UNIVERSIDAD
COMPLUTENSE
MADRID

Facultad de Informática

Departamento de Sistemas Informáticos y Computación

Trabajo de fin de grado del Grado en Ingeniería Informática

Lost2Found: Encuentra tus objetos perdidos usando open data y redes de colaboración

---

Autores:

Carolina Rivero Fernández
David Zamora Rey

Director:

Jesús Correas Fernández

8 de junio de 2018

Agradecimientos

Quiero agradecer a mi familia y a mis amigos por el apoyo mostrado en todo momento a lo largo del desarrollo del proyecto, ya que sin ellos esto no habría sido posible.

También a nuestro director del proyecto el Dr. Jesús Correas Fernández, por la confianza depositada en nosotros para llevar a cabo el proyecto y por sus enseñanzas y dedicación.

David Zamora Rey.

En primer lugar, agradecer a mi compañero David por el maravilloso trabajo, esfuerzo y dedicación que ha mostrado durante este año, por apoyarme y ayudarme en todo momento. Gracias a mi gran amiga Laura, por estar en las buenas y en las malas a lo largo de toda la carrera, animándome y levantándome siempre que lo he necesitado. Sé que me llevo una compañera de vida. Gracias a Pablo, por no cansarse de mí, de mis idas y venidas, por creer en mí, por quererme tal y como soy. Gracias a mis padres, por proporcionarme la educación y la confianza necesarias para afrontar la carrera y, sobre todo, para la vida. Gracias a nuestro director, por hacernos más llevadero este trabajo de fin de grado, por creer en esta propuesta que, al final, ha salido adelante. Gracias a todos los que han estado en estos duros años, tanto amigos y compañeros como profesores. Y, para terminar, gracias a la luz de mi vida, ejemplo de amor, perdón y superación. Mi hermano, César, el que siempre ha estado a mi lado, dándome todo sin esperar nada. Te quiero.

Carolina Rivero Fernández.

Resumen

Actualmente las entidades públicas están haciendo considerables esfuerzos por hacer accesible gran cantidad de información que manejan internamente, esto ofrece oportunidades provechosas en el uso de este tipo de datos comúnmente denominados como *open data* o datos abiertos. Esta información supone una pieza fundamental en la transparencia de una entidad y consiste en datos que, como su nombre indica, tienen una naturaleza pública y abierta, y que son ofrecidos por entidades tanto públicas como privadas, pudiendo ser usados para cualquier fin por cualquier persona que así lo desee. La publicación de datos está en auge actualmente y cada vez son más las instituciones que se suman a este tipo de acciones, ya que generan un beneficio mutuo tanto para las personas que tienen a su disposición más información, como para las entidades que los proporcionan.

Este proyecto tiene como objetivo estudiar la influencia que ejercen este tipo de datos abiertos y comprobar los beneficios que ofrecen en aplicaciones de diversos ámbitos. Como ejemplo de los resultados de la investigación inicial sobre datos abiertos, se ha desarrollado una aplicación llamada *Lost2Found* que crea una red de colaboración entre usuarios para recuperar objetos perdidos, haciendo para ello uso de información de *open data* disponible gracias a dos organizaciones: *SNCF* y *Google*. Nuestra intención con el desarrollo de esta aplicación es demostrar el potencial de este tipo de iniciativas e incentivar el uso de las mismas en aplicaciones futuras.

Abstract

Currently public entities are making considerable efforts to make accessible a large amount of information that they handle internally, this offers profitable opportunities in the use of this type of data commonly denominated as *open data* or open data. This information is a fundamental piece in the transparency of an entity and consists of data that, as its name indicates, have a public and open nature, and are offered by both public and private entities, and can be used for any purpose by any person so wish The publication of data is currently booming and more and more institutions are joining this type of action, as they generate a mutual benefit both for the people who have more information at their disposal, and for the entities that provide them.

The objective of this project is to study the influence of this type of open data and to verify the benefits it offers in applications from various fields. As an example of the results of the initial research on open data, an application called textit Lost2Found has been developed that creates a collaboration network between users to recover lost objects, using information from textit open data available thanks to two organizations: *SNCF* and *Google*. Our intention with the development of this application is to demonstrate the potential of this type of initiatives and encourage the use of them in future applications.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

La colaboración entre iguales es un mecanismo útil e ingenioso para la resolución de problemas de todo tipo.

Siempre han existido multitud de estas iniciativas basadas en el trabajo colaborativo entre usuarios, ya que la inmensa mayoría de los problemas se pueden resolver de una manera más rápida y fácil si para ello se coopera junto a otros individuos, siendo la manera más eficiente de tratar este tipo de dificultades.

Estos mecanismos no son algo novedoso y llevan existiendo desde hace mucho tiempo, usándose en temas no basados en la tecnología como por ejemplo la producción científica o la recopilación de información epidemiológica.

Es por todo esto por lo que consideramos necesario el apoyo a iniciativas que inciten la creación y/o evolución de redes de colaboración entre usuarios, con el fin de promover este tipo de mecanismos de trabajo.

Dentro de este tipo de iniciativas, es interesante observar las posibilidades de acceso a la información a través de lo que hoy en día se conoce como datos abiertos u *open data*, información de carácter público que divulgan las instituciones de manera que ésta sea accesible para cualquier persona u organización interesada.

Estos ejercicios de transparencia de los datos están de moda actualmente, y cada vez más organizaciones realizan este tipo de acciones, difundiendo información y poniéndola al alcance de cualquier usuario que desee consultarla.

Este proyecto surge del intento de demostrar el potencial y las posibilidades existentes tanto de las redes de colaboración como del *open data*, para aumentar la eficiencia en la resolución de problemas de distinta índole.

Debido a esto, en primer lugar se va a realizar una investigación, tanto del estado del arte como sobre los ejemplos prácticos de iniciativas ya existentes, para después desarrollar una aplicación como resultado final del proyecto.

Lost2Found es una aplicación *Android* que trata sobre la pérdida y devolución de objetos perdidos, y tiene como objetivo principal poner en contacto a usuarios que han

perdido o encontrado algún objeto con el fin de devolvérselo a su legítimo dueño, además hace uso tanto de técnicas de redes de colaboración como de *open data*, sirviéndonos de ejemplo empírico del estudio realizado para demostrar el beneficio que se obtiene con el uso de estos conceptos.

La aplicación está disponible en *Google Play Store* y se puede descargar bien buscándola por su nombre, *Lost2Found* o accediendo a la página correspondiente a su ficha en el siguiente enlace `https://play.google.com/store/apps/details?id=es.lost2found`.

Para probar la aplicación simplemente es necesario registrarse con un nuevo usuario y ya se podrán crear anuncios de pérdida o hallazgo de objetos, realizar búsquedas de anuncios en la aplicación, o consultar el *open data* disponible sobre objetos perdidos.

Todas las posibilidades que ofrece esta aplicación se explicarán detalladamente en los próximos capítulos de esta memoria.

