Cadena de favores

Sergio Morales Machado

Resumen—La finalidad de esta aplicación es el intercambio de habilidades entre usuarios, por ejemplo: pintar; limpiar; servicio de fontanería, etc. Estos servicios serán favores, por lo tanto, totalmente gratuitos entre usuarios. Un usuario podrá hacer una petición donde especificará su necesidad y los días en los que está disponible para que se le pueda ayudar. El servicio cognitivo de la aplicación interpretará los requerimientos y hará una consulta a una base de datos donde se guardarán todos los usuarios existentes con sus correspondientes disponibilidades y aptitudes. Durante esta consulta se podrán ver los usuarios disponibles con la necesidad requerida. Una vez finalizado el servicio el usuario dejará una reseña que el sistema evaluará poniendo estrellas.

Palabras clave—Favores, ayuda, servicios, aplicación móvil

Abstract—The purpose of this application is the exchange of skills between users, for example: painting; clean; plumbing service, etc. These services will be favors, therefore, totally free among users. A user will be able to make a request where he will specify his need and the days when it is available so that he can be helped. The cognitive service of the application will interpret the requirements and make a query to a database where all existing users will be saved with their corresponding availability and skills. During this consultation, users available with the required need will be seen. Once the service is finished, the user will leave a review that the system will evaluate by putting stars

Key Words —Exchange, help, services, Mobile app

1 Introducción

Durante mis prácticas externas de la universidad amplié mis conocimientos sobre las aplicaciones Web-Api cosa que me pareció muy interesante y he querido hacer una aplicación con lo aprendido que ayudara a la sociedad.

Actualmente en muestra sociedad a pesar de haber mucha variedad de profesiones no todos sabemos hacer de todo. Cada persona tiene habilidades distintas que pueden ser tecnológicas como de cualquier otro ámbito. Hay personas que saben de ordenadores, otros cocinar, temas de fontanería, electrónica, etc… Con esta aplicación una comunidad puede ayudarse entre sí.

2 Estado del arte

Wallapop es una aplicación muy usada donde un usuario vende o intercambia un producto usado, para ello crea un anuncio donde detalla qué vende, el precio, su descripción... Otros usuarios entran en la aplicación y pueden ver un tablón de anuncios con todas las publicaciones de venta o intercambio de otros usuarios cercanos a su ubicación. De esta manera la persona interesada en un producto puede abrir chat y acordar un encuentro para la compra-venta del producto en cuestión.

Otra de las aplicaciones más conocidas en el ámbito de la compra-venta es *Milanuncios*, esta es una aplicación muy similar a *Wallapop* en la que se pueden publicar ofertas de artículos que están en venta y otros usuarios contactarán para realizar el intercambio. Al igual que *Wallapop* tiene la ventaja de poder guardar una lista de deseos para los anuncios que más gusten a un usuario.

Tienes Sal es otra aplicación algo distinta a las dos anteriores, con esta un usuario se registra en su área vecinal, hay un proceso de verificación para asegurarse que se sea vecino de ese barrio y así tener acceso a eventos e intercambios dentro de la misma comunidad. Estos eventos no son solo la compra-venta del artículo sino que también son quedadas para conocer más a los vecinos y entablar amistades mediante distintas actividades.

3 Objetivos

En esta sección se definirán los objetivos, principales y secundarios, y los requisitos de la aplicación.

3.1 Objetivos principales

Se considerarán objetivos críticos aquellos que sean estrictamente necesarios para el mínimo funcionamiento de la aplicación y los relacionados con los servicios cognitivos.

- *E-mail de contacto: sergio.moralesma@e-campus.uab.cat*
- *Mención realizada: Computación.*
- *Trabajo tutorizado por: Yolanda Benítez (Dpto. Ciencias de la Computación)*
- *Curso 2019/2020*

1. Ayudar a usuarios cercanos a interactuar para hacer favores.
2. Crear un diseño intuitivo y sencillo de usar: para que los usuarios no tengan dificultades en usar la aplicación.
3. Encontrar usuarios aptos para las peticiones con una precisión de un 90% de acierto.
4. Obtener los datos necesarios para poder hacer reseñas de manera completa: para que otros usuarios puedan ver el perfil de la persona a la que van a ayudar o va a prestar su ayuda.
5. Gestionar y modificar una Base de Datos: para llevar un registro de los servicios, usuarios, reseñas.

