1.3 Enfoque y método seguido

El trabajo a realizar contempla como posibles estrategias la creación de un nuevo producto, y la adaptación de un trabajo existente, ya que en el Master se han realizado trabajos que incluye el uso de APIs de geolocalización que pueden ser de utilidad.

La aplicación, por lo tanto, reutilizará partes de código, aunque readaptadas a la temática a cubrir, y añadiendo nuevas funcionalidades.

El trabajo existente que será reutilizado, en su parte de geolocalización, será la PEC de la asignatura Desarrollo para Dispositivos iOS. Esta aplicación nos permite añadir Sitios al mapa utilizando la geolocalización actual del dispositivo, agregando a su vez, nombre, descripción y fotos al mismo. A su vez, permite al usuario encontrar en el mapa y en un listado una serie de Sitios que se encuentran en un rango de Kilómetros de la actual geolocalización del usuario.

Por una parte, para solicitar un profesional de un área laboral específica, será necesario indicar un radio en el que se desea encontrar a esos trabajadores, ya que no nos valdría de nada encontrar un informático en Barcelona si estamos viviendo en Vigo, por lo que esta parte es reutilizable en cierto modo para la realización de esta nueva aplicación.

Además, se desea mostrar un listado categorizado por oficios de distintos profesionales en nuestro rango, por si el usuario desea, en vez de solicitar el servicio a través de la app a los trabajadores disponibles, realizar un contacto directo a través de su número de teléfono. Estos listados deberán aparecer al menos en una Tabla, con la posibilidad de mostrarse en un mapa con puntos en el mismo. En esta parte pueden utilizarse partes funcionales de la PEC de la asignatura anteriormente citada.

Por último, se requerirá de añadir funcionalidades y adaptar el producto para cubrir los requerimientos específicos, como la suscripción a alertas, el uso de notificaciones, el contacto y negociación a través de la aplicación entre cliente potencial y el trabajador, etcétera.

1.4 Planificación del Trabajo

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

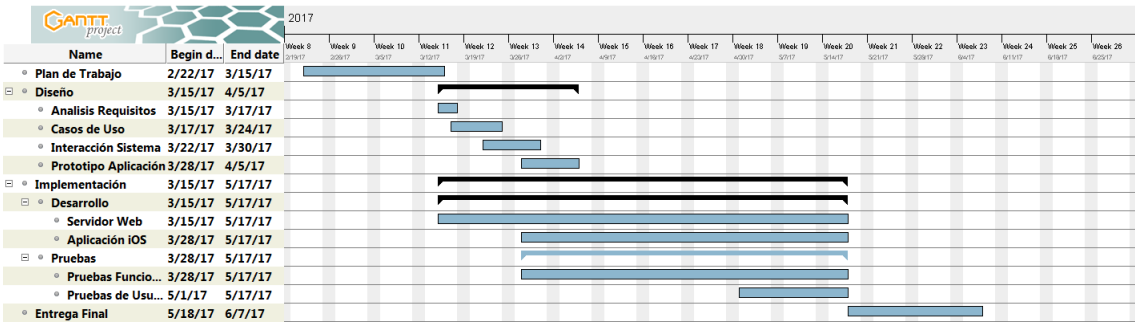
Las herramientas necesarias serán:

- Un ordenador personal Macintosh, que en este caso será un MacBook Pro, que nos permitirá desarrollar Software para dispositivos iOS.
- El Entorno de Desarrollo XCode, que nos permite desarrollar y publicar aplicaciones para dispositivos iOS, tanto en lenguaje Swift como en Objective C.
- Un Emulador para realizar pruebas en las primeras fases del desarrollo. Este, es incluido en XCode.
- Varios dispositivos móviles iOS (iPhone, iPad) para probar la aplicación en diferentes entornos y que nos permitan detectar errores y subsanarlos para realizar una aplicación multidispositivo que funcione correctamente.
- Software necesario para realizar la documentación, como editores de texto, herramientas para realizar gráficas, diagramas, y otros recursos que sean de utilidad para mostrar en la documentación.
- Hosting para poder subir el servidor web que pueda ser accesible desde la aplicación.

En el área de la planificación, el trabajo se dividirá en 4 fases que corresponderán con una entrega de cada PEC:

- Plan de Trabajo
- Diseño
- Implementación
- Entrega Final

Estas 4 fases darán lugar a un producto entregable en forma de Documentación, que en este caso será la Memoria del Trabajo Final de Master, dividido en entregas parciales, y por otra parte se obtendrá la aplicación móvil y un servidor web al que la aplicación realizará las peticiones.



1. Planificación del Trabajo: Diagrama de Gantt

Dentro de los hitos que se pretenden conseguir en cada entregable están los siguientes:

Plan de Trabajo:

- Saber definir una planificación realista para un proyecto, teniendo en cuenta su alcance y los recursos disponibles.
- Adquirir experiencia en afrontar los retos que supone sacar adelante un proyecto completo.
- Realizar una primera toma de requisitos tanto funcionales como no funcionales de la aplicación

Diseño:

- Investigar los usuarios de la aplicación y recoger requisitos, tanto cuantitativos como cualitativos, que ayudarán a conocer los usuarios y definir perfiles.
- Examinar y analizar las condiciones en que se utilizará el sistema para definir su contexto de uso.
- Elaborar escenarios de uso.
- Definición de los flujos de interacción en el sistema.
- Diseñar y construir un prototipo de alto nivel de la aplicación

Implementación:

El proceso de implementación del proyecto podrá dividirse en dos etapas:

Desarrollo

La etapa de desarrollo en nuestra aplicación se dividirá en dos partes, en el desarrollo del servidor web, donde se utilizará un Framework PHP que nos será de ayuda para la creación del mismo. Se contempla el uso de los siguientes Frameworks:

- BulletPHP
- Slim
- Silex

Si bien, el servidor requiere ser implementado en un futuro en caso de un crecimiento numeroso de los usuarios, por problemas de planificación y tiempo, se ha de utilizar un servidor backendless como Firebase de Google, que nos permite además hacer uso de las notificaciones Push tanto para dispositivos Android como iOS.

Para el desarrollo de la aplicación móvil para dispositivos iOS, se utilizará el lenguaje de programación Swift [5] (para dispositivos iOS) y se hará uso de APIs para mapas, geolocalización y mapas que nos proporciona el propio lenguaje. Además, se investigará si existen librerías externas que puedan facilitar la elaboración de la aplicación.

Pruebas

En la etapa de pruebas se elaborará una serie de pruebas para asegurar el correcto funcionamiento de la aplicación, donde tendremos:

- Pruebas Funcionales: Se elaborará una lista de funcionalidades basadas en los Requisitos Funcionales anteriormente definidos para asegurar que el comportamiento es el correcto.
- Pruebas de Usuario: Se realizará una serie de pruebas con usuarios finales para detectar errores que no pudieron ser detectados en las Pruebas Funcionales.

Entrega Final:

- Poner en práctica los conocimientos adquiridos a lo largo de todo el máster.
- Adquirir experiencia en afrontar los retos que supone llevar a cabo un proyecto completo.
- Ser capaz de documentar y justificar las decisiones tomadas en el desarrollo y los resultados obtenidos.
- Ser capaz de presentar el trabajo realizado a un público no especializado.

Las etapas en el diagrama de Gantt [6] siguen unas fechas las cuales se pueden solapar en unos casos, y en otros no.

