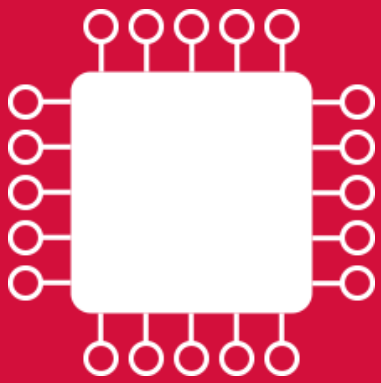
RPS

Red Social para entusiastas de Raspberry Pi



Jose M. Jiménez López

Trabajo Fin de Carrera

UOC Universitat Oberta de Catalunya

Índice

1 Resumen....3
2 Introducción....4
2.1 Razón y oportunidad del proyecto....5
2.2 Situación actual....5
3 Descripción del trabajo....7
3.1 Objetivos del proyecto....11
3.2 Resultados esperados....12
3.3 Análisis de riesgos....12
4 Planificación....14
4.1 Fases....15
4.1.1 Fase 0. Planificación. Análisis previo....15
4.1.2 Fase 1. Análisis de requisitos....15
4.1.3 Fase 2. Prototipo....16
4.1.4 Fase 3. Diseño....16
4.1.5 Fase 4. Implementación y pruebas....17
4.1.6 Fase 5. Documentación. Memoria final....17
4.2 Cronograma....18
5 Análisis de requisitos....20
5.1 Diseño orientado a usuarios....20
5.1.1 Usuarios y contexto....20
5.1.2 Diseño conceptual....35
5.1.3 Prototipado....42
5.1.4 Evaluación....53
6 Diseño técnico....56
6.1 Arquitectura del sistema....57
6.2 Modelo de datos....60
6.2.1 Parse....60
6.2.2 Modelo de datos de RPS....62
6.3 Modelo de clases....66
7 Implementación y pruebas....71
7.1 Tecnología aplicada....71
7.2 Composición del proyectos....74
7.3 Módulo principal....80
7.4 Módulo de usuarios....87
7.5 Módulo de entradas....92
7.6 Compilación y empaquetado....95
7.7 Plan de pruebas....97
8 Conclusiones....101
8.1 Retrospectiva....101
8.2 Continuación del proyecto....103
9 Bibliografía y fuentes consultadas....105

1 Resumen

El auge de los dispositivos móviles para interacción con redes sociales y el compartimiento de información es cada vez mayor, relegando a un segundo plano, cada vez más, el uso de estas redes en navegadores de escritorio. Por otro lado, los diferentes sistemas operativos obligan a realizar un desarrollo independiente para cada plataforma, lo que requiere de un gran esfuerzo desarrollar una aplicación para cada dispositivo. No obstante, existen herramientas que permiten realizar un único esfuerzo de desarrollo de una aplicación en un lenguaje concreto y 'migrarlo' a otras plataformas.

Este proyecto aplica esta tecnología para la creación de una red social de carácter específico para reunir todos aquellos proyectos que se han desarrollado o serán desarrollados basados en la tecnología de Raspberry Pi: pequeña placa de ordenador de coste reducido y con grandes posibilidades, debido al auge que ha tenido desde que se concibió y a toda la comunidad que se ha generado en torno a ella.

2 Introducción

El proyecto "RPS: Red social para entusiastas de Raspberry Pi", en adelante RPS, se desarrollará en el ámbito temático de "Desarrollo de aplicaciones móviles con HTML5". La idea principal de este proyecto se centra en dar visibilidad a la comunidad de Raspberry Pi, mediante una aplicación móvil destinada a compartir las experiencias de sus usuarios de una forma sencilla y atractiva.

Aunque Raspberry Pi se concibió como una plataforma tecnológica para la enseñanza, cada vez es mayor el número de personas que muestran interés por experimentar con esta tecnología, tanto en un ámbito particular como en el profesional. En internet hay innumerables proyectos creados por aficionados a este dispositivo, sin embargo, no existe ninguna plataforma especialmente diseñada para compartirlos o aprovechar la experiencia de sus creadores para apoyar a otros proyectos.

Por otro lado, esta idea surge de la propia experiencia de usuario y de otras personas que utilizan esta tecnología, además de la oportunidad de ofrecer a la comunidad de Raspberry Pi un entorno social donde compartir experiencias e inquietudes. Durante el desarrollo de este proyecto se ha contado con el apoyo de personas cercanas a esta tecnología que han aportado su punto de vista como usuario futuro de RPS.

2.1 Razón y oportunidad del proyecto

Raspberry Pi es una placa, de dimensiones reducidas, concebida por la Raspberry Pi Foundation, organización sin ánimo de lucro, para utilizarla en la enseñanza de la informática con un coste muy reducido, lo que la hace muy accesible. La ventaja de esta placa con respecto a la que se incorpora en un ordenador personal o un portátil es que está preparada para que se pueda modificar su configuración de recursos hardware de forma cómoda y sencilla, lo que es ideal para explicar a niños y jóvenes el funcionamiento de un ordenador.

Hay mucha información proporcionada por la propia fundación y por otras entidades o usuarios que enseñan cómo incorporar todos los recursos necesarios para obtener un ordenador con diferentes propósitos. La propia página web de Raspberry Pi incluye numerosos manuales de creación de proyectos nuevos y originales basados en esta tecnología. También existen numerosos foros y *blogs* en diferentes idiomas donde se comparte conocimiento, se apoyan proyectos y se resuelven dudas. Además, cada vez hay mayor tendencia a compartir estos proyectos en las redes sociales, como pueden ser Twitter, Facebook o Google+, lo que ha permitido que la ampliación de la comunidad de usuarios de este dispositivo crezca día a día.

Sin embargo, a pesar de que, cada vez más, la compartición de información se realiza a través de redes sociales, a día de hoy, no existe una red social específica para compartir el conocimiento sobre esta tecnología. Por este motivo, es el momento adecuado de desarrollar RPS como red social dedicada a la colaboración en proyectos relacionados con Raspberry Pi, puesto que se sienta sobre las bases de la participación del conocimiento (al igual que la Raspberry Pi Foundation) combinado con el paradigma de las redes sociales.

2.2 Situación actual

En la actualidad, existen diferentes medios digitales (como el de la página oficial de Raspberry Pi) que incorporan información de proyectos creados a partir de estas placas y componentes compatibles. Entre ellos cabe destacar los que aparecen en la tabla siguiente:

Sitio web	Descripción
Community Raspberry *http://www.raspberrypi.org/community/*	Desde la página oficial de Rasberry Pi se puede acceder a otros sitios web que exponen información de proyectos creados con esta tecnología. Son proyectos interesantes pero que está ubicados en lugares diferentes sin ninguna conexión.
RSPPI - RaSPberry PI *http://rsppi.blogspot.com.es/*	Blog particular en el que se muestran diferentes proyectos realizados por la misma persona explicados paso a paso. Es una muestra del interés que existe por esta tecnología entre sus aficionados.
Mis apuntes de... *http://misapuntesde.com/*	Otro blog particular en el que se muestran noticias que han ido surgiendo y que han llamado la atención del autor y también algún proyecto personal.
Tutoriales Raspberry Pi - Electroensaimada DIY *http://www.electroensaimada.com/raspberry-pi.html*	Otro blog que muestra el interés de los aficionados por crear nuevos proyectos y compartirlos con los demás. Este autor ha dividido sus proyectos según el grado de dificultad que pueda tener su desarrollo. En este caso, el autor muestra información de otras tecnologías, como puede ser Arduino.
Fresymetal *http://www.fresymetal.com/category/raspberry/*	En este sitio web se muestran proyectos basados en Raspberry Pi y en otras tecnologías. Al igual que en los casos anteriores, se muestra el interés por la creación de montar de forma 'casera' y barata sistemas que pueden parecer complejos o sofisticados, como puede ser el control de la domótica en casa.
Foro de la comunidad hispana de Raspberry Pi *http://www.raspberrypi-spanish.es/foro/*	En este foro se tratan múltiples temas entorno a esta tecnología, tanto a nivel hardware como a nivel de sistema operativo o de aplicaciones.
Foro de la comunidad hispana de Raspberry Pi *http://www.frambuesapi.co/*	En este sitio web se combinan foros, blogs, tutoriales y vídeo-tutoriales a través de Youtube.

La tabla anterior es una pequeña muestra de toda la información que hay en Internet actualmente. Toda esa información (proyectos caseros, manuales de instalación, ...) son una fuente inagotable de contenido para la red social que se quiere lanzar con RPS. A cambio, RPS proporciona a sus autores una mayor visibilidad entre los entusiastas de Raspberry Pi y una mayor cooperación entre ellos.

3 Descripción del trabajo

El diseño del proyecto RPS esta basado, como el área temática indica, en HTML5. HTML5 es una especificación que implica un conjunto de tecnologías que forman lo que se denomina la Plataforma Web Abierta. Esta última revisión de HTML tiene una capacidad y una versatilidad superior a la revisión anterior, de forma que proporciona acceso a los recursos de los dispositivos, como pueden ser audio y/o vídeo, mejoras en la gestión de recursos visuales, animaciones, nuevas capacidades de red, mejora semántica y recursos de accesibilidad y usabilidad.

HTML5 comprende la integración de un conjunto de tecnologías, pero nos centraremos principalmente en HTML, CSS, JavaScript y en algunos *frameworks* (palabra inglesa para definir marcos de desarrollo o librerías) creados para facilitar el uso de estas tecnologías.

La unión de estas tecnologías crea un entorno propicio para lo que hoy en día se conoce como SPA (*single-page application*), que no son más que aplicaciones web, en las que todo el código necesario, ya sea HTML, CSS o JavaScript, se descarga de una sola vez, en la carga inicial de la página. Esta forma de mostrar el contenido, ofrece al usuario una experiencia de uso mucho más fluida, similar a lo que podría ser una aplicación de escritorio, o una aplicación nativa en el caso de los dispositivos móviles. En este tipo de aplicaciones (una vez que se han descargado), las únicas peticiones que se realizan al servidor son para recuperar los datos necesarios para mostrar al usuario, como por ejemplo, las últimas noticias sobre un tema determinado, o el resultado de una búsqueda realizada.

De forma más específica, las tecnologías aplicadas al proyecto RPS son las siguientes:

- **HTML5**, **CSS3** y **JavaScript**, como especificaciones base del proyecto.

- Como *framework* de desarrollo sobre JavaScript, se ha incorporado uno en concreto que proporciona al proyecto una arquitectura definida por el patrón **M V C** (*model-view-controller*), que es un paradigma de desarrollo que separa la parte visual (*view*) o de interactividad de la lógica de negocio y almacenamiento de datos (*model*), de lo cual se encarga el controlador (*controller*).

  Aprovechando que los navegadores web modernos hace ya tiempo están implementando las características de HTML5, en los últimos años, han surgido un gran número de *frameworks* JavaScript que se benefician de ellas. Entre éstos, algunos de los más conocidos son Ember.js[1], AngularJS o Backbone.js[2].

  A continuación, se analizan varias opciones que se barajaron como *framework* principal para RPS:

  - **Ember.js,** *framework* de código abierto basado en el patrón MVC. De éste podemos destacar el concepto de Modelo, que proporciona al programador una abstracción en forma de clases de los objetos JSON que representan los datos de la aplicación, enviados y recibidos a la parte servidor de manera sencilla.

  - **AngularJS**, *framework* de código abierto desarrollado por Google. Una vez más siguiendo el patrón MVC, podemos destacar sus directivas, mediante las cuales, permite al desarrollador crear nuevas etiquetas o atributos HTML, que una vez tratados por el compilador HTML de AngularJS, añade funcionalidad de una manera sencilla y reutilizable dentro de la aplicación.

  - **Backbone.js**, una vez más, de código abierto. En este caso, no es siempre considerado como un *framework*, sino, más bien, como un conjunto de herramientas y convenciones para la creación de aplicaciones web. De él podemos destacar su pequeño tamaño, ya que su versión de producción ocupa tan solo 6.5 kB.

  Aunque los tres *frameworks* estudiados cumplían los requisitos necesarios para el

1 Ember.js - Ver página oficial: <http://emberjs.com/>

2 Backbone.js - Ver página oficial: <http://backbonejs.com/>

desarrollo del proyecto, se optó por AngularJS por varias razones: en primer lugar, es uno de los más ricos en cuanto a funcionalidad se refiere, proporcionando una gran cantidad de herramientas que permiten un desarrollo ágil; al mismo tiempo, permite una flexibilidad considerable gracias a componentes como las directivas o servicios; por último, y no menos importante, cuenta con una gran comunidad en Internet, asegurando que hay gran cantidad de contenido didáctico, y que el desarrollo y la resolución de bugs del propio *framework* cuenta con un gran número de desarrolladores alrededor de todo el mundo.

- Para la capa de visualización o vista, se utilizan los componentes que proporciona **Ionic Framework**[3], haciendo uso de un lenguaje llamado **Sass** (Syntactically Awesome Stylesheets), que permite escribir el CSS de una forma más dinámica y flexible.

- Para la lógica de datos, en lugar de crear un *backend* al que acceder directamente, siguiendo el patrón MVC, se ha optado por utilizar un servicio basado en el paradigma de *cloud computing*, es decir, un servicio público *online* de alta disponibilidad, que además, hasta un número concreto de transacciones, es gratuito. Este servicio es **Parse**[4], y proporciona la lógica de almacenamiento de información y ficheros de forma que el desarrollador de aplicaciones (ya sean nativas o no) para dispositivos se desligue del desarrollo del *backend*. El formato de los datos que se envían desde cliente al servidor y viceversa es **JSON**, ya que permite enviar la información como objetos y está soportado de forma nativa por todos los *frameworks* a utilizar y por el servicio Parse.

- Por último, se utiliza **Cordova**[5] para la encapsulación del proyecto HTML5, de forma que pueda ejecutarse desde una aplicación (o comúnmente bautizado como *app*) en nuestro dispositivo móvil, sin necesidad de abrir el navegador web.

En la siguiente ilustración, se puede ver un esquema de integración de los diferentes componentes que forma la arquitectura del proyecto, en el cual se diferencia claramente la gestión de la información (desde un servicio en nube) y el tratamiento de dicha información desde el dispositivo móvil:

3 Ionic - Ver página oficial: <http://ionicframework.com/>

4 Parse - Ver página oficial: <https://www.parse.com/>

5 Cordova - Ver página oficial: <http://cordova.apache.org/>

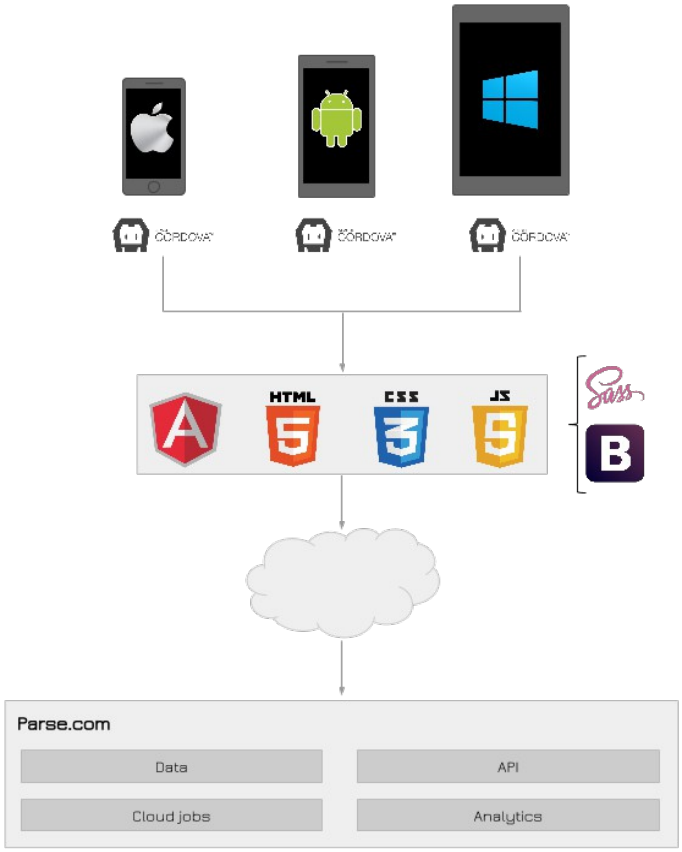


Ilustración 1: Arquitectura tecnológica de RPS

Desde el punto de vista del desarrollo en sí del proyecto, se utilizan las siguientes herramientas:

- **Node.js** como base para la creación y mantenimiento del entorno de programación del proyecto.
- **Bower**, es un gestor de paquetes para el desarrollo de aplicaciones basadas en HTML/CSS/JS.
- **Grunt**, para la automatización de tareas de desarrollo, como puede ser la concatenación y minimización de ficheros JavaScript o la generación del código CSS en base a SASS.
- **Git**, como gestor de versiones de código fuente.

3.1 Objetivos del proyecto

El objetivo principal del desarrollo de RPS es **la creación de una red social centrada en Raspberry Pi**, que está concebida para fomentar el interés de alumnos, profesores, aficionados, etc., por la tecnología y aumenta la creatividad y la innovación en el uso de dispositivos y sistemas informáticos para la resolución de problemas de diversa complejidad. Para ello, el método para compartir información es muy similar al utilizado por otras redes que ya existen en la actualidad, es decir, crear elementos de información o 'entradas', de forma que, el resto de integrantes de la red puedan visualizarlos, comentarlos y volver a compartirlos. La base social irá en consonancia con las más importantes redes sociales, para que la experiencia de usuario sea lo más satisfactoria posible.

De una forma más específica, se detallan los siguientes objetivos:

- RPS debe ser una aplicación **versátil** que pueda ejecutarse en la mayor cantidad de plataformas móviles existentes en la actualidad. Que además pueda adaptarse a nuevas configuraciones de una forma sencilla.

- Asimismo, debe ser **intuitiva** y permitir que se pueda interactuar de una forma cómoda y rápida. Es importante que la persona que la utilice no se sature ni se aburra, por lo que deberá de proporcionar una interfaz amena y atractiva.

- Por otro lado, se pretende fomentar la **comunicación** entre todos los integrantes de la comunidad de Raspberry Pi e incluso conseguir que personas que no conozcan esta tecnología, comiencen a interesarse por ella mediante la interacción con otras redes sociales.

- Un usuario podrá crear una nueva entrada en la aplicación que conlleve la incorporación de múltiples elementos (texto e imágenes). Debe permitir crear y compartir este contenido de manera que, en el caso de que el volumen de información sea considerable, la persona que lo utilice pueda ver y gestionar de forma **clara** los elementos que ya haya incluido en el artículo.

- Que sea una plataforma para conocer las últimas **novedades** relacionadas con Raspberry Pi o con cualquier dispositivo hardware o software relacionado con él.

3.2 Resultados esperados

Con el desarrollo de este proyecto se espera construir una comunidad unida en torno a la tecnología de Raspberry Pi, de forma que se genere una gran masa social que proporcione una gran base de conocimiento, disponible para cualquiera que tenga interés. El propósito es la creación de una aplicación para dispositivos móviles, que se pueda instalar en cualquiera de ellos de forma independiente de su hardware o de su sistema operativo. Debe ser adaptable, en mayor medida, a cualquier tamaño de pantalla, ya sea en un teléfono inteligente o en una tableta. De esta forma, se podrá llegar a una cantidad mayor de personas que se la instalen en su dispositivo.

En un futuro más lejano, se espera que RPS sea una herramienta de referencia para profesores, alumnos, aficionados o expertos en estos dispositivos, una vez que se haya llegado a un número considerable de proyectos que formen parte de la base de datos de la aplicación. Para ello, se seguirá trabajando de forma continuada en la mejora del proyecto, más allá de la presentación del mismo como trabajo de fin de carrera.

Por otra parte, aunque el objetivo del proyecto es llegar a los usuarios de Raspberry Pi, el desarrollo de RPS debe de ser lo más independiente posible del tipo de contenido que gestione, ya que esto permitirá en un futuro adaptarlo a otras temáticas diferentes a la compartición de información de proyectos basados en esta tecnología. Desde el inicio del proyecto se tendrá muy presente la obtención de este resultado.

3.3 Análisis de riesgos

El mayor riesgo al que se enfrenta RPS es que las personas a las que va dirigido (aficionados, profesores, alumnos, expertos, etc.) no muestren interés por él y no se involucren en la inclusión de nuevos proyectos. Para ello, será decisiva la presentación de RPS a la comunidad de Raspberry Pi, ya que como pasa con la mayoría de aplicaciones para dispositivos móviles, si no capta la atención de los usuarios desde el principio al final se va quedando en el olvido.

Por este motivo, en el momento que se lance la aplicación a las diferentes tiendas de aplicaciones, se intentará contar con la participación de la Raspberry Pi Foundation, puesto que RPS es un proyecto sin ánimo de lucro, lo que encaja perfectamente con los principios de esta fundación. Además, se intentará acceder a los profesionales (o aficionados) que tengan mayor peso dentro de la comunidad para pedir igualmente su apoyo y que utilicen RPS como una herramienta primaria.

Otro riesgo previsible está derivado del uso externo de la base de datos a través de un servicio 'en nube' de terceros, como se verá más adelante. Este servicio es gratuito hasta cierto número de peticiones que se realicen a través de RPS, por lo que, una vez superado ese número, pasará a ser de pago. En el momento del desarrollo del proyecto y, posiblemente, en su puesta en funcionamiento no será un problema; sólo lo será en el momento en el que el número de usuarios que utilicen RPS a la vez sea muy elevado.

En previsión a este problema, habrá que realizar una estimación de costes en función de la escalabilidad de RPS y buscar inversores, privados o públicos, que permitan soportar el coste del servicio. Otra posibilidad, es buscar otras alternativas de terceros más económicas o, incluso, la posibilidad de desarrollar este servicio. Aunque esta opción tiene más desventajas que ventajas: la ventaja de utilizar un servicio de terceros respecto a uno propio es básicamente la adquisición y mantenimiento de la infraestructura necesaria para soportarlo, que sobre todo al inicio de la actividad, conlleva un alto coste económico y de carga de trabajo. Por este motivo, no es una opción viable el propio desarrollo del servicio de almacenamiento de la información de RPS.

Otro riesgo derivado del uso de un servicio de terceros es que ese servicio deje de funcionar por algún motivo. En este caso, no podremos solventar nosotros mismos el problema, y en caso extremo, nos obligaría a buscar un nuevo proveedor. Por este motivo, es importante la independencia máxima del código de la aplicación con el servicio en nube para el almacenamiento.

4 Planificación

Para garantizar el cumplimiento de plazos del proyecto respecto a la planificación de entrega de las PECs de la asignatura, la planificación se ha realizado acorde a la metodología basada en el "Ciclo de Vida en Cascada", o lo que es lo mismo, el "Ciclo de Vida Clásico". De esta forma las entregas de las diferentes fases se agruparán de la siguiente forma:

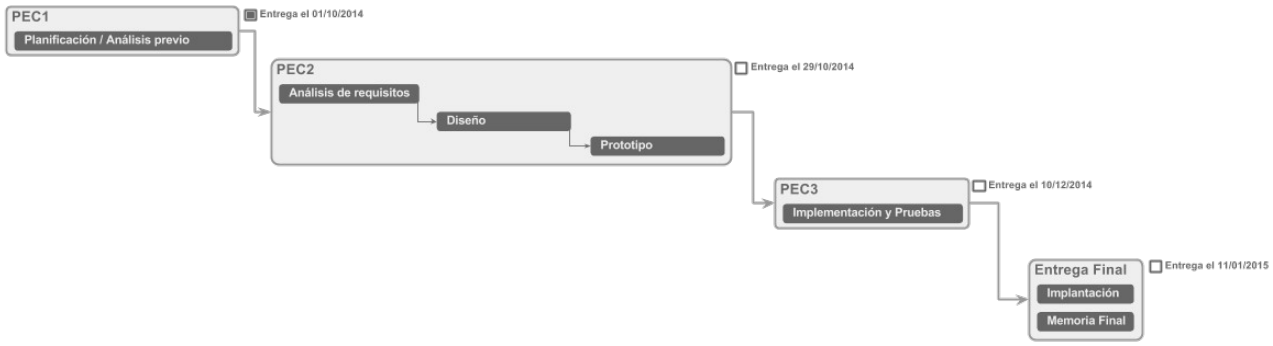


Ilustración 2: Ciclo de Vida en Cascada, adaptado a las PECs de entrega del proyecto.

De esta forma, durante las primeras etapas se han tomado todas las decisiones necesarias para la correcta construcción del proyecto. Esta es una metodología adecuada para este proyecto, puesto que tiene un ámbito bastante acotado y unos tiempos de desarrollo de las fases reducidos. Posteriormente, si después de la entrega, me decanto por continuar con el desarrollo de este proyecto, ampliando y/o modificando sus funcionalidades, es posible que cambie la metodología a

aplicar por otra adaptada a proyectos en continuo cambio, como sería una metodología ágil, por ejemplo, Scrum.

A continuación, detallo las fases y su cronología contempladas para el desarrollo del proyecto RPS.

4.1 Fases

4.1.1 Fase 0. Planificación. Análisis previo

Durante esta fase se analizan los requisitos previos que ha de tener la arquitectura de la plataforma, el conjunto de lenguajes, librerías y herramientas a utilizar y en base a esto, definir el resto de fases que conforman el desarrollo del proyecto, indicando sus tareas y elementos entregables que se generarán al finalizar las mismas, y la duración de tanto de las tareas como de las fases completas. A continuación, se muestran los objetivos, actividades y resultados de esta fase:

Fecha inicio: 17/09/14	**Fecha fin:** 01/10/14
Objetivos Esta fase debe proporcionar una guía de trabajo para el desarrollo del proyecto, estableciendo la metodología y la cronología de las actividades.	
Tareas - 0.1 Definir el ámbito funcional y tecnológico del proyecto - 0.2 Definir los recursos hardware y software requeridos para la realización del proyecto - 0.3 Definir las fases y actividades que compondrán el proyecto - 0.4 Definir cronograma del proyecto - 0.5 **Entrega PEC 1 (hito 1)**	**Entregables** - Documento del Plan de trabajo para el proyecto RPS

Al finalizar esta fase se ha completado la PEC1, por lo que la documentación generada se ha subido a la plataforma de la asignatura de la UOC.

4.1.2 Fase 1. Análisis de requisitos

Durante esta fase se han detallado las funcionalidades del proyecto, las restricciones y las diferentes casuísticas aplicadas, con un nivel de detalle que ha permitido su desarrollo. Aunque en este proyecto el papel de analista y el de programador lo realiza la misma persona, el documento de requisitos se detalla de forma que pueda utilizarlo cualquier programador y desarrollar las funcionalidades que contempla.