1.2. Objetivos y organización del trabajo

El objetivo general del proyecto consistirá en mostrar el potencial y las ventajas que se pueden obtener al hacer uso de una red de colaboración entre usuarios y de datos abiertos para la resolución de problemas.

En este caso implica la investigación sobre estos conceptos y el posterior desarrollo de una aplicación móvil, cuya finalidad se ha explicado brevemente en el apartado anterior.

Otro de los objetivos buscados es demostrar la mayor capacidad y potencial de las aplicaciones que hacen uso de datos abiertos frente a las que no utilizan este tipo de iniciativas.

A la hora de empezar con el desarrollo de la aplicación decidimos hacerlo de manera que fuera lo más sencilla e intuitiva posible, para atraer a gran cantidad de usuarios.

En cuanto al comienzo del desarrollo del proyecto tuvimos una primera reunión con nuestro director, en la que se comentaron los primeros requisitos, criterios y funcionalidades que se iban a llevar a cabo en el proyecto, además de establecer el desarrollo de una aplicación *Android* tras realizar una primera investigación inicial sobre los conceptos explicados anteriormente.

De esta manera se pretendía que el resultado del proyecto no fuera meramente una aplicación, sino que se pusiera en contexto en un ámbito en el cual se observaran las posibilidades que existen al hacer uso de estos conceptos.

Para empezar nuestro trabajo en el proyecto, en primer lugar se hará un minucioso estudio sobre los conceptos de redes de colaboración y open data.

En segundo lugar, se realizará un análisis de otras aplicaciones similares comprobando así qué factores debemos tener en cuenta a la hora de desarrollar la aplicación. De esta manera queremos asegurarnos de cubrir una necesidad que no este ya cubierta por otra aplicación existente.

Tras realizar el análisis en el capítulo 5, "Análisis de aplicaciones similares", se pudo comprobar que ninguna aplicación de objetos perdidos aprovecha información de open data, por lo tanto nuestra aplicación será única en este sentido y nos brindará un punto de originalidad clave a la hora de distinguirnos de otras aplicaciones semejantes.

Para ello se hará una búsqueda de los catálogos de datos abiertos sobre objetos perdidos disponibles, con el objetivo de seleccionar el que más se adecúe a nuestras necesidades y usar la información que nos ofrezca en nuestra aplicación.

Tras el estudio sobre los conceptos y el análisis de la competencia se elaborará un primer prototipo de la aplicación, en el cual se diseñaran las interfaces principales con la herramienta *Just In Mind*, teniendo en cuenta la guía de diseño de *Material Design* de *Google* con intención de que la aplicación sea atractiva para el usuario.

También se plantearán distintas opciones que podrá tener el usuario en la aplicación, como realizar búsquedas de otros anuncios, contactar y comunicarse con otros usuarios, especificar el lugar del anuncio de varias maneras distintas, e integrar open data en nuestra aplicación para potenciarla de la mayor manera posible.

Una vez se termine el diseño de las interfaces de las pantallas importantes y se tengan claras las opciones del usuario en la aplicación, se pasará a realizar los primeros diagramas entidad relación de la base de datos, así como sus correspondientes diagramas relacionales.

Una vez consigamos una versión final de la estructura de la base de datos de la aplicación, se volcará en un fichero *SQL*.

Tras esto se empezará a desarrollar la aplicación utilizando *Android* y ayudándonos de la documentación oficial de su página [1] y del entorno *Android Studio*.

Después se diseñarán las primeras pantallas funcionales de la aplicación en *XML*, tomando como referencia los diseños elaborados previamente en el prototipo de la aplicación, también se comenzarán a implementar algunas de las clases principales en *Java*.

Una vez se hayan conseguido varias pantallas de la aplicación, se trabajará en las conexiones con la base de datos mediante *PHP* usando *JSON* como forma de comunicación, además de implementar la lógica interna de la aplicación y de los ficheros del servidor.

Por último, una vez desarrollado el prototipo de la aplicación se investigará como utilizar la *API* que nos ofrecía el open data escogido anteriormente para realizar peticiones obteniendo los datos necesarios, así como otras *API's* que nos puedan hacer falta en nuestra aplicación.

1.3. Estructura de la memoria

La memoria está dividida en capítulos claramente diferenciados, en los cuales se da detalle del planteamiento y el desarrollo del proceso seguido en la elaboración del proyecto.

- Capítulo 1 - Introducción: Se describe la motivación que nos llevó a la realización del proyecto, los objetivos a alcanzar y la estructura de esta memoria.
- Capítulo 2 - Introduction: Es la traducción al inglés del primer capítulo.
- Capítulo 3 - Redes de colaboración: Se detalla el proceso llevado a cabo en la investigación de los conceptos fundamentales que se han usado en el proyecto, así como de los ejemplos prácticos de iniciativas ya existentes, además de la información resultante de la investigación realizada.
- Capítulo 4 - Open Data: En este capítulo se explica en profundidad el estudio realizado sobre este concepto en la investigación previa al desarrollo del prototipo *Lost2Found*.
- Capítulo 5 - Análisis de aplicaciones similares: Consiste en un análisis de las aplicaciones que se encuentran actualmente en el mercado y que tienen una funcionalidad similar a la nuestra.
- Capítulo 6 - Open Data *SNCF* y *Google APIs*: En él se explica detalladamente la investigación de las distintas *APIs* que se han usado en su implicación en el proyecto.
- Capítulo 7 - Primer prototipo: Se centra en la realización de los diseños de las primeras interfaces de la aplicación, la definición de los tipos de lugares y anuncios y el diseño de la estructura de la base de datos.
- Capítulo 8 - Diseño e implementación de *Lost2Found*: Trata sobre el trabajo realizado durante el diseño final y la implementación de la aplicación.
- Capítulo 9 - Conclusiones y trabajo futuro: Se exponen las principales conclusiones del proyecto así como el trabajo futuro posible.
- Capítulo 10 - Conclusions and future work: Es la traducción al inglés del noveno capítulo.
- Capítulo 11 - Aportaciones individuales: Se explican las contribuciones personales de cada integrante al proyecto.

Capítulo 2

Introduction

2.1. Motivation

Peer collaboration is a useful and ingenious mechanism for solving problems of all kinds.

There have always been many of these initiatives based on collaborative work among users, since the vast majority of problems can be resolved in a faster and easier way if this is done together with other individuals, being the most efficient way to deal with them. this kind of difficulties.

These mechanisms are not new and have been around for a long time, being used in topics not based on technology such as scientific production or the collection of epidemiological information.

It is for all this that we consider it necessary to support initiatives that encourage the creation and / or evolution of collaboration networks among users, in order to promote this type of work mechanisms.

Within this type of initiatives, it is interesting to observe the possibilities of access to information through what is now known as open data u *open data*, public information that institutions disclose so that this be accessible to any interested person or organization.

These data transparency exercises are currently in fashion, and more and more organizations are carrying out this type of action, disseminating information and making it available to any user who wishes to consult it.

