CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

TRABAJO FIN DE GRADO

(Diseño e implementación de un módulo de red social para apps híbridas)

Autor: Juan Gil Manjón

Director: Francisco Rosales

MADRID, ENERO 2017

Resumen

En este documento se explica el proceso de desarrollo de una aplicación web responsive como módulo aplicable para dotar a otras apps de negocio de funcionalidades de red social.

Para su implementación, utilizaré el gestor de paquetes npm de Node Js sobre Angular 2 y PHP.

Aquí se expone todo el trabajo realizado para alcanzar los objetivos derivados de las funcionalidades de una aplicación de estas características, así como las herramientas y tecnologías escogidas para ello.

También se analizarán los trabajos previos realizados para comenzar a programar. Lo que abarca el origen de esta idea, la elección del diseño sobre el que he trabajado y el análisis de la lógica de negocio que define las funcionalidades de esta app.

Para su correcto funcionamiento, se ha implementado el código de la parte servidor, del cliente y el entorno que posibilita la interacción entre ambos.

Por último, se expondrá cuál puede ser la utilidad de esta herramienta en el sector de las aplicaciones y qué tipo de aplicaciones pueden beneficiarse de las funcionalidades que se han conseguido alcanzar en el proyecto.

Abstract

This document explains developing process of a responsive web application as an applicable module to provide other business apps with social networking functionality.

For its implementation, I will use the Node Js package manager Npm to develop an application on Angular 2 and PHP.

Here it is exposed the whole realized process to reach the objectives plained for an application of these characteristics, as well as the tools and technologies chosen for it.

It will be also analyze the previous work done to start programming. This encompasses the origin of this idea, the design choices and an analysis of the business logic that defines the functionalities of this application.

For its correct operation, the code of the server side, the client side and the environment that allows the interaction between both have been implemented.

Finally, the usefulness of this tool in the applications market and what type of applications can benefit from the functionalities that have been achieved in the project will be analyzed.

Agradecimientos

A mis padres, que han hecho posible que yo estudie una carrera. A la empresa GilperTech, sin cuya formación y apoyo, no podría haber realizado este proyecto. Y a mi hermana y mis sobrinos, que me han apoyado siempre durante la realización del mismo.

Índice de Contenidos

Índice de figuras

1. Introducción

1.1 Motivación

Llegando ya a los años 20 de este siglo, la cantidad de profesionales dedicados al sector del desarrollo de aplicaciones web ha crecido exponencialmente. Convirtiéndose en uno de los sectores profesionales más prolíficos en la actualidad.

Cualquier empresa con suficientes medios recurre a desarrolladores -tanto internos como externos- para llevar a la práctica aplicaciones de distinta temática. Esto que hace que actualmente las salidas profesionales en este campo sean amplias y valoradas.

Mi experiencia laboral trabajando en dos empresas dedicadas al mundo de la publicidad ha confirmado esto. Estas tienen departamentos de IT o contratan empresas de desarrollo web con intención de crear apps propias o para terceros enfocadas a llevar el negocio a una plataforma digital que modernice el aspecto y la gestión de sus productos de cara a los clientes.

Estas empresas gestionan usuarios. Estos son el centro y el valor más importante dentro de su actividad como negocio. Y normalmente, y salvo que sea su propósito específico, estos son invisibles para el resto de usuarios dentro de la aplicación puesto que la interacción entre ellos es nula.

Por ello, existe una creciente demanda de empresas que buscan añadir la funcionalidad de una red social interna para generar comunidad dentro de la app y con ello fidelidad a la aplicación. De esta demanda nace mi proyecto de TFG.

Su realización tiene como propósito un futuro uso por parte de la empresa en la que trabajo, donde se me ha ayudado tanto en la documentación como en la orientación al desarrollo de este módulo adaptable de red social.

Tras su implementación y adaptación. Los clientes que quieran añadir las características de una red social a su aplicación para restaurantes, apuntes, coches, perros… podrán solicitar su inclusión sin tener que sumar los costes derivados de la implementación desde cero de un módulo de estas características.

1.2 Descripción del trabajo

Este proyecto de fin de grado consiste en el proceso de desarrollo de la aplicación, los resultados derivados de este, así como los pasos previos necesarios para comenzar a desarrollar la app; diseño, documentación, tecnologías escogidas, etc…

Para comenzar, listaré los distintos objetivos definidos en el planteamiento inicial. A este listado le seguirá el apartado de los trabajos previos, donde se examinaron distintas redes sociales en aras de tomar referencias sobre qué tipo era el más adecuado para mi implementación.

Tras esto, haré un breve resumen de las tecnologías sobre las que he trabajado y con una descripción simple de cada una, pretendo enmarcarlas en el contexto de la aplicación. Lo que ayudará a entender el siguiente capítulo donde expongo las decisiones de diseño que afectaron al proyecto. Esto incluye los apartados de requisitos y análisis de la base de datos.

Después de analizar el diseño comienza un capítulo de desarrollo en el que se describe un diagrama de componentes que ilustra el funcionamiento de la app, por otro lado el proceso de implementación llevado a cabo para crear la aplicación desde cero además de la metodología de trabajo que he seguido durante el mismo.

En el capítulo de resultados se describirán las funcionalidades alcanzadas, muestras de la interfaz y del aspecto de la aplicación, así como un análisis de qué sería necesario para instalar la aplicación en móvil. Terminaré con el apartado de integración donde explicaré como se llevaría a cabo su adaptación a la aplicación original que define el contenido de datos.

Por último en el capítulo de conclusiones analizaré su posible papel en el mercado de las aplicaciones, además de dar mis opiniones sobre el propio trabajo de fin de grado

.

1.3 Objetivos del proyecto

Desde el inicio, se marcaron una serie de metas en base a un análisis sobre el modelo de red social que buscaba implementar.

Este análisis se realiza en el siguiente apartado, donde también se explican las tecnologías elegidas para alcanzar los siguientes objetivos:

- Diseñar la funcionalidad de una app de red social basada en un modelo vista-controlador.
- Implementar y comprender el entorno de desarrollo necesario para poder implementar la app.
- Elegir y familiarizarme con distintos frameworks de desarrollo.
- Implementar la gestión de usuarios dentro de la app – log in, log out, alta y remember password.
- Implementar perfil de administrador con permisos totales y usuarios estándar con permisos específicos para cada uno.
- Implementar un perfil de usuario con información propia elegida por este.
- Implementar un sistema de relaciones de amistad interno que haga que los usuarios interactúen y se conozcan entre sí.
- Implementar un muro de noticias donde se muestren los contenidos públicos o los que hayan sido subidos por amigos.
- Implementar un sistema de valoraciones y comentarios sobre los contenidos subidos por parte de usuarios de la app.
- Implementar Ranking de contenidos más valorados durante X tiempo.

Como he comentado, estos objetivos obedecen a la idea de desarrollar un tipo de red social específico cuyas características son algunos de los objetivos marcados en el proyecto y que son desglosados de forma más concreta en el apartado de requisitos.

2. Planteamiento Inicial

2.1 Trabajos Previos

Uno de los aspectos más importantes para el desarrollo del proyecto, es planificar las tareas que se van a realizar. Para ello tuve que adquirir unos conocimientos previos que me ayudaron entender sus distintas fases y saber cómo y cuánto tiempo llevaría poner en práctica los objetivos definidos.

Estos conocimientos han de ser tanto tecnológicos como sociales, por lo que primero tuve que hacer un pequeño análisis de las redes sociales existen y cuáles son las funcionalidades adecuadas para mi aplicación.

Una vez tenía una idea de en qué iba a consistir la aplicación, basándome en lo que había aprendido en el trabajo, hice la elección de las tecnologías que iba a utilizar. El conocimiento de algunas de estas tecnologías me ayudó a la hora de solucionar los errores que me iban surgiendo a en el desarrollo de la misma.

2.1.1 Redes sociales existentes

Atendiendo a su definición una red social es una estructura integrada por personas, entidades u organizaciones las cuales se encuentran conectadas entre sí por una o varios tipos de relaciones como pueden ser: amistad, parentesco, económicas o intereses comunes, entre otras posibilidades.[1]

"*El concepto* ***red social*** *alude a aquella estructura o forma de interacción social que involucra a un conjunto de personas relacionadas a partir de afinidades, similitudes a nivel profesional, amistad y o parentesco".*[2]

Dentro de los distintos tipos de redes sociales, existen las redes sociales verticales**,** que se refieren a aquellas redes dirigidas a un público determinado. Es decir, son especializadas. Las personas acuden a ellas debido a un interés en común. Lo que cuadra perfectamente con la idea genérica de la aplicación, que pretende que la interacción de los usuarios gire en torno a un producto.

Dentro de esta clasificación encontramos las redes sociales verticales mixtas. Este tipo ofrece al público un lugar concreto donde desarrollar actividades profesionales y personales, de tal forma que el contenido puede ser profesional o de ocio, pero tiene que haber un contenido concreto sobre el que gira su actividad. De acuerdo a estas características de cabe destacar Linkedln.

Por otro lado, en función del sujeto encontramos las redes sociales de contenido: el centro de interés reside en el contenido de aquello que se publica en la red. Es decir que las relaciones establecidas allí dependerán de los archivos a los que tengan acceso el resto de los usuarios. Algunas de las redes sociales más populares contenidas en esta clasificación es Flickr, Instagram Youtube y Vimeo.

Las aplicaciones de red social más conocidas son aquellas que entran dentro de las redes sociales horizontales. Esta clase de red social no fue creada para alojar a un tipo específico de usuario o un tópico concreto. De modo contrario, permiten la libre participación de quien así lo desee, proporcionándole una herramienta para la interacción a nivel general. Ejemplos de este tipo de red social son Facebook, Twitter, Google +, etc. Se trata de algunas de las redes sociales más concurridas y cabe destacar su facilidad de manejo de cara al usuario y sencillez.

2.1.2 Modelo escogido

En base a lo referido en el apartado anterior, mi decisión fue que el proyecto consistiera en una red social vertical mixta de contenido, donde mis referencias principales fueron redes sociales como instagram, facebook y twitter.

El porqué de estas referencias obedece a la idea de que la aplicación ha de ser sumamente sencilla de usar y con poca sobreinformación, algo que podría hacer que el usuario se viera desbordado en primera instancia y no fuera captado por la idea en sí.

En el caso de instagram, la referencia es clara, se pueden valorar contenidos, comentarlos y ver de forma sencilla un perfil y un muro sin complicaciones que hace que los contenidos sean llamativos y claros. Además el muro de contenidos puede ser público o sólo para ver contenidos subidos por amigos. Estos conceptos lo introduje en mi red social.