En el caso de la etapa de diseño, podemos observar que se realizar un solapamiento con la implementación en la fase de la creación del prototipo de alto nivel. En este caso, se realizará un prototipo utilizando Mockups [7], pero también se realizará una primera versión a muy alto nivel de la aplicación, la cual será utilizada para la etapa del desarrollo de la aplicación, donde a su vez es requerido que el servidor esté mínimamente operativo.

También, en la etapa de la implementación, las etapas de desarrollo y pruebas se solaparán durante todo el desarrollo, ya que cada vez que se añada una nueva funcionalidad se requerirá hacer pruebas funcionales para asegurarnos de que esa parte está libre de errores en buena medida.

La última etapa, la Entrega Final, debería corresponder a una etapa de entrega, realización de Vídeos y presentaciones del producto, por lo que las etapas de Diseño e Implementación deberían haber sido concluidas totalmente.

1.5 Breve sumario de productos obtenidos

Los productos obtenidos al final de este trabajo serán:

- Aplicación Móvil para dispositivos iOS (iPhone, iPad).
- 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:

- Diseño
- Implementación

Dentro del diseño, se incluyen diversos diagramas que sirven de ayuda para la construcción del producto, entre los cuales tendremos un árbol de navegación, un prototipo hecho con wireframes, diagrama de casos de uso, y de arquitectura, tanto de la base de datos como de las entidades y clases.

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.

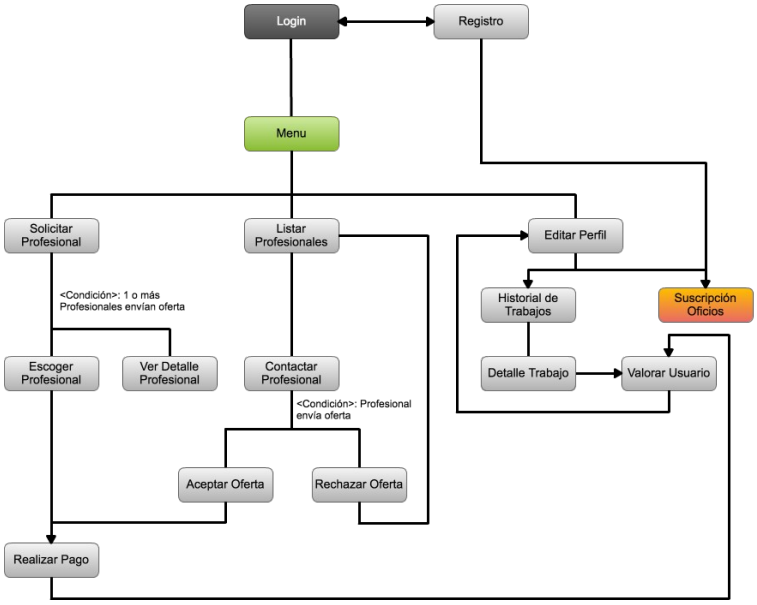
Por otra parte, el usuario también debe disponer de la posibilidad de ver un listado de profesionales cercanos para poder contactar y plantear dudas sin ningún tipo de compromiso, y desde ahí tener la posibilidad de llegar a acuerdos.

2.2 Diseño Conceptual

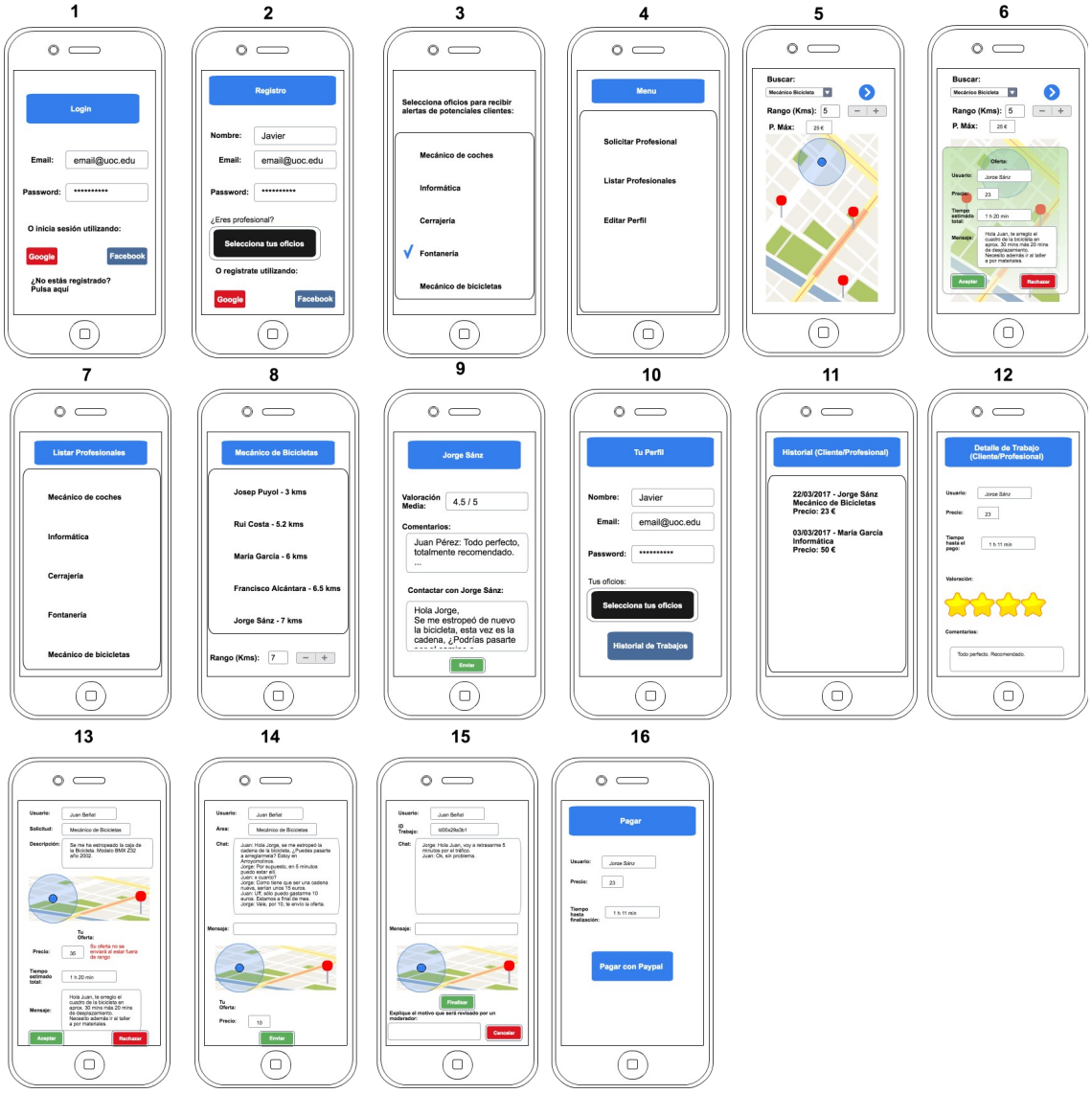
Para el diseño conceptual se ha realizado un árbol de navegación donde se transita entre diferentes estados, que están representados en diferentes colores según su tipo:

- 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.