3.1 Objetivos secundarios

Se considerarán objetivos secundarios los relacionados con la rama de Software, ya que este es un proyecto que se centra más en la parte de ingeniería de la computación.

1. Las peticiones se harán vía audio: para una mayor comodidad del usuario.
2. Que el usuario sea capaz de gestionar y modificar su perfil propio de un usuario: para modificar las propias disponibilidades y habilidades.
3. Mostrar los datos de los usuarios aptos de manera más gráfica para aumentar la efectividad y velocidad de selección en un 50%.
4. Registrar usuarios reales.

4 Planificación

Esta es la planificación del proyecto a alto nivel

Actividad	Inicio	Final
Módulo de carga de la base de datos	1 marzo	15 marzo
Módulo encargado de gestionar el servicio cognitivo de búsqueda de usuario	16 marzo	31 marzo
Módulo encargado de gestionar la reseña	1 abril	15 abril
Tests	16 abril	20 abril
Primera versión de la aplicación	1 mayo	2 mayo
Corrección de errores de la primera versión	2 mayo	15 mayo
Segunda versión	16 mayo	17 mayo
Corrección de errores	18 mayo	31 mayo
Documentación	1 marzo	1 julio

Tabla 1: Planificación

5 Metodología

El desarrollo de aplicaciones móvil se suele dar con metodologías ágiles, pero en este caso dado que el equipo está desarrollado por una sola persona, que es la misma que el *product owner* y que el *project manager*, no sería correcto decir que se usará una metodología ágil como tal.

La metodología utilizada será una adaptación entre metodología ágil y metodología tradicional.

Se utilizarán algunas herramientas, en este caso *Trello*, para definir conjuntos de tareas como *tareas a empezar*, *en proceso, finalizadas*, *a testear* y *revisadas* para tener una mayor organización.

En la *Fig 1* se puede ver un ejemplo del estado de Trello a mitad de proyecto

Fig 1: Trello a mitad de proyecto

6 Requisitos

En este apartado se explicarán los requisitos funcionales y no funcionales.

6.1 Requisitos funcionales

Los requisitos funcionales son los comportamientos que ha de tener la aplicación ante distintas situaciones.

6.1.1 Hacer petición

En este apartado se explican los requisitos de la parte del servicio cognitivo de creación de la petición.

- RF1: El sistema detectará el servicio pedido.
- RF2: El sistema detectará la fecha pedida.

- RF3: El sistema buscará usuarios que cumplan las características pedidas.
- RF4: El sistema controlará las palabras similares por si no están bien escritas.
- RF6: Las peticiones se almacenarán en la Base de Datos del sistema para gestionarlas posteriormente.
- RF7: Solo reconocerá servicios que estén en la base de datos anteriormente.
- RF8: El usuario ha de estar registrado y con la sesión iniciada para poder hacer una petición.

6.1.2 Crear reseña automática

En este apartado se explican los requisitos de la parte del servicio cognitivo de creación de la reseña.

- RF9: Se introducirá input de texto.
- RF10: El sistema reconocerá los inputs del usuario para orientar la encuesta.
- RF11: Se dará una reseña en estrellas respecto a las palabras escritas.
- RF12: Se reconocerán solo caracteres en inglés.
- RF13: El usuario ha de estar registrado y con la sesión iniciada para poder crear una reseña.

6.1.3 Iniciar sesión

En este apartado se explican los requisitos de la parte del inicio de sesión.

- RF14: El campo usuario será único y aceptará caracteres alfanuméricos y símbolos especiales.
- RF15: El campo contraseña será alfanumérico.
- RF16: Todos los campos serán requeridos para iniciar sesión.
- RF17: Si no se introducen los campos bien serán pedidos otra vez.
- RF18: El botón iniciar sesión te llevará a la pantalla inicial si las credenciales son correctas.
- RF19: El botón borrar dejará en blanco los campos usuario y contraseña.

6.1.4 Registrarse

En este apartado se explican los requisitos de la parte del registro.