Para el caso de Facebook, se trata de una red social que permite compartir todo tipo de contenidos. Para este módulo, he diseñado inicialmente que acepte sólo imágenes pero la idea es que del mismo modo que se van a compartir imágenes para las pruebas, podría tratarse de otro tipo de archivos para distintas temáticas de la comunidad de usuarios. Además, Facebook permite poner un título al post y comentar dentro de él y las imágenes se muestran sin necesidad de pinchar dentro del post. Algo con lo que cuenta mi aplicación.

En cuanto a twitter, el listado de tweets es muy eficiente, esto hace que la aplicación cargue los contenidos sin consumir muchos recursos. Esta idea de sencillez me atrajo.

Con todo esto la idea que saqué en claro es que quería hacer una aplicación con un perfil, un muro de contenidos, relaciones de amistad entre usuarios, posibilidad de publicar posts/hilos/ publicaciones públicas y filtrarlas para que se muestren al usuario según si quiere que sea en base a todos los usuarios o solo aquellos con los que se tiene una relación dentro de la aplicación.

2.2 Tecnologías sobre las que se sustenta el proyecto

Puesto que mi TFG consiste en el desarrollo de una app, se van a necesitar nociones sobre diferentes ámbitos de la parte de la informática orientada a servicios web. Algunos de ellos los he estudiado en la carrera y otros lo he ido adquiriendo en el trabajo o por cuenta propia durante la realización del proyecto.

Existen distintas formas de plantear la arquitectura de un servicio web cliente-servidor. En este caso mi patrón de diseño está basado en un modelo vista controlador con algunas particularidades que residen en la forma en que el cliente gestiona las peticiones REST hacia el servidor, cuyo intercambio de datos se realiza en formato JSON.

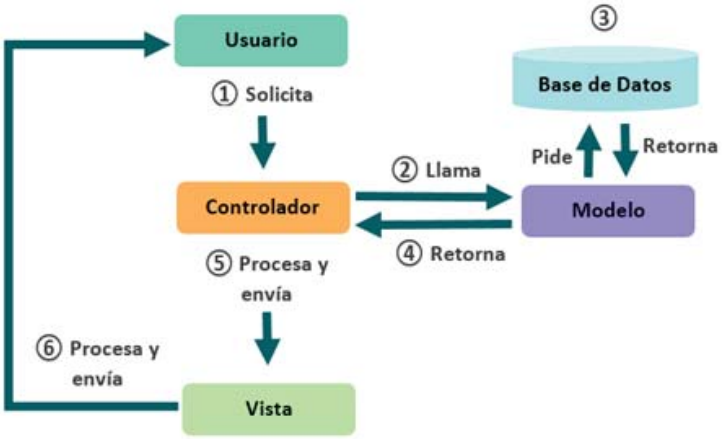

Figura 1: Modelo MVC clásico

En la imagen superior podemos ver el esquema del MVC clásico. Un usuario carga una vista mediante un navegador web, esta vista contiene datos que han sido obtenidos del servidor. Del mismo modo, el usuario envía información mediante actions que lanzan peticiones post o get sobre un recurso. El servidor accede a través de la uri al controller necesario que se encargará de llamar a un service que gestiona las acciones de un repositorio donde se manejan las bases de datos.

Las consultas o modificaciones pertinentes devuelven esta información al usuario cargando de nuevo la vista con las plantillas html y los datos devueltos en dicha operación. Este proceso se repite continuamente y el volumen de peticiones que se realiza al servidor suele ir de acuerdo al número de veces que se carga la vista. En esto excluyo las llamadas asíncronas como Ajax de Javascript.

En mi caso, utilizaré node js y Angular 2 para gestionar la interacción cliente-servidor. La particularidad de este framework de desarrollo web es que permite al cliente hacer comprobaciones, redirecciones, obtención de datos y cargar vistas sin necesidad de hacer peticiones al servidor, o por lo menos realizar las mínimas necesarias.

Estos significa que la parte cliente va a manejar más datos, lo que conlleva que la aplicación consumirá más recursos del lado del cliente de forma inicial. Pero también, significa que una vez se realiza la carga inicial la cantidad de peticiones al servidor disminuirán notablemente, lo que también repercutirá positivamente en la experiencia de navegación del usuario dentro de la app.

A continuación explicaré cada una de las tecnologías usadas en el proyecto mediante una breve descripción de cada una de ellas con un apunte de para qué van a ser utilizadas en mi proyecto. Para facilitar esta review, voy a dividirlas en dos grupos. Las que se utilizan para el front, y las que se usarán en el back de la aplicación, dado que en este proyecto he desarrollado ambos extremos de la aplicación.

Front

Figura 2: logotipo Node JS

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto para la capa del servidor. Construido con el motor de JavaScript V8 de Chrome Node.js usa un modelo de operaciones E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente. El ecosistema de paquetes de Node.js, npm, es el ecosistema más grande de librerías de código abierto en el mundo.[3]

En este caso, recurrí a npm para instalar algunos de estos paquetes Node Js.

Figura 3: Logotipo NPM

Npm es el manejador de paquetes por defecto para Node.js.

Se utiliza para instalar los distintos componentes que proporciona Node Js. Cabe destacar los que generan componentes y ejecutan la aplicación desde la propia línea de comandos permitiendo primero instalar Angular 2 y después los distintos servicios utilizados para llamar a las aplicaciones generadas mediante esta tecnología, los Ng services.

Figura 4: logotipo Angular

Angular 2 es un framework basado en TypeScript, un superconjunto de JavaScript que proporciona principalmente tipografía estática opcional, clases e interfaces. Uno de los grandes beneficios es permitir un entorno más rico para detectar errores comunes mientras escribe el código [4]. Ambos engloban parte cliente, lo que se refiere a los templates HTML y los estilos CSS. Por otro lado llevan a cabo labores del servidor, como redirección o manejo de datos.

La carga de contenidos se produce de forma mucho más selectiva. Se puede navegar por dentro de una aplicación sin realizar una sola llamada al servidor del back y sólo en las partes realmente necesarias se realizan peticiones y transferencia de datos en formato JSON. Además, este flujo en la parte del cliente mediante typescript es perfecto para depurar código.[5]

Angular 2, se basa en componente y promesas. Una promesa es un estado de espera que adquiere el componente –sin dejar de ejecutarse-. Esta se debe a que se necesitan datos que se encuentran en el servidor, por lo que se envía una petición y cuando estos llegan de vuelta al cliente genera un evento con el valor de retorno de la petición. Una vez le llega esta respuesta se manejan estos datos conforme a la lógica de navegación diseñada.

Estas llamadas de los componentes se hacen utilizando services, que son los proveedores de servicios. Lo que se conoce como providers. Estos se incluyen en la estructura global de la aplicación mediante importaciones al módulo principal de la app.

En la siguiente imagen se ilustra la secuencia de acción por parte del usuario, que interactúa con una vista conectada a una clase que genera un modelo que es enviado por el service en formato json al servidor.

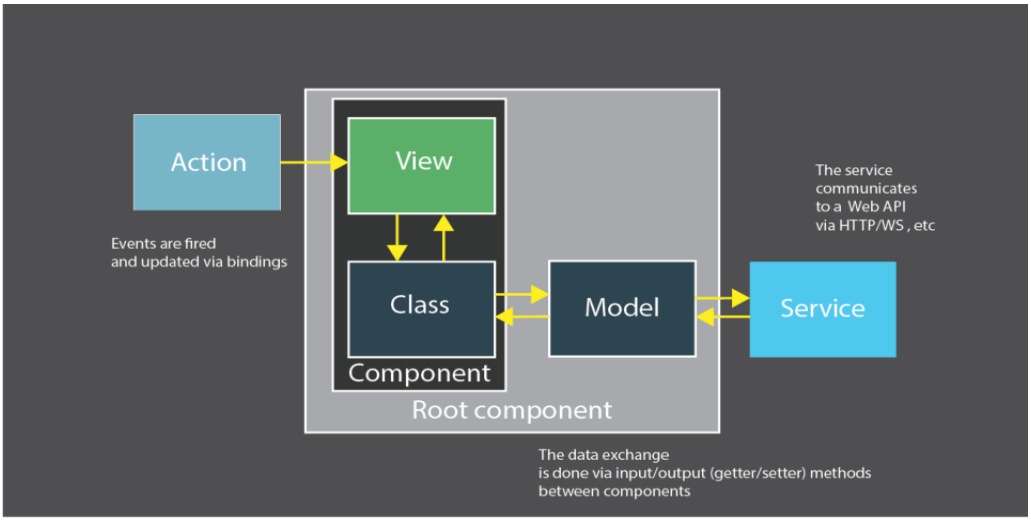

Figura 5: Esquema componente Angular 2

Como podemos observar en la imagen. Cada componente puede estar relacionado a una vista, esta no tiene por qué ser una página complete como tal sino simplemente una parte de esta. Estos componentes tienen "en propiedad" tres tipos de archivos typescript(class), css y html. Además podemos importar jquery en cualquiera de estos.

Los componentes pueden ser hijos o padres de otros componentes. Esta relación padre hijo es totalmente equivalente a la herencia en java. Puesto que las funciones del padre no tienen por qué tener nada que ver con las funciones del hijo. Y para hacer referencia a variables entre componentes antes hay que hacer referencia explícita al component que se desea acceder.

Cabe destacar el app.component, que es el componente que carga el resto de componentes, el app.routes que define la redirección dentro de la app y el app.modules contiene los diferentes módulos de componentes que se quieren insertar en la aplicación.

Por otro lado, es importante remarcar el uso de modelos o clases para definir objetos que podemos definir y usar a nuestro gusto dentro de la app. Estos modelos serán los objetos que contengan los datos que "rellenan" las vistas html.

Todo lo anterior define sobre qué estructura base está montada la aplicación. Con este entorno de programación el manejo de datos y la conexión cliente servidor están montados sobre tecnologías eficientes y la app utilizará entornos de programación modernos que se usan hoy en día para la implementación de aplicaciones para grandes empresas.

Aún con esto, existen unos **lenguajes básicos** sobre los cuáles irá montada la vista de la aplicación que estructuran y estilizan el DOM – Los diferentes elementos de la vista-. Estos son html, css y javascript.

Figura 6: logotipo HTML 5

Siglas en inglés de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web. La última versión permite etiquetar elementos utilizando cualquier término. Algo que no permitían las anteriores versiones. [6]

Figura 7: Logotipo CSS

Hojas de Estilo en Cascada (Cascading Style Sheets), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.[7]

En este proyecto me he servido del framework css más utilizado, bootstrap.

Figura 8: Logotipo Jquery

jQuery es una biblioteca multiplataforma de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX para web.