This project arises from the attempt to demonstrate the potential and the existing possibilities of both collaboration networks and *open data*, to increase efficiency in solving problems of different kinds.

Due to this, in the first place an investigation is going to be carried out, both of the state of the art and on the practical examples of already existing initiatives, to later develop an application as the final result of the project.

Lost2Found is an application *Android* that deals with the loss and return of lost objects, and its main objective is to put in contact users who have lost or found an object in order to return it to its rightful owner In addition, it makes use of collaboration network techniques as well as *open data*, using an empirical example of the study carried out to demonstrate

the benefit obtained with the use of these concepts.

The application is available in the *Google Play Store* and can be downloaded by searching for it by its name, textit Lost2Found or by accessing the page corresponding to its file in the following link `https://play.google.com/store/apps/details?id=en.lost2found`.

To test the application you simply need to register with a new user and you can create ads for loss or finding of objects, search for ads in the application, or consult the *open data* available about lost objects.

All the possibilities offered by this application will be explained in detail in the next chapters of this report.

2.2. Objectives and work organization

The general objective of the project will be to show the potential and the advantages that can be obtained by making use of a collaboration network between users and open data to solve problems.

In this case, it involves research on these concepts and the subsequent development of a mobile application, whose purpose has been briefly explained in the previous section.

Another of the objectives sought is to demonstrate the greater capacity and potential of applications that make use of open data compared to those that do not use this type of initiative.

When we started with the development of the application, we decided to do it in a way that was as simple and intuitive as possible, to attract a large number of users.

Regarding the beginning of the development of the project, we had a first meeting with our director, in which the first requirements, criteria and functionalities that were going to be carried out in the project were discussed, as well as establishing the development of an application *Android* after making an initial initial investigation about the concepts explained above.

In this way it was intended that the result of the project was not merely an application, but put into context in an environment in which the possibilities that exist when making use of these concepts were observed.

To start our work on the project, we will first make a thorough study about the concepts of collaboration networks and open data.

Secondly, a study of other similar applications will be carried out, thus verifying what factors we must take into account when developing the application. In this way we want

to make sure we cover a need that is not already covered by another existing application.

After carrying out the study of other similar tools in the chapter 5, "Análisis de aplicaciones similares", it was possible to verify that no application of lost objects takes advantage of open data, therefore our application will be unique in this sense and will give us a point of key originality when it comes to distinguishing us from other similar applications.

To do so, a search of the open data catalogs on lost objects available will be made, with the aim of selecting the one that best suits our needs and using the information that we offer in our application.

After the study on the concepts and the analysis of the competition a first prototype of the application will be elaborated, in which the main interfaces will be designed with the tool *Just In Mind*, taking into account the design guide of *Material Design* de *Google* with the intention of making the application attractive to the user.

It will also consider different options that the user may have in the application, such as searching for other ads, contacting and communicating with other users, specifying the place of the advertisement in several different ways, and integrating open data in our application to enhance it from the greater possible way

Once the design of the interfaces of the important screens is finished and the user's options are clear in the application, the first diagrams of the database will be made, as well as their corresponding relational diagrams.

Once we get a final version of the database structure of the application, it will be dumped into a file *SQL*.

After this, the application will be developed using *Android* and using the official documentation of the Android [1] page and the *Android Studio* environment.

After the first functional screens of the application will be designed in *XML*, taking as reference the designs previously elaborated in the prototype of the application, some of the main classes in *Java* will also be implemented.

Once several screens of the application have been achieved, the connections to the database will be worked through *PHP* using *JSON* as a form of communication, in addition to implementing the internal logic of the application and the server files.

Finally, once the prototype of the application has been developed, it will be investigated how to use the *API* offered by the open data chosen previously to make requests obtaining the necessary data, as well as other *API's* that we may need in our application.

2.3. Structure of the document

The report is divided into clearly differentiated chapters, which detail the approach and the development of the process followed in the development of the project.

- Chapter 1 - Introduction: The motivation that led us to the realization of the project, the objectives to be achieved and the structure of this memory are described.
- Chapter 2 - Introduction: It is the English translation of the first chapter.
- Chapter 3 - Collaboration networks: It details the process carried out in the investigation of the fundamental concepts that have been used in the project, as well as the practical examples of existing initiatives, as well as the information resulting from the research carried out.
- Chapter 4 - Open Data: This chapter explains in depth the study carried out on this concept in the research prior to the development of the prototype *Lost2Found*.
- Chapter 5 - Analysis of similar applications: It consists of a study of the applications that are currently in the market and that have a similar functionality to ours.
- Chapter 6 - *SNCF* and *Google APIs*: It explains in detail the research of the different *APIs* that have been used in their involvement in the project.
- Chapter 7 - First prototype: Focuses on the realization of the designs of the first interfaces of the application, the definition of the types of places and announcements and the design of the structure of the database.
- Chapter 8 - Design and implementation of *Lost2Found*: It deals with the work done during the final design and implementation of the application.
- Chapter 9 - Conclusions and future work: The main conclusions of the project are presented as well as the possible future work.
- Chapter 10 - Conclusions and future work: It is the English translation of the ninth chapter.
- Chapter 11 - Individual contributions: The personal contributions of each member to the project are explained.

Capítulo 3

Redes de colaboración

En este capítulo se explicará, tras haber sido investigado para la realización de este proyecto y de este capítulo en concreto, en qué consiste el concepto de redes de colaboración y cuáles son sus tipos y factores relevantes, así como ejemplos prácticos.

Según la RAE (Real Academia Española) una red es un "*conjunto de elementos organizados para un determinado fin*", esta definición se ajusta perfectamente al concepto de red de colaboración, ya que consiste en una o más asociaciones de interesados cuyo objetivo es lograr resultados acordados conjuntamente mediante la participación y la colaboración entre ellos.

En la gran mayoría de los casos, las colaboraciones de los integrantes de este tipo de redes están principalmente basadas en el altruismo, es decir, no esperan recibir recompensa alguna por su contribución y se tratan de entidades sin ánimo de lucro.

La idea de este tipo de redes es funcionar como si fueran una unidad, integrada por multitud de individuos pero funcionando al unísono para maximizar la productividad. Estos mecanismos de trabajo compartido producen muy buenos resultados a gran velocidad y la aplicación de los mismos a toda clase de industrias ha ocasionado un cambio en sus paradigmas, produciendo mejoras muy notables en temas de productividad en los últimos años.

En el caso concreto de nuestra aplicación las redes de colaboración nos sirven para poner en contacto a personas, pudiendo de esta manera colaborar y ayudarse entre sí con el fin de encontrar los objetos que se hayan podido perder y devolvérselos a su dueño original.

3.1. Factores relevantes de una red de colaboración

Dentro de este tipo de redes cabe destacar una serie de factores importantes a tener en cuenta, como por ejemplo, las dificultades que pueden surgir producidas por el desigual compromiso de sus integrantes, lo cual puede influir en el interés de los participantes y ocasionar problemas para las posibilidades que ofrece un espacio cooperativo que se basa en la voluntariedad y el beneficio mutuo.