- RF20: El campo usuario será único y aceptará caracteres alfanuméricos y símbolos especiales.
- RF21: El campo contraseña será alfanumérico.
- RF22: La contraseña se encriptará en la base de datos.
- RF23: El campo email deberá ser un email existente.
- RF24: El campo ciudad serán caracteres numérico.
- RF25: El campo teléfono será numérico.
- RF26: Todos los campos serán obligatorios para registrarse
- RF27: Si no se introducen los campos correctamente se mostrará un mensaje de error.
- RF28: El botón iniciar sesión te llevará a la pantalla inicial si las credenciales son correctas.
- RF29: El botón borrar dejará en blanco todos los campos.

6.2 Requisitos no funcionales

Los requisitos funcionales son aquellos que definen características del comportamiento de la aplicación:

- RNF1: Responsive.
- RNF2: Intuitivo.
- RNF3: El sistema proporcionará mensajes informativos de las acciones que se lleven a cabo.
- RNF4: Los datos serán actualizados en tiempo real

7 DISEÑO

Está aplicación está pensada para ser desarrollada en Android, aunque puede ser desarrollado en otro sistema operativo como *iOS* en un futuro. Por eso se separa el código en módulos con partes reutilizables.

La aplicación será un servicio web api junto a dos servicios cognitivos y constará de diversas pantallas.

7.1 Servicio cognitivo de búsqueda de usuario

Este servicio recibirá por parámetro una cadena del tipo: "Se necesita un fontanero para el lunes a mediodía" y devolverá los valores necesarios para hacer la búsqueda del usuario con dicha habilidad y su disponibilidad, siguiendo el ejemplo anterior: "fontanero" y "lunes mediodía".

7.2 Servicio cognitivo de creación de reseña

Este servicio creará a través del texto del usuario, una reseña que el sistema analizará y le dará un valor numérico del 0 al 5.

7.3 Pantallas

Las pantallas de las que será compuesta la aplicación serán:

1. Página principal: Esta es la página que se abrirá al encender la aplicación, desde aquí se podrá iniciar sesión, registrarse, ver la información de la aplicación o salir.
2. Inicio de sesión: Introducirás tus credenciales (nombre de usuario y contraseña) para acceder a tu sesión. Si las credenciales son correctas te llevará a una nueva página principal, en caso contrario se volverán a solicitar.

3. Registro: Será la página de registro de un nuevo usuario, se introducirá el nombre de usuario, contraseña, nombre real, apellidos, teléfono, y habilidades.
4. Información: Aquí se podrá ver la información de la aplicación y su funcionamiento.
5. Página principal una vez iniciada sesión: una vez iniciada sesión hay varias opciones: hacer una nueva petición, ver el perfil, ver la información de la aplicación y salir.
6. Crear petición: en esta pantalla habrá un campo de texto donde se creará la petición indicando la especialidad requerida y la disponibilidad necesitada, por ejemplo: "Se necesita un fontanero para el lunes a mediodía". También habrá dos botones: uno para limpiar la búsqueda y otro para hacerla.
7. Resultados de la búsqueda: Se mostrarán los usuarios coincidentes con la búsqueda anterior.
8. Perfil: se podrán ver los datos introducidos del propio perfil.
9. Confirmación de usuario: al clicar en un resultado de la búsqueda de usuario aparecerá esta pantalla para ver más detalles del usuario seleccionado.
10. Registros (histórico): para controlar los servicios aceptados, pendientes, rechazados y acabados.

8 DESARROLLO

En el apartado de desarrollo, se definirá un poco las herramientas usadas para el desarrollo general y más adelante el desarrollo individual de cada uno de los módulos del servicio cognitivo de búsqueda, de reseña y de la aplicación front-end y back-end.

8.1 Desarrollo general

Para el entorno se ha creado un proyecto en *Visual Studio 2019*, se ha usado el framework *Xamarin* que usa el lenguaje *C#* para la aplicación Android. Para la visualización y estilo de las pantallas se ha *usado xml (Application Markup Language)*. El servicio cognitivo de búsqueda de usuarios está totalmente desarrollado en *C#*, en cambio el servicio de creación de reseñas está parcialmente desarrollado en *python* y en *C#*.

El patrón de diseño de la aplicación es *MVVM (Model-View-ViewModel*) que consiste en desacoplar el máximo de interfaz de usuario para que se pueda actualizar a otros sistemas en un futuro, como por ejemplo a *iOS*.