Fecha inicio: 02/10/14	**Fecha fin:** 19/10/14
Objetivos Documentar de forma concisa los requisitos que tendrá el proyecto, tanto funcionales como no funcionales, sin entrar en el cómo se hará. Para realizar esta tarea se aplicado la metodología de diseño orientado a usuario, en la que se ha tenido en cuenta principalmente la interacción de los usuarios con la aplicación para que sea atractiva y usable.	
Tareas - 1.1 Definición de los usuarios y los contextos de uso - 1.2 Diseño conceptual del proyecto	**Entregables** - Documento de Análisis de Requisitos

4.1.3 Fase 2. Prototipo

Esta fase permite crear un esbozo de lo que será el resultado final del proyecto, de forma que antes de programar se tenga claro que lo que se va a desarrollar está acorde con las necesidades del proyecto y para que se pueda analizar la adecuación del mismo.

Fecha inicio: 20/10/14	**Fecha fin:** 29/10/2014
Objetivos Tener una referencia del resultado del proyecto antes de desarrollar el mismo.	
Tareas - 2.1 Desarrollo del prototipo - 2.2 **Entrega PEC 2 (hito 2)**	**Entregables** - Capturas de pantalla del prototipo y *sketches*

4.1.4 Fase 3. Diseño

En esta fase, se ha concretado cómo se resuelven las cuestiones planteadas durante la fase de análisis. Para ello, se especifica cómo se va a diseñar la arquitectura global del proyecto, el API de comunicación con el backend, el diseño de clases, las estructuras de datos, interfaces de usuario, etc.

Fecha inicio: 30/10/14	**Fecha fin:** 11/11/14
Objetivos Describir todos los elementos que van a formar parte del sistema, así como describir las pruebas que serán requeridas para garantizar el correcto funcionamiento del software desarrollado.	
Tareas - 3.1 Diseño de la arquitectura global del proyecto. Diseño de clases	**Entregables** - Documento de Especificación del

- 3.2 Diseño lógico de datos y API de acceso al backend - 3.3 Diseño de las interfaces de usuario. Aplicación de usabilidad. - 3.4 Diseño de las pruebas de integración y de comunicación.	Diseño - Documento de Especificación de Pruebas

4.1.5 Fase 4. Implementación y pruebas

En esta fase, se desarrolla todo el código requerido en base al diseño y el prototipo creados anteriormente. Además se realizan todas las tareas de configuración requeridas tanto a nivel de *hardware* como de *software* para que el sistema funcione correctamente. Por otra parte, se implementan y ejecutan todas las pruebas pertinentes, tanto a nivel unitario como a nivel de integración de componentes y comunicaciones.

Fecha inicio: 12/11/14	**Fecha fin:** 10/12/14
Objetivos Desarrollar el software RPS según las directrices funcionales y de diseño especificadas en las fases anteriores.	
Tareas - 4.1 Implementación de servicios de integración con la lógica de servidor. - 4.2 Implementación de la gestión de usuarios y seguimiento de usuarios - 4.3 Implementación de la gestión de entradas y contenidos. - 4.4 Implementación y ejecución de pruebas unitarias. - 4.5 Pruebas de integración y comunicación - 4.6 Empaquetado de la aplicación para Android - 4.7 **Entrega PEC 3 (hito 3)**	**Entregables** **- Código fuente** - Binarios empaquetados (para Android) - Documento de Resultado de Pruebas

4.1.6 Fase 5. Documentación. Memoria final

Por último, durante esta fase se comprueba que el *software*, principalmente el *backend* que, como se ha indicado, será un servicio en nube, esté disponible para que pueda probarlo el tribunal durante toda la fase de revisión. Además, se redacta la memoria final y se incorporan todos los documentos generados durante el desarrollo del proyecto (se omiten documentos o partes de los mismos que no proporcionen información relevante para la memoria).

Fecha inicio: 11/12/14	**Fecha fin:** 11/01/14
Objetivos - Poner a disposición del tribunal todos los medios necesarios para que puedan evaluar el proyecto. - Concluir la Memoria Final del Proyecto, incluyendo toda la información necesaria generada durante el transcurso del proyecto.	
Tareas - 5.1 Preparación Memoria Final - 5.2 **Entrega Memoria Final (hito 4)**	**Entregables** - Memoria Final

4.2 Cronograma

A continuación, se muestra el esfuerzo requerido para realizar cada una de las tareas del proyecto:

Nombre	Duración	Inicio	Terminado
⊟Fase 0. Planificación. Análisis previo	**14 days**	**18/09/14 8:00**	**1/10/14 17:00**
0.1 Definir el ámbito funcional y tecnológico del proyecto	3 days	18/09/14 8:00	20/09/14 17:00
0.2 Definir los recursos hardware y software requeridos...	4 days	21/09/14 8:00	24/09/14 17:00
0.3 Definir las fases y actividades que compondrán el pr...	5 days	25/09/14 8:00	29/09/14 17:00
0.4 Definir cronograma del proyecto	2 days	30/09/14 8:00	1/10/14 17:00
0.5 Entrega PEC 1	0 days	1/10/14 17:00	1/10/14 17:00
⊟Fase 1. Análisis de requisitos	**5 days**	**2/10/14 8:00**	**6/10/14 17:00**
1.1 Definición de los requisitos funcionales	3 days	2/10/14 8:00	4/10/14 17:00
1.2 Definición de los requisitos no funcionales	2 days	5/10/14 8:00	6/10/14 17:00
⊟Fase 2. Diseño	**17 days**	**7/10/14 8:00**	**23/10/14 17:00**
2.1 Diseño de la arquitectura global del proyecto	3 days	7/10/14 8:00	9/10/14 17:00
2.2 Diseño lógico de datos	6 days	10/10/14 8:00	15/10/14 17:00
2.3 Diseño de las interfaces de usuario. Aplicación de us...	5 days	16/10/14 8:00	20/10/14 17:00
2.4 Diseño de las pruebas de integración y de comunicac...	3 days	21/10/14 8:00	23/10/14 17:00
⊟Fase 3. Prototipo	**6 days**	**24/10/14 8:00**	**29/10/14 17:00**
3.1 Desarrollo del prototipo	6 days	24/10/14 8:00	29/10/14 17:00
3.2 Entrega PEC 2	0 days	29/10/14 17:00	29/10/14 17:00
⊟Fase 4. Implementación y pruebas	**42 days**	**30/10/14 8:00**	**10/12/14 17:00**
4.1 Implementación de servicios de integración con la lóg...	8 days	30/10/14 8:00	6/11/14 17:00
4.2 Implementación de la gestión de usuarios y seguimie...	8 days	7/11/14 8:00	14/11/14 17:00
4.3 Implementación de la gestión de entradas y contenid...	15 days	15/11/14 8:00	29/11/14 17:00
4.4 Implementación y ejecución de pruebas unitarias.	4 days	30/11/14 8:00	3/12/14 17:00
4.5 Pruebas de integración y comunicación	5 days	4/12/14 8:00	8/12/14 17:00
4.6 Empaquetado de la aplicación para Android	2 days	9/12/14 8:00	10/12/14 17:00
4.7. Entrega PEC 3	0 days	10/12/14 17:00	10/12/14 17:00
⊟Fase 5. Documentación. Memoria Final	**32 days?**	**11/12/14 8:00**	**11/01/15 17:00**
5.1 Preparación Memoria Final	32 days?	11/12/14 8:00	11/01/15 17:00
5.2 Entrega Memoria Final	0 days	11/01/15 17:00	11/01/15 17:00

Por último, se muestra la cronología anterior en el siguiente Diagrama de Gantt, donde se pueden ver además las dependencias entre tareas:

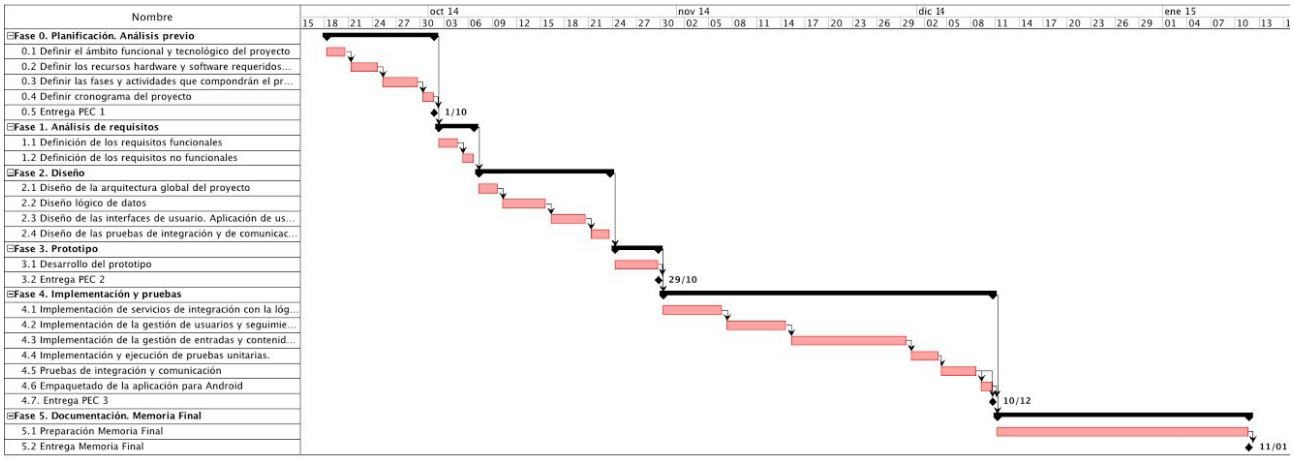


Ilustración 3: Diagrama de Gantt del proyecto RPS.

5 Análisis de requisitos

En este capítulo se describe el comportamiento y las características del proyecto en base al análisis realizado de los usuarios potenciales y de entorno, así como de un estudio de otros productos similares del mercado. Para ello, para la definición y diseño del proyecto se ha aplicado la metodología de **diseño centrado en el usuario**.

5.1 Diseño orientado a usuarios

El diseño orientado a usuarios (DCU) permite orientar el desarrollo del proyecto para proporcionar una experiencia de usuario adecuada para el tipo de dispositivo y de usuarios a los que va dirigida la aplicación. Por este motivo, para la obtención de requisitos se ha contado con la colaboración de un grupo de usuarios de un perfil técnico que conocen, en mayor o menor medida, la tecnología Raspberry Pi y las redes sociales, de forma que se contase con diferentes puntos de vista para la definición de la aplicación.

5.1.1 Usuarios y contexto

Esta fase del proyecto se centra en la obtención de los requisitos que conforman el proyecto, en base a los propios usuarios y a la experiencia de usuario esperada para dispositivos móviles. Para ello se siguen las metodologías contempladas en el diseño centrado en el usuario. La adecuación del desarrollo del proyecto al DCU nos permitirá dar una mejor aproximación al enfoque que los

propios usuarios puedan tener del proyecto antes de diseñarlo.

Esta fase de la análisis se centra en responder principalmente a las siguientes cuestiones:

- ¿A quién va dirigida la aplicación?
- ¿Qué se persigue con esta aplicación?
- ¿Cuál es el nivel de conocimiento de los usuarios a quienes va dirigida?
- ¿Qué se espera de esta aplicación?
- ¿Qué información podrían necesitar los usuarios de la aplicación?
- ¿Cuál es la mejor forma de mostrar la información a los usuarios?

Para responder a las preguntas anteriores se utilizan dos métodos distintos de indagación que se complementan entre sí, ya que uno de los métodos es cualitativo, la dinámica de grupos; y el otro método es cuantitativo, el *benchmarking*.

- Se ha elegido el método de las dinámicas de grupo, puesto que es una forma bastante eficiente de encontrar un punto común de los criterios de varios usuarios haciendo uso del debate entre ellos.
- Por otra parte, desde el punto de vista cuantitativo, se hace un análisis competitivo de varias aplicaciones existentes, tanto para plataforma Android como Apple (como mayores referentes de aplicaciones móviles), como en el propio *store* de Raspberry Pi, de las que se extraen conclusiones sobre su experiencia de usuario y la respuesta de los usuarios ante estas aplicaciones.

5.1.1.1 Dinámicas de grupo

Para llevar a la práctica este tipo de entrevista a un grupo de usuarios potenciales, se ha hecho uso de una de las redes sociales estudiadas en este proyecto: Google+. Para ello, se ha creado lo que en Google+ se denominan comunidades, la forma de proceder ha sido la siguiente:

- Una vez creada la comunidad, se ha invitado a varios amigos, compañeros de profesión y conocidos que tienen en menor o mayor grado interés por la tecnología relacionada con

Raspberry Pi, ya sea como usuarios básicos o bien, como desarrolladores y técnicos que trabajan con esta tecnología o forma parte de sus hobbies.

- Posteriormente, se han creado una serie de entradas, de forma que cada una de ellas es una pregunta de la entrevista común. De esta forma, los usuarios pueden participar aportando sus opiniones como comentarios en esas entradas. Esto posibilita que cada usuario pueda aportar su opinión una o más veces, según se esté debatiendo cada pregunta.

- Se mantiene el debate abierto durante una semana, para facilitar a los usuarios el poder responder según sus horarios y disponibilidad.

- Finalmente, se extrae toda la información proporcionada para realizar el análisis de la misma.

A continuación se puede ver una captura de pantalla de la comunidad creada en G+ con un extracto de las preguntas realizadas en la entrevista grupal. La comunidad puede ser consultada en el siguiente enlace: https://plus.google.com/communities/114410348605400109334

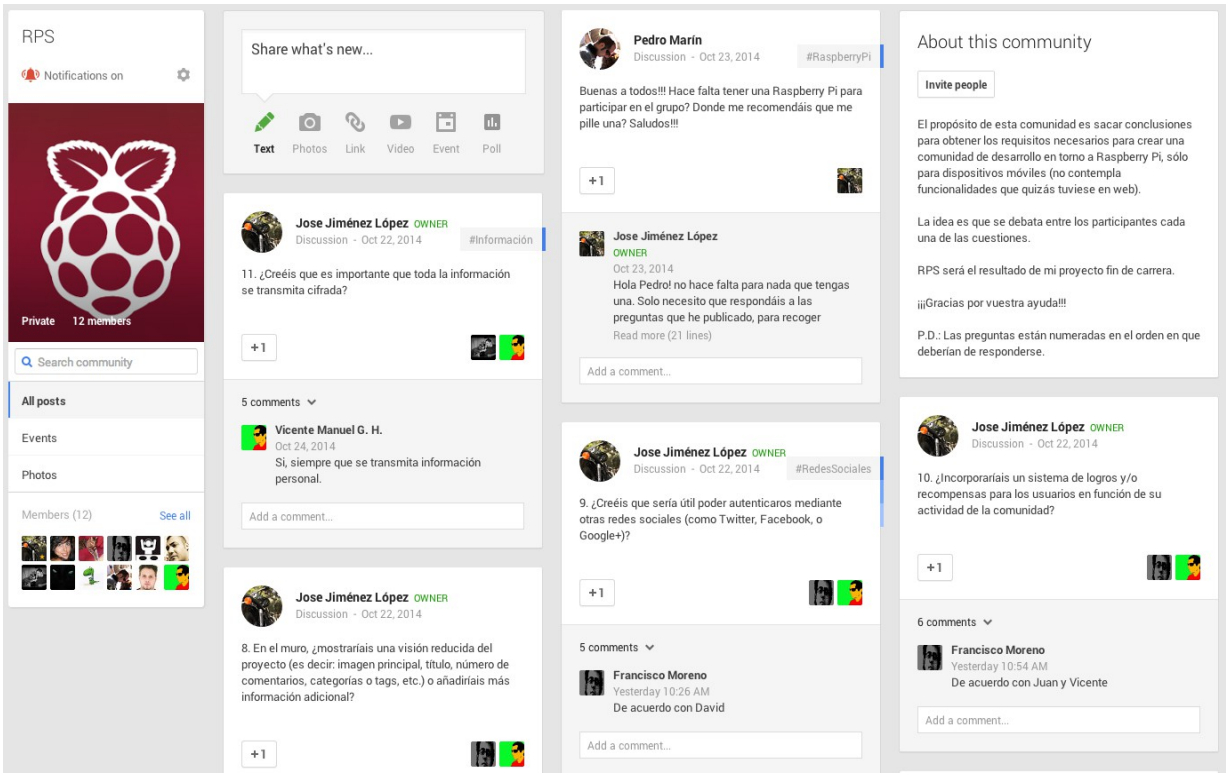


Ilustración 4: Comunidad Google+ usada para el debate.

En la siguiente tabla muestro todas las preguntas realizadas y expuestas de forma que sus respuestas sean claras y concisas y que tengan relevancia a la hora de definir los requisitos de una forma u otra:

Pregunta	Respuestas destacadas
1. ¿Creéis que es una buena idea crear una red social para realizar el seguimiento de novedades y proyectos que se pueden realizar con Raspberry Pi? En caso de que no os parezca una buena idea, ¿qué haríais en su lugar?	· Puede ser una buena idea tener una red social propia, ya que hay muchos blogs y muchas páginas de tutoriales en la web, pero no hay nada pensado para móviles. · A mi me parece bien, habría que ver que no haya otros proyectos iguales, lo mismo participando en algún otro que ya haya se podría aportar mucho más que comenzando uno desde el principio.
2.a. En el caso de que queráis añadir un proyecto que hayáis realizado con Raspberry Pi, ¿Creéis que sería mejor poner toda la información de una vez? ¿o quizás separarla en diferentes pasos que se pudieran editar de forma individual?	· Podría quedar más claro si se hace paso a paso, de forma que puedas ir desplazándote, por ejemplo, horizontalmente por cada uno, deslizando el dedo lateralmente. · Me parece mejor por pasos, tanto para seguir el tutorial como par ir haciéndolo con la app.
2.b. Siguiendo la pregunta anterior: La mayoría de las redes sociales no te permiten guardar un borrador de tu entrada, ¿creéis que en este caso sería necesario crear un borrador e ir rellenándolo a medida que se va avanzando en el proyecto?	· Es interesante poder ir guardando los cambios a medida que vas añadiendo nuevos componentes y no esperar a tenerlo todo. · Sería una gran idea. · Un borrador nunca estorba y es fácil de implementar.
3. Además de imágenes y texto, ¿creéis que sería útil añadir URLs de vídeos accesibles desde otros servicios como Youtube?	· Indispensable. · Yo creo que sería más interesante subir los vídeos directamente desde el móvil, pero no sé si eso implica contratar algún servicio de almacenamiento para los vídeos. Si no es posible, lo de las URLs está bien. · Subir los videos directamente es lo mejor, en caso de no poderse las URL,creo que el tener que usar un servicio externo retiene mucho a los usuarios a la hora de subir videos.
4. ¿Cuál creéis que es mejor método para organizar las entradas: usar categorías o etiquetas?	· Las etiquetas son más fáciles para el usuario y parece ser la tendencia actual. Además de que puedes usarlas en publicaciones en otras redes sociales. · De acuerdo con los demás. Las etiquetas resultan más fácil de aplicar si eres un usuario y es la tendencia en las redes sociales. · A mi me resultan más útiles las etiquetas.
5. ¿Creéis que para acceder al contenido desde la aplicación móvil deberíais estar	· No, en absoluto. Prefiero tener la opción de

siempre autenticados? (La mayoría de las redes sociales actuales, aunque permiten ver desde web artículos públicos, obligan al usuario a estar autenticado antes de poder usar la aplicación).	identificarme. · Si, la autenticación la veo necesaria para escribir, puesto que si no la red se llenaría de comentarios anónimos. La gente es vaga por naturaleza.
6. Estoy pensando que sería interesante poder indicar al resto de la comunidad si un proyecto que has visto en la aplicación lo has realizado tu también por tu cuenta, indicarlo con un "flag" que diga "Yo lo he hecho" (o similar). ¿Qué opináis? ¿Os parece interesante?	· Sí, es interesante. Quizá 'Yo lo he hecho' no sería la mejor frase, pero sí. · A parte de "lo he hecho" podría puntuarse si es bueno o malo el tutorial, no? Puede que con una puntuación típica (ejemplo 5 estrellas o así) mates dos pájaros de un tiro. Solo pueden puntuar los que lo han hecho en casa y a parte si les ha parecido útil el manual. · Pues es una gran idea, a parte de las ideas que habéis propuesto yo añadiría un ranking con los mejores tutoriales o proyectos basado en la opinión de la gente. · Quizás pondría un icono más que una frase. Y por supuesto, añadir comentarios y +1s como el G+. Prefiero los +1 a puntuar de 1 a 5 estrellas (como indica Pedro) porque imagino que también dependerá de la pericia de cada uno o del interés que pueda tener un proyecto para unos y para otros... · Aquí si que vería interesante el sistema de logros. Si tu pones un tutorial y yo lo hago, pongo fotos de cómo ha quedado mi proyecto en los comentarios de tu tutorial y me llevo un logro de haber hecho ese proyecto 'también'. ¿Que os parece?
7. A la hora de mostrar una entrada de un proyecto, ¿mostraríais los comentarios, likes, y "Yo lo he hecho" por separado, o como hace Tumblr en una única línea de tiempo?	· Como menos estorbe :) Prefiero por separado. · A mí me gusta Tumblr. · Tumblr style!! · Por separado, no creo que sea una información determinante si de entrada me ha resultado atractivo el proyecto. En caso de querer hacerlo seguro que busco más información y encontraré la gente que lo ha hecho y si ha sido interesante.
8. En el muro, ¿mostraríais una visión reducida del proyecto (es decir: imagen principal, título, número de comentarios, categorías o tags, etc.) o añadiríais más información adicional?	· Sí, y una descripción corta. · Para mi lo ideal seria Foto, titulo y resumen. · Le pondría foto, titulo, autor y los tags.
9. ¿Creéis que sería útil poder autenticaros mediante otras redes sociales (como Twitter, Facebook, o Google+)?	· Supongo que será útil. Personalmente nunca lo uso. · Creo que eso provocaría más participación, cuanto más fácil se lo pongas a la gente, mejor. · Personalmente no soy amigo de vincular mis cuentas sociales a otras plataformas; sin embargo sé que a la gente le resulta más sencillo. Para mi no es

	imprescindible, pero si deseable de cara a mayor participación.
10. ¿Incorporaríais un sistema de logros y/o recompensas para los usuarios en función de su actividad de la comunidad?	· Me parece buena idea. · Pues personalmente nunca le he hecho mucho caso a las medallas y logros de los foros, así que para mi es prescindible. · Es buena idea pero también podría algo que no hace nadie y es poner un sistema de penalizaciones. Por ejemplo si alguien pone un tutorial donde se salte pasos, cosas que no funcionan o alguien que etiquete sus proyectos con información que no tiene que ver con lo lo que escribe.
11. ¿Creéis que es importante que toda la información se transmita cifrada?	· Indispensable. · Si es información sensible es totalmente necesario.

5.1.1.2 Conclusiones de las entrevistas

Como puede verse en los comentarios de la tabla superior, en general el *feedback* recibido por las personas que participaron en el debate es positivo respecto a las cuestiones planteadas. En este debate, también se plantean algunas cuestiones que en un principio no formarán parte de la implementación que se realizará para la entrega del trabajo final de carrera, por limitaciones de tiempo para la entrega del proyecto, aunque fuera de plazo, una vez entregado el proyecto, se llevarán a cabo, puesto que le darán un gran valor añadido a la aplicación. Dichas funcionalidades son la inserción de vídeos grabados directamente desde el dispositivo móvil del usuario, el acceso a la plataforma utilizando servicios de terceros (como Google+ o Twitter), o la obtención de logros.

Para resumir el resultado del debate, se disponen a continuación las funcionalidades anteriores tal y como se han desarrollado en el proyecto:

1. La aplicación funciona como una red social para compartir información sobre Raspberry Pi y proyectos derivados.

2. La información de cada proyecto se muestra en artículos llamados 'entradas' y cada entrada consta de múltiples pasos que se añaden de forma individual para facilitar la inserción de los mismos y su posterior visualización.

3. Cada vez que se hace una modificación en una entrada se permite crear un borrador de la

misma de forma que se pueda añadir posteriormente nueva información antes de publicarla.

4. A la hora de filtrar las entradas, ha tenido más aceptación el etiquetado que la categorización.

5. Respecto a la incorporación de flags del tipo 'Yo lo he hecho' o 'Me gusta', se ha notado mayor disparidad de opiniones aunque casi todas tienen en común el indicar de alguna forma que los proyectos son interesantes. Por este motivo se incluyen estos dos flags y aunque se sale de la planificación del proyecto posteriormente se realizará una vista en la que se visualicen los proyectos mejor valorados por los usuarios.

6. Aunque en un primer momento, la idea para mostrar comentarios, 'Me gusta' y 'Yo lo he hecho' era siguiendo una línea temporal independientemente del tipo, en vista de las respuestas de los usuarios, se decide reorientar esta funcionalidad de forma que cada tipo de interacción se visualice por separado.

7. En el muro aparece la información estrictamente necesaria de la entrada, es decir: título, imagen principal, etiquetas, número de comentarios, número de 'Me gusta' y número de 'Yo lo he hecho'.

8. Toda la información es transmitida de forma segura. Por tanto la transmisión de información desde la aplicación hasta el servidor está siempre cifrada.

A continuación se muestran los requisitos que aunque los usuarios han visto deseables, se salen de la planificación por limitaciones de tiempo:

1. Añadir a una entrada vídeos grabados directamente desde el dispositivo móvil (aunque inicialmente no se incluye como funcionalidad, como una aplicación para dispositivos móviles debe de estar en continuo cambio, no se descarta que posteriormente a la entrega del proyecto pudiese añadirse como una mejora).

2. Aunque varios usuarios creen que estaría bien poder ver entradas públicas sin estar autenticado, por limitaciones de tiempo inicialmente la aplicación requiere siempre estar autenticado y posteriormente, se volverá a analizar la necesidad de ver publicaciones sin estar autenticado. Esta decisión también está acorde a la forma que tienen de mostrar su información las principales redes sociales en sus aplicaciones para dispositivos móviles.

3. Aunque la mayoría de los usuarios cree interesante vincular las cuentas de otras redes sociales, por limitaciones de tiempo para la presentación del proyecto no se lleva a cabo, aunque es posible que sea una mejora para una versión posterior de la aplicación.

4. Es interesante incorporar un sistema de logros o recompensas, y por este motivo se consultó a los usuarios. No obstante, por limitaciones de tiempo, quedará relegado para una versión futura de la aplicación.