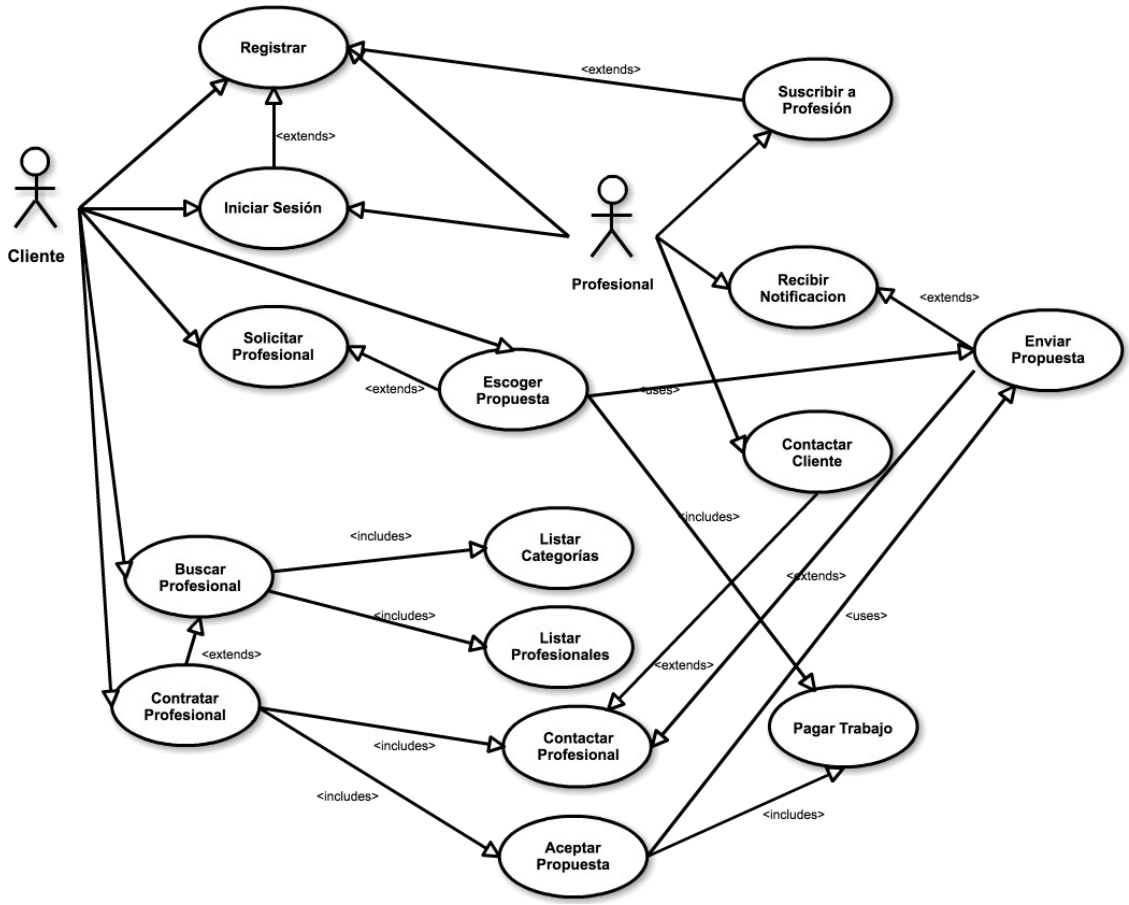
Esto ha ayudado a corregir en primer lugar, la falta de botones y otros elementos de la Interfaz necesarios en ciertas pantallas del prototipo. Citando algunos ejemplos:

- 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:

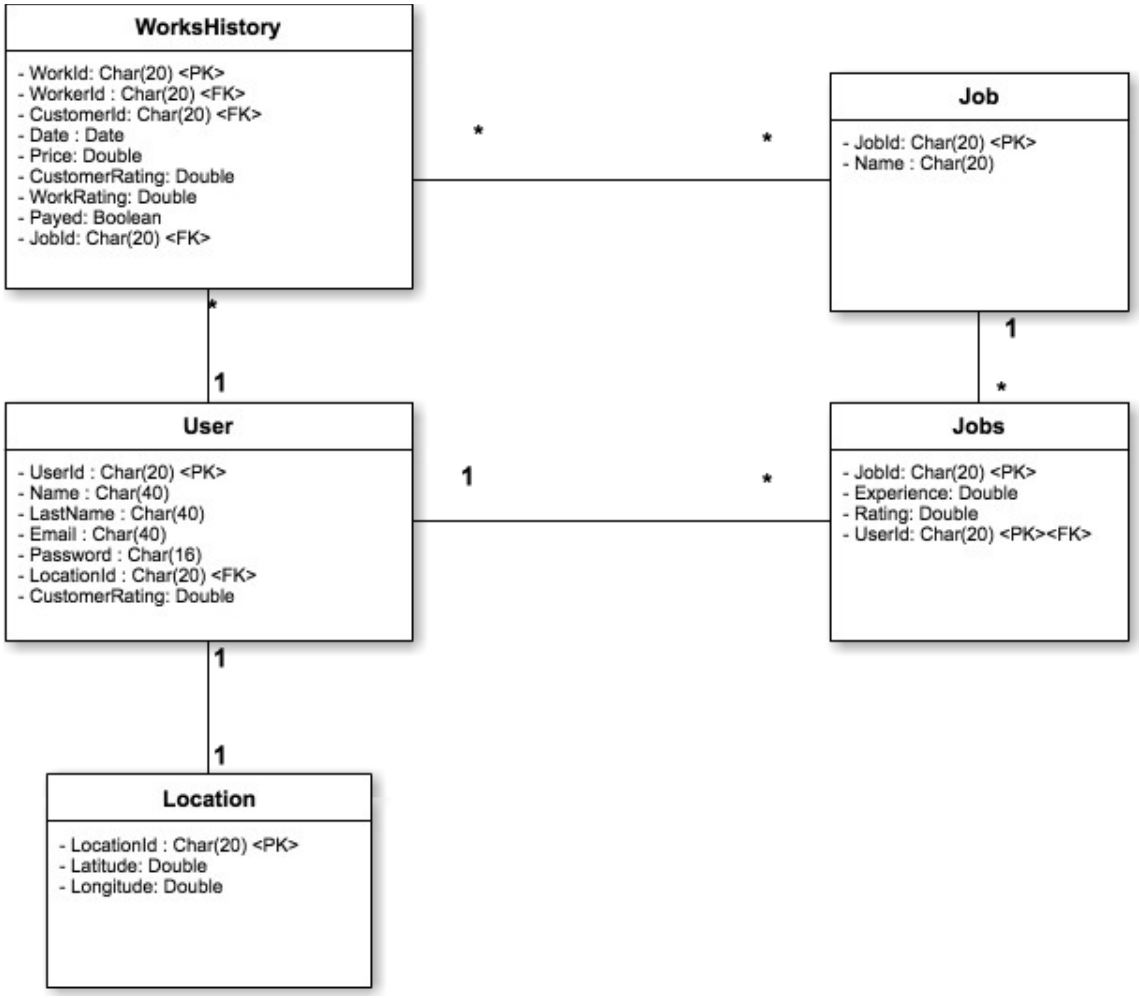
- CU001 – Registrar
  - Actores: Cliente, Profesional.
  - Precondiciones: Ninguna.
  - Postcondiciones: Ninguna.
  - Descripción: Permite al Usuario registrarse en la aplicación mediante el uso de email, contraseña y datos personales, o una cuenta de Google o Facebook.
- CU002 - Iniciar Sesión
  - Actores: 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:

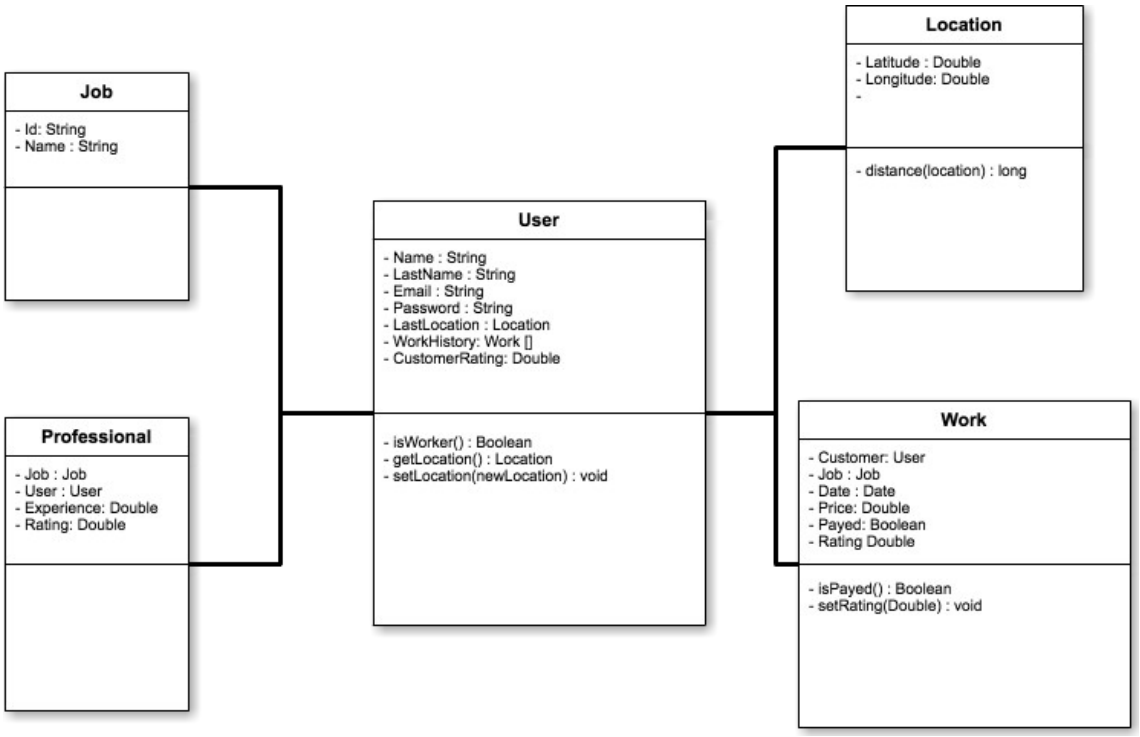
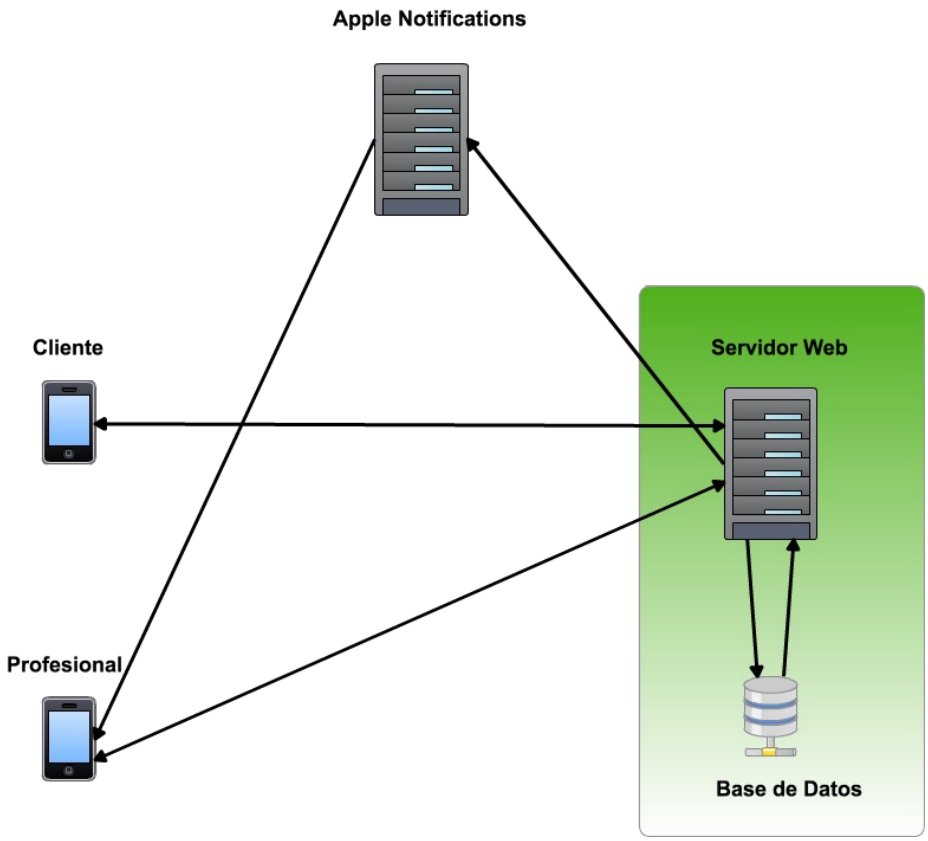
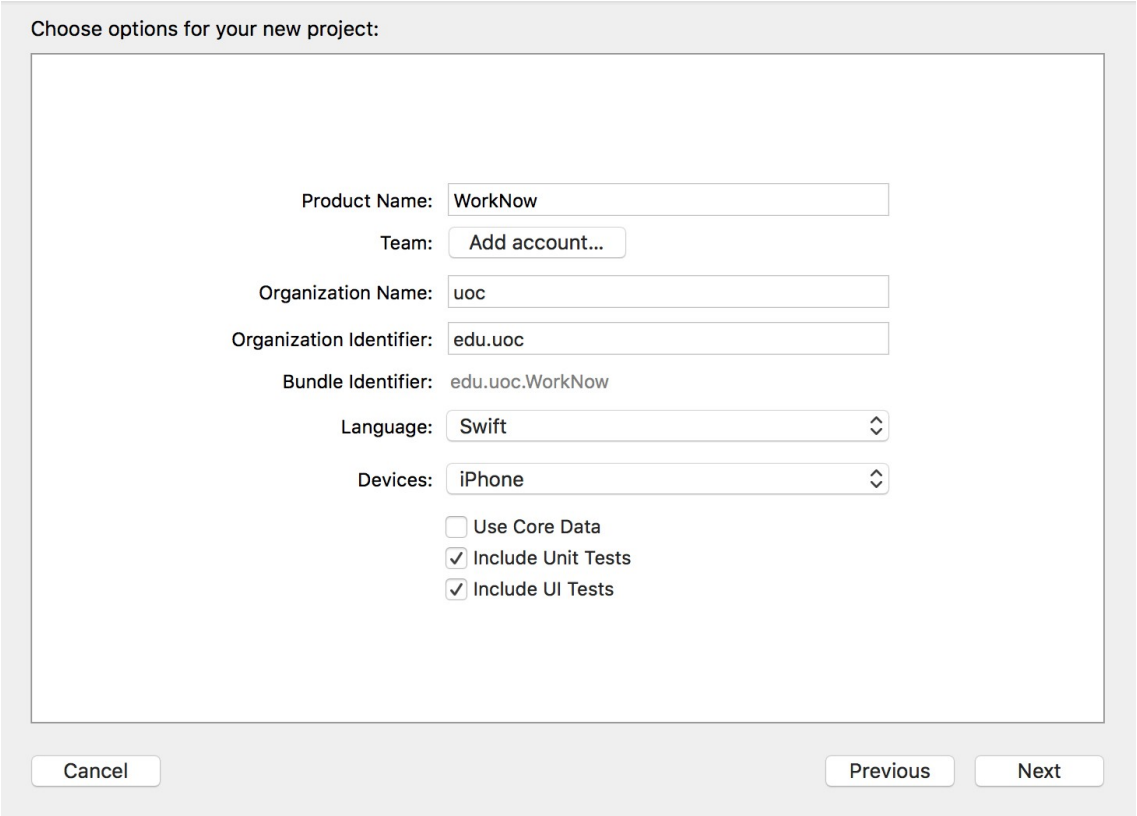