La base de datos utilizada es *Firebase DataBase en tiempo real*, esta base de datos es no-relacional y guarda los datos en forma de árbol en formato *JSON*, se actualiza a tiempo real para todos los usuarios conectados en ese momento a la plataforma. Esta última es la principal razón de utilización de esta base de datos ya que la idea principal era usar SQLite, una extensión de Visual Studio que creaba una base de datos local, pero daba muchos problemas respecto al tiempo de actualización de los datos para todos los usuarios.

8.2 Desarrollo servicio cognitivo de búsqueda

Para hacer el servicio cognitivo de búsqueda de usuarios se plantearon estos pasos:

1. Obtener el texto del servicio.
2. Obtener el nombre del servicio.
3. Obtener la fecha del servicio.
4. Obtener respuesta completa.

En el primer paso se obtiene la petición escrita por el usuario.

Para la obtención el nombre del servicio hay una estructura con todos los servicios disponibles en la base de datos y se hace un porcentaje de similitud entre lo introducido por el usuario y el texto de la estructura almacenada. El porcentaje de similitud sirve para reconocer palabras similares que han sido escritas de forma incorrecta o sin alguna tilde.

Seguidamente obtenemos la disponibilidad. Para las fechas de disponibilidad se busca en 3 bloques: el periodo del día: tarde, mañana, noche,…; el día de la semana: lunes, martes, miércoles, etc., y finalmente por el número de día del mes. Una vez encontramos uno de los 3 bloques en la lista de palabras lo guardamos en una variable.

El último paso lo conseguimos juntando los resultados obtenidos de los dos anteriores.

El pseudocódigo de la parte principal queda así:

```
Foreach (frase escrita)
{
  Foreach (frase.split(" "))
  {
    servicio = ObtencionNombreServicio(palabra)
    fecha = ObtenerFechaServicio(palabra)
  }
}
Objeto respuesta  = { servicio = servicio, fecha = fecha}
```

Para las funciones de semejanza comentadas anteriormente se utiliza este otro pseudocódigo:

```
Float Similaridad(palabra1, palabra2)
{
  Foreach (cada carácter palabraMasLarga)
  {
    If(caracteres son iguales)
    {
      Aciertos++;
    }
  }
  return Aciertos / LongitudPalabramasLarga
}
```

Se utiliza la palabra más larga como iteración del for porque suele ser la más completa y que contiene más información.

Y por último el módulo obtener la fecha:

```
ObtenerFechaServicio(frase.Split(" "))
{
  Disponibilidad[3]: string;
  Bool dia = false, numero = false, periodoDia = false;
  Foreach (palabra)
  {
    If(¡dia)
    {
      If(ComprobarPalabra(palabra, listaDiasSemana))
      {
        Dia = true;
        Disponibilidad[0] = palabra
      }
    }
    If(¡numero)
    {
      If(ComprobarPalabra(numero, listaNumerosMes))
      {
        Numero = true;
        Disponibilidad[1] = palabra
      }
    }
    If(¡periodoDia)
      {
      If(ComprobarPalabra(numero, listaPeriodosDia))
      {
        periodoDia = true;
        Disponibilidad[2] = palabra
      }
    }
}
```

En la función ComprobarPalabra(palabra, lista), el término lista depende del módulo, por ejemplo si es día, irá del 1 al 31, si es día semana de lunes a domingo y si es periodo del día será mañana, mediodía, tarde, noche.

8.3 Desarrollo servicio cognitivo de reseña

Para la creación de este servicio se siguen estos pasos:

1. Obtención de los datos clasificados.
2. Clasificación, separación y criba de cada palabra.
3. Análisis de un nuevo texto.

Para el desarrollo de este servicio se ha buscado una base de datos, ya existente, con poco más de un millón *Tweets*, ya clasificados, donde se indica si tienen una connotación positiva o negativa.

Primero se separan todos los *tweets* en palabras y posteriormente se crea un diccionario con todas las palabras existentes. Al recorrer todos los *tweets* se mira si las palabras existen o no en el diccionario, en caso de que no se añade, en caso de que sí se incrementa en uno el valor de la connotación (negativa o positiva) dependiendo de la clasificación de la palabra.