5.1.1.3 Benchmarking

El segundo método aplicado ha sido el benchmarking o análisis competitivo, que consiste en realizar un estudio de otros productos similares en el mercado. Este estudio nos proporciona una idea de lo que puede captar la atención para los usuarios, según su interés y respuesta a estos otros productos.

5.1.1.3.1 Aplicaciones con un objetivo similar orientado a Raspberry Pi

Raspberry Pi para Android (BerkinAlex)

Aplicación que muestra información existente en Internet. Esta aplicación muestra diferentes proyectos en un listado directamente al abrirla. Cuando se quiere abrir un tema nos redirige al navegador web para abrir la información.

Ventajas

- Muestra la información de forma ordenada lo que permite ver rápidamente los proyectos.

Desventajas

- Hay que abrir el navegador para ver la información del proyecto.
- No permite la participación ni discusiones respecto al proyecto.
- Los temas no están categorizados.

Opiniones de los usuarios

- La opinión de los usuarios en general es bastante buena, por lo que deduzco que es una buena idea hacer una aplicación de este tipo.

- La mayoría de los usuarios agradece poder acceder a tantos proyectos de Raspberry de forma rápida.

Pi Projects de Rigney's

Aplicación para Android que tiene varios objetivos parecidos a los del proyecto RPS, es decir, pensada para compartir con otros usuarios ideas y proyectos basados en la tecnología de Raspberry Pi.

Ventajas

- No se han podido determinar puesto que no tiene acceso al *backend* en estos momentos.

Desventajas

- Aplicación poco atractiva con un diseño descuidado.
- Su dependencia con el backend imposibilita su ejecución. Existen alternativas free en estos momentos, como es el caso de Parse.com, que permiten disponer de un backend sin necesidad de que conlleve un coste para el desarrollador (siempre que no se superen los límites que indican).

Opiniones de los usuarios

- La mayoría de los usuarios no han visto adecuada esta aplicación, debido fundamentalmente a que no funciona.
- Como usuario, a simple vista (por la información proporcionada por el autor) creo que la experiencia de usuario y la usabilidad de esta aplicación podría ser bastante pobre, puesto que al dar de alta un nuevo proyecto sólo almacena una URL, lo que indica que requiere abrir un navegador web para ver el contenido de los proyectos.

5.1.1.3.2 Aplicaciones basadas en redes sociales

Tumblr

Tumblr es una red social bastante extendida en la que se puede compartir imágenes, vídeos, música, texto, etc. Tiene una estructura bastante sencilla para explorar y compartir los llamados *tumblelogs* que son entradas tipo blog pero bastante más sencillas. RPS tiene muchas similitudes

con este tipo de red social, acorde a las ventajas que proporciona.

Ventajas

- Aplicación bastante atractiva y sencilla para compartir contenido de diferente naturaleza.
- Permite etiquetar los tumblelogs para realizar filtrado en las búsquedas.
- Permite crear el tumblelog y guardarlo para publicarlo en un momento posterior.

Desventajas

- La aplicación móvil no permite buscar y seguir a usuarios conocidos.

Opiniones de los usuarios

- Esta es la opinión de algunos usuarios que se refieren a la app en sí:
    - 'Es una red social muy buena, puedes compartir fotos, videos, frases, etc. Conoces los gustos de gente de muchos países. Es una red social muy completa y divertida.'
    - 'Me gusta porque tengo un blog para publicar libremente mis noticias sin censura y al alcance de todo el mundo.'
    - 'Tiene cosas muy divertidas.. XD. Y lo mejor es que las puedes rebloguear y compartir!'
- Esta es la opinión respecto al uso de la aplicación:
    - 'la verdad es que esta muy lento y deja mucho que desear en cuanto a la velocidad. Dejando la lentitud de fuera la app esta buena.'
    - 'Ya he probado con mi Internet y en wifi y es lo mismo, los gifs y algunas imágenes tardan mucho en cargar.'
    - 'Me encantó que aplicaran el Material Design, es una de las primeras aplicaciones en cambiar su apariencia a Material, está buenísimo'
    - 'Es tercera vez que tengo que instalar de nuevo la aplicación ya que no se actualiza el número de seguidores, mensajes, en cola y borradores. Aparte de esto, la aplicación es

muy buena y siempre la uso.'

- De todo lo anterior, se deduce que a los usuarios le gusta la flexibilidad y versatilidad de esta red social, les gusta en general el diseño de la misma, aunque parece ser que suele tener bastantes problemas con el contenido multimedia y la carga de las vistas.

Google +

Esta es la red social de Google está pensada como un complemento a los servicios que ofrece, proporcionando a éstos una capa social, en la que compartir información. Además, está integrada con otros servicios como *Hangout*, el servicio de subida de fotos desde el móvil, el calendario, etc.

Ventajas

- Ordena a los contactos en círculos (uno o más) de forma que permite compartir información de una forma muy flexible, ya que además se puede compartir a usuarios y a círculos a la vez.
- Al estar integrado con otras herramientas de Google, tiene una gran versatilidad, de forma que podremos usarlo para crear galerías de fotos, como para crear eventos, compartir ubicaciones o crear citas.

Desventajas

- No se puede estar dado de alta en Google+ y no estarlo en otros servicios, como por ejemplo, el correo.

Opiniones de los usuarios

- La opinión de los usuarios respecto a la aplicación en sí, por lo general, es bastante mala puesto que con cada actualización parece ser que requiere más recursos y en algunos modelos de dispositivos con Android suele fallar bastante.
- Respecto al concepto en sí de G+ como red social, las opiniones son bastante variadas, con una buena aceptación, aunque no se puede evitar la comparativa con Facebook. Los usuarios lo que más valoran es la capacidad de subida automática de fotos, los filtros y las composiciones automáticas que realiza G+ con el contenido del propio usuario.

5.1.1.3.3 Conclusiones sobre el *benchmarking*

De las aplicaciones plasmadas en este punto y de otras que también se han visto durante la fase de estudio, se sacan las siguientes conclusiones:

- Hay mucha información en Internet sobre Raspberry Pi y las tecnologías que lo rodean: existen foros específicos, para gente experta o principiantes, tutoriales, etc. Sin embargo, para dispositivos móviles hay muy pocas aplicaciones dedicadas a compartir información sobre esta tecnología. De las pocas aplicaciones existentes, sus usuarios se muestran interesados en este tipo de interacción. Esto me permite deducir que RPS podría tener una gran aceptación entre los usuarios de Raspberry Pi.

- Las aplicaciones que existen para compartir información de esta tecnología, no tienen un componente social, como lo tendrían en web los foros o los *blogs*, los cuales están muy extendidos para la compartición de información de Raspberry Pi. Esto me permite concluir que los usuarios potenciales agradecerán poder crear la aplicación orientada a la participación y a las interacciones entre usuarios.

- Por otro lado, cada red social tiene un fin y un tipo de usuario al que se quiere captar según sus intereses o inquietudes. De las redes sociales estudiadas, tanto Tumblr y Google+, como las no plasmadas en este documento: Twitter, Facebook, Pinterest, LinkedIn, etc., pero de las que soy conocedor por mi propia experiencia de usuario, he determinado que la red social que surja de este proyecto tiene que tener la siguientes características mínimas:

    - Debe de ser intuitiva y simple de utilizar.

    - Debe presentar claramente los contenidos y a ser posible, de forma atractiva, siguiendo las tendencias de cada momento. En este momento, con la salida de iOS 8 y Android 5.0 *Lollipop*, me decanto por crear una interfaz que siga, hasta cierto punto, las pautas indicadas por estos sistemas, teniendo en cuenta también las tendencias actuales que existen para el diseño de aplicaciones en HTML5.

    - Debe tener la información estructurada en dos vertientes:

        - La información que se comparte y cómo se comparte

        - Los usuarios con los que se comparte y cómo relacionarlos entre ellos

- Debe de ser ligera (almacenamiento) y requerir un uso de recursos lo más básico posible, puesto que en las aplicaciones estudiadas, la mayoría de los usuarios (con diferentes dispositivos) se quejaban de que las aplicaciones tardan en cargar, o son demasiado 'pesadas'.
- La presentación y búsqueda de la información debe ser obtenerse siempre en el menor número de pasos, de forma que el usuario no se 'aburra' de ella.

5.1.1.4 Perfiles de usuario

De los métodos de indagación aplicados, se extraen los requisitos que se consideran más importantes para crear una aplicación atractiva y que pueda llevar al mayor número de usuarios. Sin embargo, esta extracción de requisitos se hace de una forma genérica (dinámicas de grupo y benchmarking) sin tener demasiado en mente el tipo de usuario que va a utilizarlas. Por lo que el siguiente paso, es determinar si todo estos requisitos son útiles independientemente del perfil de los usuarios, o al contrario, a la hora de determinar los diferentes perfiles, se detecta que algunos requisitos no son tan importantes y, sin embargo, otros no contemplados hasta ahora sí lo son.

La determinación de estos perfiles de usuario aplicables para el uso de la aplicación RPS, permite responder algunas de las cuestiones que se indican al inicio del apartado que son:

- ¿A quién va dirigida la aplicación?
- ¿Cuál es el nivel de conocimiento de los usuarios a quién va dirigida?
- ¿Qué se espera de esta aplicación?
- ¿Dónde se utilizará esta aplicación?

Para poder responder a estas preguntas he estudiado el comportamiento de algunos usuarios de foros y blogs dedicados a la tecnología que rodea a Raspberry Pi. A continuación, se determinan las respuestas a las preguntas anteriores definiendo inicialmente los tipos de usuarios que utilizarán RPS en base a diferentes formas de categorizarlos:

- Los tipos de usuarios detectados en función del objetivo del uso de la tecnología Raspberry Pi:

- Educación. En sus inicios, la placa Raspberry Pi fue concebida para el ámbito educativo, por lo que existen muchos usuarios que la utilizan para aprender (o enseñar), por ejemplo, a programar.
- Entretenimiento. Gracias a que la placa cuenta con un chip gráfico capaz de reproducir contenido en alta definición, existe una gran cantidad de usuarios que utilizan esta placa como centro de entretenimiento (vídeo, música, fotografías, etc.).
- Domótica. Existen usuario que la utilizan para montar un sistema de videovigilancia, o para controlar la iluminación o temperatura de la vivienda, etc.

- Según la motivación por la que utilizan esta tecnología:
  - Por ahorro económico, ya que al ser un mini-ordenador que permite ejecutar cierto tipo de software, es una opción mejor que otras que puedan tener un coste mayor y que realice las mismas tareas.
  - Por ampliación del conocimiento TIC, puesto que trabajar con esta tecnología permite profundizar en algunos conocimientos, como puede ser configurar una pantalla LCD, convertir la Raspberry en un NAS casero, en un punto de acceso WiFi, etc.
- Según el interés por la tecnología, estarían:
  - Los que nunca han utilizado Raspberry Pi pero tienen interés por hacerlo, quizás porque se han marcado alguno de los objetivos anteriores.
  - Los que han realizado alguna configuración básica para un uso concreto, interesados en aplicar esta tecnología a otros ámbitos, o bien, que quieran compartir su experiencia con otros usuarios.
  - Los entusiastas de esta tecnología que la aplican a cualquier cosa que se les pueda ocurrir, muy interesados en compartir su experiencia y en aumentar el número de usuarios que pueda llegar a tener la comunidad.
- Por otro lado, según la experiencia:
  - Aficionado, aquella persona que tiene interés en usar esta tecnología sólo por el placer

de hacerlo, o bien, porque le proporciona una solución que necesita (mediacenter, domótica, etc.).

- Profesional, ya que cada vez es más extenso el uso de esta tecnología entre empresas, ya sean de base TIC o no.

- De la categoría anterior, se puede extraer otra del lugar donde se utilizará la aplicación:
  - En el hogar, donde la utilizarán la mayoría de aficionados para realizar sus propios 'experimentos'.
  - En el lugar de trabajo, como tarea propia de su actividad.
  - En menor medida, en otros entornos, en los cuales la aplicación no se utilizará por lo común para agregar nuevos proyectos, pero sí que se utilizará para visualizar y comentar los proyectos de otros usuarios.

De todas las categorizaciones anteriores, se extraen a continuación los perfiles de usuario más relevantes, que marcarán las diferencias de uso entre unos y otros usuarios:

- ***Usuario casual***, usuario que ha oído hablar de esta tecnología y de sus ventajas y quiere embarcarse en este 'mundillo'.
- ***Estudiante TIC***, usuario con necesidad de conocer la tecnología.
- ***Aficionado***, que ha hecho alguna que otra instalación, basándose en tutoriales que han hecho otros usuarios y que poco a poco va ampliando sus conocimientos hasta que él mismo es capaz de compartir sus experiencias.
- ***Experto***, profesional o no, que lleva mucho tiempo dedicado a explorar la capacidad de esta tecnología y que proporciona su conocimiento al resto de la comunidad.

En el siguiente apartado, se detallarán las fichas de cada uno de los perfiles de usuario detectados, en base a las características de su perfil, a los contextos de uso aplicables, a las tareas que se requieren para estos usuarios y cualquier característica que se haya determinado durante la fase de indagación, junto con la información de los escenarios de uso, que forma parte de la fase de diseño conceptual.

5.1.2 Diseño conceptual

En esta fase del proyecto se va a unificar la ficha de los perfiles de usuario junto con la creación de los escenarios de uso, aplicables a cada uno de los perfiles.

Usuario casual	
Características del perfil	Este usuario, aunque no tiene un conocimiento avanzado en tecnología, conoce Raspberry Pi por su entorno, ya sean familiares, compañeros o amigos. Es un usuario habitual de la tecnología, que probablemente usa el ordenador personal y los dispositivos móviles de una forma habitual, y usa habitualmente redes sociales para compartir experiencias con otras personas. La edad de estos usuarios comprenderá entre jóvenes y personas de mediana edad principalmente.
Contextos de uso	Este tipo de usuario usará una aplicación de este tipo principalmente en un ambiente doméstico y principalmente en sus horas libres. Será un usuario aficionado por lo general, que compartirá sus proyectos o los proyectos que más le interesan con amigos de un mismo perfil tecnológico.
Análisis de tareas	Este usuario se centrará principalmente en la visualización de proyectos de otros usuarios, marcará los proyectos que más le interesen y en alguna que otra ocasión probará a llevar a cabo él mismo alguno de los proyectos. Respecto al componente social, buscará usuarios del tipo experto que pueda mostrarle nuevos proyectos que puedan ser de interés para él. También es posible que busque entre sus contactos a amigos conocidos con los que le pueda interesar compartir sus experiencias en una red social de este tipo.
Características requeridas	• Intuitiva y simple de utilizar. • Acceso rápido a la información deseada. • Debe ser ligera (almacenamiento) y requerir pocos recursos del dispositivo. • Interacción como red social para compartir información. • Visualización por pasos de cada uno de los proyectos. • Etiquetado de las entradas. • Incorporación de *flags* 'yo lo he hecho' y 'me gusta'.
Escenario de uso	Juan ha ido a visitar a su hermana María para comer juntos. Después de la comida, en la sobremesa, María le ha enseñado a Juan su nuevo centro multimedia desde el cual puede seleccionar películas y series de su biblioteca personal. Juan queda fascinado y siente curiosidad por el dispositivo que ha utilizado María para disponer del centro multimedia: Raspberry Pi (¡qué nombre más raro!). Este dispositivo, según le comenta ella, es un mini ordenador que cuesta unos 40 euros y que es muy fácil de instalar, puesto que ella lo ha hecho siguiendo unos tutoriales que ha encontrado en Internet. Juan, una vez en casa, busca

	en su dispositivo móvil si hay alguna app que le permita buscar de forma cómoda información y tutoriales para poder montar el mismo su propio centro multimedia. En el store ha encontrado una aplicación llamada RPS, que según indica, es una red social para entusiastas de Raspberry Pi. Se instala la aplicación y una vez abierta, se sorprende de la cantidad de cosas que se pueden realizar con esta tecnología (¡¡hasta se podría crear su propio smartphone!!). Además, comprueba que hay mucha interacción entre los usuarios, a los cuales puede seguir para ver los proyectos que estos publican. Está decidido: se va a comprar una Raspberry Pi y en su tiempo libre intentará, en casa, configurarla para crear su propio centro multimedia, y quien sabe, … Este es el comienzo de una gran amistad...

Jose Antonio Morales

34 años | Médico de familia

Interés por la tecnología:
- Adicto a las nuevas tecnologías.
- Le gusta comprarse lo último en smartphones y tablets.

Otros *hobbies*:
- El cine y la música jazz.
- Salir a correr.

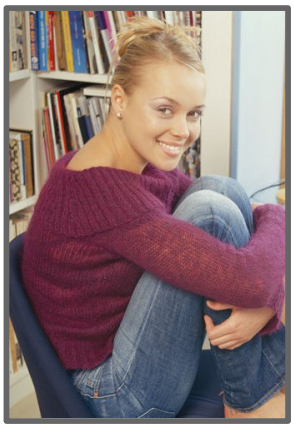
Objetivo:
- Conocer nuevos ámbitos de aplicación de las tecnologías para saber como aplicarlas a la medicina.
- Conocer otras personas con estos mismos objetivos.

Comportamiento:
- Embarcado en crear un centro multimedia para su consulta por sus propios medios para proporcionar información interactiva sobre tratamientos y terapias a sus pacientes, de forma que mejore el vínculo médico-paciente.

Estudiante TIC	
Características del perfil	Este tipo de usuario estará más centrado en el conocimiento de los componentes y de la tecnología que rodea a Raspberry Pi que de la propia creación de los proyectos. Es un tipo de usuario casual, con un nivel de conocimiento sobre la tecnología mayor que el resto y que enriquecerá las discusiones sobre los proyectos con sus dudas y planteamientos. La edad de estos usuarios comprenderá principalmente a jóvenes en edad de formación a nivel tecnológico, ya sea una formación profesional o en estudios superiores.

Contextos de uso	Este tipo de usuario utilizará la aplicación principalmente durante sus horas de estudio y en un ámbito doméstico y en su centro de enseñanza. Es probable que incluso con su profesor y sus compañeros creen entre ellos proyectos específicos según requiera la temática correspondiente.
Análisis de tareas	Este usuario se centrará principalmente en la búsqueda de información y en la posible resolución de sus dudas por parte de usuarios expertos en la materia. Menos interesados en el componente social y más en la búsqueda de conocimiento.
Características requeridas	• Acceso rápido a la información deseada. • Visualización por pasos de cada proyecto. • Etiquetado de las entradas. • Mostrar en el muro información estrictamente necesaria para saber de qué se habla. • La inclusión de comentarios.
Escenario de uso	Ana tiene una práctica que entregar: tiene que crear un entorno para la publicación de páginas web sobre Linux, que además incorpore un control de versiones y otras herramientas. Sin embargo, no puede utilizar su portátil con Windows porque no tiene espacio para hacer una partición de Linux. En este momento, no puede costearse comprarse otro ordenador, así que no sabe qué hacer. Su amigo Héctor le comenta que hay una solución asequible que le resolverá este problema y que en un futuro podría serle de utilidad para otras cosas. Ana es un poco escéptica, así que se pone a investigar esto del Raspberry Pi. 'Es increíble lo barato que es', piensa. 'Quizás con esto solvente mi problema'. Ahora que se lo ha comprado, necesita sabe cómo montar el entorno que le reclaman en la práctica. Para ello ha buscado desde su móvil y ha encontrado una red social llamada RPS, en la que resulta que esta su amigo Héctor. Hay otro usuario, llamado H4ck3r, que ha hecho algo parecido a lo que ella necesita. '¡Estupendo!'. Ana está en su habitación de la residencia configurando su Raspberry, siguiendo el manual de este usuario pero tiene un problema. Llegado a un paso, el resultado que obtiene no es el mismo y no funciona bien el servidor web. Por lo que añade un comentario en el tutorial con el problema que tiene, y unos minutos después le responde otro usuario, Chapuzas, que le indica cual ha sido el fallo que ha cometido Ana al configurar el servidor web. Por fin, Ana ha terminado su práctica y piensa: 'Si no fuese por Raspberry Pi y por la comunidad que tiene detrás me hubiese sido muy complicado desarrollar esta práctica. Creo que me gusta esto, a ver si aprendo algo más…'

Eva García



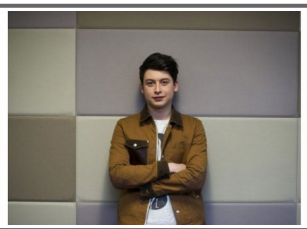
21 años | Estudiante de Telecomunicaciones de tercer curso

- Quiere centrar su carrera en la robótica
- No tenía grandes conocimientos de tecnología antes de comenzar la carrera y ahora le fascina la idea de construir sus propios equipos mecanizados
- Le gusta la literatura de fantasía y ciencia ficción
- Siempre quiere sorprender a sus profesores y compañeros, por lo que suele embarcarse en proyectos inesperados.
- No se conforma con lo primero que le dicen y siempre busca profundizar en los temas que se le presentan.
- Le encanta hacer nuevas amistades, ya sea en vivo o a través de redes sociales.

Aficionado	
Características del perfil	Este es un tipo de usuario con un mayor conocimiento de la tecnología que accederá a la aplicación para buscar nuevos retos a realizar y a compartir sus experiencias con otros usuarios aficionados y expertos. Es probable que este usuario tenga un conocimiento TIC bastante elevado que use el ordenador personal y el dispositivo móvil de una forma habitual y que busque estar al tanto de las nuevas tecnologías disponibles. La edad de estos usuarios comprenderá entre jóvenes y personas de mediana edad principalmente, aunque el rango de edad sea probablemente un poco más elevado que el del usuario casual.
Contextos de uso	Este tipo de usuario puede aplicar el conocimiento obtenido del uso de estas tecnologías para proyectos realizados como hobby para ponerlos en funcionamiento en su lugar de trabajo en el caso de que éste tenga un nivel tecnológico adecuado para poder aplicarlo. Por tanto el ámbito en el que se mueve este tipo de usuario puede ser tanto doméstico (en su casa) como en un ámbito laboral.
Análisis de tareas	Este tipo de usuario hará un uso mayor que los dos tipos anteriores puesto que su interés por esta tecnología también es mayor. En este caso, su participación en la incorporación de nuevos proyectos será mayor de forma proporcional a su conocimiento al igual que su participación en las discusiones sobre los mismos. El factor social también tendrá un papel mayor puesto que su interés por la compartición de información será mayor, y por tanto, su círculo de amistades relacionadas con esta tecnología será mayor.
Características requeridas	• Intuitiva y simple de utilizar. • Acceso rápido a la información deseada. • Interacción como red social para compartir información. • Visualización por pasos de cada uno de los proyectos.

	• Guardar como borrador la creación de entradas para poder editarlos posteriormente. • Etiquetado de las entradas. • Incorporación de flags 'yo lo he hecho' y 'me gusta'. • Mostrar en el muro información estrictamente necesaria para saber de qué se habla. • La inclusión de comentarios.
Escenario de uso	A Joaquín le encanta hacer cosas nuevas con su Raspberry Pi. Esta semana se ha propuesto conectar una pantalla LCD de un Nokia 3220 antiguo que tenía guardado mediante las conexiones GPIO de la placa Raspberry Pi. Ha creado un programa que le permitirá mostrar la información del sistema como: el porcentaje de CPU o la memoria usados, o la temperatura del dispositivo, etc. A medida que ha ido realizando modificaciones (desmontar la pantalla del móvil, soldar los *pines* a un cable conector que irá a la placa, probar la pantalla, crear el programa para la lectura de la información, etc.) ha ido haciendo fotografías y tomando notas para poder luego compartirlas con otros usuarios. En estos últimos días ha empezado a probar una aplicación para móviles que permite añadir tutoriales y artículos de forma que puedan verlo otros usuarios, como si del Facebook se tratase. La verdad es que el formato le gusta, así que va a realizar su primera aportación. Lo primero es crear 'la entrada' (como dicen los creadores). Después va metiendo toda la información ('¡Anda! ¡Si permite separar los pasos y guardar un borrador! Genial porque se me ha olvidado hacer una foto de uno de los pasos', piensa) y por último, publica su proyecto. Un par de días después comprueba que hay varios usuarios que han comentado su publicación, e incluso hay un usuario que indica que lo ha hecho. Joaquín está muy contento, así que se pone a pensar cual podría ser su próximo proyecto.

Ignacio Avilés



24 años | Programador Senior en C++ en la empresa INVOSOFT desde hace 3 años

Alto conocimiento en sistemas operativos y programación a bajo nivel.

Expectativas:

- Crear su propia empresa de desarrollo de software para seguridad y cifrado de datos en la nube.

Hobbies:

- Inventar nuevos proyectos combinando diferentes componentes con Arduino, Raspberry Pi y su consola PS Vita.
- Jugar en su PS4 y su PS Vita.
- La música techno.

Comportamiento

- No le gusta demasiado relacionarse con la gente aunque le gusta compartir sus proyectos y que se le valore por ello.

Experto	
Características del perfil	Este tipo de usuario tiene un conocimiento profundo de esta tecnología y lo aplica en diferentes aspectos de su vida cotidiana, como sería, en su vida personal en la creación de sistemas domóticos, centros multimedia, etc. como a un nivel profesional para la mejora de procesos existentes en su entorno laboral, independientemente de que este tenga un ámbito tecnológico bajo o elevado. La edad de estos usuarios estará comprendida principalmente por personas de mediana edad, puesto que la experiencia viene en parte determinada por la cantidad de tiempo dedicada a esta tecnología.
Contextos de uso	Este tipo de usuarios aplicará esta tecnología en cualquier momento que le surja la oportunidad, puesto que la mayoría de los planteamientos que realice, los transformará para poder aplicarlos. La mayoría de estos usuarios serán profesionales TIC, especializados en esta materia. Por tanto, le dará un mayor uso a nivel profesional que el resto de usuarios.
Análisis de tareas	Este tipo de usuarios se centrará principalmente en la publicación de sus proyectos y en la resolución de dudas que puedan surgir a otros usuarios en los debates iniciados, tanto para sus proyectos, como para los proyectos de otros usuarios. Estarán más centrados en la propia tecnología y menos centrados en el componente social de la aplicación (conectar con otras personas).
Características requeridas	• Intuitiva y simple de utilizar. • Acceso rápido a la información deseada. • Guardar como borrador la creación de entradas para poder editarlos posteriormente. • La inclusión de comentarios. • Etiquetado de las entradas. • Mostrar en el muro información estrictamente necesaria para saber de qué se habla. • Transmisión de la información de forma segura.
Escenario de uso	Antonio y Fernando son dos de los administradores del CPD de la multinacional Oráculo. Llevan muchos años trabajando en este sector, y aunque cuentan con presupuestos astronómicos para mejorar la

	seguridad del CPD, tienen que configurar un servidor independiente del resto del CPD y por tanto no cuenta con esta capacidad. Ambos llevan años trabajando además con Raspberry Pi, desde que salió el primer modelo de la placa. Tal es su fascinación que se han propuesto utilizar una Raspberry pi para aumentar la seguridad en las conexiones a Internet del servidor aislado. Para ello han pensado montar un firewall de forma que el RP sea el que se conecte directamente a la red, además se alimentará directamente del servidor por usb. Además, montarán un servidor VPN de forma que los usuarios que se conecten lo hagan a través de una conexión cifrada. Desde los foros de RP les han comentado que publiquen este proyecto en una nueva aplicación para móviles que ha surgido y que está teniendo bastante aceptación. Ellos, aunque no les interesa mucho el tema social, creen adecuado añadir su proyecto y orientar al resto de usuarios en todo lo posible para que también sean capaces de llevarlo a cabo.