Diagrama de Arquitectura del Sistema:



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 nuestro PodFile:

```
pod ‘JSQMessagesViewController’
```

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

```
pod install
```

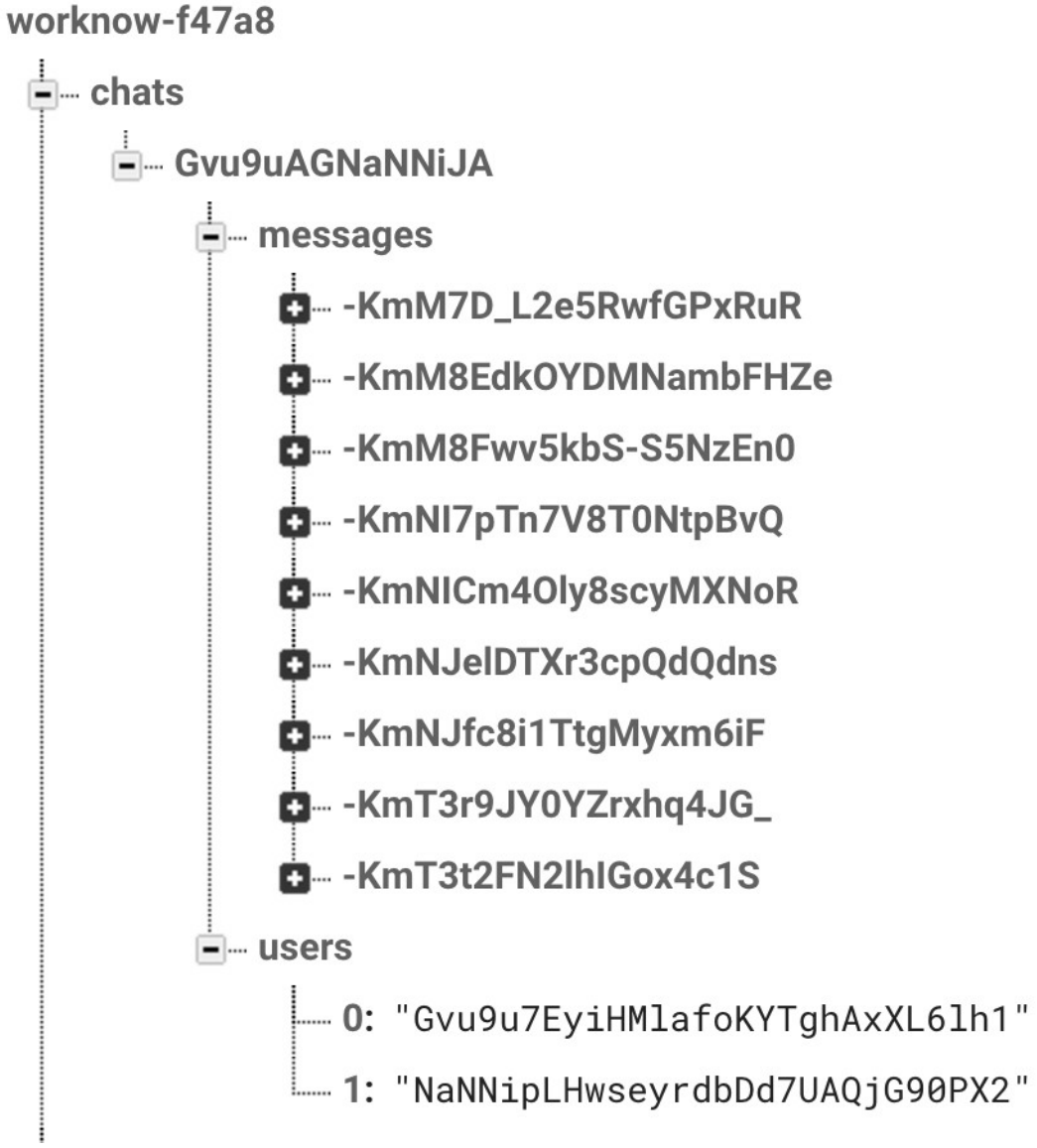
Un elemento importante a destacar de la implementación del ViewController que hereda del incluído en la librería es la estructura del chat en Firebase. Este se guarda en la base de datos como un objeto JSON al igual que los Usuarios, Oficios y otros objetos. Todos los chats derivan se guardan bajo la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

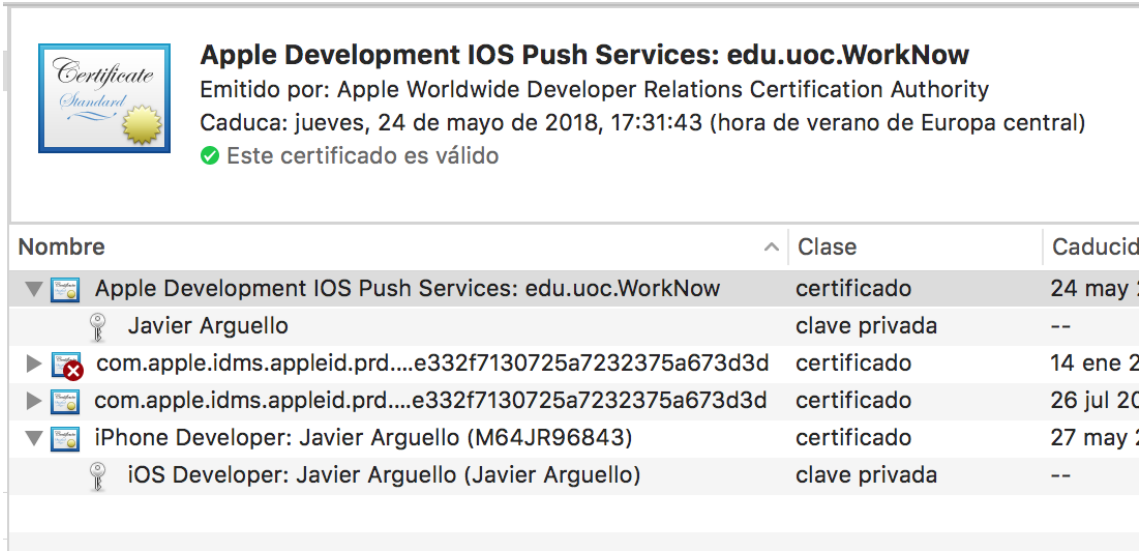
Por último, comentar que al ser un ViewController de un tercero, la interfaz gráfica requiere ser adaptada a la nuestra desde código, donde se ha cambiado el fondo, se han eliminado botones no necesarios en esta aplicación, y otros cambios para adecuar el diseño al resto de la aplicación.