El pseudocódigo de la creación del diccionario queda así:

```
Foreach (todos los tweets)
{
  Foreach (tweet.split(" "))
  {
    If(palabra existe en dict)
    {
      dict[palabra][label] = dict[palabra][label] +1
    }
    Else
    {
      dict.update(palabra: { negative: 0, positive: 0})
    }
  }
}
```

Algunos ejemplos del diccionario aparecen en la *Fig 2:*

11	dict	3	{'negative':1011, 'positive':779, 'total_count':1790}
2	dict	3	{'negative':21486, 'positive':15557, 'total_count':37043}
30mins	dict	3	{'negative':54, 'positive':29, 'total_count':83}
730	dict	3	{'negative':235, 'positive':110, 'total_count':345}
a	dict	3	{'negative':178089, 'positive':193580, 'total_count':371669}
already	dict	3	{'negative':9098, 'positive':5256, 'total_count':14354}
at	dict	3	{'negative':63771, 'positive':50169, 'total_count':113940}
been	dict	3	{'negative':20018, 'positive':13167, 'total_count':33185}

Fig 2: Diccionario de palabras

Dado que hay muchas palabras que se presentan de manera excepcional se hace una criba: si una palabra aparece mínimo 5 veces no será considerada, esto reduce mucho el tamaño del listado de palabras de 750.000 a 250.000. De esta manera se consigue una mayor fluidez y rapidez al consultar el listado sin variar el resultado de futuras reseñas, ya que estas palabras se supone que están mal escritas o son conceptos muy concretos y poco usados como *tags* a otros usuarios:

Algunos ejemplos de esta criba se pueden ver en la *Fig 3:*

@hatermagazine	dict	3	{'negative':1, 'positive':1, 'total_count':2}
@haugern	dict	3	{'negative':0, 'positive':0, 'total_count':0}
@hellopnsdear	dict	3	{'negative':1, 'positive':0, 'total_count':1}
@hidingesther	dict	3	{'negative':1, 'positive':0, 'total_count':1}
@hijackking7	dict	3	{'negative':0, 'positive':0, 'total_count':0}
@hippojuicefilm	dict	3	{'negative':0, 'positive':1, 'total_count':1}
@hitzproductions	dict	3	{'negative':3, 'positive':1, 'total_count':4}

Fig 3: Criba de palabras

Ya que esto está hecho en lenguaje *python* y el resto del proyecto en un lenguaje C#, se guarda el listado de palabras en un archivo .txt que se lee desde la aplicación móvil para saber si una nueva cadena tiene una connotación positiva o negativa.

El proceso de clasificar una frase es el siguiente:

```
Float positivo = 1.0, negativo = 1.0

Foreach (listaPalabras)
{
  Objeto palabraClasifi  = ObtenerPalabraDict(palabra)
  If(palabraClasifi ¡ = null)
  {
    negativo = negativo * palabraClasificada['negativo']
    positivo = positivo * palabraClasificada['positivo']
  }
}
negativo = negativo * (numeroTweetsNegativosDB/total)
positivo  = positivo * (numeroTweetsPositivosDB/total)
```

Una vez se obtiene la cantidad de positivo y de negativo de una frase, dividimos la cantidad positiva entre la cantidad negativa para así obtener valores relativos para todas las reseñas.

A partir de estos datos se crean frases propias y se clasifican de 0 a 5 estrellas, una vez clasificadas manualmente se calcula la relación entre la connotación positiva y negativa de cada frase. Con este cálculo se hace una correspondencia entre la cantidad de estrellas y el valor obtenido. (Véase la *Tabla 2* del apartado de pruebas para algunos ejemplos)

8.4 Desarrollo pantallas

Para empezar el desarrollo se planificaron los distintos módulos a construir y se hicieron los diseños de la aplicación en una aplicación de Windows llamada *MockUps* ya que era sencilla e intuitiva. Seguidamente se aplicó ese diseño en el proyecto y se crearon todos los layouts necesarios. La parte de front-end está hecha en lenguaje .xml con los estilos correspondientes a cada elemento. La parte de backend, los controladores de botones y la interacción entre pantallas está programada en C#.

9 Resultados

Esta sección se divide en resultados del servicio cognitivo de búsqueda, servicio cognitivo de reseña y finalmente resultados de la parte software.

9.1 Servicio cognitivo de búsqueda

Esta funcionalidad se encarga de enviar un texto con una petición y obtener una respuesta con todos los usuarios disponibles para realizar la tarea.

En la *Fig 4* se encuentra la petición y aparecen todos los usuarios que cumplen las condiciones de ser cocineros y tener una disponibilidad para el sábado a mediodía.