Alejandro J. Torres

42 años | Profesor de arquitectura de computadores de la UOC

Objetivos

- Su vocación siempre ha sido la enseñanza.
- Quiere crear una asignatura específica para trabajar con Raspberry Pi.

Comportamiento

- Le encanta compartir su conocimiento. Siempre participa en foros y redes relacionadas con cualquier campo de la tecnología.
- Siempre anima a sus alumnos a compartir ellos mismos sus experiencias.
- Siempre a la última de las novedades que aparecen por la red en aplicaciones relacionadas con la tecnología.
- Es exigente tanto con su trabajo como con el de los demás.

5.1.2.1 Flujos de interacción

A continuación se muestra el flujo de interacción que representa la navegación del usuario entre las distintas pantallas de la aplicación.

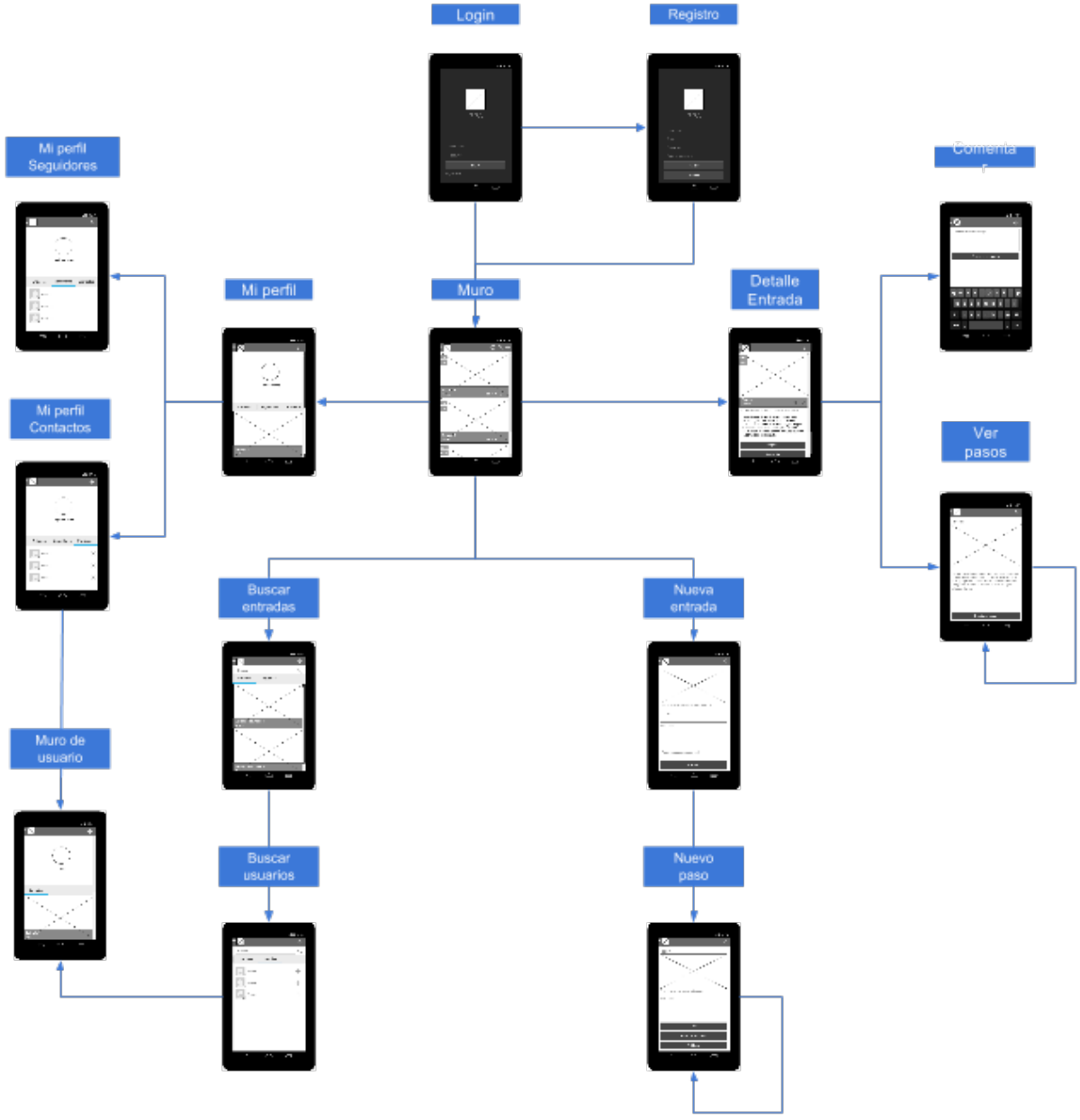


Ilustración 1: Flujo de interacción

5.1.3 Prototipado

El prototipado se ha realizado de manera incremental en tres fases:

5.1.3.1 Sketches

En primer lugar, se han realizado unos *sketches* hechos a mano alzada, que representan las distintas pantallas de la aplicación. En ellos, se incluyen los principales componentes que forman las pantallas, así como algunas notas sobre la interacción del usuario. En este paso, se trata de tener en cuenta la diferencia existente entre móviles y tablets. En este sentido, el diseño se define de forma que la orientación del dispositivo siempre sea vertical, por lo que las diferencias entre ambos dispositivos serán mínimas (adecuación de los espacios entre los elementos, la disposición de algunos elementos, como pueden ser botones, que en el móvil están uno encima del otro, y en el tablet están uno al lado del otro para aprovechar mejor el espacio y que no resulten demasiado grandes en este tipo de dispositivo). Los muros, se decide que tengan la misma visualización, ya que como el elemento principal de estos es una imagen, tanto en móviles como en tablets permiten captar mejor la atención del usuario cuanto mayor es la imagen.

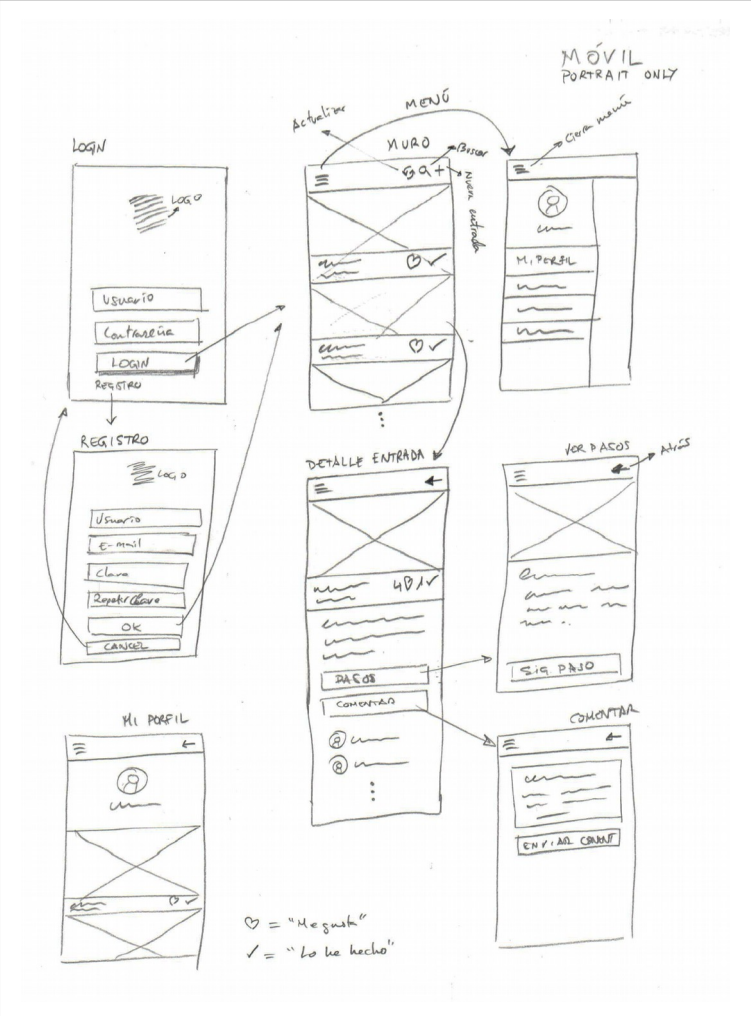


Ilustración 5: Sketches. Página 1.

En esta página se puede observar la transición que se produce al abrir la aplicación:

- Comienza con la página de acceso o *login*.
- Desde ahí se puede entrar a la aplicación, a la página del 'Muro', con los datos de autenticación adecuados.
- O bien, acceder a la página de registro.
- Dentro de la aplicación, se podrá acceder al menú lateral y de ahí a 'Mi perfil'
- Desde el 'Muro' se podrá acceder al 'Detalle de la entrada', y a sus 'pasos' de la entrada y a la vista para la inserción de comentarios.

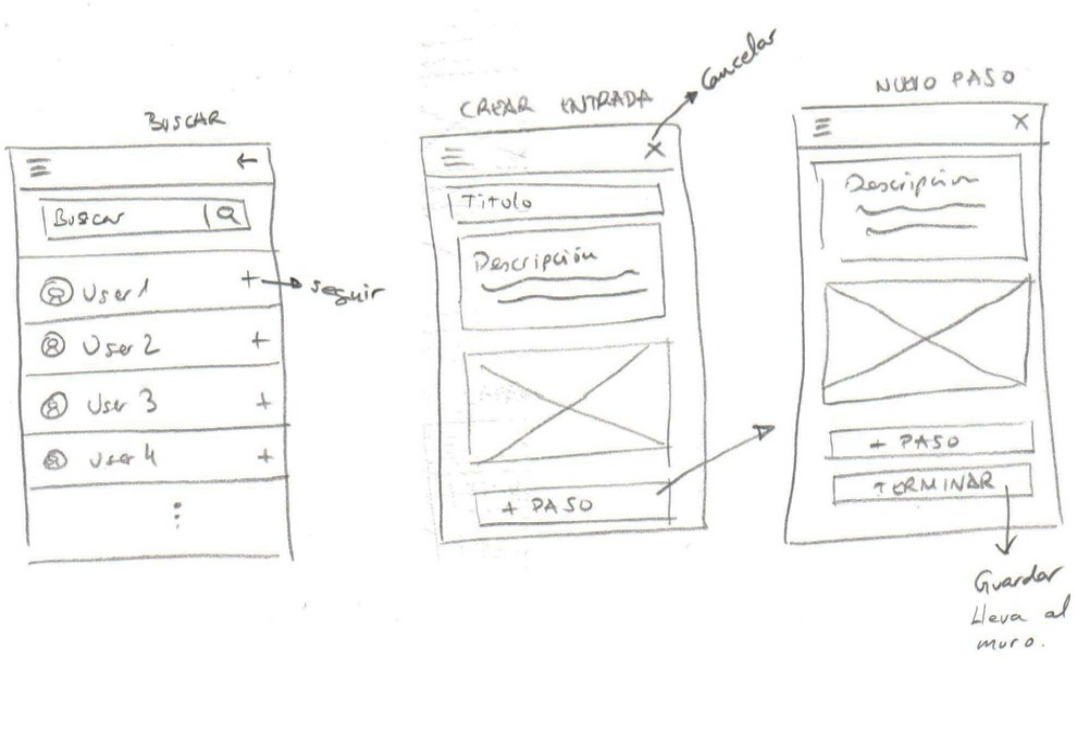


Ilustración 6: Sketches. Página 2.

En la ilustración anterior, se muestran las vistas: 'Buscar usuarios' para localizar y 'seguir' a otros usuarios; 'Crear entrada' con la información necesaria para su creación y la incorporación de un nuevo paso a la entrada que se está creando.

5.1.3.2 Prototipado en papel

A continuación, se realiza un prototipado en papel, usando una de las herramientas nombradas en los materiales (Sneakpeekit). En esta fase, se añade la colocación de los elementos en pantalla, así como los componentes que aparecerán en cada una de ellas.

'*Login*' y 'Registro'

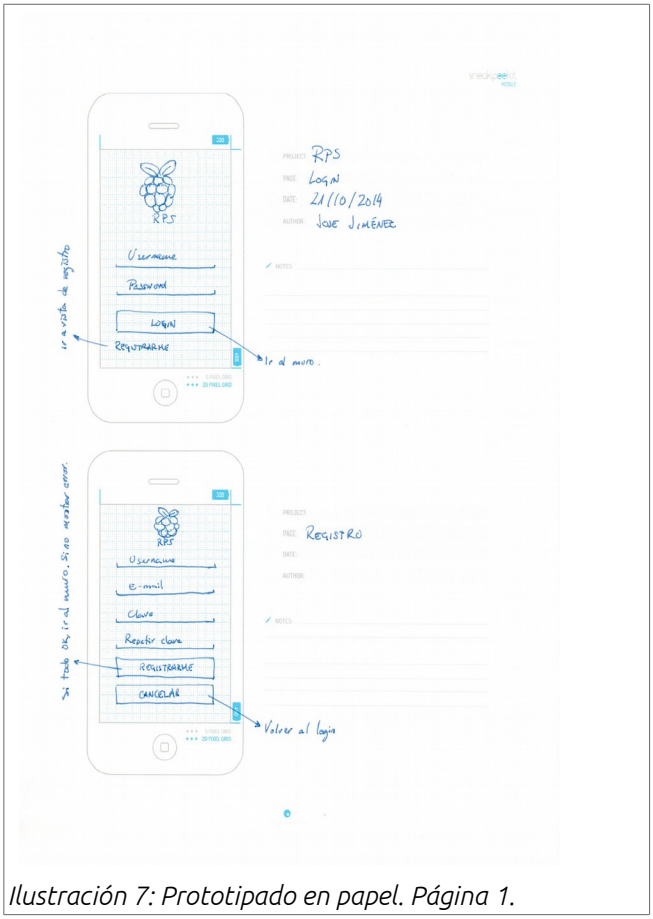


Ilustración 7: Prototipado en papel. Página 1.

'Muro' y menú lateral

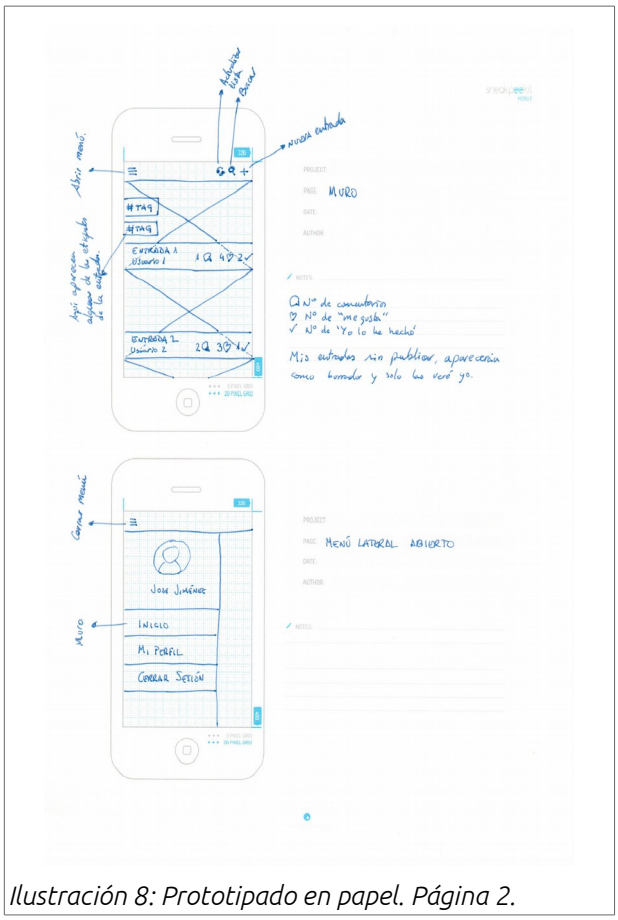


Ilustración 8: Prototipado en papel. Página 2.

'Mi Perfil' - vista de 'Mis entradas' y 'Detalle de entrada'

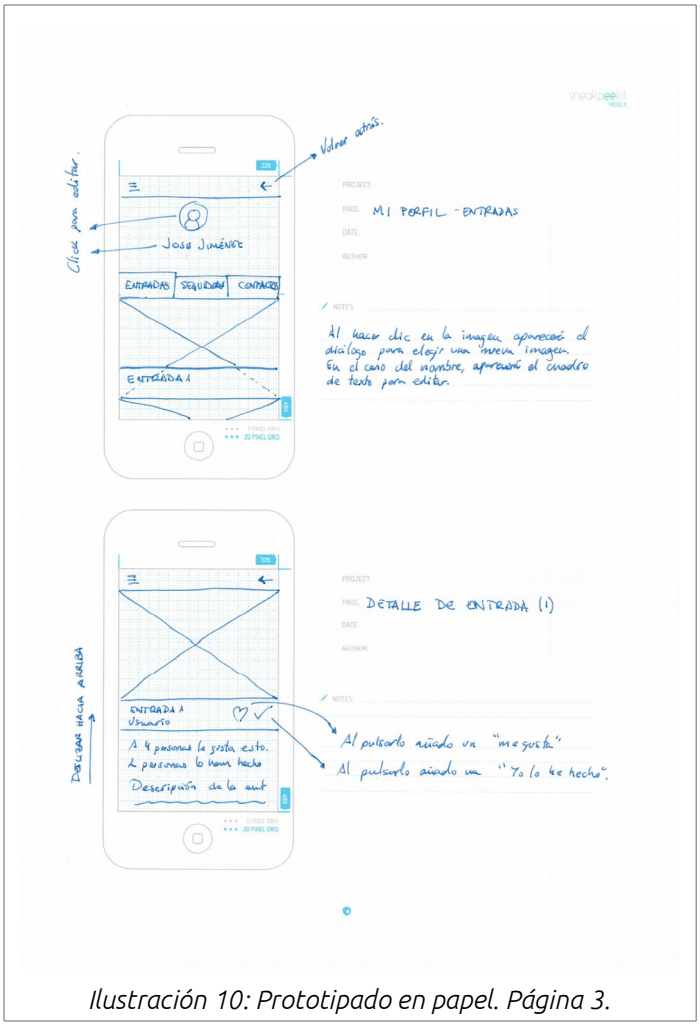


Ilustración 10: Prototipado en papel. Página 3.

'Mi Perfil' - vista de 'Mis seguidores' y vista de 'Mis contactos'

Ilustración 9: Prototipado en papel. Página4.

'Detalle de entrada' (continuación)

y 'Comentar entrada'

Ilustración 11: Prototipado en papel. Página 5.

'Buscar entradas'

y 'Buscar usuarios'

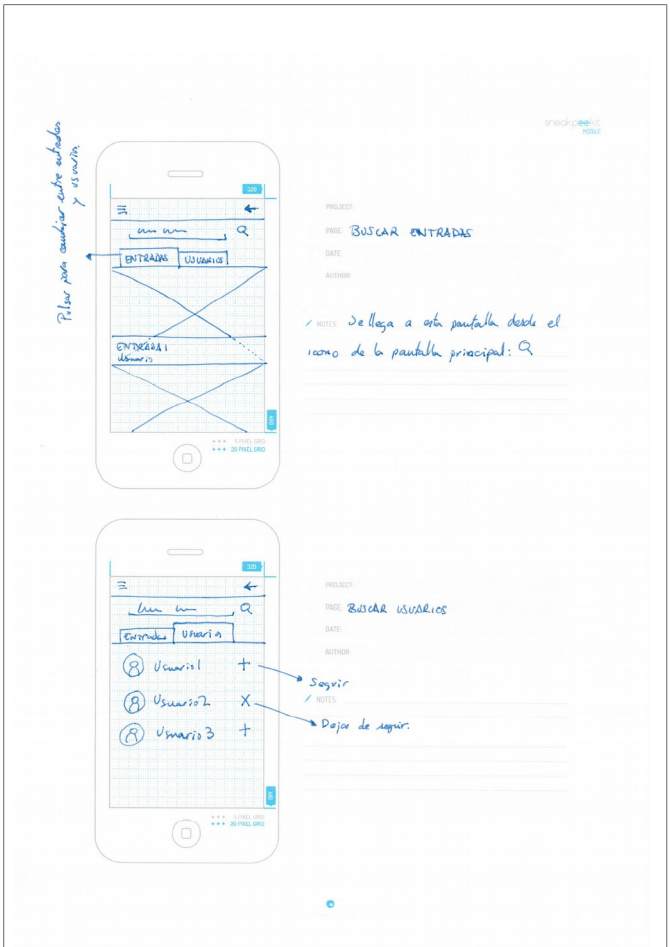


Ilustración 12: Prototipado en papel. Página 6.

'Nuevo paso'

y 'Ver paso'

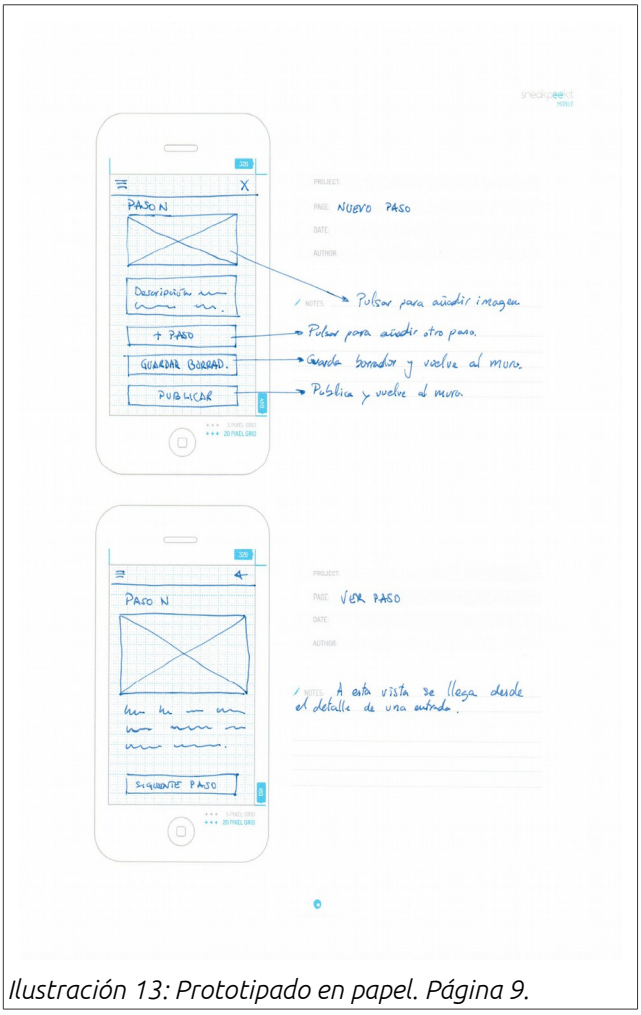


Ilustración 13: Prototipado en papel. Página 9.

'Crear entrada (I)'

y 'Crear entrada (II)'

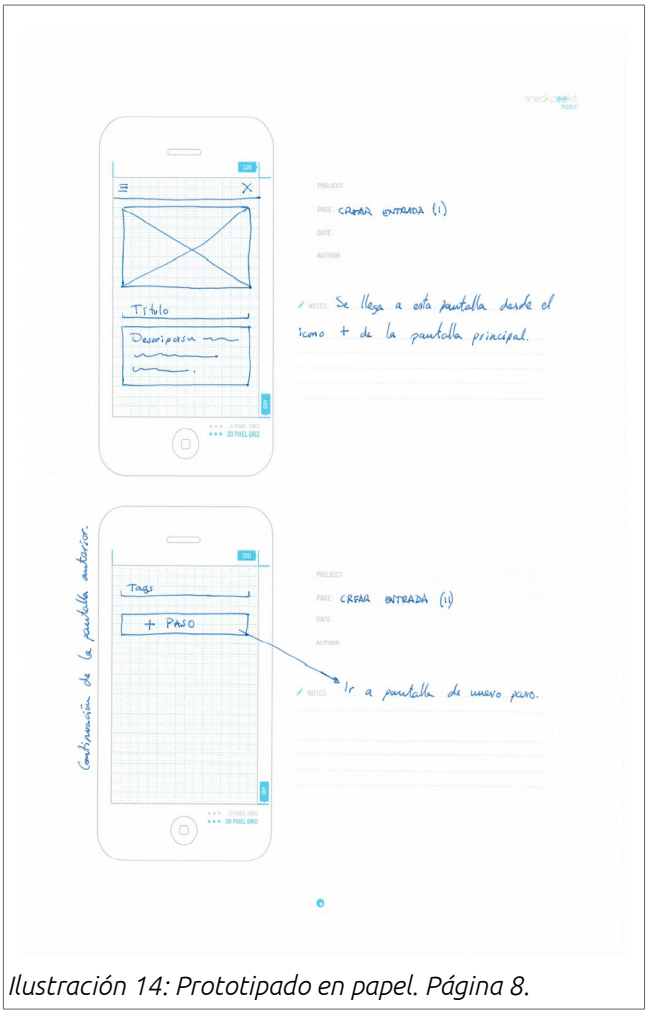
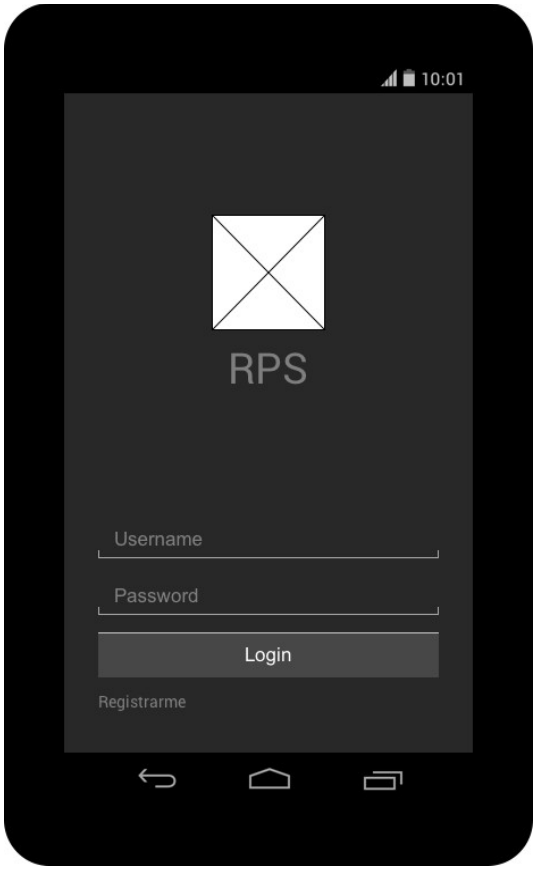


Ilustración 14: Prototipado en papel. Página 8.