También existen factores que favorecen el éxito o progreso de este tipo de redes, como por ejemplo:

- Tener un objetivo definido y concreto.
- Seleccionar con criterio los participantes de la red.
- La eficiencia y eficacia en la coordinación y la gestión de la red.
- La actitud participativa.
- El cumplimiento de los compromisos.
- La existencia de un esquema claro y aceptado por los integrantes.
- El sentimiento de compartir el trabajo y los beneficios de la red.

3.2. Ejemplos de redes de colaboración

Todas las sociedades han utilizado redes de colaboración a lo largo de su historia. Ejemplos tradicionales de ello son: la información epidemiológica, que nos permite conocer información entre los sistemas sanitarios de diferentes países, evitando así la propagación de distintas enfermedades epidémicas o; la producción científica, según la cual los investigadores de distintos países e instituciones han colaborado y compartido conocimientos entre ellos, con el único objetivo de potenciar el desarrollo en sus investigaciones científicas.

La tecnología acual y en particular el caso de internet han potenciado las redes de colaboración entre personas. Ejemplo ilustrativo de estas redes es *Wikipedia* [40], que consiste en una enciclopedia libre editada de manera colaborativa en la que cualquier usuario que lo desee puede modificar artículos y aportar información extra sobre cualquier tema.

Fue creada en 2001 y, desde entonces, ha crecido de una manera inmensurable. Actualmente contiene más de 1.360.000 artículos creados por usuarios de todo el mundo sobre todo tipo de temas. Es un claro ejemplo del potencial que tienen este tipo de iniciativas.

Otro ejemplo claro es *Stack Overflow* [37], que consiste en una comunidad de desarrolladores en la cual se pueden encontrar soluciones a problemas de programación en diferentes lenguajes.

Fue lanzada en agosto de 2008 y resulta de gran ayuda para cualquier duda o consulta que se tenga sobre prácticamente cualquier lenguaje y, actualmente, cuenta con casi 9 millones de usuarios registrados.

Por último, en nuestro caso concreto la red de colaboración creada en nuestro proyecto tiene como fin devolver los objetos perdidos a los dueños legítimos de los mismos, de manera altruista y sin obtener ningún tipo de beneficio a cambio.

Capítulo 4

Open Data

Además de lo visto en el capítulo 3, "Redes de colaboración", en este capítulo se detalla en profundidad otro concepto del que hacemos uso en nuestro prototipo *Lost2Found* que se conoce como *open data*, o datos abiertos.

También se explicará cuáles son los objetivos y principios de esta iniciativa, en qué formatos se suele publicar la información, de qué manera se debe hacer y algunos ejemplos actuales de aplicaciones que hagan uso de esta información.

4.1. Introducción

Los datos abiertos u *open data* son una iniciativa global que consiste en que la información esté disponible de forma libre para todo el mundo, sin ningún tipo de restricción de acceso, copyright u otros mecanismos de control. De esta manera pueden ser modificados o integrados en otros conjuntos de datos.

Este tipo de información se sustenta en diversos fundamentos como la transparencia, la colaboración y la participación, aportando beneficios tanto para las entidades que hacen públicos estos datos generando confianza en las instituciones, como para las personas que tienen acceso libre a esta información. De esta manera se ayuda al desarrollo económico y a la creación de nuevos sectores y servicios para los ciudadanos. También es importante destacar que, debido a su naturaleza pública, estos datos nunca contendrán información personal sobre individuos específicos.

En los últimos años, cada vez son más las instituciones que se suman a este tipo de iniciativas, ligadas a las políticas de gobierno abierto, facilitando de esta manera sus datos a cualquier usuario. La administración pública pionera fue data.gov [6] en EE.UU y, poco a poco, estas iniciativas se han ido extendiendo por todo el mundo.

Pese al incremento internacional de este tipo de iniciativas, las organizaciones que promueven la adopción de datos abiertos a nivel internacional nos advierten de que la mayoría de los datos están desactualizados, incompletos o son de baja calidad, debido a que se publican de forma compleja y dispersa, lo que da lugar a una difícil comprensión de los mismos.

4.2. Objetivos y principios

A nivel nacional existe una iniciativa llamada RISP (Reutilización de la Información del Sector Público) [34], cuyo objetivo consiste en hacer disponible los datos del sector público en formatos estándar abiertos, con el fin de facilitar tanto el acceso como la reutilización de esta información, sin importar si sus fines son comerciales o no.

Es importante que los datos cumplan con una serie de principios para su accesibilidad y usabilidad. Para ello se establecen una serie de cualidades, tales como:

- Disponibilidad y acceso: deben estar disponibles en una forma modificable, además de tener un acceso sencillo.
- Reutilización y redistribución: se deben poder reutilizar y redistribuir, por ejemplo para integrarlos junto a otros conjuntos de datos.
- Participación universal: cualquier persona que así lo desee debe poder usar, modificar y/o compartir estos datos.
- Interoperabilidad: deben existir sistemas capaces de interoperar juntos permitiendo que distintos componentes colaboren.

4.3. Formatos

La información se publica en diferentes formatos estructurados para que pueda ser utilizada de forma automática por los lenguajes de programación. Los formatos utilizados son los siguientes:

- CSV. Ficheros con valores separados por coma, son un tipo de documento en formato abierto sencillo para representar datos en formato de tabla.
- DWG. Formato de archivo informático utilizado principalmente para almacenar la información de dibujo en tres dimensiones de forma vectorial.
- XLS. Formato de Microsoft que muestra la información en celdas organizadas en filas y columnas. Cada celda contiene datos o fórmulas, con referencias relativas o absolutas a otras celdas.
- GTFS. Permiten que las empresas de transporte público publiquen sus datos de transporte y que los programadores escriban aplicaciones que consuman esos datos de manera sencilla, definiendo un formato común para los horarios de transporte público y la información geográfica asociada a ellos.

- JSON. Es un formato ligero para el intercambio de datos basado en la notación literal de objetos de JavaScript. Facilita el tratamiento en los navegadores debido a su reducido tamaño de flujo de datos entre cliente y servidor, en el caso concreto de nuestra aplicación *Android* hacemos uso de este formato debido a su tratamiento sencillo junto con código *Java*.

- KML. Utilizado para compartir lugares e información entre aplicaciones, es un formato de archivo para la creación de modelos y el almacenamiento de funciones geográficas como puntos, líneas, imágenes, polígonos, etc.

- XML. Es un metalenguaje de etiquetas que permite definir lenguajes para diferentes necesidades, es el estándar para el intercambio de información estructurada entre diferentes plataformas.

- RSS. Es un formato XML que facilita la publicación de información actualizada a los usuarios suscritos a la fuente RSS sin necesidad de usar un navegador, se usa para la distribución de contenidos de páginas web.

- SHP. Shapefile es un formato propietario estándar de datos espaciales que almacena tanto la geometría como la información alfanumérica.