En este proyecto he utilizado jquery para diseñar la animación de aparición de la cabecera además de algunas funciones relacionadas con el autologin mediante cookies.[8]

Back

El back-end maneja los datos y los introduce, actualiza y borra en base de datos. Esta parte de la app está desarrollada en PHP utilizando un framework propio de la empresa. Es aquí donde vamos a recurrir a bastante conocimiento aprendido en la facultad relacionado con las bases de datos y las técnicas de programación eficientes.

El primer paso para poder generar las clases que dotan de la lógica de negocio a la aplicación, es el diseño de la base de datos relacional que representa las relaciones que existen entre los datos que maneja la aplicación. Este diseño, descrito más adelante, es un capítulo específico en el que se analiza la base de datos en profundidad. A continuación detallo las tecnologías utilizados del lado servidor.

Figura 9: logotipo Easy PHP

Easy PHP server es un programa utilizado para montar un servidor web en nuestro equipo compatible con el lenguaje PHP.

En realidad habría que ser más específicos si decimos que Easy PHP no es un solo programa, sino que en realidad son tres en uno. Por un lado tenemos a Apache, el servidor más popular de páginas web. Por otro lado a MySQL, la base de datos más extendida de código libre y por otro PHP, el lenguaje o tecnología más extendido para realizar páginas con programación en servidor, acceso a bases de datos, etc. [9]

Easy PHP es un programa que permite disponer de los tres componentes indispensables para programar con PHP en nuestro propio ordenador, con una descarga rápida y una instalación sin ningún tipo de problemas o necesidades de configuración adicionales.

Figura 10: logotipo PHP

PHP es uno de los lenguajes más requeridos en el sector, ya sea para desarrollar un sitio web o reformar o brindar soporte a la web de alguna empresa que lo necesite. [10]

El uso de algunos frameworks resulta útil para entender y realizar mejores trabajos de manera mucho más práctica. Algunos de los más famosos son Laravel, CodeIgniter, Yii, KumbiaPHP o Kohana. En mi caso he utilizado un framework propio de la empresa con el que he estoy acostumbrado a trabajar.

La empresa me ofreció la posibilidad de contar con él sin compromiso alguno, siempre y cuando el código no se publicara.

Una vez supe qué tecnologías usar, hay que definir cómo vamos a comunicar ambas partes. Teniendo montados dos servidores –uno web y otro de datos-. Hay que redefinir el archivo host del equipo. En este caso, al haber usado un Windows se encontraría en C:\Windows\System32\drivers\etc\hosts. Donde se añade la ruta escogida para acceder a la aplicación. Todo esto para trabajar en local, en modo desarrollo.

Además, hay que configurar el archivo httpd-vhosts.conf del servidor http para que asocie la carpeta donde se encuentra el proyecto a una dirección establecida dentro de la máquina. Tras esta configuración. Ya tenemos un cliente conectado a un servidor web y este a un servidor de base de datos. Lo necesario para empezar a desarrollar la aplicación ya está instalado.

3. Diseño

3.1 Requisitos

Uno de los aspectos fundamentales son las funcionalidades específicas a implementar en la aplicación, lo que se recoge en el listado de requisitos. Estos definen qué ha de poder hacer la aplicación y con ello qué detalles habrá que tener en cuenta a la hora de programar.

En este apartado, se listan estos requisitos en base a lo que serían a priori los diferentes apartados de la app.

En cuanto al acceso

- Un usuario tiene que poder acceder a la pantalla de login de la aplicación
- Un usuario ha de poder registrarse en caso de no estar registrado con anterioridad
- Un usuario ha de poder recurrir al servicio remember password para que se le envíe un email con su contraseña
- Un usuario ha de poder acceder mediante un login a la pantalla home/muro
- Si hay fallo en el login, registro o remember, se tiene que informar al usuario de cuál ha sido el motivo del error

Para todas las pantallas

- la app ha de mostrar una cabecera con las diferentes opciones de navegación dentro de la aplicación, incluido el log out
- Un usuario debe de poder hacer log out desde cualquier pantalla
- La aplicación tiene que darte la opción de crear un nuevo hilo o post

Al crear un post o hilo

- Se tiene que dar opción de publicar un post con título –obligatorio-, un texto acompañando al título y adjuntar una imagen.
- Al publicar un post se debe de avisar con una notificación de que todo ha ido correctamente.
- Además los post de cada usuario pueden ser editados y borrados desde el muro.

Definición requisites diferentes pantallas

Muro

- Ha de haber un mensaje de bienvenida para el usuario
- Se debe de mostrar en una pantalla todos los contenidos más recientes publicados por otros usuarios de la app
- Se tienen que poder filtrar estos contenidos para mostrar sólo los relacionados con usuarios amigos en la app
- Al publicar un post ha de verse como publicado dentro de esta pantalla
- Se tiene que dar la opción de borrar el post
- Se tiene que dar la opción de valorar el post en forma de like/dislike –no hay valoraciones negativas-
- Se tiene que dar la opción de ir al perfil del usuario que ha publicado cada post
- Se tienen que poder ver las imágenes publicadas en cada post sin necesidad de pinchar en el post
- Se tiene que poder comentar el post
- Si el hilo ha sido creado por el usuario de sesión, puede eliminarlo
- El administrador peude borrar cualquier contenido

Perfil

- El perfil puede ser visto desde tres perspectivas, usuario propio, usuario amigo o usuario desconocido
- El usuario propio ha de poder modificar su foto de perfil
- El usuario propio ha de poder modificar su bio
- El usuario propio ha de poder acceder a sus hilos y eliminar los que él considere oportuno
- El usuario propio ha de poder acceder a sus amigos y verá en esta pantalla sus peticiones de amistad
- El usuario amigo ha de poder visualizar la información y los contenidos publicados por su amigo
- El usuario amigo podrá comentar y valorar los contenidos publicados
- El usuario desconocido sólo podrá visualizar la imagen de perfil y la bio del usuario
- El usuario desconocido tendrá la opción de agregar amigos, y si el otro usuario acepta se le notificará

Ranking

- El ranking debe de mostar los post más valorados en la última semana
- El ranking debe de mostrar los post más valorados en el último mes

3.2 Diseño Base de datos

Después de definir los requisitos, de estos se desprendían cuáles iban a ser los principales actores dentro de la app. En base a ellos realicé un diseño de una base de datos marcada por las funcionalidades que se querían alcanzar y los datos que esta app iba a manejar.

La definición de la base de datos, resulta fundamental para entender el funcionamiento de esta. En mi caso utilicé la herramienta Navicat, que gestiona bases de datos SQL.

Se trata de un *modelo relacional* que contempla usuarios, relaciones de amistad entre estos, hilos creados por los usuarios, contenidos posteados por cada usuario sobre un hilo y los likes que recibe un hilo.

Para analizarla la BD, explicaré cada una de las 5 tablas que la forman:

- Usuarios

Contiene la información que se almacena de los usuarios de la app. Se guarda en ella un identificador único autoincremental, un email, password -hash resultante de pasar la password del usuario por un hash MD5- , un nombre de usuario, un campo de texto para una pequeña bio, un campo para guardar el nombre del archivo que será la imagen de perfil y dos campos creation_time y update_time de tipo timestamp que se actualizan al introducir un nuevo elemento y al actualizarlo.

El campo identificador será la clave primaria y junto con el creation_time y el update_time serán comunes a las 5 tablas, por lo tanto no los voy a nombrar en los siguientes puntos.

- Amigos

Esta tabla contiene las relaciones de amistad entre usuarios, puesto que la amistad entre dos usuarios es una relación de orden n a n, hay que hacer una tabla específica para esta. Consta de dos claves foráneas que relacionen dos id_usuario y un estado para saber si la relación está pendiente de aceptar o ha sido rechazada o aceptada.

- Hilos

Un hilo es el contenido principal de la app, debe contener un título y puede contener un comentario o un archivo adjunto. Además contiene una clave foránea apuntando a un id_usuario, donde se establece una relación de orden 1 a n. Un hilo sólo puede haber sido creado por un usuario.

- Contenidos

Los contenidos son los comentarios dentro de un hilo que aportan otros usuarios. Tiene un campo comentario, un campo archivo y dos claves foráneas de

relacionando esta tabla con un usuario y un hilo. Esta relación es una relación 1:1:n. Puesto que un contenido está relacionado a un único usuario y a un único hilo.

- Likes

Al igual que pasa con la tabla amigos, la relación de likes es de orden n a n, puesto que un usuario puede dar me gusta a varios hilos y varios hilos pueden tener varios likes de usuarios. Por lo tanto se hace una nueva tabla con dos claves foráneas a usuario e hilo.

Tras definir esta base de datos Sql, generamos el esquema que representa la BD.

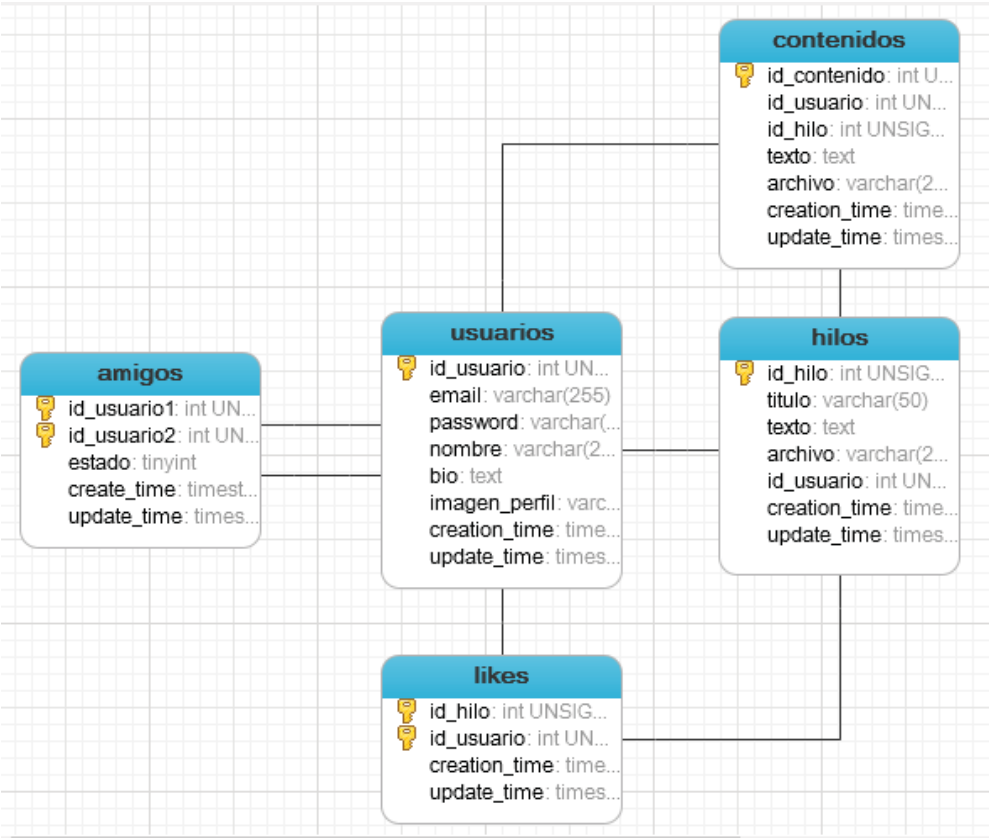

Figura 11: Esquema de la BD del sistema

Para representar una base de datos o las relaciones entre los distintos agentes del sistema podría haberse utilizado un diagrama E-R. Como la herramienta Navicat generaba el Schema SQL. Utilizo esta figura para ilutrar la BD.