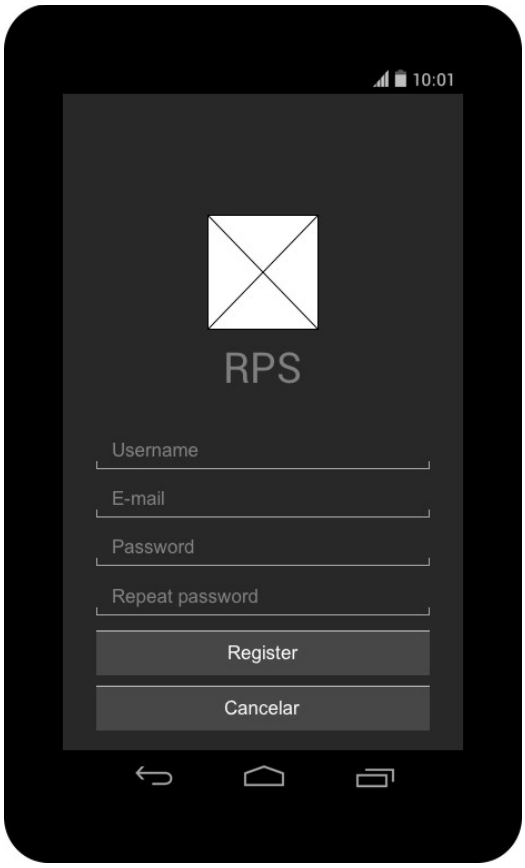
5.1.3.3 Prototipo de alta fidelidad

En este último paso, usando el software nombrado en los materiales (Just in mind), se realiza el prototipo de alta fidelidad. El resultado final de la aplicación deberá parecerse en gran medida a este prototipo. En este paso, se añade alguna vista nueva que no se había incluido en los pasos anteriores.

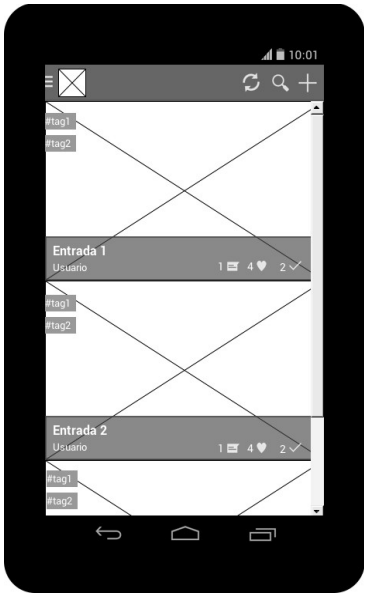
Login



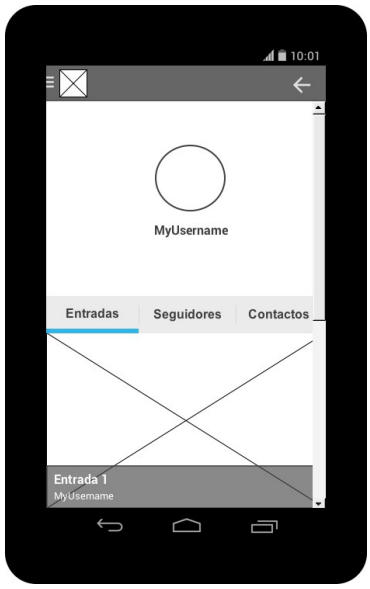
Registro



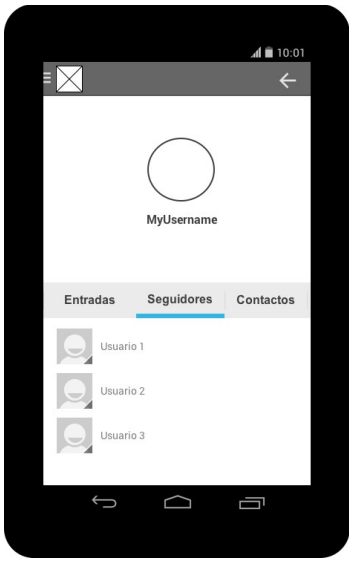
Muro



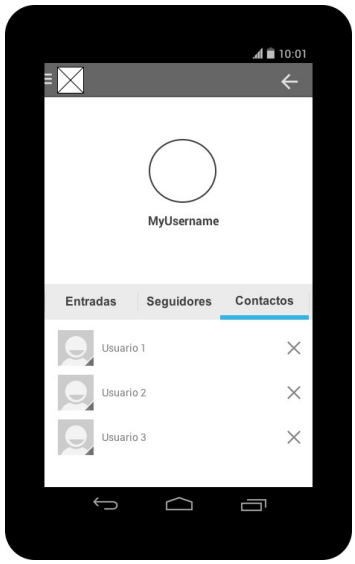
Mi perfil - Entradas



Mi perfil - Seguidores



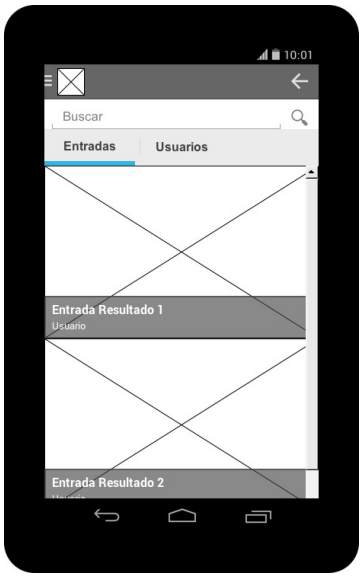
Mi perfil - Contactos



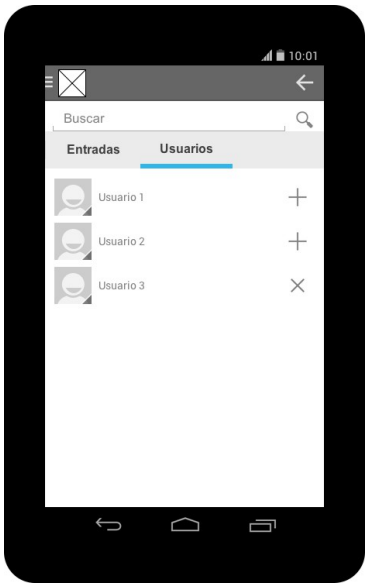
Detalle de entrada

Comentar entrada

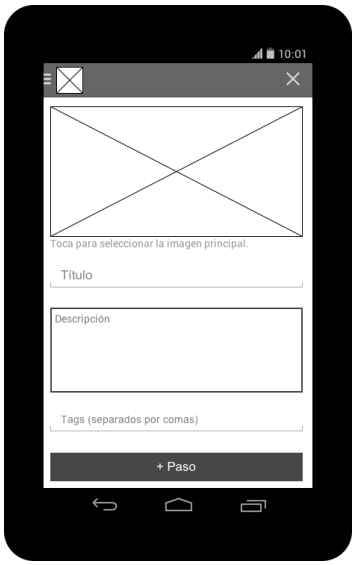
Buscar entradas



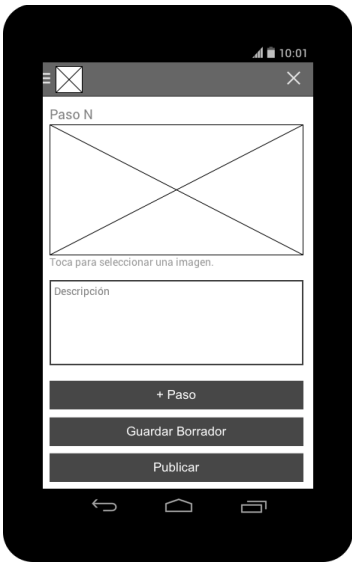
Buscar usuarios



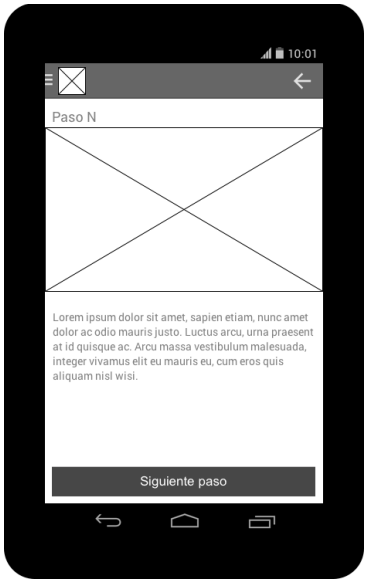
Nueva entrada



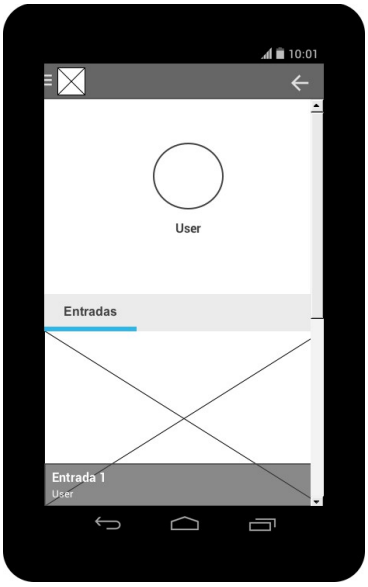
Nuevo paso



Ver paso



Entradas de usuario



5.1.4 Evaluación

En esta fase del diseño centrado en el usuario se aplica la técnica del test con usuarios, que proporciona información cualitativa sobre la interacción del usuario con la aplicación, lo que permite, durante el desarrollo de la aplicación RPS, analizar la usabilidad de la misma, mediante la observación de los usuarios que realizarán una serie de tareas, y analizar la adecuación respecto a la experiencia de usuario y la usabilidad de RPS.

Para ello, a continuación se definen los componentes del test estructurados en:

- Las preguntas a realizar a los usuarios para determinar su grado de interés, motivaciones, disponibilidad y otros aspectos del propio usuario, de forma que permitan analizar el grado de interacción que pueda tener dicho usuario con la aplicación y, por tanto, incorporarlo a uno de los perfiles de usuario determinados previamente.
- La descripción de las tareas que se espera que realicen estos usuarios
- Las preguntas a realizar relacionadas con la ejecución de las tareas.

5.1.4.1 Preguntas sobre el usuario

¿Conoces Raspberry Pi? ¿Qué es lo que más te llama la atención de esta tecnología?
Si tuvieras una Raspberry Pi, ¿sabrías lo que hacer con ella?
¿Qué tipo de fuentes consultarías para obtener información sobre esta tecnología?
¿Te instalarías una aplicación en el móvil para obtener información sobre esta tecnología?
¿Te instalarías una aplicación basada en una red social entorno a esta tecnología?
¿Cuáles son las tareas que realizarías con una aplicación como esta? ¿Buscarías proyectos (artículos, tutoriales, etc.) de otros usuarios? ¿Publicarías tus propios proyectos?
¿Te gustaría enlazar con otros usuarios dentro de esta red social para seguirles y ver sus publicaciones?
¿Crees que las publicaciones de los usuarios deben de ser públicas o privadas?
En tu entorno social, real y/o virtual, ¿hay personas interesadas en esta tecnología?
¿Animarías a gente de tu entorno a utilizar esta aplicación?

¿Qué te motivaría a utilizar esta aplicación?
¿Dónde crees que utilizarías generalmente esta aplicación?
¿Tratarías esta aplicación como una herramienta más de trabajo o de estudio?

5.1.4.2 Tareas a realizar

1.	Accede a la aplicación. Si no estás registrado, hazlo y después accede a la aplicación.
2.	Cambia los datos de tu perfil de usuario.
3.	Visualiza todo el contenido de una entrada concreta que haya publicado un usuario cualquiera.
4.	Busca y visualiza una entrada relacionada con un concepto relacionado con esta tecnología.
5.	Busca a otros usuarios por su nombre de usuario o email.
6.	Sigue a un nuevo usuario.
7.	Deja de seguir a un usuario.
8.	Visualiza las publicaciones de un usuario.
9.	Crea una nueva entrada con todos los pasos que correspondan.
10.	Guarda la entrada sin publicarla.
11.	Publica una entrada en el momento de crearla.
12.	Recupera una entrada guardada como borrador y publícala.
13.	Comenta una publicación.
14.	Haz "Me gusta" en una publicación.
15.	Marca una publicación como que tú 'la has hecho'.
16.	Cierra tu sesión de usuario.

5.1.4.3 Preguntas sobre las tareas

¿Te ha resultado intuitivo el registro y acceso a la aplicación?
Dentro de la aplicación, ¿has entendido para qué era cada uno de los componentes de la pantalla principal?
¿Te ha costado mucho esfuerzo averiguar cómo cambiar los datos de tu perfil?
¿Te ha resultado sencillo buscar entradas ya publicadas por otros usuarios?
¿Te parece clara la información que aparece en las entradas? ¿Cómo mostrarías esta información?
¿Te resulta cómoda la presentación de la información? ¿La asociarías a como estás acostumbrado a verla en otras redes sociales?
¿Te ha resultado cómodo buscar a otros usuarios y agregarlos a tus contactos? ¿Cómo crees que sería más cómodo?
¿Crees que la aplicación tiene los suficientes atajos para acceder a toda su funcionalidad o debería de tener más atajos?
¿Crees que el muro de las entradas tiene un exceso o deficiencia de información?
¿Te parecen intuitivos los iconos? ¿O crees que habría que sustituirlos por otros? ¿Y por texto?
¿Crees que es útil almacenar las entradas como borradores, sin publicarlas?

6 Diseño técnico

El presente capítulo tiene como objetivo describir el desarrollo llevado a cabo del proyecto RPS, correspondiente a la definición previa realizada en las fases anteriores, y se centra en la definición del paradigma de desarrollo aplicado al proyecto, en el modelado de la información que se trata y almacena en el sistema, en la definición y relación existente entre las clases que componen la aplicación.

Antes de realizar el diseño del proyecto, se han tomado una serie de decisiones que marcarán el desarrollo del mismo. Entre ellas, cabe destacar las siguientes:

- Desde el inicio del proyecto, se tomó como base de su arquitectura el uso de un servicio en nube que permitiera deslocalizar el almacenamiento de la información del proyecto. Esta decisión está fundamentada en la idea de trabajar, una vez desarrollado el proyecto, con un servicio estable, de alta disponibilidad y que soporte un número elevado de peticiones por segundo. Es por esto que se decidió trabajar con Parse, puesto que, además de cumplir con los requisitos marcados, es un servicio gratuito mientras no se superen las 30 peticiones por segundo que establecen como límite.

- El desarrollo de la aplicación, finalmente, está basado en Ionic Framework sobre AngularJS, aunque inicialmente la idea era realizar el desarrollo con EmberJS, puesto que es un framework con el que estoy familiarizado y es muy ágil a la hora de desarrollar aplicaciones

HTML5. Finalmente, la decisión fue utilizar AngularJS porque tiene una comunidad y un número de componentes propios y de terceros muy superior, entre ellos Ionic Framework, que está enfocado al desarrollo de aplicaciones HTML5 para móviles.

- Inicialmente, todo el código se iba a ejecutar en el dispositivo móvil utilizando un servidor de respaldo propio, pero tras estudiar el comportamiento de Parse, se ha observado que éste ofrece la posibilidad de ejecutar código en su propio servidor. Aprovechando esta funcionalidad, algunas de las acciones (principalmente, las que requieren un mayor número de consultas) se ejecutan en remoto, en lugar de hacerlo en el dispositivo.

6.1 Arquitectura del sistema

El proyecto RPS está basado en una arquitectura cliente-servidor, de esta forma, las tareas a realizar se distribuyen entre la máquina que realiza las funciones de servidor (almacenamiento de información y clases que trabajan directamente para el tratamiento de información) y la que realiza funciones de cliente, es decir, tareas orientadas a la experiencia de usuario. Adicionalmente, se ha aplicado MVC como paradigma de modelado del sistema.

MVC (*model-view-controller*) es un patrón para la definición de una arquitectura cliente-servidor que permite separar la lógica de la aplicación de la interfaz de usuario, lo que tiene bastantes ventajas:

- Al separar la vista del modelo, se puede cambiar fácilmente un componente sin afectar al otro. De esta forma, permite un desarrollo más independiente de cada uno de estos componentes, e incluso, sustituirlos por otros sin que el otro se vea afectado.

- Por otro lado, permite tener una separación estructural del código de cada componente, lo que mejora su comprensión.

- También facilita la reutilización de código, lo que permite reducir redundancias y errores derivados por las mismas.

MVC son las siglas de las palabras en inglés: *model, view y controller*, que se traducen a 'modelo', 'vista' y 'controlador'. Estas son las capas que definen una aplicación de este tipo:

- El modelo o *model* es la capa que gestiona la información de la aplicación. En el caso de este

proyecto, el modelo está dividido en dos subcapas:

  - Hay una capa que se ejecuta en el cliente formada por clases denominadas *services* que son las encargadas de solicitar información o realizar modificaciones en Parse. En el cliente, sólo desde estos servicios se accede a las clases que proporciona Parse para acceder a su información.

  - Hay una capa que se ejecuta en remoto, en el servidor de Parse, que se utiliza para realizar consultas complejas o múltiples sobre la información de Parse, por ejemplo, para obtener todos los datos de una entrada y sus entidades asociadas (pasos, comentarios, usuarios que han marcado '*like*' o '*done*', etc.). La decisión de hacer estas consultas en remoto está basada en agilidad, reducción de peticiones al servidor y sobre todo para mejorar el tiempo de respuesta de dichas consultas.

- El controlador o *controller* es el responsable de la comunicación entre las vistas y los modelos. Esta capa permite realizar comprobaciones y revisiones de la información que se gestiona antes de realizar una petición a los *services*. Por comodidad y claridad de código, se ha establecido como patrón en este proyecto que cada *controller* gestione únicamente la información que aparece en una única vista, de esta forma, habrá una relación uno a uno entre la vista y el controlador.

- La vista o *view* es la responsable de mostrar la información que le proporciona el *controller*. Estas vistas se corresponden, en este caso, a un fichero HTML que contendrá código HTML5, CSS, Javascript y directivas especiales de AngularJS e Ionic Framework, además de componentes de éstos últimos.

A continuación, se puede ver una representación gráfica de la aplicación del paradigma MVC para el proyecto RPS:

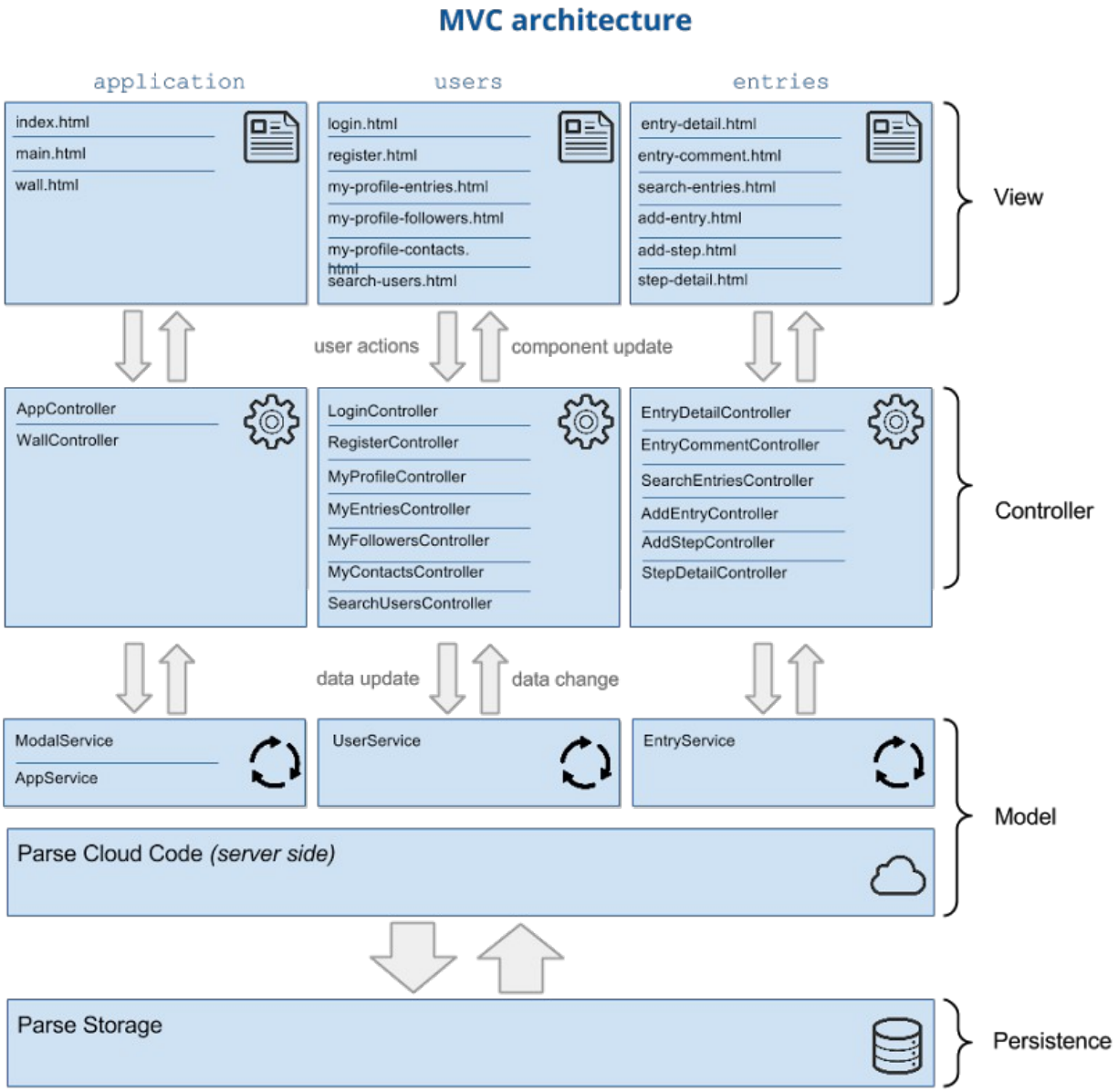


Ilustración 15: Arquitectura del proyecto basada en MVC

Como se puede ver en el diagrama anterior, además de la separación entre las capas de MVC, se ha realizado una división de los componentes según su finalidad:

- *application*, hace referencia a clases y componentes globales a toda la aplicación. Entre ellas, cabe destacar los que gestionan el muro de entradas, que es la funcionalidad principal de la aplicación.

- *users*, hace referencia a todos los componentes utilizados para la gestión de usuarios, como

puede ser el acceso a la aplicación o la acción de seguir a otro usuario.

- *entries*, hace referencia a todos los componentes necesarios para la gestión de entradas, como puede ser, crear una entrada o añadir comentarios a la misma.

6.2 Modelo de datos

El modelo de datos del proyecto RPS está basado en el paradigma del almacenamiento en nube, de forma que, la información se envía y recibe de un servicio, en este caso de un proveedor externo, lo que permite separar la gestión del *backend* (es decir, de los datos, consultas, transacciones, etc.) de la creación de una aplicación para dispositivos móviles. Para ello, la alternativa que más se ha adecuado al propósito de este proyecto es Parse.

6.2.1 Parse

Parse está compuesto por una serie de servicios complementarios entre sí. Sin embargo, para este proyecto se han utilizado dos de ellos: Parse Core y Parse Cloud Code. Éste último se detallará en el siguiente apartado. Otros servicios que proporciona, no contemplados para el proyecto, son: servicio de notificaciones *push*, que permite configurar una aplicación para que reciba en el dispositivo una notificación cuando se produzca un evento, y un servicio de análisis de rendimiento y acceso que proporciona información de interés para conocer el impacto de la aplicación.

Parse Core es un servicio que permite almacenar y consultar información. En varios aspectos, funciona de forma similar a una base de datos convencional, con la diferencia de que permite desentenderse de su gestión. Este servicio, orientado a objetos, permite guardar de forma segura (todas las conexiones con HTTPS) datos básicos y complejos, ficheros e imágenes, relaciones entre objetos, etc.

Para acceder y guardar la información a través de su servicio, proporcionan unas librerías de acceso (o *SDK - software developer kit*) en diferentes lenguajes y para las plataformas más extendidas (Android, iOS, Javascript, Unity, PHP, .Net, etc.). Es decir, a través de este SDK se tiene un acceso completo al que será el *backend* del proyecto.

6.2.1.1 Modelo de datos en Parse

Parse es un servicio orientado a objetos, y por tanto, su modelo de datos está basado en clases, y la creación del mismo se realiza a través de una herramienta *online* que proporcionan para este cometido. En Parse existen los siguientes tipos de datos:

Tipo	Descripción
String	Permite añadir un campo como una cadena
Number	Permite añadir un campo como un número real
Boolean	Permite añadir un campo como valor lógico
Date	Permite añadir un campo como una fecha
File	Permite añadir un campo como un fichero
GeoPoint	Permite añadir un campo como una localización geográfica
Array	Permite añadir un campo como una lista de elementos
Object	Permite añadir un campo como un object genérico
Pointer	Permite añadir un campo como una relación uno-a-uno y uno-a-muchos..
Relation	Permite añadir un campo como una relación muchos-a-muchos.

En este caso, el tipo de dato file será usado para subir las imágenes del proyecto. No está previsto utilizar para este proyecto los tipos *GeoPoint* y *Object*, aunque en una futura versión del mismo podría ser interesante su uso.