- PDF. Es un formato multiplataforma cuyo principal inconveniente consiste en la complejidad a la hora de procesarlos automáticamente, no se recomienda publicar los datos abiertos en este formato.

  La figura 4.1 se ha extraído de la página de datos abiertos del gobierno de España [12]

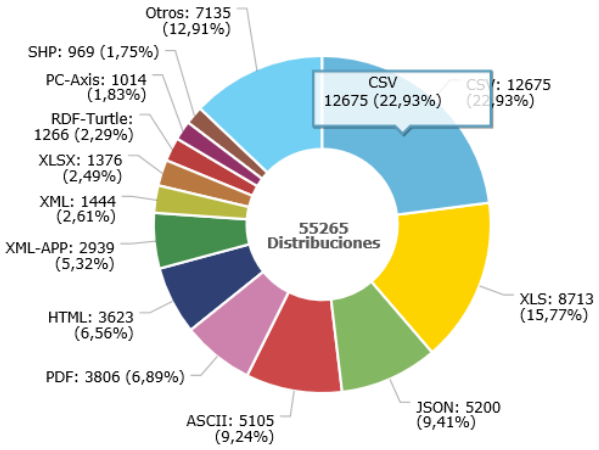


Figura 4.1: Formatos de Open Data

4.4. Formas de publicación de *open data*

Existen varias pautas a seguir a la hora de difundir *open data*. Lo primero es disponer de un conjunto de datos abiertos haciendo uso de una licencia abierta para definir los derechos de propiedad intelectual y soportar el carácter público del *open data*.

Tras esto, se hace disponible la información en su conjunto y en un formato que sea útil para los usuarios finales que vayan a utilizar esos datos. Se pueden ver los distintos formatos existentes en la sección 4.3, "Formatos".

Es importante hacer visible el conjunto de datos creado con el fin de que los usuarios puedan extraer la información que necesiten y trabajar con ella para crear aplicaciones.

Uno de los factores principales a la hora de publicar *open data* es la frecuencia con la cual se actualizan los datos, que necesariamente debe ser alta, ya que de esta manera se saca un mayor provecho del uso de datos abiertos y no se reduce a la mera obtención de información anticuada. También es importante destacar la integridad de los datos que, de otra manera, no serían útiles para el sistema.

4.5. *Open data* en el gobierno local: *Smart Cities*

Un ejemplo característico del uso de *open data* se encuentra en las administraciones municipales por su cercanía al ciudadano. De estas entidades y de la necesidad de hacer las ciudades más sostenibles, a la par que eficientes, surgen las *smart cities*, dando una respuesta a las necesidades de las empresas e instituciones en el plano económico, social y ambiental, otorgando así las estrategias adecuadas para la sostenibilidad.

Se usan las comunicaciones y las tecnologías para crear ciudades inteligentes, innovando en infraestructuras que aporten servicios y recursos para renovar lo existente, mejorar la calidad de vida, disminuir costes y ahorrar energía.

Para llegar a ser una *smart city* se debe de tener una visión holística de toda la ciudad y de sus necesidades, ayudando en diversos ámbitos como el medio ambiente, la sanidad, educación, cultura y economía, proporcionando herramientas para la interacción del usuario y del gobierno con la misma, promoviendo así su uso.

Los datos abiertos son una herramienta para la gestión adecuada de las *smart cities* y sus recursos, de manera que, compaginando ambos, se facilita que la ciudad inteligente sea una plataforma colaborativa donde todos aquellos que lo deseen puedan participar en su desarrollo.

Un ejemplo de la combinación *open data* y *smart city* es *Realtime Rome* [33], un experimento realizado por el MIT en la ciudad de Roma (Italia) en el cual se unen datos proporcionados por los habitantes y datos de las administraciones públicas y privadas.

En este caso, las administraciones solicitan a los ciudadanos poder localizar su posición mediante su teléfono móvil. De esta manera, y con esa información, se facilitan los datos sobre los desplazamientos de las personas mediante autobuses y taxis, con el fin de considerar cómo intervienen los eventos turísticos con la vida cotidiana de sus residentes.

Otro ejemplo que hace uso de este tipo de iniciativas es *Smartappcity* [35], una aplicación ganadora de un concurso a nivel mundial que recoge distintos servicios de una ciudad gracias a los conjuntos de datos proporcionados por *open data*, aportando, de esta manera, inmediatez y valor al ciudadano, como por ejemplo información turística, cultural, de movilidad, etc. Las empresas públicas y privadas también han cedido sus datos para colaborar en este proyecto, con miras a un crecimiento de popularidad y una mejora de su imagen. Se ayudan de una base de datos estructural, donde todos los dueños de comercios y negocios pueden publicar sus productos y ofertas mediante una plataforma web que autogestiona su contenido. Cada ciudad que quiera implantar sus servicios a la aplicación puede hacerlo. Para ello se necesitan partners locales, que son socios que tengan conocimiento del sector y capaces de aportar ideas y ayuda a su instauración. Existen hoy en día diversos países que hacen uso actualmente de *Smartappcity* como, por ejemplo, Chile, Costa Rica, España, etc.

Aunque Madrid no se pueda considerar como una *smart city*, otro ejemplo cercano de *open data* en nuestro país es la sección que tienen en la página web del ayuntamiento de Madrid [29], en la cual se publica un catálogo con un listado de todos los conjuntos de datos que el ayuntamiento pone a disposición de los ciudadanos. Esta información se puede descargar en distintos formatos. Además, en la página se puede filtrar los datos por sector o por formato y se pueden hacer propuestas para solicitar conjuntos de datos.

Figura 4.2: Open data ayuntamiento de Madrid: Logo

4.6. Aplicaciones con datos abiertos

En esta sección se detallan ejemplos prácticos de aplicaciones que hacen uso de *open data*, como son:

- *Moovit* [26]: una aplicación que, a través del acceso a un portal de datos abiertos, ofrece al usuario información acerca de líneas de metro, autobuses, cercanías, interurbanos, metro ligero, bicicletas públicas y, no sólo en España, sino en más de 1500 ciudades de 78 países. Para hacernos una idea de la magnitud y exactitud de la información a la que puede acceder, *Moovit* incorpora una nueva ciudad cada 15 horas. Algo muy destacable es que realiza continuas actualizaciones debido a la imprevisibilidad del transporte público.

- *Patea la Palma* [30]: otra aplicación que hace uso de los datos abiertos para proporcionar información de senderos y rutas de la isla de La Palma, teniendo siempre en cuenta los gustos del usuario para brindarle los datos que más se adecuen a sus necesidades.

- *Inmuebles de alquiler Aragón* [18]: en ella se pueden consultar precios de alquileres de viviendas y locales de Aragón. Esta información la han obtenido del portal de datos abiertos de Aragón.

- *Pharmaraba* [31]: es una aplicación donde se pueden encontrar todas las farmacias de guardia del País Vasco junto con sus horarios. Ha extraído sus datos de *open data* Euskadi.