Fig 4: Petición y respuesta de la búsqueda de usuarios 1

En este otro ejemplo de la *Fig 5* se ve como gracias a la función de semejanza no hace falta recalcar el "oficio" requerido.

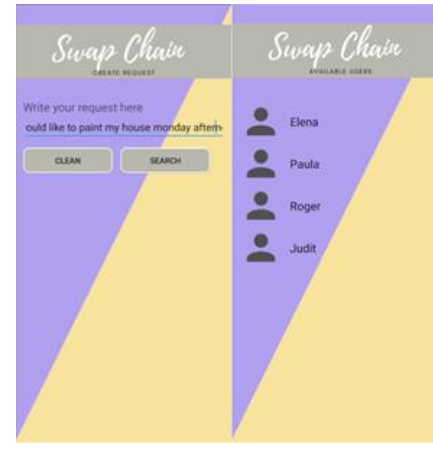

Fig 5: Petición y respuesta de la búsqueda de usuarios sin especificar el oficio

9.2 Servicio cognitivo de reseña

Esta funcionalidad consta de un input de texto donde el usuario valora el servicio dado o recibido. La aplicación da la reseña acorde a las palabras utilizadas.

En la *Fig 6* se pueden observar los dos procesos de escritura de la reseña y la obtención de las estrellas.

Fig 6: Obtención de una buena reseña

Se puede observar que para la reseña "The service was awesome and the guy was kind" el sistema devuelve una puntuación de 4 estrellas, cosa que se ajusta con la reseña dada.

Fig 7: Obtención de una mala reseña

En la *Fig 7* se puede observar que la reseña "I hate the service I got and it was so bad" es claramente mala, algo que sería entre 0 estrellas y 1,5-2 estrellas. El sistema calcula que es de 1 estrella y por lo tanto se ajusta a nuestros pensamientos.

Gracias a este sistema de reseña se evita que haya usuarios que puedan escribir una reseña muy buena y luego pongan una estrella o viceversa, que escriban una reseña muy mala y luego den 5 estrellas.

9.3 Otras funcionalidades

A parte de los servicios cognitivos se han desarrollado funcionalidades de crear cuenta, acceder a cuenta, salir, registros, pantalla principal, confirmar usuario seleccionado,…

En las *Fig 8 y 9* se puede ver la página principal de la aplicación con sesión no iniciada y con sesión iniciada.

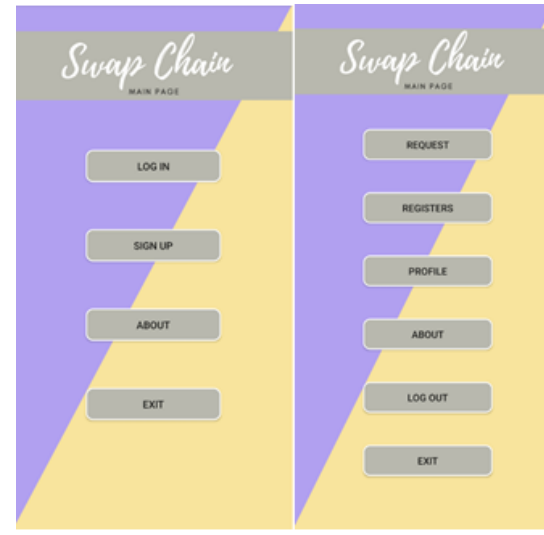

Fig 8 y 9: Página principal con sesión no iniciada y sesión iniciada respectivamente

Se puede ver la implementación del historial de registros y del inicio de sesión en las *Fig 9 y 10.*

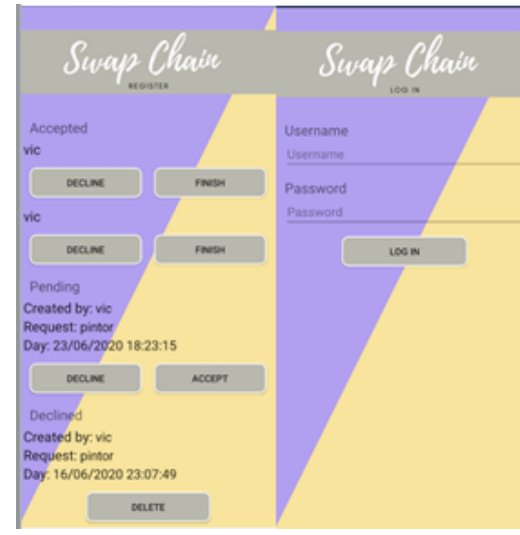

Fig 9 y 10: Página de registros e inicio de sesión respectivamente

En las imágenes 11 y 12 se puede ver las pantallas de creación de cuenta y la información de la aplicación.