4. Desarrollo

Tras haber expuesto la estructura del proyecto en el diagrama de componentes. En este capítulo de desarrollo se analizará cómo ha sido la evolución del proyecto y cuál ha sido la metodología de trabajo seguida durante el mismo.

En este caso, no haré especial énfasis sobre el apartado de planificación, ya que este se ha tratado en entregas anteriores durante la realización del TFG. En cualquier caso podría decirse que durante cuatro meses he sacado una aplicación adelante compaginando el trabajo con la realización del proyecto por las tardes, lo que me ha llevado en torno a las 320 horas planificadas.

Como era de esperar la planificación inicial sufrió diversos cambios, empezando por el orden de realización de las tareas. Este se fue modificando a medida que se iba avanzando y planteándose situaciones de dependencia entre componentes que hacía que algunos tuvieran que estar implementados antes que otros.

En este capítulo hare un análisis global del funcionamiento de la app en el apartado de componentes, hablaré de los problemas que he encontrado en el desarrollo y sobre cuál era mi metodología de trabajo.

4.1 Diagrama de componentes

El siguiente esquema representa la jerarquía de componentes de un proyecto en Angular 2. Podría asemejarse al diagrama de clases en java pero en este caso sólo se representan las dependencias y navegación dentro de los distintos componentes de la app. Los métodos específicos pueden ser únicos de cada componente y la herencia de métodos no es igual que en java, que es lo que más hemos estudiado en la carrera.

Aquí se muestra la lógica de las distintas pantallas de la app. Cada una de las cuáles se corresponde con uno o más componentes, que en este caso contienen otros componentes, que llaman a servicios y manejan modelos.

Estos últimos podríamos considerarlos la unidad de datos que se intercambia entre el servidor y el cliente. En realidad es algo que sólo ve el cliente de angular ya que a la hora de enviar estos modelos se pasan primero a formato Json, y de vuelta se produce un casting en el servicio appDataService del cliente sobre el objeto Json que llega. Este casting genera un nuevo modelo para pasárselo al componente que ha realizado la llamada al servicio y está esperando un evento llamado promise con id único para el proceso que lo ha convocado.

Estos modelos pueden ser específicos, como es el caso del modelo usuario, el modelo like, comentario e hilo. O por otro lado puede ser un modelo resultado o error que es común a otros proyectos de angular. El modelo error se produce cuando el objeto que se recibe al desencapsular el Json tiene una propiedad llamada error. El modelo resultado se utiliza para cuando por ejemplo se hace un borrado o un update sobre una tabla, en él se muestra el status de la operación y el número de filas afectadas dentro de la bd.

Los modelos específicos se corresponden con las entidades de la base de datos. Pero también pueden generarse nuevos modelos con más información. Por ejemplo, un objeto hilo devuelve los campos que hay en la base de datos. Un objeto hilo_con_user se utiliza para un mensaje de respuesta del servidor con el campo id_usuario convertido en un objeto usuario.

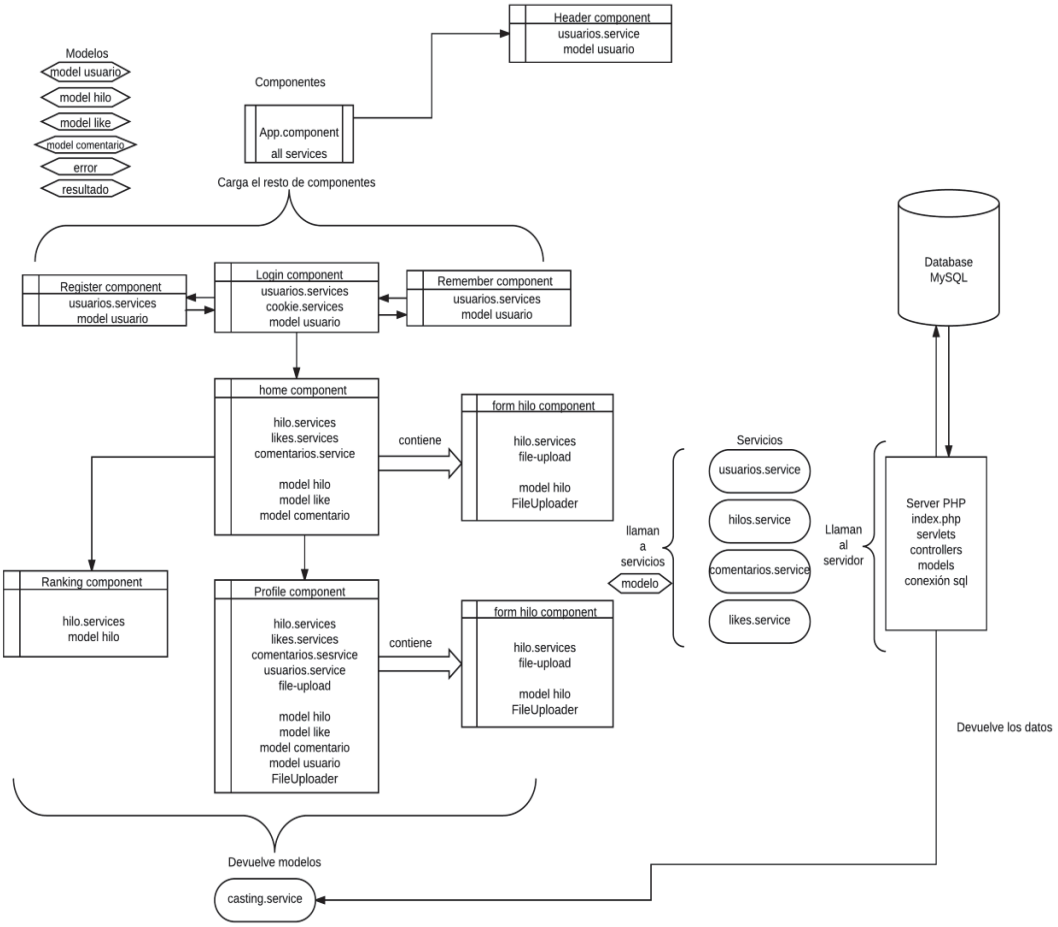

Figura 12: Diagrama de componentes

En cuanto a los componentes, cargan modelos en la vista html y navegan dentro del sistema de redirección de la app intercambiando información entre sí o adoptando como hijos otros componentes. Como es el caso del form hilo tanto para la pantalla de perfil como el muro o el single-component que carga la vista de un post individual.

Los componentes se cargan en primera instancia por el app.component que en este caso carga también el header para las pantallas que no sean el login, el remember password o el register. Es importante destacar que el componente header porque está disponible dentro de las pantallas de la aplicación y sirve para realizar la redirección dentro de la app.

Por último cabe destacar los servicios. Que son utilizados por los componentes para realizar las llamadas necesarias al servidor encapsulando los modelos que los componentes les pasan –puede ser vacío- para hacer la llamada.

En cada servicio se especifica la URL del servidor sobre la cual se quiere actuar y el valor del índice del applicationDataService que queremos que haga el casting del objeto devuelto por el servidor.

En el diagrama se representa todo lo que se ha citado en este apartado. Relacionando cada componente con el modelo de datos que utiliza y el servicio que importa para realizar llamadas al servidor. También se representa el servidor y la base de datos.

Sobre este servidor habría que destacar que recibe el flujo de datos en el archivo index.php. Aquí conecta el primer parámetro pasado por url, con el servlet correspondiente. Existen 5 servlets, uno por cada tabla de la BD, es decir, usuarios, hilos, likes, comentarios y relaciones de Amistad.

Estos servlet son los que llevan el flujo a un controller específico o genérico para realizar las peticiones a la BD y obtener así los datos requeridos y manipularlos.

Existen peticiones muy concretas pero las más utilizadas son las que hacen alta, borrado, listado de datos de la bd. En el caso del usuario, también estarían entre estas peticiones las de login, log out, register o remember.

4.2 Problemas encontrados

Durante la realización del TFG han ido surgiendo numerosas situaciones que han hecho que me quedara parado y no pudiera continuar con el desarrollo.

En algunos casos, sobre todo al principio del proyecto, tuve que recurrir a la ayuda de la empresa para solucionar algunos de estos errores. A medida que he ido avanzando dentro del desarrollo he ido aprendiendo cómo solucionarlos algo clave para mi formación como desarrollador de apps.

Los primeros problemas surgieron a la hora de montar el entorno sobre el que se iba a programar. Todavía no tenía todos los conocimientos necesarios sobre easy php y tuve ciertos problemas con archivos de configuración para montar el host dentro de mi máquina.

Una vez estaba ya montado el entorno, el primer problema vendría con el sistema de mailing de gmail. Dentro del back, para gestionar la petición de la pantalla de remember password para enviar un email con la contraseña. Había ciertas limitaciones que google ponía dentro de los archivos de configuración mailing de PHP.

Otro problema surgió a la hora de montar el header de la aplicación. Pensé que como los metadatos del head se incrustaban desde el index.html el header, que se incluiría en todas las pantallas, podría ser introducido de esta forma. La correcta colocación de este elemento header consistía en albergarla en el componente app.component que carga cada componente después de él haberse cargado al inicio de la ejecución.

El problema más correoso que tuve fue que cuando quería instalar con npm algún paquete extra para añadirlo al node-modules y poder acceder a algunas de sus características como provider. No me dejaba instalar estos paquetes. Se debía al escudo del sistema de archivos del antivirus que no alertaba de nada, pero bloqueaba la instalación automatic de algunos de estos archivos.

En cuanto al diseño de la base de datos o el diseño de requisitos no tuve grandes problemas ya que me acordaba de lo que había estudiado en la facultad y tenía modelos de referencia en la empresa.