(a) Moovit (b) Patea la Palma (c) Inmuebles Aragón (d) Pharmaraba

Figura 4.3: Aplicaciones con datos abiertos

4.7. Uso de datos abiertos en nuestra aplicación

Para la aplicación *Lost2Found*, que es el objetivo de este trabajo de fin de grado, se han evaluado las distintas posibilidades de *open data* de ese tema. Desafortunadamente, al no tener el ayuntamiento de Madrid ningún conjunto de datos relativo a objetos perdidos en su portal, lo que sería interesante para la elaboración del prototipo *Lost2Found*, hemos tenido que buscar otras alternativas:

- Pamplona. Existe una publicación de objetos perdidos en San Fermin del año 2013, tan solo contiene 168 objetos y está en una hoja excel *XLS*.
- Gijón. Hay un conjunto de datos relacionado con objetos perdidos creado en enero de 2017. Su última actualización ha sido en septiembre, está en una hoja *CSV* y apenas contiene entradas.
- Nueva York. Son objetos perdidos en los taxis de Nueva York. Se creó en enero de 2014 y se actualiza frecuentemente.
- Londres. La oficina de objetos perdidos divulga un documento en formato PDF en el que hay una lista de los objetos extraviados. Se creó en 2014 y su última actualización ha sido en 2017.
- Francia. Ofrece un catálogo con varios conjuntos de datos de objetos perdidos y encontrados en los trenes y en las estaciones de toda Francia. Se creó en mayo de 2015 y la frecuencia de actualización es de tres veces al día.

Al terminar la búsqueda, nos decantamos por el portal de datos abiertos de Francia, dado que está en constante actualización, algo muy interesante y de gran valor para nosotros y para los usuarios, ya que la aplicación estaría también actualizada. Además, cuenta con dos conjuntos de datos abiertos, uno para objetos perdidos y otro para objetos encontrados, con un número más que considerable de elementos (745000 y 450000 respectivamente).

Capítulo 5

Análisis de aplicaciones similares

5.1. Introducción

En este capítulo se explica detalladamente el análisis que hemos realizado, teniendo en cuenta las herramientas o aplicaciones que actualmente se encuentran en el mercado y qué funcionalidades ofrecen.

Con esto pretendemos observar qué servicios suelen brindar estas aplicaciones para dar un enfoque distinto a nuestra aplicación, con el fin de hacerla destacar frente a las demás y, de esta manera, no desarrollar algo que ya esté disponible.

De la misma manera, este análisis nos ayuda a descubrir alguna característica interesante que no hubiésemos pensado antes y realizar, así, una aplicación completa y funcional.

Tras buscar aplicaciones que traten temas de objetos perdidos en la plataforma de Google Play Store [17], hemos hecho una selección de las cinco que más nos han llamado la atención por sus características interesantes y que vamos a comparar para realizar este análisis.

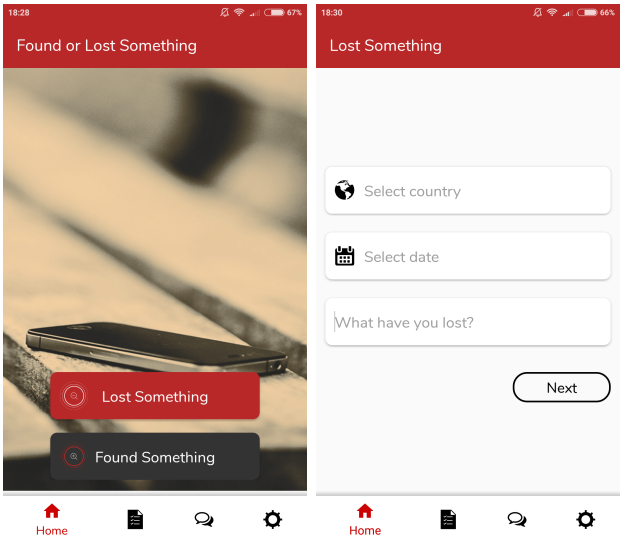
5.2. Estudio de otras herramientas similares

5.2.1. *MissingX*

MissingX [25] es una aplicación para anunciar si se ha encontrado o perdido algún objeto. Su funcionamiento es muy simple y consta de un menú inferior con 4 secciones, en cada una de las cuales se integra una funcionalidad distinta.

5.2.2. Interfaz

Como vemos en la figura 5.1, *MissingX* muestra nada más iniciarse dos botones con los textos *Lost Something* y *Found Something* en el apartado principal (a). Al pinchar en cualquiera de ellos nos redirige a un formulario en el cual se nos preguntan datos como nuestro país o la fecha, además de qué objeto hemos perdido o encontrado (b).

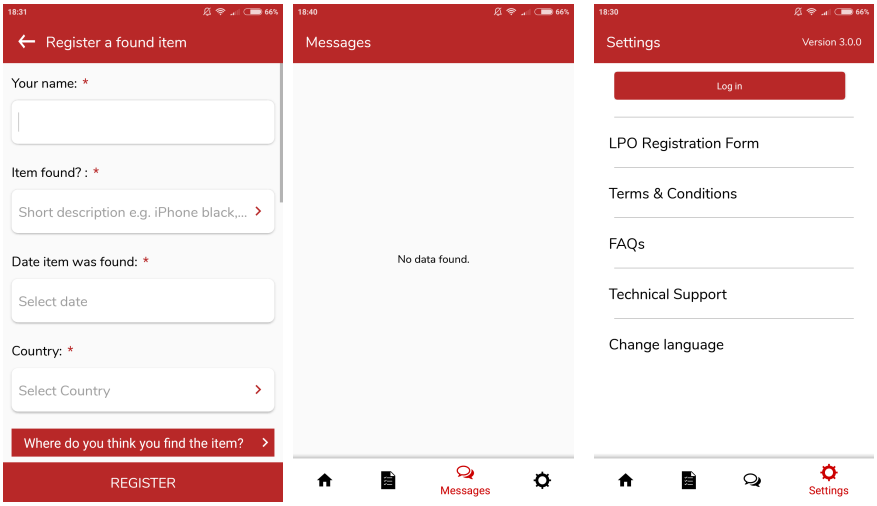


(a) Apartado principal (b) Formulario objetos

Figura 5.1: *MissingX*: Sección principal

El tiempo de carga de la aplicación es lento en general y no se aprovecha toda la pantalla en muchas secciones.

Además de esta sección principal, la aplicación cuenta con otra para registrar objetos encontrados como vemos en la figura 5.2(a), otra para chatear con usuarios (b) y otra de configuración (c).



(a) Registrar objetos encontrados (b) Chat (c) Configuración

Figura 5.2: *MissingX*: Resto secciones

5.2.3. Qué nos interesa

Tras analizar esta aplicación nos han gustado varios aspectos como los dos botones de la sección principal. En el diseño preliminar de nuestra aplicación ya teníamos pensado algo muy parecido, dado que aporta rapidez a la hora de la interacción con el usuario, que es fundamental en este tipo de servicios, sobre todo si el usuario quiere publicar el anuncio de la pérdida de su objeto lo antes posible.