6.2.1.2 Seguridad de datos en Parse

En este proyecto se utiliza el SDK de Javascript para el acceso a Parse. Esto, a simple vista, tiene un inconveniente de seguridad de datos, ya que el código Javascript se ejecuta íntegramente en el cliente, en este caso, en el dispositivo móvil, lo que podría provocar que alguien que 'espíe' la transmisión de información desde el dispositivo a Parse, y viceversa, accediera a nuestras credenciales y nos suplantara para obtener información delicada.

Sin embargo, esto no es posible puesto que Parse cuenta con varios mecanismos para proteger los

accesos a la información:

- Conexión cliente-servidor. Cuando una aplicación se conecta a Parse, lo hace con dos claves: *applicationID* y *clientKey*. Estos datos no son secretos y por tanto, por sí solos no garantizan la seguridad de los datos. Sin embargo, existe una clave maestra que, como su nombre indica, permite disponer o anular el resto de mecanismos de seguridad. Esta debe de estar siempre a buen recaudo porque da acceso a todo el servicio. La filosofía de Parse es limitar las acciones que pueden llevar a cabo los usuarios. Esta llave maestra se utiliza únicamente en las acciones llevadas a cabo desde Cloud Code, al que no puede acceder ningún usuario. Además, todas las conexiones son HTTPS y SSL.

- Permisos de *class-level*. Este nivel de seguridad se da en el esquema de datos, al que sólo puede acceder el propio desarrollador. Esto permite aplicar restricciones a las acciones que se pueden realizar desde una aplicación. Por ejemplo, se puede limitar que una aplicación cree nuevas clases (o tablas), o que incluya columnas, etc. También puede limitar el acceso a nivel de modificación o consulta de datos. Estos permisos pueden aplicarse a nivel de aplicación, de usuario o de roles.

- Restricciones a nivel de objeto. Esta restricción permite restringir los datos de un usuario de los de otro, de forma que se pueda controlar quién y cómo (escritura / lectura) puede acceder a los mismos. Este mecanismo está basado en un campo que se añade al crear una clase. Este campo se llama ACL (*Access Control Lists* o Listas de Control de Acceso). Cada objeto tiene una lista de usuarios y roles junto a los permisos que cada uno de ellos tiene lo que proporciona una gran flexibilidad y control sobre el acceso a la información.

- Por último, siempre está la posibilidad de realizar las consultas a través del servicio Cloud Code que sí se ejecuta de forma remota, por lo que no hay ninguna posibilidad de que se pueda apropiar alguien de forma indebida de los datos de la aplicación.

6.2.2 Modelo de datos de RPS

El modelo de datos que aparece a continuación es un modelo entidad-relación de la base de datos que tiene RPS, y que es fácilmente extrapolable al modelo orientado a objetos de Parse.

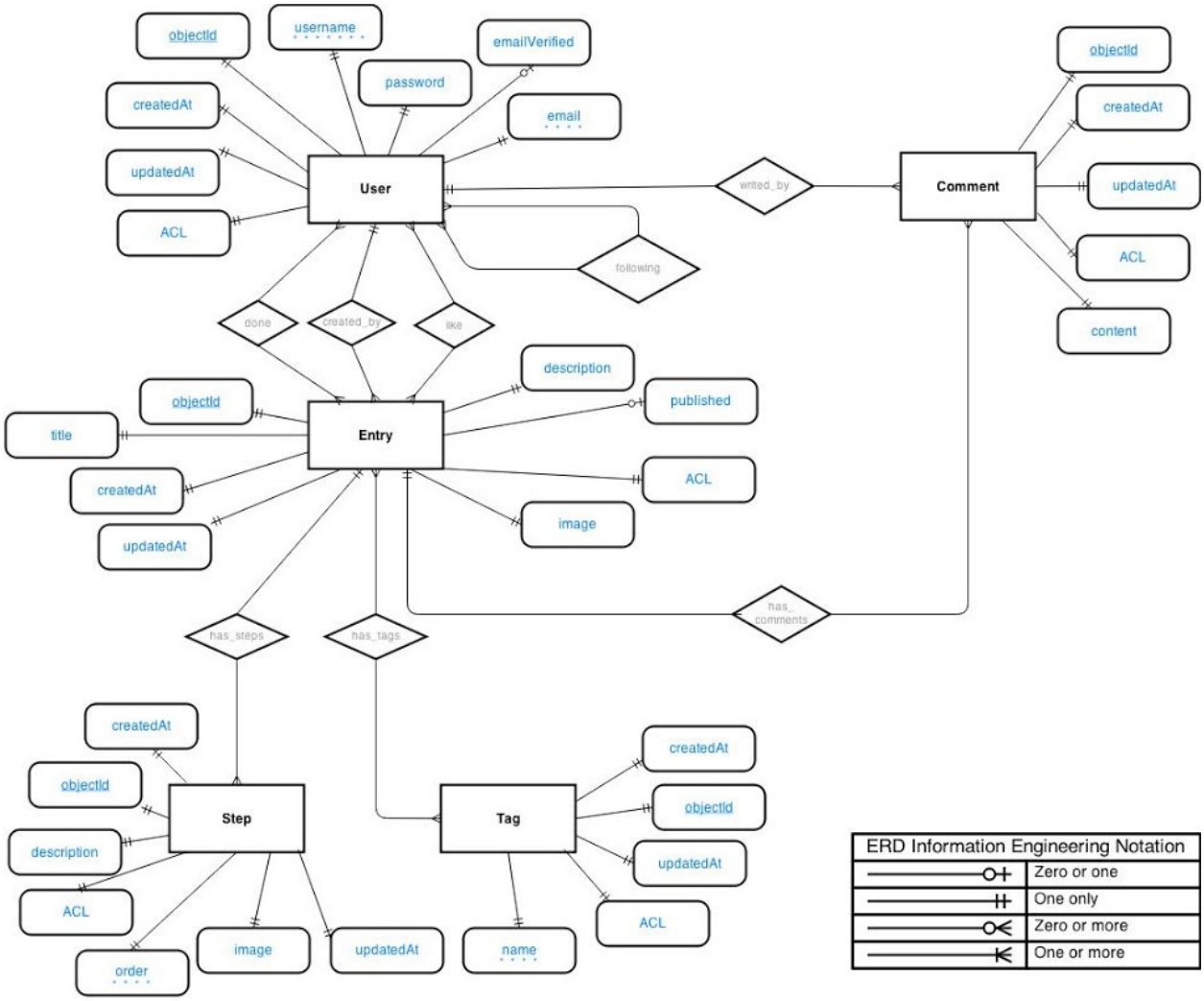


Ilustración 16: Diagrama Entidad-Relación del proyecto RPS.

Como se puede apreciar, hay 5 entidades que forman la base de datos de RPS. Todas ellas contienen, al menos los siguientes campos:

- *objectId* (String): es la clave primaria de todas las clases. Se autogenera cuando se crea un nuevo objeto.
- *createdAt* (Date): cuando se crea un objeto, Parse añade la fecha y hora de su creación
- *updatedAt* (Date): cada vez que se modifica un objeto, Parse actualiza la fecha y hora del mismo
- *ACL* (ACL): es la lista de control de acceso. Por defecto está vacía.

User

Define al usuario. Contiene información relativa al usuario.

Campo	Tipo	Descripción
username	String	Identificador único de usuario (obligatorio)
password	String	Clave de acceso del usuario (obligatorio)
authData	String	Dato de control de la sesión (interno de Parse)
email	String	Dirección de correo del usuario (obligatorio)
emailVerified	String	Confirmación de que se ha metido correctamente el correo (no obligatorio en esta versión)

Entry

Define una entrada. Contiene la información relativa a la entrada.

Campo	Tipo	Descripción
title	String	Título de la entrada (obligatorio)
description	String	Descripción de la entrada (obligatorio)
publish	Boolean	indica si la entrada se ha publicado o no en el muro.
image	File	Imagen principal de la entrada (obligatorio)
createdBy	Pointer<User>	Usuario que ha creado la entrada (obligatorio)
tags	Relation<Tag>	Lista de tags asociados a la entrada
done	Relation<User>	Lista de usuarios que han hecho click en 'done'
like	Relation<User>	Lista de usuarios que han hecho click en 'like'

Step

Define un paso de una entrada. Contiene la información relativa a un paso.

Campo	Tipo	Descripción
description	String	Descripción de la entrada (obligatorio)
entry	Pointer<User>	Referencia a la entrada de la que depende (obligatorio)
image	File	Imagen de este paso (obligatorio)
order	Number	índice del paso

Comment

Define un comentario. Contiene la información relativa a un comentario realizado por un usuario para una entrada concreta.

Campo	Tipo	Descripción
content	String	Contiene el comentario en sí. (obligatorio)
entry	Pointer<User>	Referencia a la entrada de la que depende (obligatorio)
createdBy	Pointer<User>	Usuario que ha introducido el comentario (obligatorio)

Tag

Define un comentario. Contiene la información relativa a un comentario realizado por un usuario para una entrada concreta.

Campo	Tipo	Descripción
name	String	Contiene el nombre del tag. (obligatorio)

6.3 Modelo de clases

En este apartado se describen las decisiones de diseño que se han realizado respecto al modelo de clases de la aplicación. Como se indicó en el apartado de 'Arquitectura del sistema', las clases están diferenciadas en función de dos categorías: la capa a la que pertenecen y los datos que gestionan. A continuación se muestra el diagrama de clases correspondiente a este proyecto:

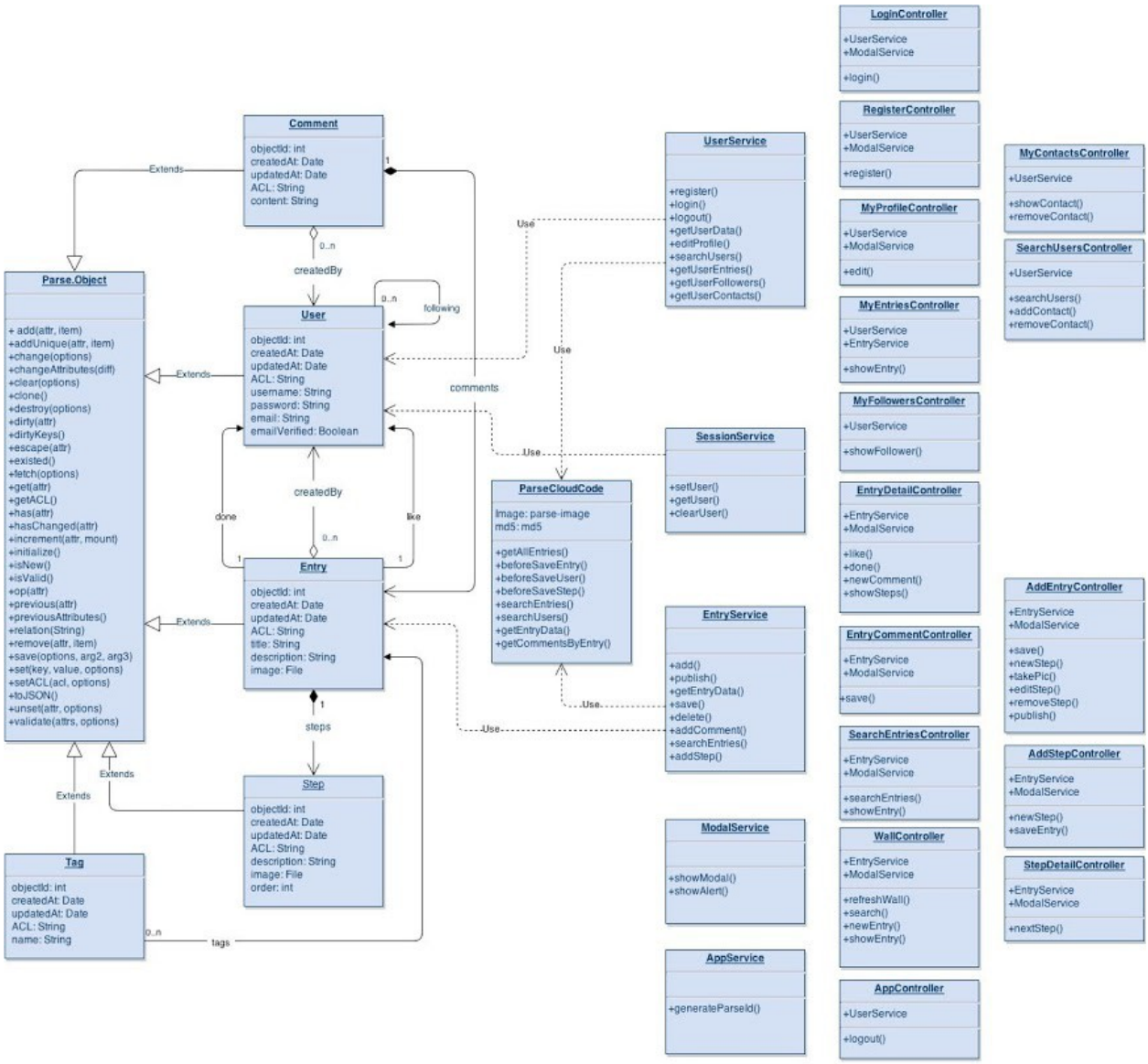


Ilustración 17: Diagrama de clases del proyecto RPS.

En el diagrama anterior, podemos distinguir las clases según la capa a la que corresponden:

- **Clases del SDK de Parse.** Estas clases definen los objetos que están almacenados en Parse. Extienden de Parse.Object. Esta clase tiene la información organizada en pares de 'clave-valor' compatibles con JSON. De esta forma, las clases que extienden de ella no tienen atributos específicos para cada una de ellas. La representación anterior no es exacta al funcionamiento de estas clases, pero es la mejor forma de proporcionar un vistazo a la información que manejan. Estas clases no están definidas previamente como tales, sino que se instancian en el momento de su uso y todos los métodos que contienen son los proporcionados por Parse. No es necesario agregar nuevos métodos para la interacción con Parse. Las clases que extienden Parse.Object son:
    - **User**, que instancia un objeto con la información del usuario, almacenada en Parse como '_User' ya que es una clase predefinida por este servicio para facilitar la gestión de usuarios.
    - **Entry**, que proporciona la información de las entradas almacenadas en Parse en objetos tipo 'Entry'.
    - **Step**, que proporciona la información de los pasos correspondientes a entradas almacenadas en Parse en objetos tipo 'Step'.
    - **Comment**, que proporciona la información de los comentarios que realizan los usuarios en las entradas, almacenadas en Parse en objetos tipo 'Comment'.
    - **Tag**, que proporciona la información de las etiquetas que se incorporan a las entradas, almacenadas en Parse en objetos tipo 'Tag'.

En el caso de que se requiera trabajar con los datos de una entrada, deberán ejecutarse las siguientes líneas:

```
var Entry = Parse.Object.extend('Entry');
var entry = new Entry();
```

A partir de aquí, se podrá trabajar con el objeto *entry* como con cualquier objeto.

- **Clases del modelo.** Como se indicó previamente, existen dos niveles de clases para la lógica del proyecto:

- **En cliente:** se definen las clase de tipo *service*. Se llaman desde los *controllers* y son las clases encargadas de realizar todas las consultas y gestiones de los datos almacenados en Parse (y otras rutinas). Para ello, dentro de los servicios se instancian las clases correspondientes a las entidades almacenadas en remoto. La información que proviene del servidor se transforma para que sea legible desde la vista. Las clases *service* son las siguientes:
  - **UserService:** su labor es gestionar toda la información de los usuarios, desde el control de acceso hasta el seguimiento de otros usuarios.
  - **SessionService:** encargada de gestionar la información de usuario en la sesión actual.
  - **EntryService:** su labor es gestionar toda la información de las entradas, su creación, la inserción de pasos en la entrada, comentarios de usuarios, etc.
  - **ModalService:** es un servicio especial que no accede a la información de Parse, sino que controla la forma en la que se muestran los mensajes en los dispositivos.
  - **AppService:** es otro servicio especial que se utiliza para incluir acciones que no corresponden a ninguna funcionalidad concreta. En este caso, se utiliza para generar un identificador aleatorio que se empleará, por ejemplo, para nombrar las imágenes que se suban a Parse.
- **En remoto:** existe una clase predefinida por Parse, llamada Parse.Cloud a la que se pueden incorporar nuevos métodos para poder realizar consultas complejas o que requieran de un grado de seguridad mayor que las ejecutadas en el cliente. A esta clase sólo se la podrá llamar desde las clases del modelo *services*, lo que permitirá una mayor abstracción para las llamadas realizadas desde el controlador.

- **Clases del controlador.** Son los intermediarios entre los servicios y las vistas de la aplicación. Responsables de la validación de la información de entrada del usuario y de la actualización de la información del modelo. Los métodos que se ejecutan en las vistas corresponden a su controlador. Se ha definido un controlador por cada vista, para que resulte más cómodo y sencillo trabajar con estas clases. Los *controllers* definidos son:

- **AppController:** Es un controlador transversal, ya que sus métodos se ejecutarán desde el menú principal de la aplicación.
- **LoginController**: Encargado de comprobar los datos de usuario para que pueda acceder a la aplicación.
- **RegisterController**: Encargado de comprobar los datos que inserta el usuario para registrarse en la aplicación.
- **WallController**: Encargado de mostrar la información de las entradas que deben aparecer en el 'muro' así como permitir que la información que aparece se pueda actualizar.
- **AddEntryController**: Encargado de comprobar que los datos de una nueva entrada sean los correctos. Permitirá agregar imágenes y nuevos pasos a la entrada.
- **AddStepController**: Encargado de gestionar la información que se suba para la creación de un nuevo paso de una entrada.
- **EntryDetailController**: Obtiene la información de una entrada que ya está publicada en el muro.
- **EntryCommentController**: Encargado de la inserción de los comentarios que realizan los usuarios en las entradas ya publicadas.
- **StepDetailController**: Obtiene la información de un paso de una entrada para mostrarlo en la vista.
- **MyProfileController**: Obtiene la información del perfil de un usuario para mostrarlo en la vista.
- **MyEntriesController**: Obtiene un listado de las entradas que el usuario conectado ha publicado.
- **MyFollowersController**: Obtiene un listado de seguidores del usuario conectado.
- **MyContactsController**: Obtiene un listado de contactos del usuario conectado.

- **SearchEntriesController**: Proporciona los mecanismos de búsqueda y obtiene los resultados para encontrar entradas ya publicadas.
- **SearchUsersController**: Proporciona los mecanismos de búsqueda y obtiene los resultados para encontrar usuarios registrados en el sistema.

- Clases de la vista. La vista no está formada por clases, sino por documentos HTML5 (HTML, CSS y Javascript) que utilizan directrices e inyectores de AngularJS, Ionic Framework para recuperar y solicitar información de los controladores.

7 Implementación y pruebas

El desarrollo del proyecto ha sufrido algunos pequeños cambios respecto al diseño original debido a restricciones de los *frameworks* o de algunas de las librerías utilizadas, o incluso a la conveniencia de refactorización de algunas clases que han quedado obsoletas durante el desarrollo. En este apartado se describen las tecnologías usadas, así como detalles sobre la implementación.

7.1 Tecnología aplicada

A continuación, se indican los elementos utilizados, distinguidos por su cometido o por el tipo de componente:

- A nivel *hardware*, meramente informativo, se han utilizado dos máquinas diferentes: un Macbook Pro (i5, 16GB RAM) y un PC (i7, 8GB RAM). Los sistemas operativos han sido Mac Os X 10.10.1 y Linuxmint respectivamente. Las pruebas en navegador se han realizado con Chrome y Chromium ambos en su versión 39.

- Como IDE de desarrollo he utilizado **Atom**[6], un editor bastante flexible que es un *fork* libre de Sublime Text. En este editor, los principales *plugins* que me han facilitado el desarrollo del proyecto son:

6 Atom: https://atom.io/

- AngularJS - para trabajar con AngularJS.
- Ask Stack - para encontrar respuesta en StackOverflow cuando he tenido algún 'atranco' que no me permitía continuar.
- Otros para resaltar y colorear el código en base a su contenido y también para el autocompletado.
- Git Plus y Git Diff - para subir los cambios a mi repositorio personal.
- JShint - para validar el código Javascript.
- Language CSS, Javascript, Sass, Git, Html, Json, etc. para soportar estos lenguajes en Atom.
- Otros menos relevantes.

- Respecto a la tecnología aplicada al proyecto se puede distinguir:
  - **NodeJS**[7] - es un entorno de programación pero que realmente sólo he utilizado como apoyo para trabajar con herramientas de integración continua que dependen de este entorno.
  - **npm**[8] - es el gestor de paquetes de NodeJS (Node Package Manager) con el que gestiono la descarga de paquetes necesarios para el proyecto.
  - **Bower**[9] - es un gestor de paquetes, dependiente de npm, que gestiona paquetes de desarrollo del *frontend*. La mayoría de los paquetes utilizados en el proyecto se pueden descargar de Bower.
  - **Apache Cordova**[10] - es un conjunto de APIs que permiten a los desarrolladores acceder a funciones nativas de los dispositivos móviles a través de Javascript. También permite encapsular una *webapp* en una aplicación que se instala de forma nativa en el dispositivo, lanzando la aplicación a través de un componente *WebView*, en el caso de

7 NodeJs: http://nodejs.org/

8 npm: https://www.npmjs.com/

9 Bower: http://bower.io/

10 http://cordova.apache.org/

Android basado en el navegador Chrome.

- **WebView**[11] - es el componente encargado de mostrar la aplicación usando el motor de renderizado de Google Chrome. Precisamente, este componente será el responsable del buen funcionamiento de la interacción de la aplicación con los recursos del dispositivo. Más adelante explicaré algunos casos en los que este componente no funciona de forma similar a Chrome, lo que ha provocado que el uso de algunas librerías, que funcionan correctamente en Chrome, ya sea en escritorio o en móvil, no funcionen bien en el WebView.

- **Grunt**[12] - es una plataforma para la automatización de las tareas que hay que realizar para compilar y empaquetar el código, lanzar el servidor, enviar al dispositivo la aplicación, etc.

- **JSHint**[13] - utilizado tanto por Grunt como por el editor de texto usado para comprobar errores de programación en Javascript.

- **AngularJS**[14] - es un *framework* de Javascript desarrollado y mantenido por Google. Está basado en el patrón MVC. AngularJS tiene muchas extensiones, propias o de terceros, que proporcionan soluciones para casi todos los problemas planteados hasta el momento en el desarrollo de aplicaciones web.

- **Ionic Framework**[15] - *framework* que proporciona un conjunto de librerías y utilidades para la creación de aplicaciones web atractivas para el usuario. Tiene integración total con AngularJS y Cordova. Además , Ionic proporciona todos los iconos utilizados en la aplicación mediante **Ionicons**[16].

- **Sass**[17] - (Syntactically Awesome StyleSheets). Como su nombre indica, es un lenguaje de programación de estilos, basado en CSS, que amplía y mejora la capacidad de éste de la

11 WebView: http://developer.android.com/reference/android/webkit/WebView.html
12 Grunt: http://gruntjs.com/
13 JSHint: http://jshint.com/
14 AngularJS: https://angularjs.org/
15 Ionic Framework: http://ionicframework.com/
16 Ionicons: http://ionicons.com/
17 Sass: http://sass-lang.com/

gestión de estilos, proporcionando la posibilidad de crear variables, métodos (*mixins*) y otras características que mejoran la manipulación de los estilos de las aplicaciones.

7.2 Composición del proyectos

Estructura de directorios

Para la generación de la estructura de carpetas de la aplicación se utiliza una herramienta llamada Yeoman . Ésta es una herramienta que, mediante *plugins*, permite la generación de 'esqueletos' o 'andamios' (*scaffolds* en inglés) para proyectos con diferentes *frameworks*. En este caso, se ha utilizado el generador llamado generator-ionic . Las principales carpetas de este esqueleto son:

- ***app***. Aquí se encuentra el código fuente de la aplicación, tanto scripts como recursos y plantillas. En este caso se ha introducido un pequeño cambio en la estructura de carpetas usada por defecto por el generador, ya que éste hacía una separación por tipo de componentes (controladores, servicios, etc.), y se modificó para organizarlo según módulos (rps, users y entries).

Ilustración 2: Estrucutura de carpetas principal del proyecto.

En la ilustración anterior se distinguen las carpetas según su funcionalidad:

- ***fonts***: contiene las fuentes utilizadas por Ionic para mostrar los iconos de la aplicación.
- ***lib***: contiene el código fuente de todas las librerías utilizadas en la aplicación: angular, angular-animate, angular-gravatar, angular-mocks, angular-moment, angular-sanitize, angular-scenario, angular-ui-router, collide, ionic, jasmine, jr-crop, moment, ngImageEditor, ngImgCrop, parse-js-sdk, sinon-browser-only
- ***resources***: conjunto de elementos visuales utilizados en el desarrollo: imágenes y fuentes.
- ***scripts***: contiene las clases del proyecto. Se distinguen entre controladores (*controllers*) y servicios (*services*). Estas clases están divididas en carpetas según su funcionalidad.
- ***styles***: contiene los ficheros Sass que se utilizarán para modificar los componentes estéticos de la aplicación.

- **templates**: contiene el código HTML de la aplicación. Estos ficheros están divididos en carpetas según su funcionalidad o utilidad.

- ***libs***. Aquí se ubican las librerías externas utilizadas, en este caso solo se encuentra Crosswalk.

- ***tests***. Como su nombre indica, aquí se encuentran los tests unitarios de nuestra aplicación. Una vez dentro de esta carpeta, se reproduce la misma estructura de ficheros y carpetas que dentro de *app/scripts*, para seguir una misma nomenclatura y hacer más fácil al programador el encontrar los ficheros que necesita.

El esqueleto del proyecto incluye, también, una serie de herramientas y configuraciones que facilitan al programador el desarrollo de la aplicación, como es gruntjs, el cual, también mediante *plugins*, permite automatizar tareas repetitivas que, de otra forma, absorberían bastante tiempo del desarrollo.

AngularJS

El código fuente Javascript de la aplicación se compone de dos tipos de elementos:

- **Servicios**. Son los encargados de la lógica de la aplicación, así como de la conexión con los servicios externos, en este caso el *backend* en Parse. Estos servicios, que se inyectan en los controladores, publican métodos que permiten la ejecución de la lógica y la obtención de datos remotos. Su definición se realiza de la siguiente forma:

```
rpsApp.factory('EntryService', ['UtilsService', 'ErrorsService',
    function(UtilsService, ErrorsService) {
        var service = {};
       service.list = function () {};
       return service;
    }
]);
```

En este ejemplo, se define un servicio llamado *EntryService*, al que se le inyectan otros dos servicios (*UserService y ErrorsService).* Todos los atributos del objeto *service* son publicados para su uso fuera de este servicio.