3.1.3 Notificaciones Push iOS mediante Firebase

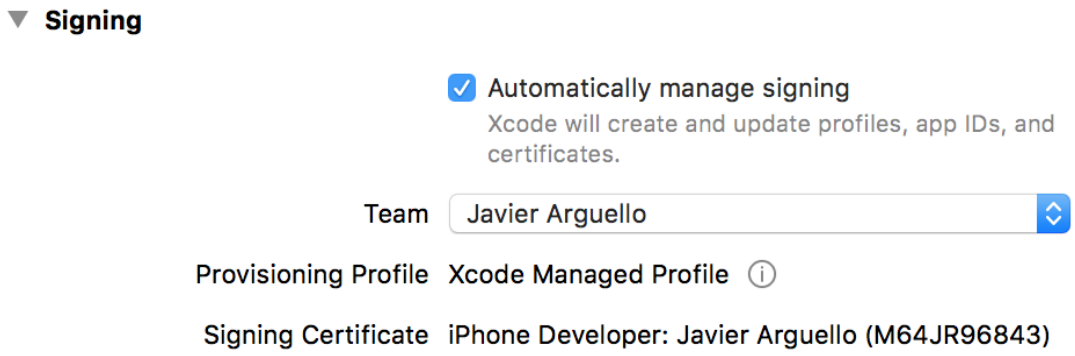
La parte más costosa a nivel de implementación ha sido sin duda el uso de las notificaciones push.

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

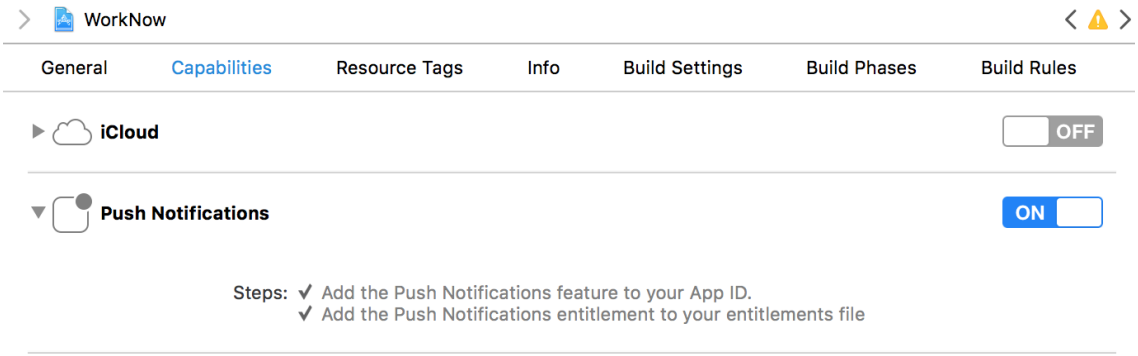
Después de completar el registro como desarrollador de Apple, se requiere crear un certificado de desarrollador iOS (para poder utilizarlo en dispositivos) y otro para notificaciones push de iOS.



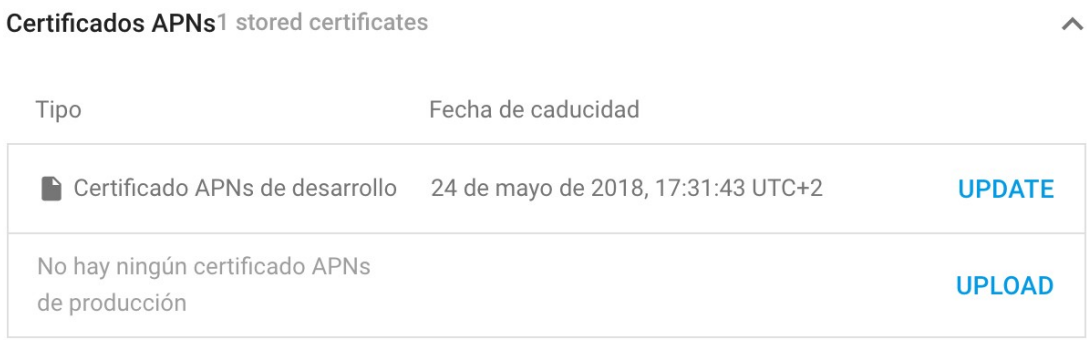
Ambos certificados han de estar instalados en el Administrador de Llaveros de nuestro Mac para habilitar las notificaciones. Además, el certificado de desarrollador de iPhone debe estar incluído en el proyecto de Xcode en **Proyecto > General > Signing.**



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.

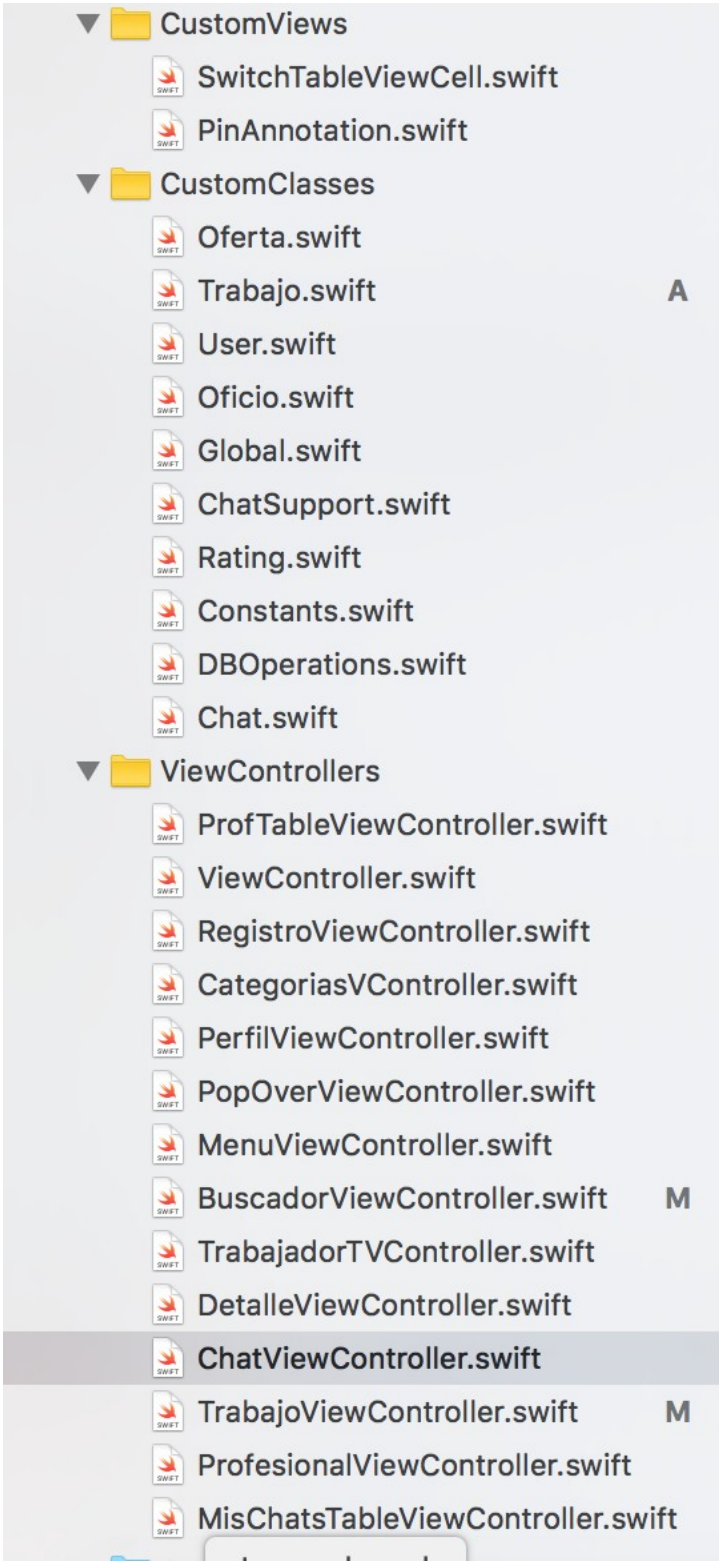
3.2 Organización del proyecto XCode

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:

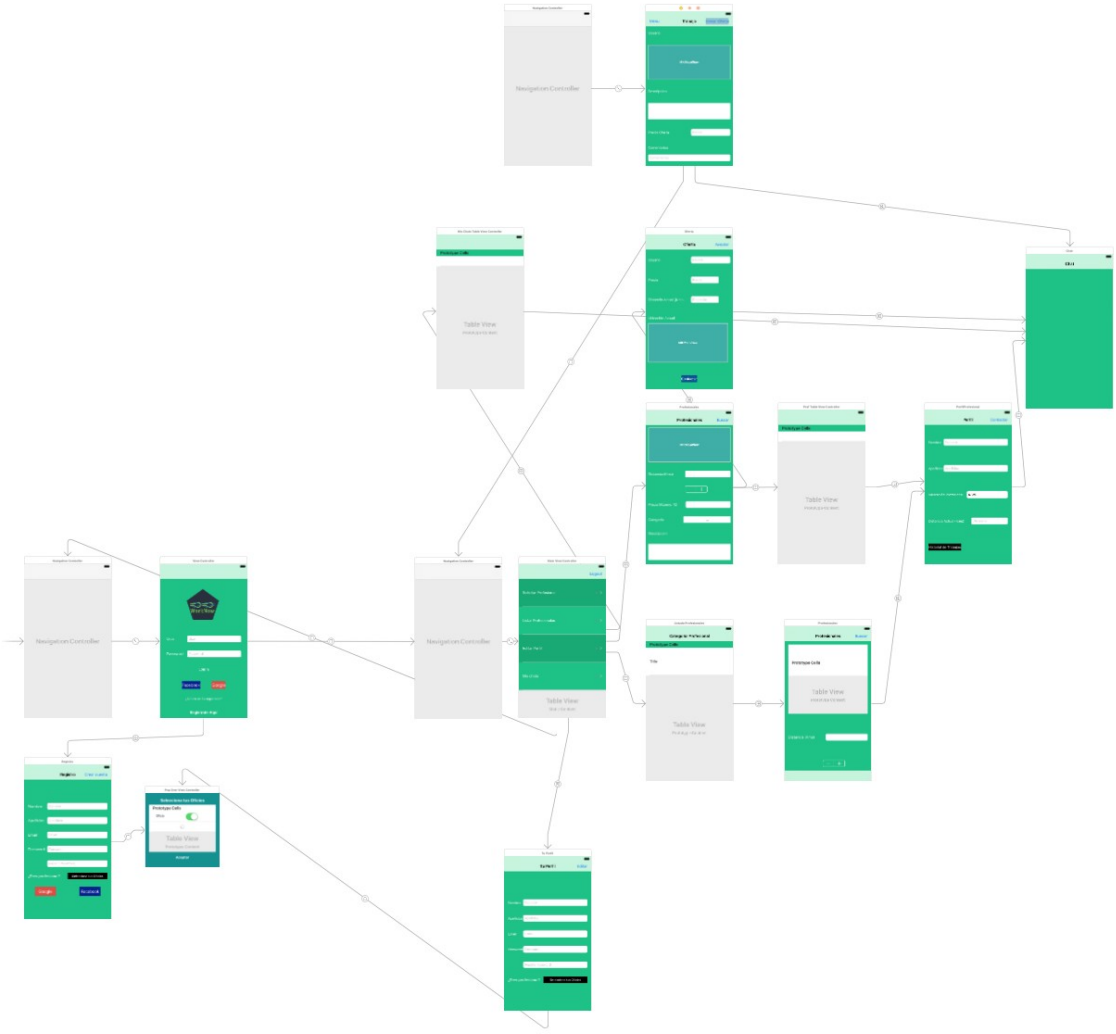
- CustomViews: Son controladores de vistas simples personalizadas para una funcionalidad muy concreta. En esta etapa del proyecto sólo se ha

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.