4.3 Metodología de trabajo

La metodología de trabajo era la misma a la hora de desarrollar la app. En primer lugar había que arrancar la aplicación desde el terminal, donde se ejecutaba npm.

Al ejecutar ng serve, la salida compilada se sirve desde la memoria, no desde el disco. Esto significa que la aplicación a la que se está sirviendo, para que sea más rápido el acceso a los recursos, no se encuentra en el disco.

Una vez están aplicación y los dos servidores de bases de datos y http arrancados podemos acceder al proyecto mediante una url propuesta en el archivo virtual hosts del servidor http.

Para la edición de código he utilizado eclipse php neon para la parte de back y jetbrains web storm para la parte de front. Esta segunda herramienta me ha requerido un pequeño desembolso para adquirir la licencia durante un periodo de 3 meses.

Para monitorizar las peticiones REST, he utilizado la herramienta Charles, que permite ver el contenido de las petciones y respuestas hacia el servidor. Además permite editar estas peticiones para hacer comprobaciones de seguridad. Cualquier flag o alerta que colocara en el código para depurar, se iba a mostrar a través de este programa.

En cuanto al diseño el propio navegador me servía para editar estilos y comprobar los elementos que modificaba. En mi caso Google Chrome- me ayudaba a demás a depurar errores por consola tanto con alertar creadas por mí en el código como por alertas que están implementadas dentro de los paquetes npm.

Por otro lado llevaba un control de versiones con source tree, que me permitía acceder a bitbucket donde tenía dos repositories donde guardadaba dos proyectos, uno que incluía la parte del cliente y otra para la parte del servidor. Con este programa podía llevar una organización y seguimiento de mi proyecto y revertir cambios si era necesario.

Esta metodología ya estaba bastante automatizada y tardaba muy poco tiempo en ponerme a trabajar. Además, aprovechando cada uno de los elementos aquí citados, la depuración de errores era rápida lo que me ha mostrado que he aprendiendo a manejar herramientas de desarrollo web con soltura.

5. Resultados

En este apartado se analiza la aplicación una vez terminada. Desde la funcionalidades alcanzadas y su apariencia final hasta su adaptabilidad a otras aplicaciones y entre dispositivos.

5.1 Funcionalidades alcanzadas

Tras el desarrollo del código de la aplicación, se pueden valorar las funcionalidades alcanzadas en base a los objetivos definidos al principio del proyecto.

La mayoría de las funcionalidades planificadas las he podido llevar a la práctica, y en algún caso he valorado implementar otras que finalmente no he desarrollado. A continuación listo las funcionalidades alcanzadas.

En primer lugar la gestión de usuarios dentro de la app –el log in, log out, alta y remember password- se realizan correctamente. El acceso de usuario está supeditado a la comprobación de si se es o no el administrador.

Como normalmente se suele hacer un back-office para el administrador, en este caso para hacer la comprobación del rol del usuario en vez de un campo en base de datos, existe un listado de usuarios en el lado cliente que reconoce si se es administrador y restringe el número de funcionalidades a las que puede acceder este. Por lo tanto para ser administrador se tiene que definir previamente al registro de este tipo de usuarios, esto habría que mejorarlo oañadiendo un campo perfil en la table de usuarios.

En cuanto al muro de noticias/posts donde se muestran los contenidos públicos o los que han sido subidos por amigos. Se ha implementado correctamente, dando la opción de editar o eliminar aquellos posts que han sido creados por el usuario. Se pueden crear posts desde este muro que son introducidos según se crean atualizando el contenido de la tabla que los contiene. Al pinchar en cada elemento se muestra un pop con el contenido del post, que yo he tratado como un componente single presente en todas las pantallas.

El perfil del usuario contiene información básica sobre este, además de los posts y amigos que tiene el usuario. La vista está restringida según el usuario que accede al perfil -como se definió en los requisitos- y desde esta pantalla se da la opción de agregar amigos o crear posts.

Las relaciones de amistad son almacenadas en bases de datos con tres posibles estados, 0 pendiente, 1 aceptada y -1 rechazada. En caso de estar pendiente la vista del perfil sigue estando como estaba antes de ser aceptada, es decir vista restringida a la info del perfil. Si ha sido aceptada mostrará el perfil con la info del perfil y los posts de ese usuario con posibilidad de verlos y varolarlos. En caso de acceder al perfil del usuario de sesión se podrán editar y borrar contenidos. Además, se mostará un listado de amigos

con su foto de perfil y se tendrá un buzón de peticiones de Amistad en caso de tener alguna pendiente.

Todos estos post se pueden valorar y comentar. Cualquier usuario puede comentar los hilos que han sido publicados en el muro de posts y valorarlos. Una funcionalidad que se podría haber añadido es poder elegir qué post son públicos o están restringidos a usuarios amigos.

Además se ha implementado un ranking que permitir ver los post más valorados en la última semana o el ultimo mes. Desde esta pantalla se puede acceder a la creación de un hilo o ver en detalle un post.

Una de las funcionalidades no alcanzadas y que desde un principio estaba en el aire era un sistema de mensajería interno entre los usuarios de la aplicación.

5.2 Interfaz

En este capítulo se mostrarán gráficamente las pantallas implementadas. Empezando por el acceso:

Login:

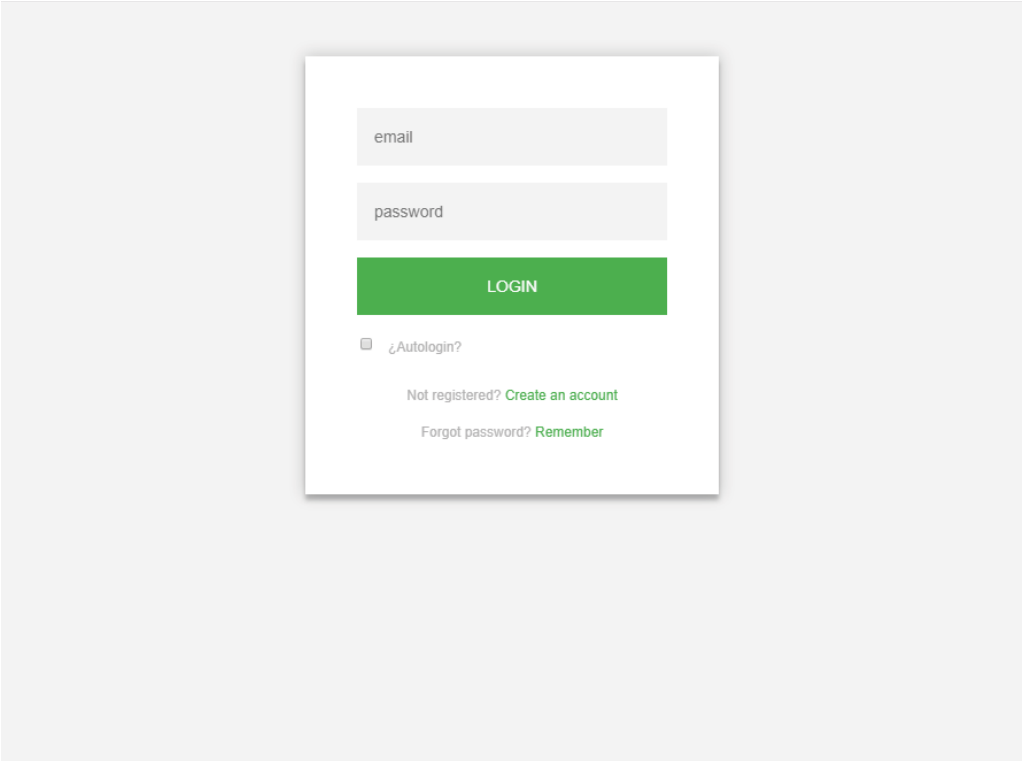

Figura 13: Pantalla de login

Register:

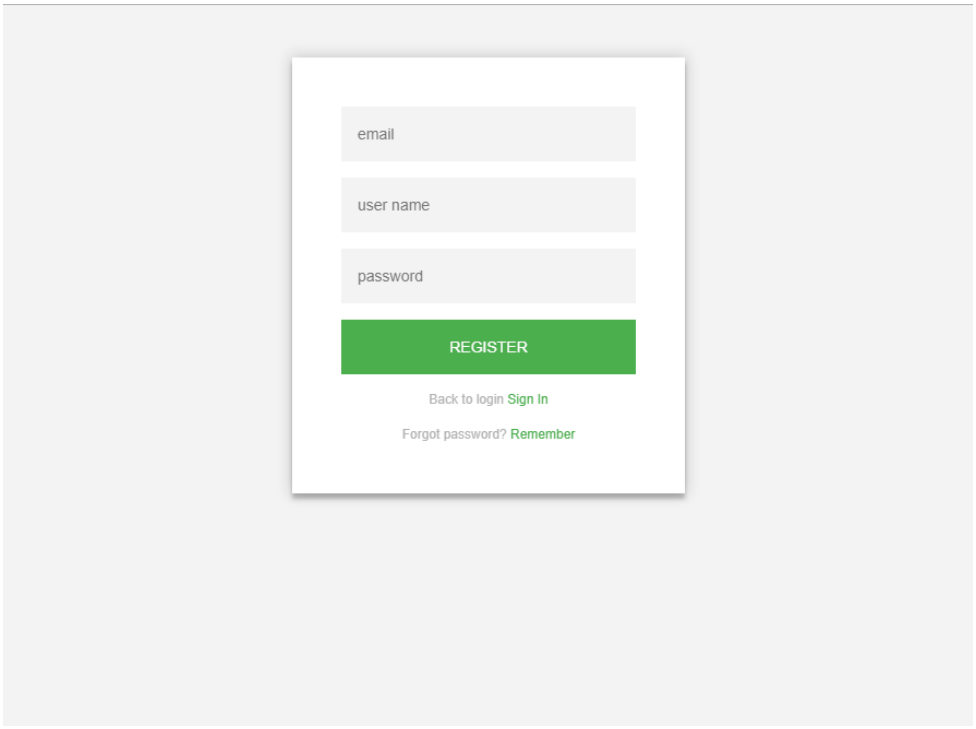

Figura 14: Pantalla de registro

Remember password:

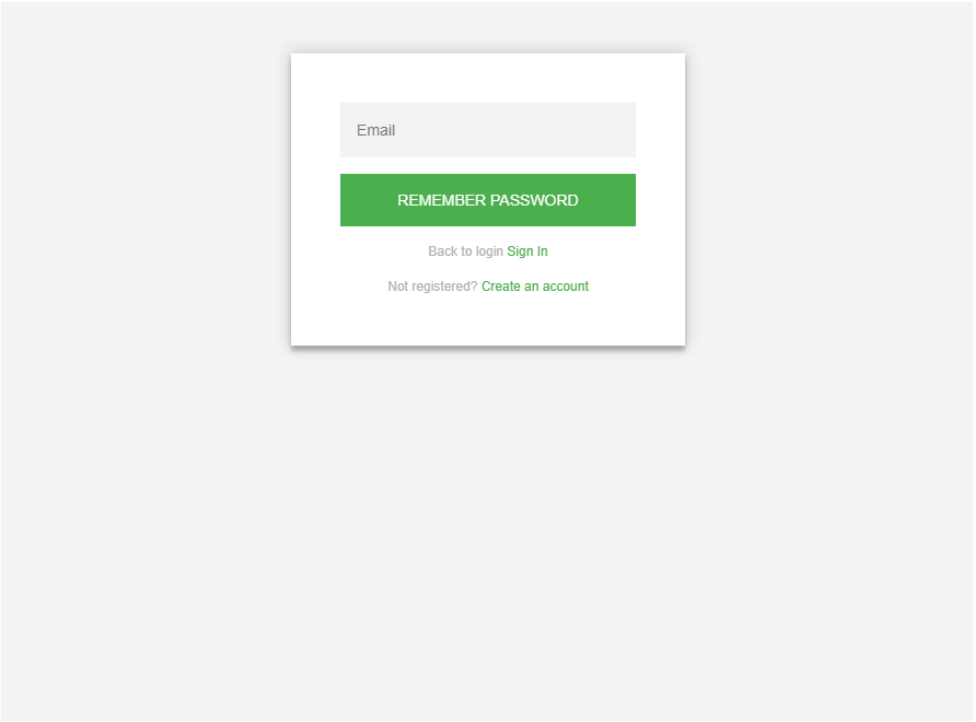

Figura 15: Pantalla de recordar contraseña

Menú desplegable del header:

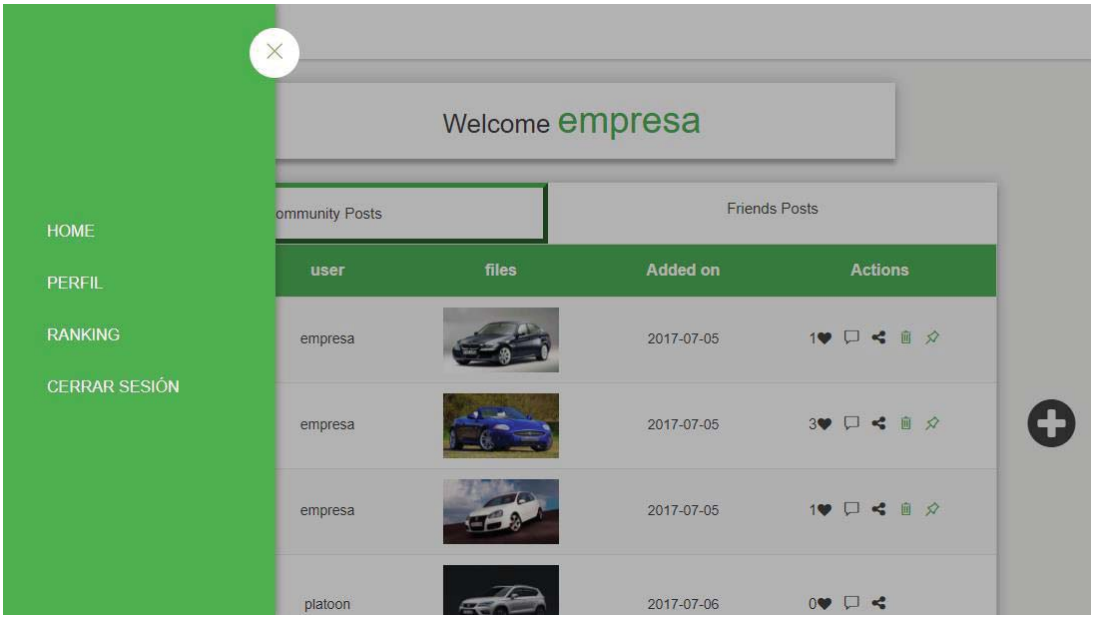

Figura 16: Pantalla de menú-header

Home:

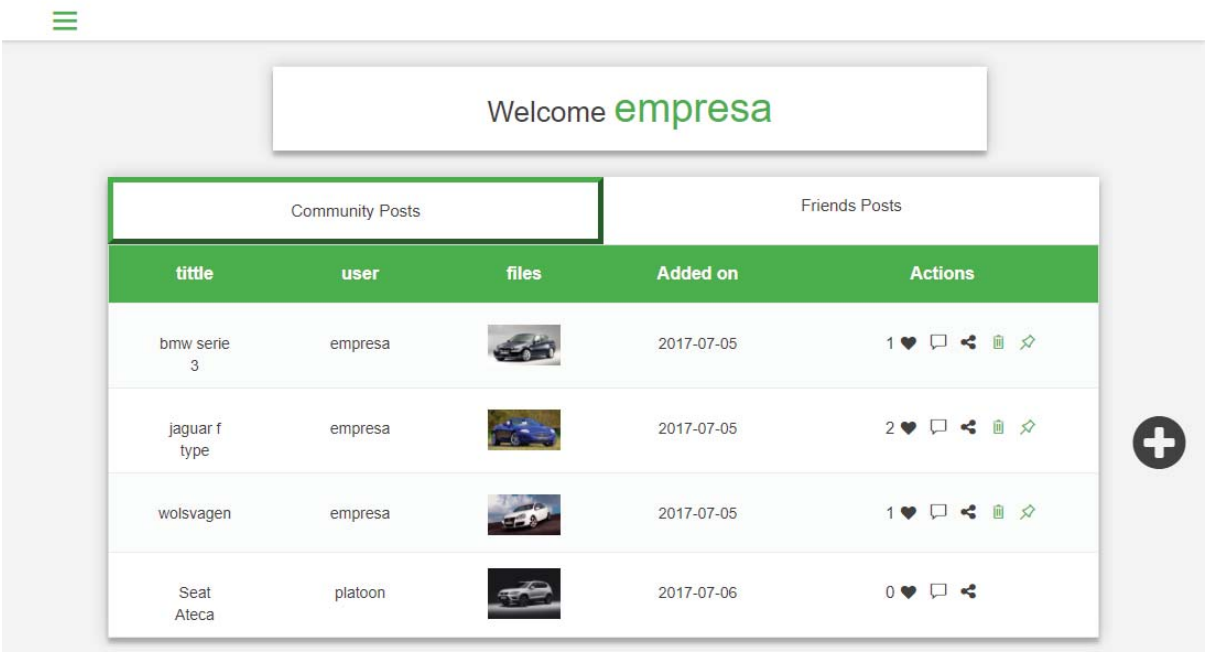

Figura 17: Pantalla de muro de post

Fornulario para crear un post:

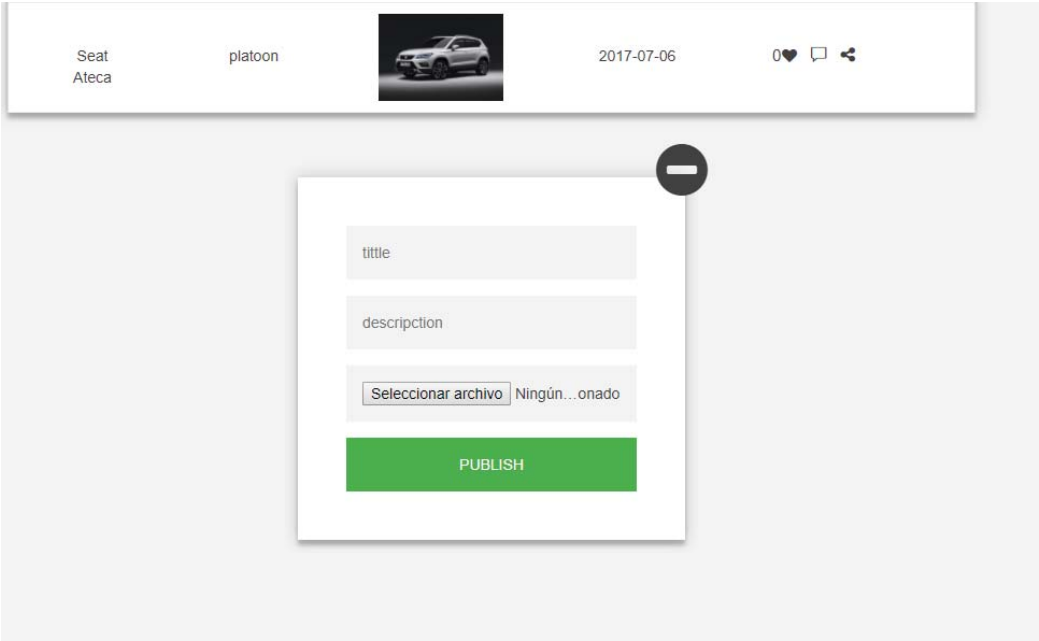

Figura 18: Form para crear un post

Vista en detalle del post:

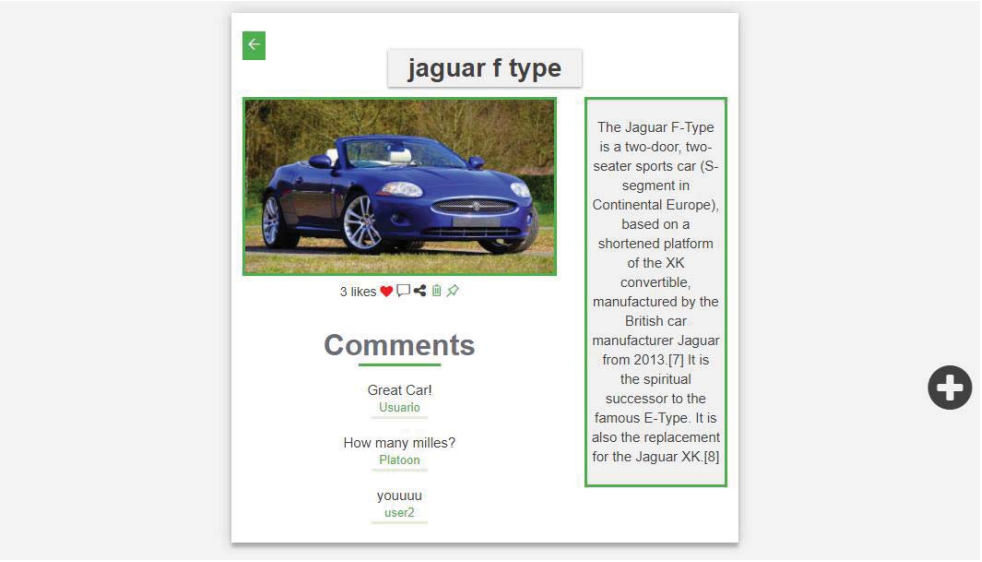

Figura 19: Detalle Post

Perfil vista 1:

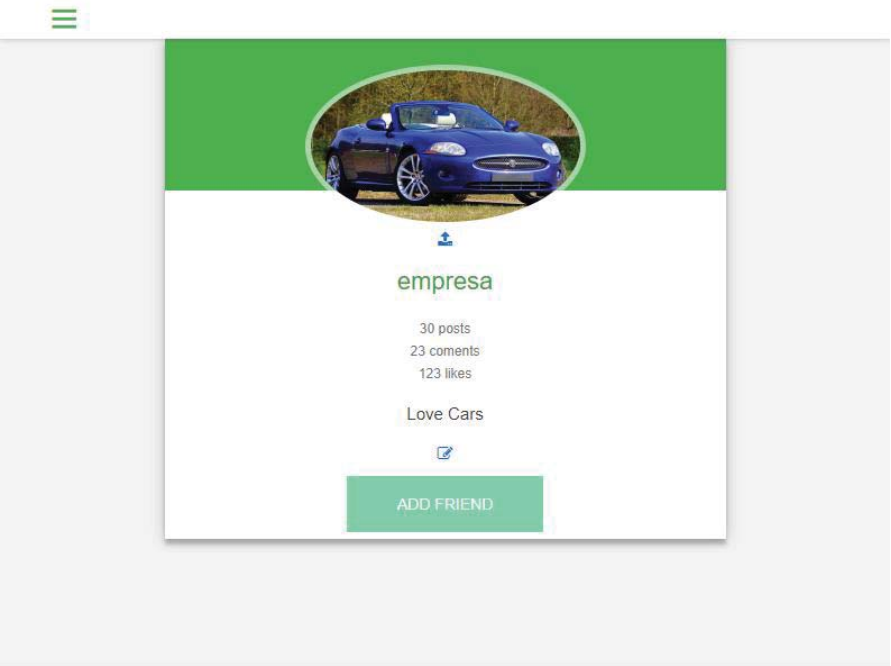

Figura 20: Acceso perfil usuario desconocido

Perfil vista 2:

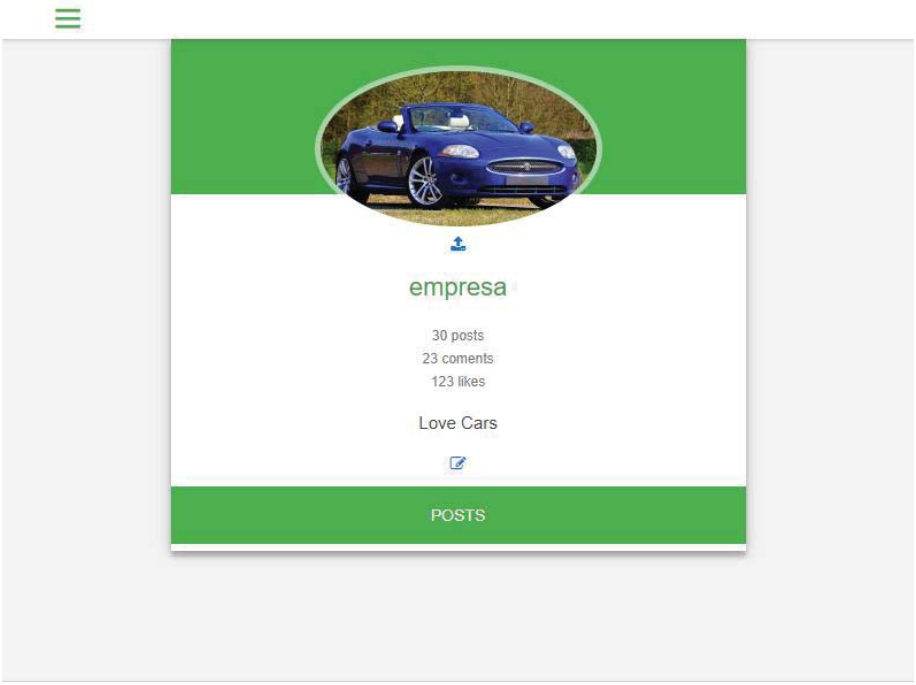

Figura 21: Acceso perfil usuario amigo

Perfil vista 3:

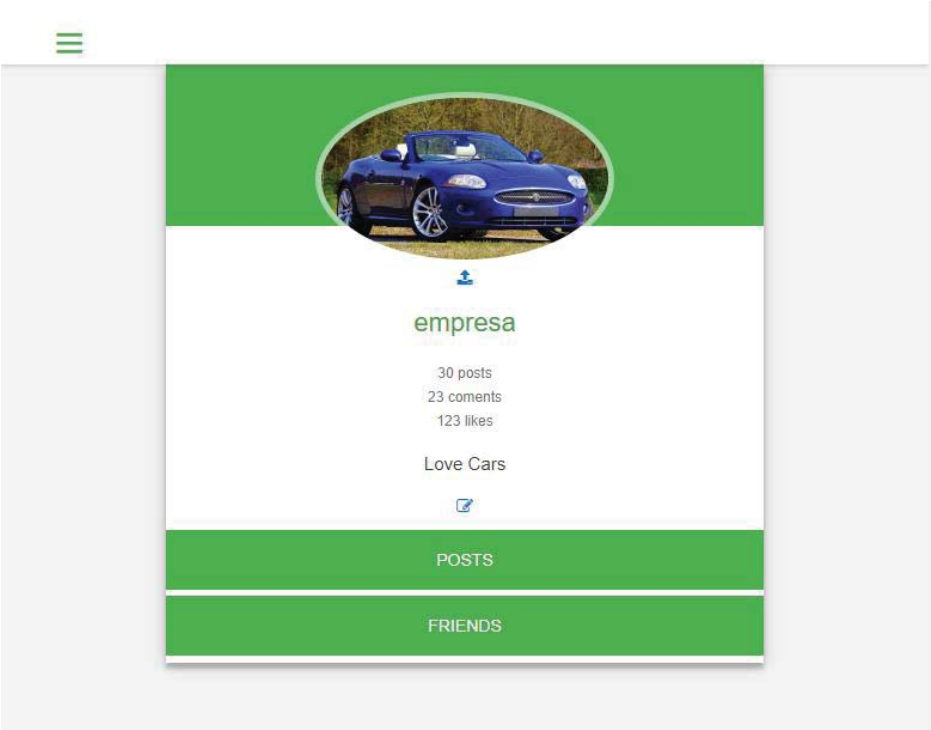

Figura 22: Acceso perfil usuario propio

Vista posts desde profile:

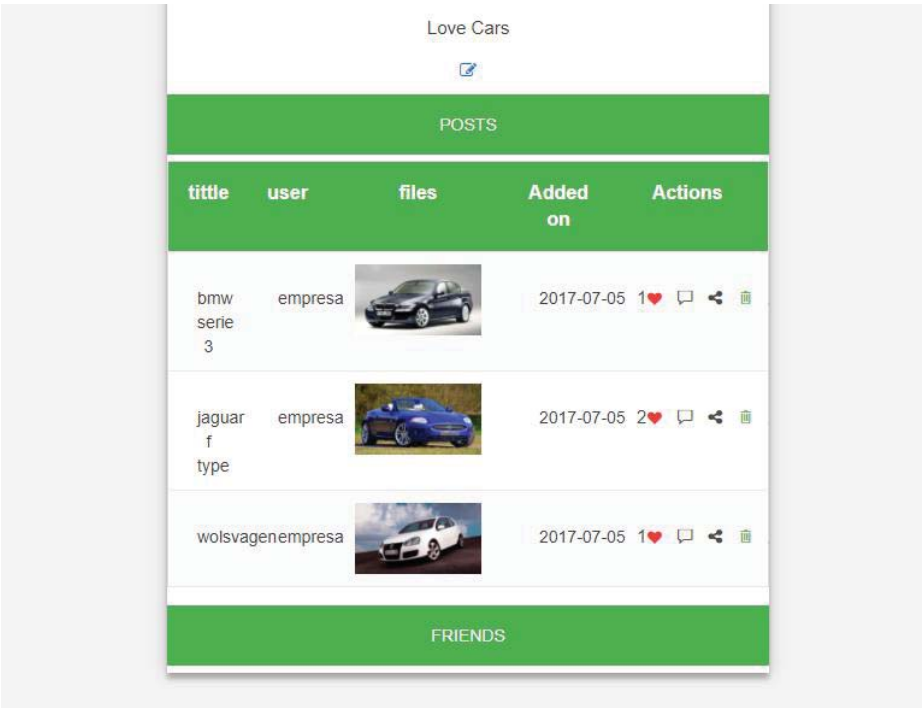

Figura 23: Posts perfil usuario

Vista amigos desde profile:

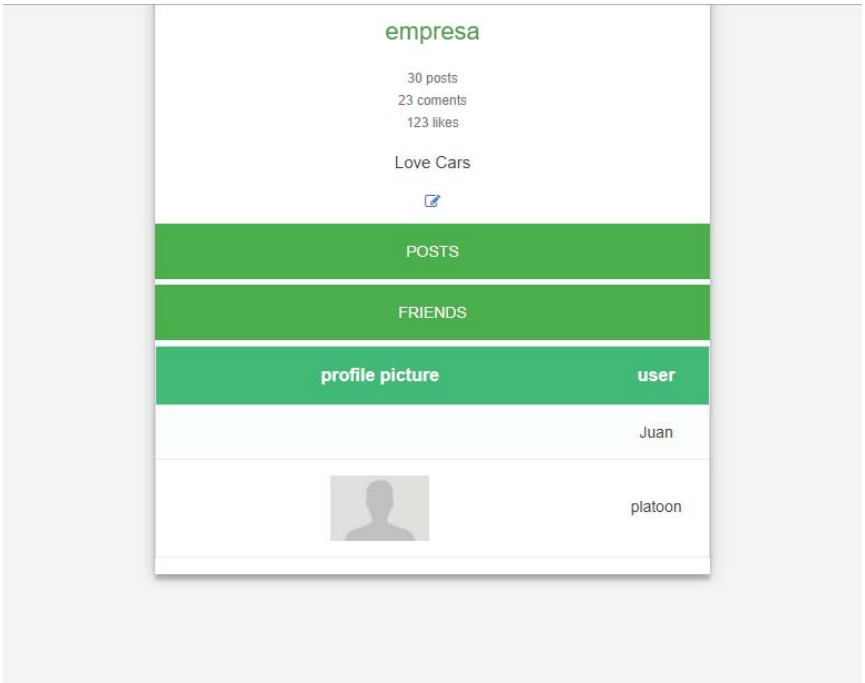

Figura 24: Amigos perfil usuario

5.3 Adaptabilidad a distintos dispositivos

En mi caso, todas las pruebas y desarrollo se han hecho desde mi equipo personal de sobremesa. Para montar la aplicación para en móviles habría que salvar el problema de los lenguajes nativos de cada teléfono, ya que el lenguaje sobre el que está programada la app no es el específico de IOS ni de Android.