Por otro lado, nos ha parecido muy interesante dar la opción al usuario de indicar en un mapa en qué punto se ha perdido o encontrado el objeto, ya que es una manera rápida e intuitiva de indicar el lugar donde se ha dado la situación.

Por lo tanto, y teniendo en cuenta estos aspectos, a la hora de desarrollar nuestra aplicación y especificar el lugar donde se ha perdido o encontrado un objeto haremos una distinción de hasta tres maneras distintas de hacerlo. Una cuando se trate de una dirección pública con calle y número, otra que se especifiquen en un mapa las coordenadas GPS y, por último, otra si el lugar es en un medio de transporte como puede ser metro, autobús, tren, etc, donde forzamos al usuario a que nos especifique claramente la línea y la estación, ya que señalarlo en un mapa no serviría de mucho.

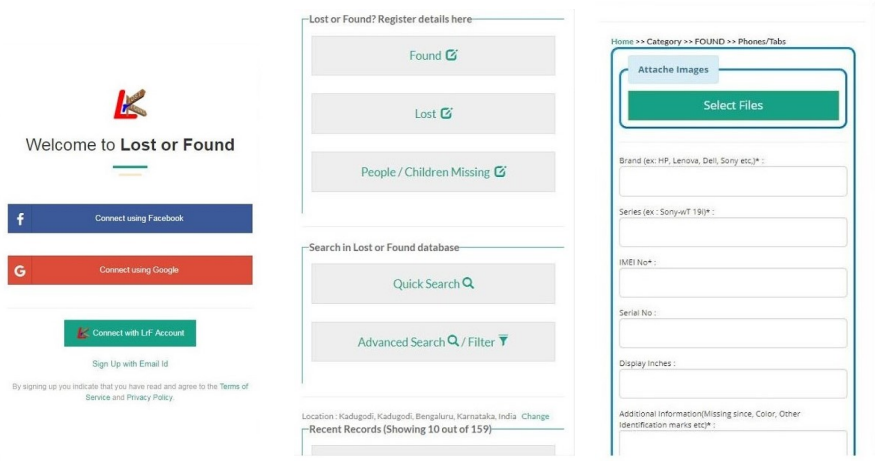
5.2.4. *Lost or Found*

Lost or Found [21] es una aplicación en la que hay información tanto de objetos perdidos y encontrados como de personas que han desaparecido. También existe una sección donde se puede ver la cantidad de anuncios con sus respectivos objetos.

5.2.5. Interfaz

Como vemos en la figura 5.3, el registro en la aplicación es el mismo para todos, no distingue entre las personas que encuentran algo y las que lo pierden (a), por lo que es necesaria la siguiente pantalla (b) donde el usuario debe indicar qué tipo de anuncio quiere publicar, si uno de pérdida u otro donde indique lo que ha encontrado.

En cualquier caso, es ineludible la sección 5.3(c) donde se piden los datos del objeto en cuestión, ya sea en el caso de haberlo perdido como en el de haberlo encontrado. Antes de este formulario, hay un filtro de categorías, donde el usuario indica la correspondiente con su objeto encontrado o perdido.

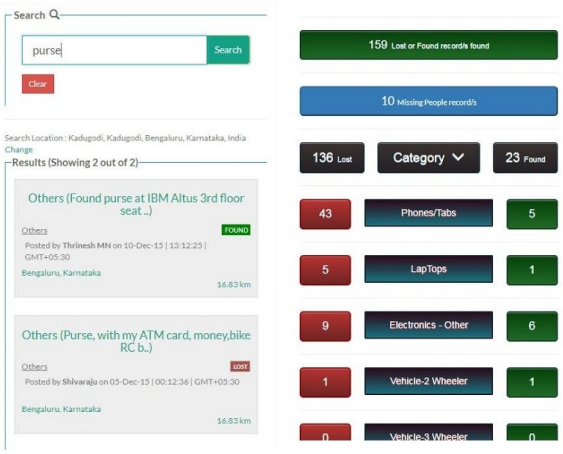


(a) Registro en la aplicación (b) Pantalla de selección (c) Formulario objetos

Figura 5.3: *Lost or Found*: Sección principal

La aplicación cuenta también con un apartado de búsqueda donde puedes poner la categoría del objeto que estás buscando, como se muestra en la figura 5.4(a). Los resultados pueden ser de objetos todavía sin hallar o de los ya encontrados anteriormente.

Hay otra sección interesante donde se indican las diferentes categorías junto con el número de objetos perdidos y encontrados de la misma 5.4(b). Además, hay un contador que señala la cantidad de objetos perdidos que se han encontrado gracias a la aplicación.



(a) Búsqueda de objetos (b) Listado de categorías

Figura 5.4: Lost or Found: Resto de secciones

5.2.6. Qué nos interesa

Al analizar esta aplicación hemos identificado varios aspectos relevantes: el primero es que distingue dos tipos de registro, uno para las personas que pierden objetos y otro para las personas que los encuentran. En nuestra aplicación haremos esto para simplificar el registro de objetos encontrados, con el fin de poner el menor número de obstáculos posibles a la persona que encuentra algo y tiene la intención de devolverlo, incentivando así el uso de la aplicación por ambas partes. El segundo aspecto relevante es que no haremos visible para todos los usuarios los objetos perdidos, ya que esto puede llevar a que haya personas tentadas de entrar en la aplicación única y exclusivamente para ver los lugares donde se han perdido objetos y pretender quedárselos o venderlos. Por último, otra característica interesante de esta aplicación es una sección con un listado de categorías tanto de los objetos que más se han perdido como de los que más se han encontrado y, además, un registro de los objetos encontrados gracias a la aplicación.

5.2.7. Find My Lost

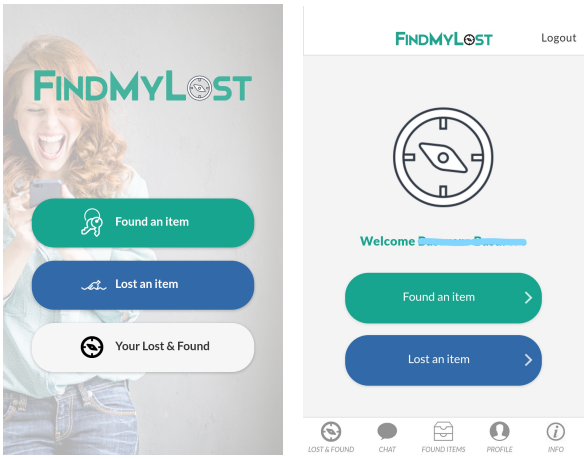
Find My Lost [21] es una aplicación italiana que ayuda a las personas que han perdido un objeto en cualquier sitio, ya sea en la calle, en un centro comercial, en cualquier transporte público, hoteles, etc, además de recoger anuncios de personas que han encontrado algo. A estas últimas se les da la opción de elegir si quieren o no recompensa, dar el objeto en persona o enviarlo mediante una empresa de transporte.