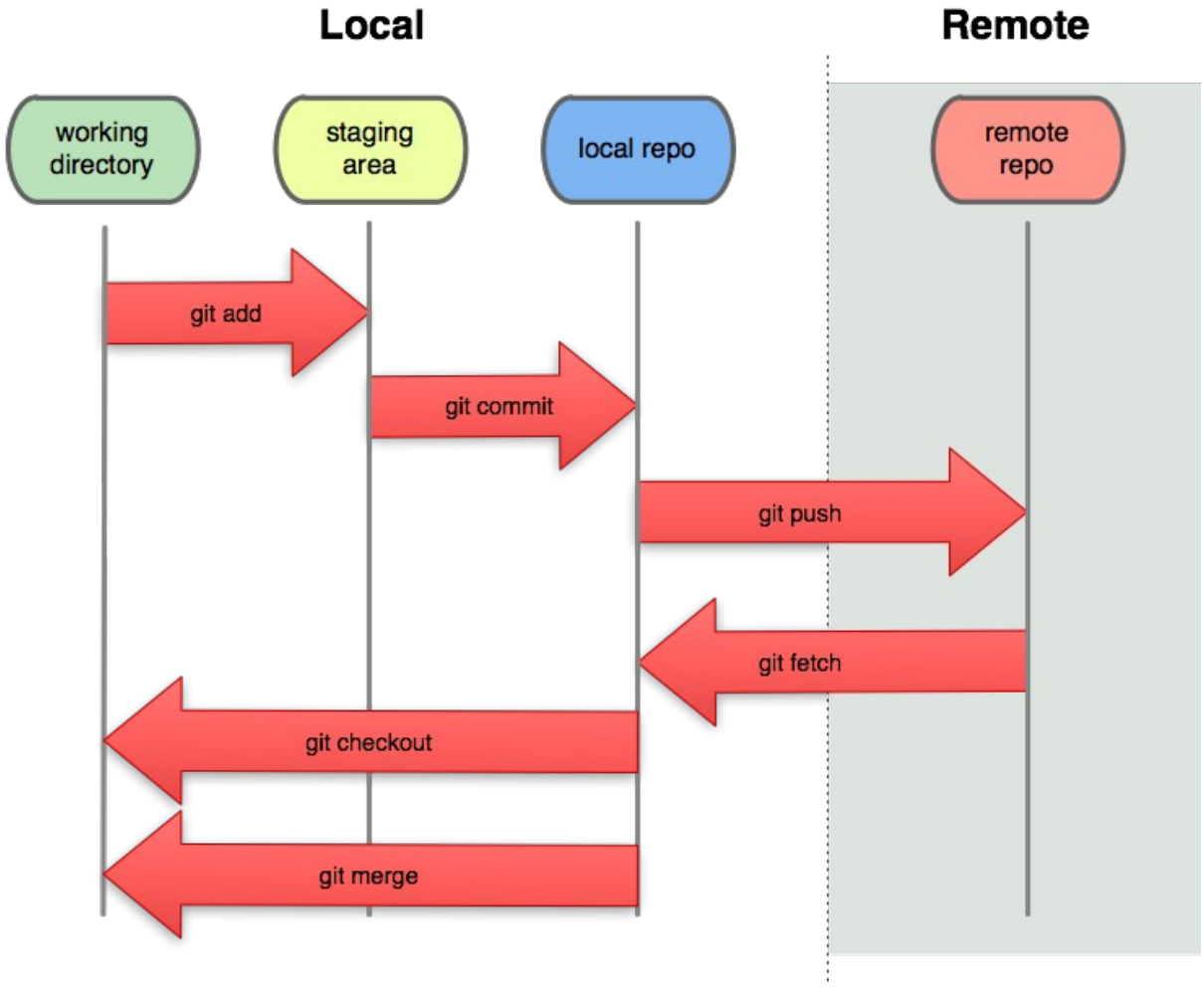
3.3 Git

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

Para el desarrollo de este proyecto se ha realizado una batería de pruebas funcionales que cubren los aspectos fundamentales de la aplicación. Un resumen de las pruebas realizadas es el siguiente:

Login:

- No se permiten campos vacíos.
- No se permite un password de menos de 8 caracteres.
- No se permite un formato de email incorrecto.
- Comunicación al usuario de los errores en el servidor.
- Logado correcto.

Registro:

- No se permiten campos vacíos.
- No se permite email y contraseña con formato incorrecto.
- Se nos muestra un Pop up cuando queremos seleccionar oficios.
- Se permite seleccionar oficios.
- Se nos reportan errores de registro.
- Se comunica el registro correcto en la aplicación.

Menu:

- Se muestran las 4 secciones y se permite navegar en ellas.

Mis Chats:

- Se muestran todos los chats que se tuvieron anteriormente con usuarios
- Se permite acceder a los chats.

Perfil:

- No se permite cambiar email, nombre y apellidos.
- Se permite cambiar contraseña.
- Se permite cambiar los oficios a los que se está suscrito.

Listado de Profesionales:

- Se muestran todas las categorías posibles.
- Se permite al usuario buscar a usuarios en un rango de 1-100 Kms.
- Se muestra una alerta avisando de que no se encontraron usuarios.
- Se muestran los usuarios en una tabla en caso de encontrarse.
- Se permite acceder al perfil de un usuario en concreto.

Perfil Profesional:

- Se muestran todos los campos del usuario
- Se permite contactar con el usuario.

Chat

- Se permite iniciar una sesión de chat.
- Se reciben los mensajes en tiempo real en cada extremo.

Buscar usuarios

- Se permite navegar por el mapa.
- Se permite seleccionar categoría
- Se permite indicar un precio.
- Se indica errores en el precio o en la categoría.

Notificaciones:

- Se reciben notificaciones en caso de estar suscritos a la categoría profesional.
- Se reciben notificaciones en caso de estar en la vista de “Buscar usuarios” si recibimos una oferta.

4. Conclusiones

4.1 Conclusiones del trabajo

El Trabajo Final de Master que nos ocupa ha sido bonito, ya que ha permitido poner en práctica muchos de los conceptos que se han aprendido en este Master de Desarrollo de Aplicaciones Móviles en su conjunto, realizando un proyecto desde su planificación, hasta su implementación y pruebas, teniendo que realizar el diseño de por medio.

El trabajo me ha permitido aprender además de realizar una planificación de un proyecto, al uso de nuevos conceptos de programación iOS no utilizados anteriormente, como puede ser el uso de notificaciones Push, que permite la comunicación en tiempo real entre dos o más dispositivos.

También he podido aprender lo importante que es realizar y seguir una buena planificación para entregar a tiempo y con calidad un trabajo de esta envergadura.

4.2 Resultados de los Objetivos del trabajo

Los objetivos del trabajo se han conseguido en parte. El proyecto es ambicioso y quizás en el tiempo que se ha tenido no se ha podido llevar a cabo todo el trabajo.

Lo más importante se ha conseguido, que es realizar una aplicación móvil que permita obtener localizaciones de usuarios, comunicación con un servidor, y la intercomunicación en tiempo real entre dispositivos mediante el uso de Chat y Notificaciones.

Hay trabajo sin concluir que debería haberse incluído en este trabajo, como puede ser el historial de trabajos, la conclusión de un trabajo entre cliente y trabajador, valoración de usuarios, etcétera. El principal motivo para no poder concluir ese trabajo ha sido básicamente el tiempo, como supongo que le habrá ocurrido a otros alumnos del Master en cierta medida.

Otro trabajo que no se realizó fue el servidor, que estaba planificado, pero no se pudo terminar a tiempo, y está en estos momentos aparcado para continuarlo en un futuro.

4.3 Seguimiento de la planificación

El proyecto se retrasó en ciertas ocasiones. Gasté demasiado tiempo en desarrollar un servidor que se quedó incompleto, y me obligó a cambiar de idea y utilizar Firebase para poder realizar lo principal, que es una aplicación iOS.

Quizás se tendría que haber evaluado el número de horas diarias que se podía emplear en el Trabajo para poder estudiar si era posible realizar el servidor en este tiempo y poder emplear más tiempo en el desarrollo puro de la aplicación.

4.4 Líneas de Trabajo futuro

Entre las líneas de trabajo futuro se incluyen las siguientes:

- Continuar con las secciones incompletas de la aplicación entre las que se incluyen:
    - Historial de Trabajos.
    - Valoración de Usuarios.
    - Notificaciones Push para mensajes de Chat (actualmente no se avisa de nuevos mensajes en las conversaciones de Chat).
- Estudiar APIs para realizar pagos a través de la aplicación.
- Cambio y mejora del diseño de la aplicación.
- Desarrollo completo de un servidor web propio para poder
- Subir la aplicación a App Store de Apple cuando la aplicación sea más estable
- Estudiar la monetización de la aplicación.

5. Glosario

Aplicación en demanda: Del inglés ‘On demand’. Son un tipo de aplicaciones informáticas, más concretamente para dispositivos móviles, que nos permiten realizar peticiones de servicios en cualquier momento y lugar, poniendo en contacto a cliente y trabajador que más concuerdan en estas condiciones.

6. Bibliografía

[1] Top 2015 de aplicaciones del tipo 'On Demand' - https://www.blowltd.com/19-on-demand-apps/ - Marzo 2017
[2] Páginas Amarillas España - https://www.paginasamarillas.es/ - Marzo 2017
[3] QDQ - http://es.qdq.com/ - Marzo 2017
[4] Google Business - https://www.google.com/business/
[5] Swift – Apple Developer – http://developer.apple.com/swift - Marzo 2017
[6] Diagrama de Gantt (Wikipedia) - https://es.wikipedia.org/wiki/Diagrama_de_Gantt - Marzo 2017
[7] 9 Design Mockup Tools - http://mashable.com/2012/06/07/mockup-tools/#Cl5iCAkx3Zqk - Marzo 2017
[8] Firebase - https://firebase.google.com/
[9] JSON - http://www.json.org/json-es.html
[10] Firebase Tutorial Chat - https://www.raywenderlich.com/140836/firebase-tutorial-real-time-chat-2
[11] Apple Developers – http://developer.apple.com

6. Anexos