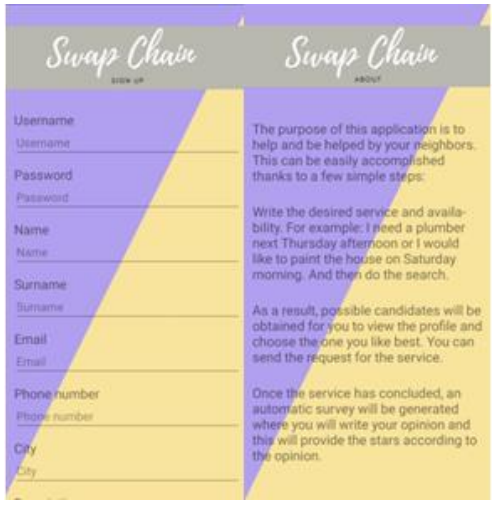

Fig 11 y 12: Página de creación de cuenta e información de la aplicación

10 PRUEBAS

En esta sección se detallan las pruebas hechas al desarrollar la aplicación, las principales pruebas son exploratory testing y pruebas de resultados de la reseña del servicio cognitivo.

10.1 Exploratoy Testing

Una vez implementada la aplicación se comprueba pantalla a pantalla cada una de las opciones y clicando todos los botones posibles para hacer que la aplicación deje de funcionar.

Dado que los accesos a base de datos son métodos asíncronos se da gran importancia a clicar varias veces a un mismo botón, esto fue uno de principales los errores que hubo ya que no se controlaban los clics que se daban y se corrigieron gracias a este test. También se verificaron que todos los mensajes *pop-up* sean correctas.

10.2 Ver resultados

Para la aproximación de las reseñas se hicieron varias pruebas con valoraciones propias, por ejemplo:

Reseña de test	Valoración manual	Cálculo posi-negati
I loved the service I get and I would repeat	4	0,46340357
I hate the application so much because the paintor has not come	1	0,00821023
The service was not bad but the the cooker made a mess and stinks	2	0,02577145
I liked the speed that the cleaner has done all the job I would repeat again	3	0,05171848
I would not repeat the plumber was late and made a mess under the sink	1	0,03354966
The service was really good and the plumber was so nice	5	1,46391647
I dont like how it ended I was expecting more	2,5	0,04132759
I dont like how it ended I was expecting more	1	0,00914682

Tabla 2: tests reseña

Dados estos resultados se decide que

- las 5: valor > 1
- 4 estrellas : valor > 0.5 && valor < 1
- 3 estrellas: valor > 0,3 && valor < 0,05
- 2 estrellas: valor > 0.02 && valor < 0.03
- 1 estrella: valor < 0.02

11 CONCLUSIONES

El objetivo de este Trabajo de fin de grado es poder consolidar mis conocimientos adquiridos en la carrera y además crear una aplicación útil para la sociedad.

Respecto la creación de la aplicación la he realizado con éxito ya que he implementado todos los requisitos en el término establecido. He conseguido ejecutar los objetivos principales, pero no todos los secundarios que se dejan para segundas versiones futuras y que explico en el apartado de líneas abiertas.

A nivel educativo este proyecto me ha aportado nuevos conocimientos y herramientas nuevas ya que nunca había utilizado Xamarin Forms para el desarrollo de una aplicación móvil. He tenido algún que otro atraso pero gracias a la gran comunidad y a los documentos de *Microsoft* he podido solventar todas las dudas e inquietudes relacionadas con el proyecto. También he aprendido sobre las Bases de Datos no-relacionales, ya que durante la carrera siempre se han utilizado relacionales como *PHPMyAdmin* u *Oracle.*

*G*racias a la metodología usado he podido desarrollar bien el proyecto ya que esta es ágil y permite poder cambiar los requisitos y funcionalidades a medida que se va avanzando en el tiempo, es el ejemplo de la base de datos

que se planificó para hacerlo con *SqLite,* pero se acabó ejecutando con la base de datos de FireBase Real-Time Database. El cambio de base de datos fue debido a que me surgieron problemas de compatibilidad y sincronización.