- **Controladores**. Son los componentes que conectan los datos obtenidos de los servicios con los templates o vistas. Su definición se realiza de la siguiente forma:

```
rpsControllers.controller('WallCtrl', [
        '$scope',
        'EntryService',
        '$ionicLoading',
                'ModalService',
        '$state',
        function($scope, EntryService, $ionicLoading, ModalService, $state) {
                $scope.entries = [];
        }
]);
```

En este ejemplo, se define un controlador llamado *WallCtrl*, al que se le inyectan los servicios *$scope*, *EntryService*, *$ionicLoading*, etc. Todos los atributos o métodos del objeto *$scope* (objeto estándar de AngularJS), son accesibles desde las vistas.

Aunque, tanto Javascript como AngularJS permiten la definición de diferentes componentes dentro de un mismo fichero, ya sean servicios o controladores, para la implementación de este proyecto, se ha decidido separar cada componente en un único fichero, incluyendo los que constan de unas pocas líneas de código, para hacer más clara la estructura de ficheros y carpetas y facilitar así el acceso a los distintos componentes para el programador.

Templates

Para las plantillas de la aplicación, se utiliza HTML5 junto con las directivas de AngularJS e Ionic Framework. Mediante las directivas, Angular permite la extensión de HTML, pudiendo crear nuevas etiquetas, atributos, o clases CSS, que pueden no sólo añadir contenido a nuestras plantillas de una manera sencilla y generalizada, sino también añadir funcionalidad. Un buen ejemplo de una directiva, sería *ng-repeat*:

```
<div ng-repeat="entry in entries" class="list card">
        {{entry.title}}
```

```
</div>
```

Como puede verse en este ejemplo, la directiva *ng-repeat* itera sobre una colección de elementos (*entries*) publicada en el *$scope* del controlador. Permitiendo así de una forma sencilla mostrar los datos de las entradas en el muro.

Sass

Para definir el componente visual de la aplicación, hay varias opciones en la actualidad que permiten trabajar con mayor flexibilidad que directamente creando estilos CSS. Entre estas opciones está Sass (*Syntactically Awesome Stylesheets*) que permite el uso de variables, reglas CSS anidadas o el uso de un componente denominado *mixin*. Los *mixins* funcionan como reglas CSS con la particularidad de que se pueden reutilizar y parametrizar, lo que simplifica de una forma considerable la composición de estilos CSS.

En el proyecto RPS se han tomado como base las hojas de estilos proporcionadas por Ionic Framework para el estilo de la aplicación. Ionic incluye un conjunto de ficheros *.scss* que contienen los estilos definidos para cada uno de los componentes que existen en su *framework*. Para poder utilizar estos estilos desde otro fichero basta con añadir la siguiente sentencia, en la que habrá que indicar la ruta del fichero 'ionic.scss' sin especificar su extensión:

```
@import "../lib/ionic/scss/ionic";
```

Una de las utilidades que proporciona Ionic es la incorporación de iconos configurables en color, posición y tamaño a través de etiquetas '<i>' en los *templates* del proyecto. Para ello, hay que definir una variable *$ionic-font-path* con la ruta a las fuentes que proporciona este *framework*, como aparece a continuación:

```
$ionicons-font-path: "../lib/ionic/release/fonts" !default;
```

Las fuentes que se han incluido en la hoja de estilos pertenecen a Ionicons[18] y la forma de incorporarlos al fichero HTML es muy sencilla:

```
<i class="icon ion-home"></i>
```

Al incluir las clases 'icon' e 'ion-home' se consigue que se muestre un icono en forma de casa que servirá para indicar un enlace a la página de inicio.

18 Ionicons: http://ionicons.com/

Por otro lado, también es útil la declaración de variables, por ejemplo, para definir los colores que se van a utilizar en la aplicación. Las variables se declaran de la siguiente forma:

```
$rps-red-pi: #D30F3B;

$rps-red-pi-80: rgba(211, 15, 59, 0.8);
```

Ambos ejemplos son válidos, y permiten definir dos colores del mismo tono, con la diferencia de que el segundo tiene un 80% de opacidad. De esta forma, cuando se quiere definir un estilo con el color elegido, se puede declarar de la siguiente forma:

```
.red-pi { color: $rps-red-pi !important; }
```

De esta forma, se ha creado una clase CSS que modificará la etiqueta a la que se asigne con el color indicado por la variable *$rps-red-pi*.

Además de lo anterior, se realizan anidaciones de estilos para combinar estilos en algunos casos o restringir la aplicación de estilos en otros casos. Esta segunda opción se ha aplicado en diferentes componentes para restringir algunas de las propiedades que proporcionan los estilos de Ionic Framework. El motivo ha sido que aunque la base de la composición de los elementos de Ionic son los que se necesitaban para RPS, la elección de colores, sombreados, márgenes, etc. no son los que se han definido para este proyecto.

Una vez definidas todas las clases de los ficheros '.scss' se podrán transformar en hojas CSS3 válidas realizando la conversión mediante un intérprete de Sass. En el caso de RPS, se ha utilizado un plugin de Grunt llamado 'grunt-sass'. Para utilizarlo hay que indicar a Grunt la ruta donde está el fichero principal que se quiere compilar.

Crosswalk

Durante la implementación y pruebas de RPS se ha detectado que el componente WebView por defecto de Android, el renderizado de Cordova tiene algunas carencias:

- La fluidez requerida para mover páginas con muchas imágenes u otros contenidos que tengan un volumen considerable o la ejecución de código Javascript para controlar algunos eventos, es bastante limitada.

- El comportamiento de algunos elementos con dispositivos con Android 4.x no es el correcto, o al menos, no es el mismo que se puede comprobar en dispositivos con Android

5 o con navegadores como Chrome.

Por estos motivos, se ha estudiado la posibilidad de sustituir este componente por otro existente llamado Crosswalk. Los resultados que eran mediocres con WebView han mejorado sustancialmente con este otro componente, aunque en el futuro se buscarán nuevas soluciones de cara a mejorar el rendimiento en todo tipo de dispositivos.

7.3 Módulo principal

RPS es una aplicación para dispositivos móviles bastante sencilla en la que se distingue dos elementos principales: los usuarios de la aplicación y las entradas que estos usuarios agregan a la aplicación. Sin embargo, para poder trabajar con estas entidades es necesario disponer de un componente principal que será el núcleo de la aplicación.

En este caso, el módulo principal está compuesto por varios elementos:

- **index.html:** Es la página base de la aplicación. Esta página contiene todos los enlaces introducidos manualmente o a través de Grunt. Al ejecutarse la aplicación, se cargarán todos los elementos HTML, Javascript, CSS, fuentes, etc. definidos en esta página. Por la configuración que se ha definido mediante Grunt, cuando se crea un nuevo componente Javascript de AngularJS, al compilar el proyecto, si ese fichero no estaba incluido en el index.html lo incluye de forma automática.
- **main.html:** Contiene los componentes visuales principales de la aplicación. Estos componentes están basados en Ionic Framework:

```
<ion-side-menus>

    <ion-pane ion-side-menu-content>
        <ion-nav-bar class="bar-assertive rps-bar">
            <ion-nav-back-button class="button-clear"><i class="icon ion-ios7-arrow-back"></i></ion-nav-back-button>
        </ion-nav-bar>
        <ion-nav-view name="appContent" animation="slide-left-right"></ion-nav-view>
    </ion-pane>

    <ion-side-menu side="left">
        <ion-content class="menu-bg">
            <ion-list class="menu-list">
                <ion-item menu-close ui-sref="app.profile.entries({userId: currentUser.id})" class="ion-item-no-padding profile-item">
                    <div id="menu-user">
                      <img class="circle border-circle" ng-src="http://gravatar.com/avatar/{{currentUser.md5Email}}?d=mm">
                      <h2>{{currentUser.username}}</h2>
                    </div>
                </ion-item>
                <ion-item nav-clear menu-close ui-sref="app.wall">
                    <i class="icon ion-home" style="margin-right: 10px;"></i>Inicio
                </ion-item>
                <ion-item menu-close ui-sref="app.addEntry">
                    <i class="icon ion-plus" style="margin-right: 10px;"></i>Crear entrada
                </ion-item>
                <ion-item menu-close ui-sref="app.search.entries">
                    <i class="icon ion-search" style="margin-right: 10px;"></i>Buscar
                </ion-item>
                <ion-item nav-clear menu-close ng-click="logOut()">
                     <i class="icon ion-log-out" style="margin-right: 10px;"></i>Salir
                </ion-item>
            </ion-list>
        </ion-content>
    </ion-side-menu>
</ion-side-menus>

```

Ilustración 3: Código fuente de main.html

Como se puede apreciar, en este fichero están configurados el panel principal *<ion-pane>*, donde se mostrará toda la funcionalidad de la aplicación y el panel que define el menú lateral desplazable *<ion-side-menu>*. Para que funcione correctamente ambos componentes deben de estar dentro del componente *<ion-side-menus>*.

- **app.js:** en este fichero Javascript se realiza la descripción del módulo general de la aplicación, de los cuales dependen las llamadas a los controladores y a las librerías necesarias para que el proyecto funcione correctamente.

```
'use strict';

/**
 * RPS App
 * Definición del módulo principal de la aplicación.
 */

window.rpsApp = angular.module('rps', ['ionic', 'rps.controllers', 'angularMoment', 'ngAnimate']);

```

Ilustración 4: Definición del módulo principal de la aplicación.

Contiene dos métodos que son básicos para el funcionamiento de la aplicación:

- **run**: Método de ejecución en la fase 'run' de la aplicación, donde se inicializan algunos de los componentes necesarios como las librerías Parse o Cordova.

- **config**: Método que se ejecuta en la fase 'config' de la aplicación. En este método se configuran todos estados (*states*) a los diferentes controladores, asociándolos a los

templates correspondientes para la comunicación entre ambos.

```
rpsApp.config(function($stateProvider, $urlRouterProvider) {
    $stateProvider

    .state('login', {
        url: '/login',

        templateUrl: 'templates/users/login.html',
        controller: 'LoginCtrl',

        data: {
            authenticate: false
        }
    })
```

Ilustración 5: Definición de un estado en el método config del módulo.

Como se puede ver en la ilustración anterior, en los estados se le pasan los parámetros necesarios para asociar el controlador con la vista, así como un atributo *data* que indica si dicha vista requiere autenticación o no.

- **Gestión de errores:** Para la gestión de los errores provenientes del *backend* se ha creado un servicio llamado *ErrorService*. Dicho servicio contiene una lista con los códigos de error devueltos por el *backend* y una descripción para cada uno de ellos. Este servicio es inyectado en los servicios, que cuando reciben un código de error desde el servidor, obtienen su mensaje correspondiente usando *ErrorsService.getMessage()*, y lo devuelven al controlador que realizó la llamada, pudiendo este último mostrar directamente al usuario dicho mensaje.

También forman parte del módulo principal los *scripts* (ficheros .js) y *templates* (ficheros .html) contenidos en las subcarpetas 'rps'. Son funcionalidades que tienen un carácter global y se muestran a continuación:

- **Muro de entradas:** Esta es la pantalla principal del proyecto. Aquí es donde se muestran las entradas que los usuarios publican para compartirlas con el resto de usuarios. El controlador para esta vista es *wall-controller*.

Ilustración 6: Pantalla principal. Muro de entradas.

El comportamiento de esta vista permite actualizar las entradas nuevas deslizando la parte movible (o *scrollable*) hacia abajo. De esta forma se evita tener un botón para recargar las entradas. De la misma forma, si desplazamos las entradas hacia arriba, cuando se llega al final de la pantalla se hace una nueva petición y muestra entradas más antiguas con un límite indicado. Esto se puede hacer hasta que no queden más entradas.

- **Salir:** Esta funcionalidad, aunque debería estar asociada al usuario, está en el módulo principal, puesto que se puede acceder a ella desde el menú lateral. Para ello, se deberá pulsar el icono de menú lateral y aparecerá el menú con la opción Salir. El controlador de esta acción es *app-controller* ya que es una acción global desde cualquier vista de la aplicación.

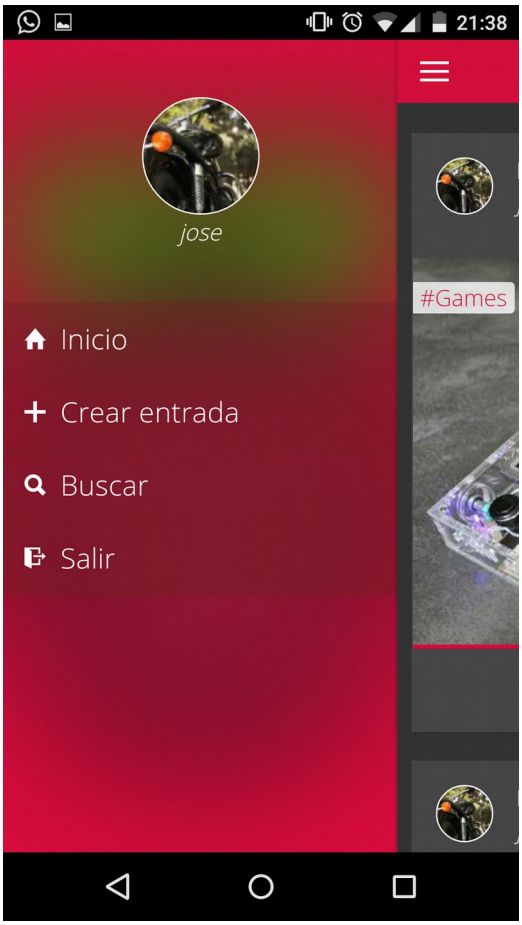


Ilustración 7: Menú lateral de la aplicación.

Al pulsar en el botón Salir se cerrará la sesión de usuario y aparecerá la pantalla de Acceso.

- **Búsqueda de entradas de usuario:** La búsqueda de entradas es otra de las opciones que está disponible desde el menú lateral de la aplicación. Permite buscar entradas por su nombre o descripción. Su controlador es *search-entries-controller*.

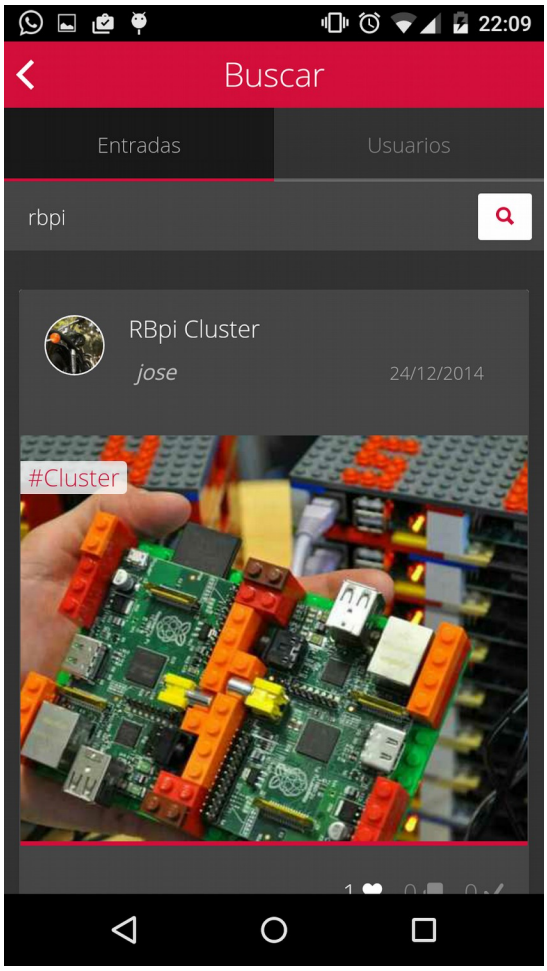


Ilustración 8: Búsqueda de entradas.

Esta pantalla, al igual que el muro, tiene la funcionalidad de mostrar entradas antiguas al desplazarse al final de la lista de entradas.

- **Búsqueda de usuarios:** Para acceder a esta funcionalidad se debe haber accedido a la funcionalidad de 'Búsqueda de entradas de usuarios' y desde allí seleccionar la pestaña 'Usuarios'. Permite buscar usuarios por su nombre de usuario o por su *email*. El controlador asociado es search-users-controller.

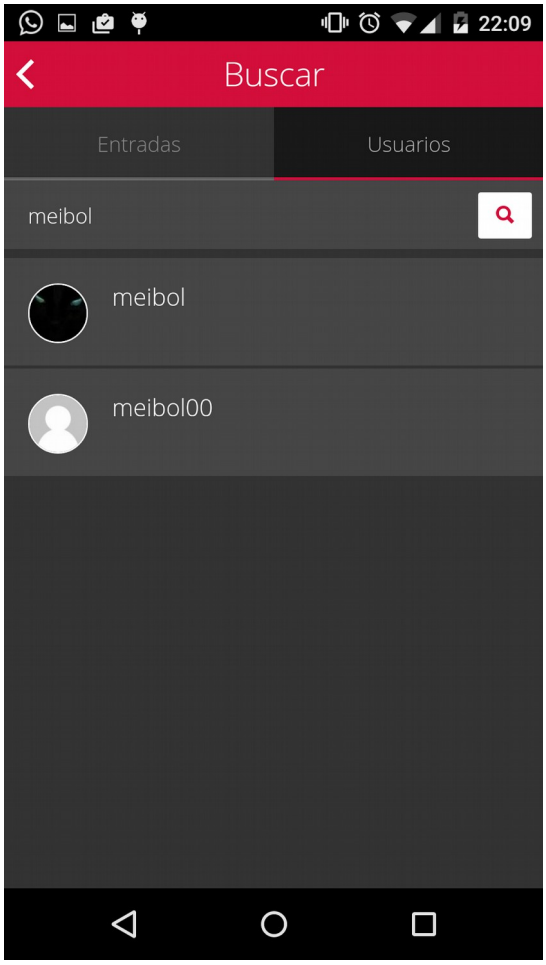


Ilustración 9: Búsqueda de usuarios.

Esta pantalla de búsqueda de usuarios también tiene la opción de cargar más nombres de usuarios al deslizar la pantalla hasta el final si no se hubiesen cargado todos en la búsqueda inicial.

7.4 Módulo de usuarios

El módulo de usuarios está compuesto por los *scripts* y *templates* que están contenidos en las subcarpetas 'users' respectivamente.

- **Login:** Esta es la pantalla de acceso a la aplicación. Para poder acceder hay que registrarse previamente. El controlador correspondiente es *login-controller*.

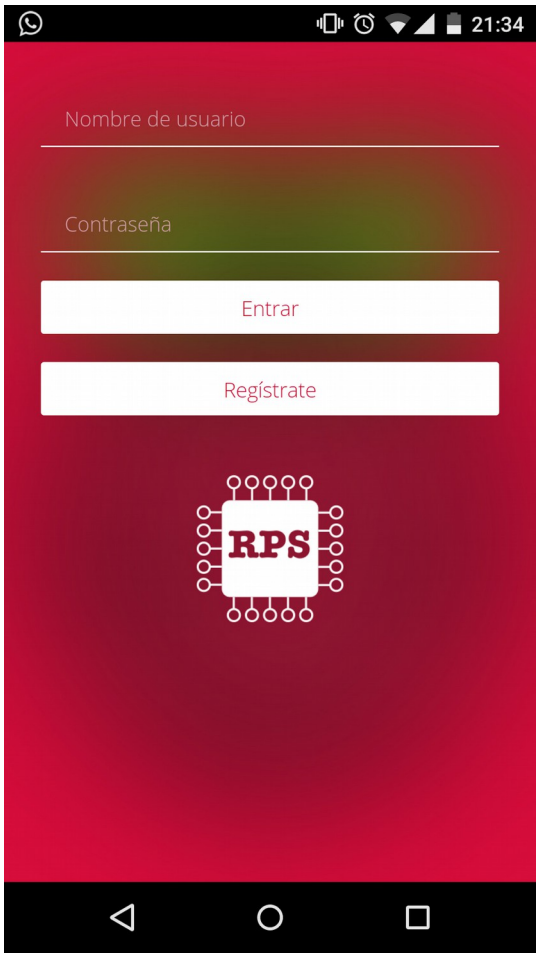


Ilustración 10: Pantalla de login o acceso a la plataforma.

En el caso de que los datos proporcionados para la autenticación no sean los correctos, mostrará un mensaje de error correspondiente al error producido.

- **Registro de usuarios:** Desde la pantalla de acceso se podrá acceder a esta pantalla pulsando el botón 'Regístrate' en el caso de que el usuario no esté registrado. El controladors de esta vista es *register-controller*.

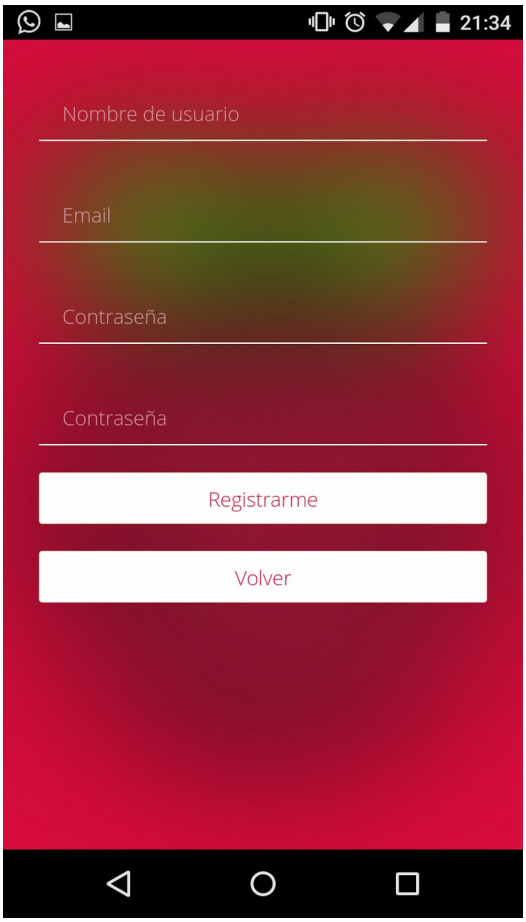


Ilustración 11: Pantalla de registro de usuarios.

En el caso de que los datos proporcionados para la autenticación no sean los correctos, mostrará un mensaje de error correspondiente al error producido.

- **Perfil de usuario:** El perfil de usuario puede ser propio o ajeno. En el caso de que sea propio, para acceder al perfil de usuario hay que pulsar en el icono de la barra superior para desplegar el menú. La carga de la información del perfil (imagen y nombre de usuario) se realiza mediante el controlador *profile-controller*.

  Tanto si el perfil es propio o ajeno, se puede acceder al mismo pulsando en la imagen del usuario, ya sea en el menú lateral (mi usuario) como en las imágenes que aparecen en las entradas del muro o en las vistas de búsqueda. Para acceder a la vista del perfil habrá que pulsar en la imagen, lo que abrirá otra vista con información detallada del usuario. El nombre de usuario y la imagen se vuelven a cargar con el controlador profile-controller, sin embargo, en esta vista además se mostrarán tres pestañas que contienen:

  - **Entradas:** muestra las entradas del usuario. Si son las entradas de otro usuario diferente al autenticado, sólo se mostrarán las entradas publicadas. Sin embargo, si es el propio usuario, aparecerán sus entradas publicadas y las no publicadas. Para distinguir unas de otras, las entradas no publicadas se muestran en otro color y con una etiqueta que indica que no están publicadas.

Ilustración 12: Entradas de Mi perfil.

- En el caso de que una entrada esté publicada, al pulsar sobre ella se mostrará la vista en modo visualización (se permitirá pulsar en 'me gusta', 'hecho' o incluir comentarios, pero no se podrá editar).
- En caso de que la entrada no esté publicada, al pulsar sobre la misma se abrirá en modo edición, es decir, se podrá modificar su contenido.

- **Seguidores:** muestra un listado de usuarios que nos siguen. Desde este listado se puede pulsar en la imagen del usuario para acceder a su perfil.

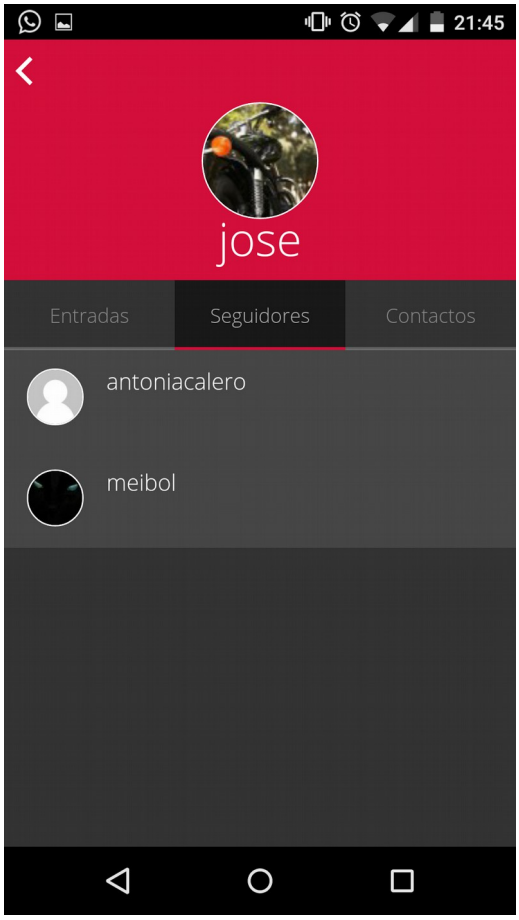


Ilustración 13: Pantalla de Mis seguidores.

- **Contactos:** muestra un listado de usuarios que nos siguen. Desde este listado se puede pulsar en la imagen del usuario para acceder a su perfil. Además se puede eliminar un contacto pulsando en el botón de borrado.