Para ello se usa Apache Cordova. Anteriormente conocido como PhoneGap se trata de un framework de desarrollo de aplicaciones móviles. Permite a los programadores de software crear aplicaciones para dispositivos móviles utilizando CSS3, HTML5 y JavaScript en lugar de depender de APIs específicas de la plataforma móvil como Android, iOS o Windows Phone, es decir, sin tener que utilizar lenguajes nativos para su funcionamiento en cada plataforma. [11]

Esto permite diseñar una aplicación web al uso y, si su vista ha sido acondicionada en modo responsive como ha sido el caso de mi app su adaptación es muy sencilla, tanto para móvil como para tablet.

5.4 Integración

La integración de la aplicación es uno de los apartados más importantes del proyecto. El complementar otras apps con este módulo ha sido la motivación principal del TFG. Aquí explicaré cuáles son las distintas opciones que hay para realizarla.

Se pueden distinguir básicamente dos opciones. Una es a priori y otra a posteriori. Es decir, si la empresa que quiere integrar las funcionalidades de red social la quieren integrar antes o después del diseño e implementación de la aplicación.

El primer caso sería que la aplicación fuera diseñada e implementada contando desde el principio con el módulo de red social, es decir, a priori. De esta forma, las complicaciones que puede dar esta adaptación son ínfimas.

Para realizar este acoplamiento bastaría con añadir los campos oportunos a la base de datos de la app, para que se pudiera relacionar un usuario con el resto de tablas que lleva inherentes la gestión de una red social y que han sido explicadas en el apartado de bases de datos.

Y es que es aquí donde reside principalmente el problema, ya que la integración del resto de la aplicación sería tan sencilla como introducirla en el proyecto y hacer las referencias oportunas en base a su localización dentro del conglomerado de carpetas que forman la app.

Las referencias a la base de datos y la capa que maneja su gestión han de ser poco cambiantes puesto que se pueden producir inconsistencia en los datos ya que otras partes de la aplicación la manejan de una forma y es recomendable que tengan toda la información posible para la correcta interacción entre los manejadores y la base de datos.

En el caso de que se quisiera adaptar la aplicación a posteriori, sería la opción en principio más compleja. Se trata de integrar un módulo a una aplicación ya existente cuya lógica y base de datos están totalmente separados de este módulo. Esto siginifica que la tabla usuarios de la aplicación origen, que pretende registrar de forma inequívoca a los usuarios de la app,ya no va a ser única ya que existirá otra tabla usuarios en la base de datos correspondiente a este módulo. Lo que nos lleva a que tenemos dos claves primarias para representar el mismo usuario.

Para solventar este problema existe lo que conocemos como Single Sign On. Se trata de un procedimiento de autenticación que habilita al usuario para acceder a varios sistemas con una sola instancia de identificación. Dentro de este existen varios tipos como el Enterprise, Kerberos, FI o OpenIdel, pero en nuestro caso cabe destacar el Web single sign-on que trabaja con aplicaciones o recursos accedidos vía web. [12]

El objetivo es permitir autenticar a los usuarios en diversas aplicaciones, sin necesidad de volver a autenticar. Los accesos a estas son interceptados con la ayuda de un servidor proxy o de un componente instalado en el servidor o aplicación destino.

Los usuarios no autenticados que hacen login son redirigidos a un servidor o servicio web de autenticación y que regresa solo después de haber logrado un acceso exitoso o con un TOKEN de autenticación para la aplicación destino. Se utilizan cookies, parámetros por GET o POST para reconocer aquellos usuarios que acceden y su estado de autenticación.

De esta forma se podría solventar el problema de utilizar una base de datos distinta para representar usuarios que ya estaban registrados en la base de datos original. Esto facilita la adaptabilidad de la app a otras aplicaciones ya existentes.

6. Conclusiones

En este apartado se desarrollaré unas conclusiones en base a los resultados alcanzados y definidos en el capítulo anterior. Aquí se hace un análisis de los posibles usos de la app y otro del todo el proceso de realización del proyecto como conclusión de la carrera de ingeniería informática que he cursado en esta facultad durante los últimos 5 años.

6.1 Posibles usos

Como ya cité en el apartado de motivación dentro del capítulo de la introducción. La realización del proyecto obedece a una demanda por parte de empresas que han preguntado en la empresa en la que trabajo si existe la posibilidad de dotar de funcionalidades de red social dentro las apps que desarrollamos para ellos.

El uso de esta aplicación va a ser el de conectar a usuarios dentro de aplicaciones comerciales. Estas aplicaciones preferiblemente han de estar desarrolladas en el mismo framework Angular 2. Que junto a Ionic, es uno de los métodos para desarrollar aplicaciones móviles híbridas más usados hoy en día.

No es obligatorio que la aplicación origen esté montada sobre el mismo lenguaje. Por supuesto que esto mejora el tiempo de adaptación y la sencillez de esta adaptación. Pero, al tratarse de un framework de javascript al final es un lenguaje que trabaja sobre html. Y si esto sucede se pueden insertar iframes de código en cualquier aplicación web.

Como ya conté en el apartado de integración, esta adaptación se puede realizar de distintas formas, y en dicho apartado se explican con detalle.

El resultado final de mi desarrollo es que está previsto que empecemos a usar la apicación a partir del próximo curso. Donde vamos a realizar integraciones a priori de dos aplicaciones web. Antes de ponerla en práctica habrá que redefinir su diseño en cuanto a la vista. Ya que mi maquetación, puede ser mejorada por un profesional de la materia.

Además, añadiremos la funcionalidad de un sistema de mensajería interno para aumentar la interacción entre usuarios. Esto es algo que me hubiera gustado alcanzar en la realización del proyecto.

6.2 Conclusiones finales

Este proyecto ha significado el último proyecto realizado en la carrera. Supone el broche final para mi primera etapa de formación dentro del mundo de la informática. En este TFG puse en práctica conceptos que he ido aprendiendo a lo largo de esta formación universitaria y que al llegar aquí no creía que pudiera llegar a alcanzar.

En mi reciente formación como desarrollador de apps todavía no me he topado con todas la situaciones posibles que se pueden dar a la hora de diseñar y encontrar soluciones para resolver los distintos retos que se producen durante el proceso de creación de una aplicación. Aún con esto, he de destacar que llevar a cabo este proyecto me ha aportado mucha experiencia y he aprendido de los problemas que me han surgiendo en su realización, lo que significa conocimiento.

Algunos de estos conocimientos no creía haberlos interiorizado hasta que los tuve que poner en práctica. Ahí es donde entra el trabajo como uno de los elementos decisivos a la hora de asentar conceptos y sentirte realizado como informático, algo que te motiva a aprender más.

En el trabajo me di cuenta que en comparación con otros compañeros que sabían informática pero no habían estudiado esta carrera, yo era capaz de solucionar problemas con más precisión y sobre todo más nivel de detalle. Por ejemplo en cuanto a eficiencia o claridad se refiere.

Todas las dudas que me surgían eran en torno a las nuevas tecnologías que aprendía o sobre decisiones de diseño que muchas veces primero tenía que consultar con mis superiores. Pero en cuanto a los conceptos de programación que al entrar en la carrera eran nuevos para mí, no tuve grandes complicaciones. Lo que sobre todo me demostró a mí mismo que lo que había estudiado sabía llevarlo a la práctica.

Ha sido laborioso y ha necesitado de gran implicación por mi parte, pero ha merecido la pena y espero agradecer haber elegido una carrera como esta en un futuro.

7. Glosario

- **App híbrida** – Aplcación válida para distintos entornos móviles (Windows, IOS y Android)
- **App responsive** – Aplicación adaptable según la resolución de la pantalla del usuario. Es decir, para móvil, tablet y ordenador personal.
- **Servidor**- Ente que recibe peticiones por parte de un cliente, procesa la petición y devuleve una respuesta.
- **Cliente**- Ente que emite peticiones al servidor para mostrar datos y envía datos al servidor para que los almacene en la BD.
- **BD**- Base de datos- Almacena los datos que maneja una aplicación y mantiene las relaciones que existen entre los diferentes datos.
- **Diagrama E-R** – Esquema gráfico que represent las relaciones del modelo de la base de datos.
- **Modelo relacional** - El modelo relacional, para el modelado y la gestión de bases de datos, es un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos.
- **Front-end** – Parte del cliente
- **Back-end** – Parte del servidor
- **Scroll** – Permite navegar verticalmente dentro de la app
- **Pop up** – Cotenido individual que se muestra por encima del contenido base.
- **Android** – Sistema Operativo
- **iOS** – Sistema Operativo de Apple
- **Windows Phone** - Sistema Operativo de Microsoft
- **Back-office administrador** – Aplicación que permite una monitorización de la actividad de una app por parte de una administrador que además puede editar contenidos.

8. Bibliografía

Redes sociales:

[1]: http://definicion.de/red-social/

[2]: http://www.tiposde.org/internet/87-tipos-de-redes- sociales/#ixzz4kShGOMm3

Tecnologías:

[3]: https://www.ibm.com/developerworks/ssa/opensource/library/os-nodejs/

[4]: http://stackoverflow.com/questions/12694530/what-is-typescript-and-why-would-i-use-it-in-place-of-javascript

[5]: https://angular.io/

[6]: https://es.wikipedia.org/wiki/HTML

[7]: https://developer.mozilla.org/es/docs/Web/CSS

[8]: https://www.w3schools.com/js/

[9]: https://desarrolloweb.com/articulos/2458.php

[10]: http://php.net/manual/es/intro-whatis.php

[11]: https://cordova.apache.org/

[12]: https://en.wikipedia.org/wiki/List_of_single_sign-on_implementations

Repositorio:

https://www.sourcetreeapp.com/

https://bitbucket.org/

De interés:

http://victorroblesweb.es/2016/09/16/instalar-angular-2-crear-componente-version-final/c>

Este documento esta firmado por

Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
Fecha/Hora	Fri Jul 07 13:18:01 CEST 2017
Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
Numero de Serie	630
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)