12 LÍNEAS ABIERTAS

En este apartado se exponen mejoras a futuro que se podrán hacer en la aplicación que principalmente son los objetivos secundarios de este trabajo.

Como principales avances se pondría un botón de audio para hacer una petición y otro para la creación de la reseña para que sean consultas más cómodas y rápidas.

Para mejorar el feedback que da la aplicación de cada usuario al hacer peticiones y ver el perfil de cada uno se le añadirán la puntuación media y las últimas reseñas de este, que ya están guardadas en la base de datos pero falta implementar visualmente.

También se puede hacer más complejo el sistema de búsqueda de usuarios haciendo que se filtre por horas y minutos, en vez de periodos de tiempo. Esto daría más exactitud y ahorraría tiempo.

13 BIBLIOGRAFÍA

[1] Xamarin Documentation (02/2020) Disponible en: https://docs.microsoft.com/en-us/xamarin/

[2] Módulo 0. Metodologías ágiles vs metodologías tradicionales. (02/2020) Disponible en: https://uv-mdap.com/programa-desarrollado/bloque-iv-metodologias-agiles/metodologias-agiles-vs-tradicionales/

[3] Balsamiq tutorials (01/2020) Disponible en: https://balsamiq.com/tutorials/

[4] Creación de diagramas (04/2020) Disponible en: https://app.diagrams.net/

[5] Xamarin Android Getting Started (3/2020). Disponible en: https://app.pluralsight.com/library/courses/xamarin-android-getting-started/table-of-contents/

[6] Xamarin-Forms working with firebase realtime-database crud operations ((05/2020) Disponible en: https://www.c-sharpcorner.com/article/xamarin-forms-working-with-firebase-realtime-database-crud-operations/

APÈNDIX

A1. DIAGRAMA DE CASOS DE USO

Figuras 13 y 14.

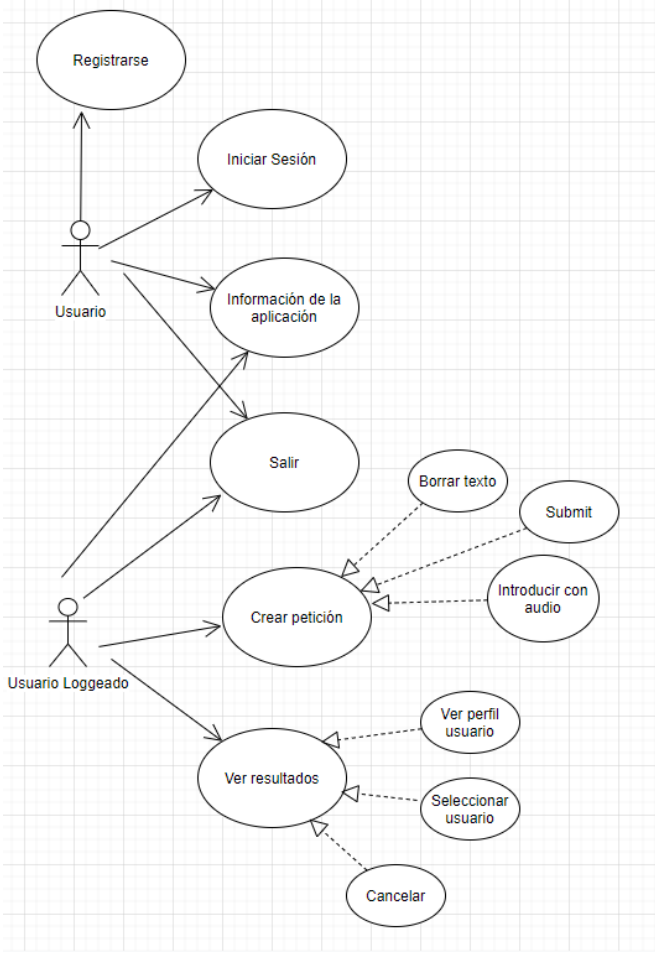

Figura 13 Diagrama de casos de uso 1

Figura 14 diagrama de casos de uso 2