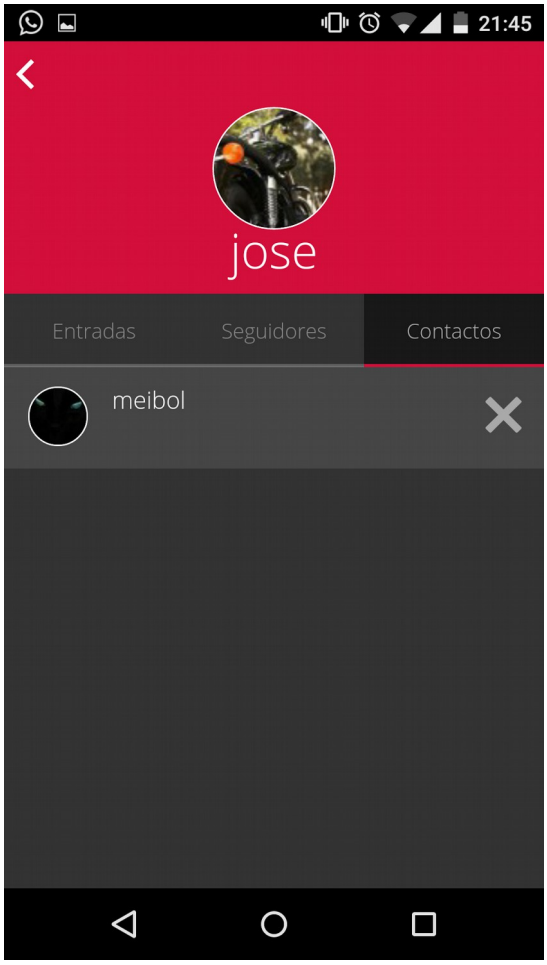


Ilustración 14: Pantalla de Mis contactos.

7.5 Módulo de entradas

En este módulo se reúnen todas las funcionalidades relacionadas con la gestión de entradas:

- **Añadir una entrada nueva**: Para acceder a esta funcionalidad, habrá que acceder al menú lateral pulsado el icono de la barra superior de la aplicación. Aparecerá una nueva vista que contiene la información necesaria para crear una entrada:

Ilustración 15: Pantalla para crear una nueva entrada.

Para agregar una imagen desde la cámara del dispositivo o de la galería se usa un plugin de Cordova llamado 'cordova-plugin' que permite el acceso a estos recursos. Una vez elegida la imagen, da igual de dónde se tome, se recorta dicha imagen, utilizando para ello un plugin llamado 'jrCrop'[19].

19 Este plugin no permitía cambiar el texto de las acciones a otro idioma, por lo que, lo modifiqué y el autor agregó mis cambios al repositorio. https://github.com/JrSchild/jr-crop

Ilustración 16: Pantalla de recorte de una imagen.

Además de la imagen, en la entrada se debe agregar el título y la descripción de la misma. Las etiquetas son opcionales. En este punto, la entrada podrá guardarse, pero no publicarse. Para ello, deberá de agregarse al menos un paso.

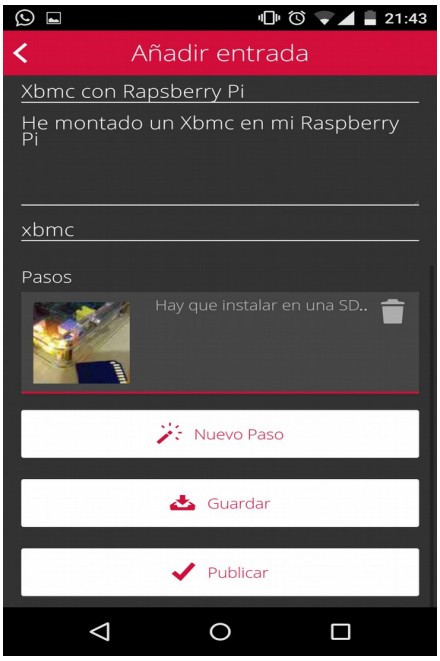


Ilustración 17: Muestra los pasos que se han añadido a una entrada.

Agregación de pasos a las entradas: Para agregar un paso a la entrada, desde su edición, podemos pulsar sobre el botón 'Nuevo paso'. En esta vista debemos de introducir una imagen de la misma manera que se hacía en la entrada, y una descripción del paso.

Ilustración 18: Pantalla en la que se añade un paso nuevo a la entrada.

En este punto podremos agregar un nuevo paso de forma que el que se está editando en este momento se guardará automáticamente, o se podrá volver a la entrada y nos preguntará si queremos guardar previamente el paso que se ha editado. En la entrada, como se vio en el punto anterior, se ve el listado de pasos que se han ido guardando.

- **Detalle de entrada:** Se puede acceder al detalle de entrada desde el muro o desde las entradas que aparecen en los perfiles de usuario. En el detalle de la entrada, se podrá agregar un 'Me gusta', un 'Hecho' o comentarios a la misma. El controlador de esta vista es *entry-detail-controller*.

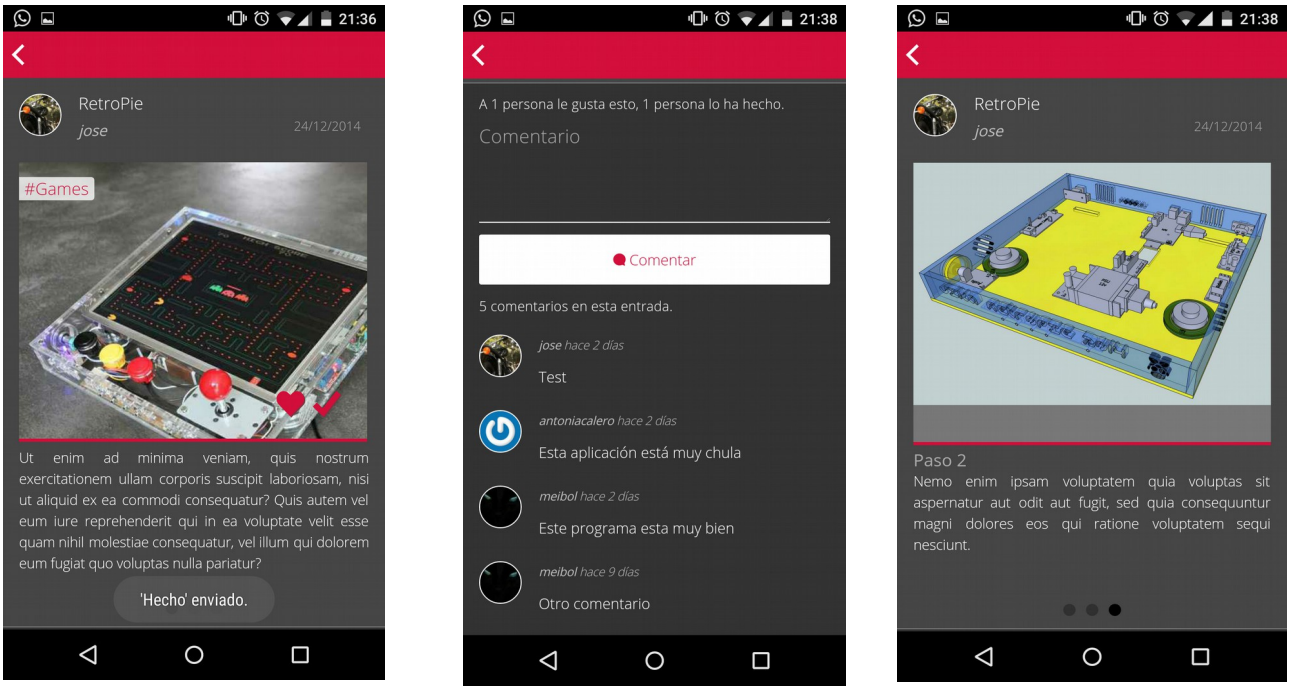


Ilustración 19: Detalle de una entrada publicada.

7.6 Compilación y empaquetado

Junto al código fuente existe un fichero 'README.md' que contiene todas las instrucciones para la instalación de las herramientas, *frameworks* y librerías necesarias para la correcta compilación de la aplicación en iOS y Android:

```
# Instalación de las dependencias

Antes de nada, instalamos las dependencias en el sistema:

    $ sudo apt install npm
    $ sudo npm install -g bower
    $ sudo npm install -g grunt-cli

Si el sistema operativo es Ubuntu, hay que tener en cuenta que el ejecutable de
nodejs se llama 'nodejs', en lugar de 'node' como en otras distribuciones o
sistemas operativos.
Para solventar esto y que las aplicaciones que usen el comando 'node' funcionen
correctamente, debemos hacer un enlace simbólico:

    $ sudo ln -s /usr/bin/nodejs /usr/bin/node

A continuación, podemos instalar las dependencias del desarrollo del proyecto. En
el directorio raíz de la aplicación, ejecutamos:
```

```
    $ npm install
    $ bower install

# Ejecución

## Navegador web

Una vez instaladas las dependencias, ejecutamos la aplicación en el navegador
mediante el comando:

    $ grunt serve

Para detener el servicio, pulsamos Ctrl+C.

## Dispositivos y emuladores

Para poder ejecutar la aplicación en dispositivos Android es necesario tener
instaladas las siguientes herramientas antes de realizar los pasos que se
describen a continuación.

    * Oracle JDK 8.
    * Android SDK.

En el caso de iOS, para poder ejecutarlo debemos trabajar en un sistema Mac OS X,
y tener instalado y actualizado XCode.

A continuación, añadimos las plataformas en las que queremos ejecutar la
aplicación, por ejemplo:

    $ grunt platform:add:android
    $ grunt platform:add:ios

Para ejecutar estos pasos, debemos previamente haber ejecutado la aplicación en el
navegador, ya que necesita crear algunos ficheros para reconocerlo como un
proyecto cordova.

## Crosswalk

Para usar [Crosswalk](https://crosswalk-project.org/) en lugar del WebView por
defecto de Android, es necesario sustituir y compilar librerías de Cordova. Ésto
podemos hacerlo ejecutando el script que se encuentra en la carpeta libs. Antes de
ejecutarlo, debemos asegurarnos de que las dependencias están instaladas (JDK y
Android SDK). El script también asume que en el sistema está instalada una imagen
de la versión 19 de Android.

    $ cd libs
    $ ./install-crosswalk.sh

### Comprobar si estamos usando Crosswalk

Para comprobar que se está usando Crosswalk, una vez ejecutado en Android, podemos
hacer debug con Chrome usando chrome://inspect, y en la consola, imprimiendo esto:

    navigator.userAgent

La salida, debería ser algo como esto:

    "Mozilla/5.0 (Linux; Android 4.2.2; ZTE V975 Build/JDQ39)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.116
    Mobile Crosswalk/7.36.154.12 Mobile Safari/537.36"
```

```
### Android

Para ejecutarlo en un emulador Android:

    $ grunt emulate:android

Para ejecutarlo en un dispositivo Android:

    $ grunt run:android

### iOS

Para ejecutarlo en un emulador iOS:

    $ grunt emulate:ios

Para ejecutarlo en un dispositivo iOS:

    $ grunt run:ios

# Ejecutar tests

Existen dos formas de ejecutar los tests, mediante consola, o en el navegador.

 * Para ejecutarlos en consola:

    $ grunt karma

* Para ejecutarlos en el navegador, basta con abrir el fichero test.html desde el
menú archivo del navegador.
```

Por motivos de disponibilidad, las pruebas se han realizado en dos terminales Android, Nexus 5 (Android 5.0) y OnePlus One (CyanogenMod 11.0), aunque espero poder realizar pruebas en otro tipo de terminales (móviles y tablets), incluyendo un iPhone.

7.7 Plan de pruebas

Para la definición del plan de pruebas realizado hay que tener en cuenta que el desarrollo de esta aplicación no se parece a una página web que se visualiza desde un navegador web, por lo que algunas herramientas que existen para tal cometido no podrán aplicarse. Por ejemplo, desde el dispositivo no podrá utilizarse los validadores que ofrece la W3C. Esto complica un poco la definición de dichas pruebas, pero se buscará la forma de solventarlo.

Las pruebas que están previstas definirse y ejecutarse son:

- **Pruebas de interfaces y contenidos**, en las que se revisará todo el contenido de la aplicación:
    - Que toda la información esté correcta (ortografía, sintaxis, etc.)

- Verificación de que todos los enlaces son correctos y llevan a las vistas o acciones correspondientes.
- Verificación de que todo el contenido multimedia se carga correctamente.

- **Pruebas de funcionalidad**, que determinarán que se obtienen correctamente los resultados que se pretenden conseguir. Entre estas pruebas comprobará:
  - Que se validan correctamente los campos obligatorios, su sintaxis, etc.
  - Se debe mostrar claramente los errores producidos en caso de que ocurran
  - Comprobar que el comportamiento es similar independientemente de la plataforma en la que se ejecute (esto dependerá de la disponibilidad de dichas plataformas)
- **Pruebas de seguridad,** para ello se deberá validar, en este caso, que los mecanismos que proporciona Parse para el control de acceso a la información se ejecutan correctamente.
  - En este caso no se contempla ejecutar pruebas de carga, puesto que esto implicaría hacerlas sobre Parse, y en este momento se está utilizando una versión gratuita que no permite realizar más de 30 peticiones por segundo. Además, este servicio garantiza su disponibilidad, por lo que no tiene demasiado sentido.

Para el registro de estas pruebas, exceptuando las pruebas unitarias, se contará con un documento de validación de las pruebas, en las que aparecerá la siguiente información por cada prueba:

Campo	Descripción
Tipo	Indicará el tipo de prueba que es
Módulo	Parte del sistema a la que afecta la prueba
Acciones	Indica la secuencia de acciones que han llevado a ejecutar esa prueba.
Resultado esperado	Indica el resultado que debe de obtenerse al realizar las acciones indicadas.
Resultado obtenido	Indica el resultado que se ha obtenido al realizar las acciones indicadas.
Correcto	Indica si es correcto o no el resultado
Estado	Si hay error, indica si:

	1. está pendiente de resolver 2. está corrigiéndose 3. se ha resuelto 4. no se ha podido reproducir 5. se requiere mś información
Fecha	Fecha de realización de la prueba
Prioridad	En caso de error, indica el grado de prioridad que tiene su resolución.

Los resultados de las pruebas de interfaces y contenido así como las de funcionalidad, se encuentran en un documento adjunto a esta memoria llamado RPS_pruebas.ods.

- **Pruebas unitarias**, mediante las cuales se asegura que cada componente de código funciona correctamente por separado. Para ello, se ha usado el *framework* Jasmine[20] junto con algunas herramientas como SinonJS[21] para la creación de *stubs* y *mocks*. La ejecución de estas pruebas, se puede realizar de distintas formas:

  - Mediante el navegador web, el cual ejecutará el código de las pruebas unitarias y mostrará los resultados.

  - Mediante consola, usando el *test runner* KarmaJS[22] que ejecuta el código de las pruebas sobre PhantomJS[23] que permite ejecutar código Javascript sobre el motor Webkit, sin necesidad de un navegador web.

  Para ejecutar las pruebas unitarias, con el entorno previamente configurado bastará con ejecutar el comando *grunt karma*. Esto mostrará por consola el resultado de las pruebas, así como un porcentaje de *coverage*. Este valor indica el porcentaje de nuestro código que ha sido probado. En el momento de la entrega de esta memoria, la salida de dicho comando es la siguiente:

20 http://jasmine.github.io/

21 http://sinonjs.org/

22 http://karma-runner.github.io/0.12/index.html

23 http://phantomjs.org/

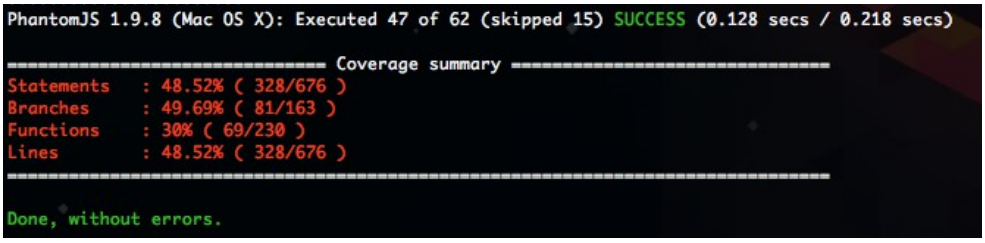


Ilustración 20: Resultado de la ejecución de las pruebas en el momento de la entrega de esta memoria.

Como puede verse, actualmente el porcentaje de *coverage* es muy bajo. Esto es debido a la falta de tiempo para la entrega del proyecto. Para la continuación del proyecto, este porcentaje deberá siempre estar por encima del 80%.

8 Conclusiones

El desarrollo de este proyecto ha sido una tarea gratificante, de la cual me he enfrentado a situaciones en las que hasta ahora no había estado y me ha ayudado a descubrir nuevas formas de desarrollar tecnología. A continuación defino las conclusiones a las que he llegado tanto en los problemas o mejoras que he visto durante el proyecto, como en las posibilidades que tiene este proyecto de cara al futuro.

8.1 Retrospectiva

Durante el desarrollo se han detectado una serie de funcionalidades que se había contemplado desarrollar y otras que no se habían contemplado y que se consideran mejoras deseables para cerrar esta versión del proyecto. También se incluyen tareas que se han detectado como incidencias y que están pendientes de su resolución. A continuación se describen algunas de ellas, ya que es posible, que surja alguna más durante el transcurso de la siguiente fase:

#	Título	Descripción	Tipo
1	Ordenar la lista de pasos desde la vista de entrada	Los pasos ahora mismo se ordenan por orden en el que se crean. Desde la edición de la entrada debe poder modificarse el orden de dichos pasos antes de su publicación.	Contemplado previamente
2	Mostrar quiénes	Esta funcionalidad sería parecida a la que proporciona	Mejora no

	han pulsado 'like' en el detalle de la entrada.	Instagram para sus fotos.	contemplada
3	Depurar el código de las entradas	Para ello, mediante AngluarJS, se quiere revisar todas las redundancias que pueda haber en el código y reducirlas lo máximo posible, para mejorar la legibilidad del mismo.	Mejora de desarrollo
5	Llamadas al método getAllEntries desde el muro	Cuando se carga el muro se realizan dos llamadas al método getAllEntries. En el muro aparece la información en diferentes páginas que se van cargando cuando visualizamos la parte inferior del mismo. En este caso, se trae las dos primeras páginas al iniciarse, en lugar de una.	Incidencia
7	Uso de collection-repeat	Es un mecanismo para mejorar el rendimiento de la lista de entradas en el muro. Internamente reutiliza los elementos que aparecen en la pantalla. Cuando un elemento desaparece por un lateral de la pantalla, lo reutiliza para pintar el que aparecerá por el otro lateral.	Incidencia. Mejora de rendimiento

Como se puede apreciar, se han quedado algunas tareas pendientes de resolver y mejoras que, aplicándose, pueden mejorar el rendimiento o la funcionalidad de la aplicación. Es posible que antes de que termine la fase final del proyecto y la entrega de la memoria surjan nuevas incidencias que haya que resolver.

Por otro lado, la entrega de la aplicación sufrió un retraso no previsto, por cuestiones personales, de dos semanas y como consecuencia se ha visto reducido de tiempo para la entrega final del proyecto. Además de este retraso, durante el desarrollo ha habido algunos problemas no previstos que indico a continuación:

#	Descripción
1	Tenía incorporada al proyecto una librería llamada 'jr-crop' que permitía la manipulación de las imágenes antes de subirlas a Parse. Aunque el autor indica en su página de Github que no está probado en Android, lo probé en un Nexus 5 con Lollipop y funciona correctamente. El problema viene cuando la aplicación se instala en un Android con una versión 4.2 o inferior, o como he podido comprobar, en otras compilaciones oficiales, como puede ser la CyanogenMod del OnePlus One. Finalmente, tras hacer algunos ajustes en el proyecto se ha podido añadir dicha librería con un rendimiento aceptable.
2	El servicio proporcionado por Parse es, en general, un servicio estable. No obstante es un servicio compartido por múltiples clientes, y durante unos días, algunas aplicaciones de otros clientes consiguieron desestabilizar el repositorio. Desde que lo solventaron no ha vuelto a ocurrir, pero esto me obligó a dejar el desarrollo parado durante un par de días.

3	La carga de contenido multimedia difiere en los dos terminales utilizados: en el One Plus con CyanogenMod no aparecía la imagen que se carga en background sin motivo aparente, puesto que al depurar con Chrome (desde 'chrome://inspect/#/devices') se puede comprobar que la imagen estaba accesible. Finalmente tras unos ajustes en el proyecto, se consiguió el funcionamiento esperado.

Por otro lado, escribo algunas mejoras que no se habían contemplado hasta el momento y que, por restricción de tiempo, no podrán incorporarse a la entrega final.

#	Título	Descripción
1	Envío de correo a los usuarios por diferentes eventos	Este envío de correo se realizará para: dar la bienvenida al servicio, saber si alguien nos ha agregado como contacto, si a alguien le gusta nuestra entrada...
2	Recordar contraseña	Es importante poder recuperar los datos de la cuenta si un usuario no se acuerda de su contraseña. Esta es una funcionalidad que debería de haber estado desde el principio pero que por limitaciones de tiempo se sale fuera de la planificación.
3	Visualización en modo *landscape*	La aplicación se puede visualizar solamente en modo *portrait* o vertical. En un futuro se implementará la opción de girar la pantalla para verlo en horizontal.
4	Tutorial de inicio	La primera vez que se utilice la aplicación deberá mostrar un pequeño tutorial interactivo para enseñar al usuario como se utiliza.
5	Autenticación con Facebook o Twitter	Parse permite que los usuarios puedan registrarse en la plataforma mediante su usuario de Facebook o de Twitter. Esta es una funcionalidad cómoda y, cada vez, más extendida.

8.2 Continuación del proyecto

Como se ha indicado a lo largo de este documento, el objetivo del desarrollo de RPS no es sólo su presentación como trabajo de fin de carrera, sino que existe un propósito de continuación del mismo y de publicación en las diferentes tiendas de aplicaciones para dispositivos móviles. La iniciativa llevada a cabo por la Raspberry Pi Foundation ha proporcionado una herramienta muy valiosa para aumentar el conocimiento y la creatividad a muchas personas (profesores, alumnos de escuela o universidad, aficionados e incluso personas con reducidos recursos económicos). Por este motivo, RPS puede mejorar bastante la visibilidad de esta tecnología si consigue llegar a muchos

usuarios y hacerse 'viral'.

Para ello, el propósito es seguir trabajando en RPS para mejorar funcionalidades existentes, o ampliarlas, como pueden ser:

- Mejorar el rendimiento de la aplicación en todos los aspectos, pero principalmente en el uso de procesador, memoria, batería y reducción del espacio requerido.
- Mejorar la navegación para que sea más fluida y se asemeje dentro de lo posible a una aplicación nativa. Durante el desarrollo actual se ha tenido que tomar la decisión de sustituir el renderización HTML que proporciona Cordova (WebView) por otro con mejor rendimiento llamado Crosswalk.
- Mejorar la experiencia de usuario, a través del uso y *feedback* proporcionado por los usuarios en sus primeras etapas de lanzamiento y posteriormente, una mejora continua que se adapte a las nuevas tendencias.
- Incorporar un gestor de correo electrónico en la parte servidor que envíe notificaciones a los usuarios cada vez que se produzca una acción que lo requiera.
- Adaptar la composición de los elementos de la pantalla para la visualización de la aplicación en pantalla en modo horizontal (*landscape*).
- Incorporar acciones para compartir contenido en otras redes sociales (Facebook, Twitter, etc.).
- Enviar notificaciones push cuando se produzca algún evento que lo requiera (por ejemplo, 'a tal persona le ha gustado tu publicación').
- Otra cosa que creo que es importante para incorporar es un tutorial interactivo que se ejecute la primera vez que se utiliza la aplicación.

9 Bibliografía y fuentes consultadas

1. Raspberry Pi Foundation members. (2012) What is a Raspberry Pi? Raspberry Pi Foundation. UK registered charity 1129409. <http://www.raspberrypi.org/help/what-is-a-raspberry-pi/>
2. RaspBerry Pi. <http://www.xataka.com/gadgets/altavoces/raspberry-pi>
3. W3C members. (2012) "Standards". W3C (MIT, ERCIM, Keio, Beihang). Fuente: <http://www.w3.org/standards/>
4. Single-page application, ¿qué son? <http://en.wikipedia.org/wiki/Single-page_application>
5. Ember.js - Ver página oficial: <http://emberjs.com/>
6. Backbone.js - Ver página oficial: <http://backbonejs.com/>
7. Parse - Ver página oficial: <https://www.parse.com/>
8. Cordova - Ver página oficial: <http://cordova.apache.org/>
9. Campderrich, B. (2003, abril). "Ingeniería del Software". Editorial UOC. Fundació per a la Universitat Oberta de Catalunya. ISSN: 84-8318-997-6.
10. Aguilar, J.M. (2007, junio). Catalogo.NET. Sistema multicliente con arquitectura SOA basado en tecnología .NET. TFC - Memoria final. Universitat Oberta de Catalunya.
11. Sass, el manual oficial: http://librosweb.es/libro/sass/
12. Mixins ¿Y eso con qué se come?: http://ivanmendoza.net/diseno-web-2/mixins-y-eso-con-que-se-come
13. Raspberry Pi Community (http://www.raspberrypi.org/forums/)
14. Foro de la comunidad hispana de Raspberry Pi (http://www.raspberrypi-spanish.es)
15. Raspberry Shop en español: http://www.raspberryshop.es/
16. App Raspberry Pi (Play Store de Google): https://play.google.com/store/apps/details?id=com.berkinalex.raspberrypi
17. Pi Projects (Play Store de Google): https://play.google.com/store/apps/details?id=com.rigney.raspberrypiprojects
18. RasPiConnect (iTunes Preview de Apple): https://itunes.apple.com/us/app/raspiconnect/id642069959?mt=8
19. Raspberry Pi Geek (iTunes Preview de Apple): https://itunes.apple.com/us/app/raspberry-pi-geek/id690438156?mt=8
20. App Store > Redes sociales > Apps populares (iTunes Preview de Apple): https://itunes.apple.com/es/genre/ios-redes-sociales/id6005?mt=8
21. Tumblr para iPhone/iPad: https://itunes.apple.com/es/app/tumblr/id305343404?mt=8
22. Tumblr para Android: https://play.google.com/store/apps/details?id=com.tumblr
23. Google + para iPhone/iPad: https://itunes.apple.com/es/app/google+/id447119634?mt=8
24. Google+ para Android: https://play.google.com/store/apps/details?id=com.google.android.apps.plus
25. Just in mind (Software para prototipado): http://www.justinmind.com/
26. Sneakpeekit (Prototipado en papel): http://sneakpeekit.com/
27. Fotografía médico: http://salud.uncomo.com/articulo/como-reconocer-a-un-buen-medico-18317.html
28. Fotografía estudiante: http://4.bp.blogspot.com/-ozNjExUASdI/UjZBh_zmoCI/AAAAAAABIrw/DylO1DsH_Ac/s1600/Dia+del+Estudiante+15.jpg
29. Fotografía del aficionado: http://cdn01.am.infobae.com/adjuntos/163/imagenes/005/860/0005860045.jpg
30. Fotografía del experto: http://img.xataka.com/2013/10/profesor.jpg