5.2.8. Interfaz

Aunque en la pantalla principal distinga a los usuarios, el formulario de registro sigue siendo el mismo para ambos, por lo que, cuando entras en la aplicación, hay otra pantalla preguntando de nuevo al usuario si ha perdido o encontrado un objeto.

En la pantalla de inicio, que se muestra en la figura 5.5(b), el icono de "*Found items*" de la parte inferior resulta muy útil, siempre y cuando se pueda filtrar de alguna manera para que al usuario no le salgan listas interminables de anuncios.

Por otra parte, el icono de chat también es adecuado tenerlo en la pantalla principal para un acceso rápido a posibles conversaciones con otros usuarios que hayan perdido un objeto.

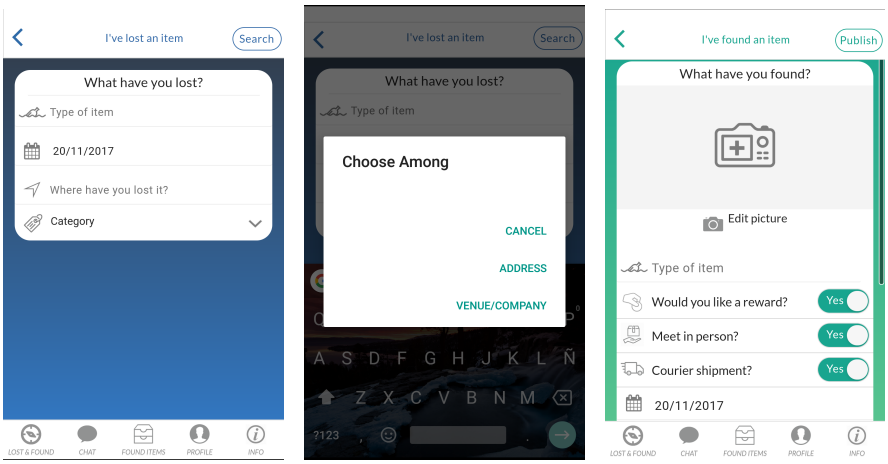


(a) Registro en la aplicación (b) Pantalla de inicio

Figura 5.5: Find My Lost: Sección principal

Las capturas de la figura 5.6 muestran lo que debe rellenar el usuario si ha perdido o encontrado algo. Se puede comprobar que al usuario que ha perdido un objeto no se le da la opción de elegir si quiere dar una recompensa por su objeto (a).

Además, hemos comprobado la utilidad de permitir varias opciones en el campo "*Where have you lost/found it?*" de la figura 5.6(b), una con la dirección y otra para elegir un lugar público de una lista desplegable que aparece en la aplicación.



(a) Formulario a rellenar (b) Opciones de lugar (c) Formatos de lugar

Figura 5.6: Find My Lost: Otras secciones

5.2.9. Qué nos interesa

De esta aplicación hemos aclarado varias ideas que ya teníamos, como dar la oportunidad al usuario que ha perdido algo de elegir si quiere dar una recompensa al que encuentre su objeto perdido.

También nos ha parecido relevante la posibilidad de filtrar por categorías las búsquedas de los objetos que más se han perdido o encontrado, para no tener que buscar entre multitud de anuncios.

Como ya hablamos en 5.2.1, "*MissingX*", otra idea interesante consiste en dar la posibilidad de ubicar un lugar concreto únicamente pinchando en un mapa, además de ofrecer una lista de lugares públicos o, simplemente, dar la opción de escribir la dirección.

Por último, el usuario puede editar su perfil y añadir datos, como su número de teléfono, su cuenta de *PayPal* para el posible pago o recompensa, etc. Ya teníamos pensado incluir una sección de perfil de usuario pero este análisis nos ha dado más motivos todavía para hacerlo.

5.2.10. Lost-Tag

Lost-Tag [22] es una aplicación alemana que permite poner etiquetas a los objetos personales. De tal manera que, cuando alguien encuentre ese objeto, solo debe escanear el código QR o introducir el ID de la etiqueta, y la aplicación se encargará de poner en contacto a esa persona con el propietario para que intercambien el objeto por la posible recompensa.

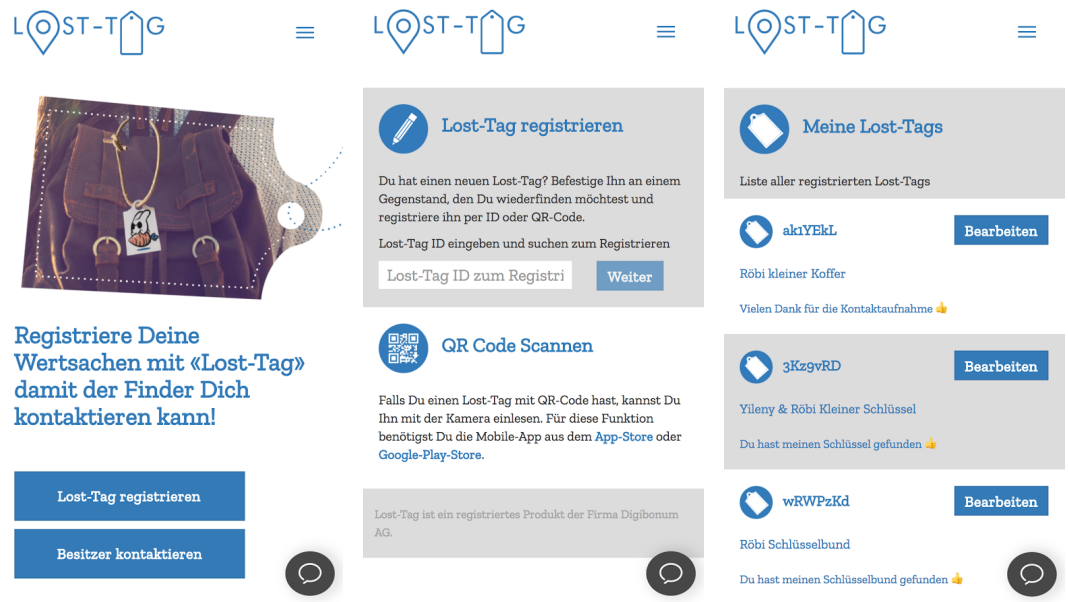
5.2.11. Interfaz

A la hora de registrarse, el usuario indica si quiere guardar una etiqueta nueva o si ha encontrado algo. Como observamos en la figura 5.7(a), ésta puede ser un ID o un código QR (b), y además se puede modificar cuando el usuario quiera (c).

Si el usuario ha perdido algún objeto, solo debe poner el ID de la etiqueta que haya asignado a ese objeto en cuestión, una recompensa a elegir por él mismo y otros valores como la categoría. Todo esto se ve en la figura 5.8(a).

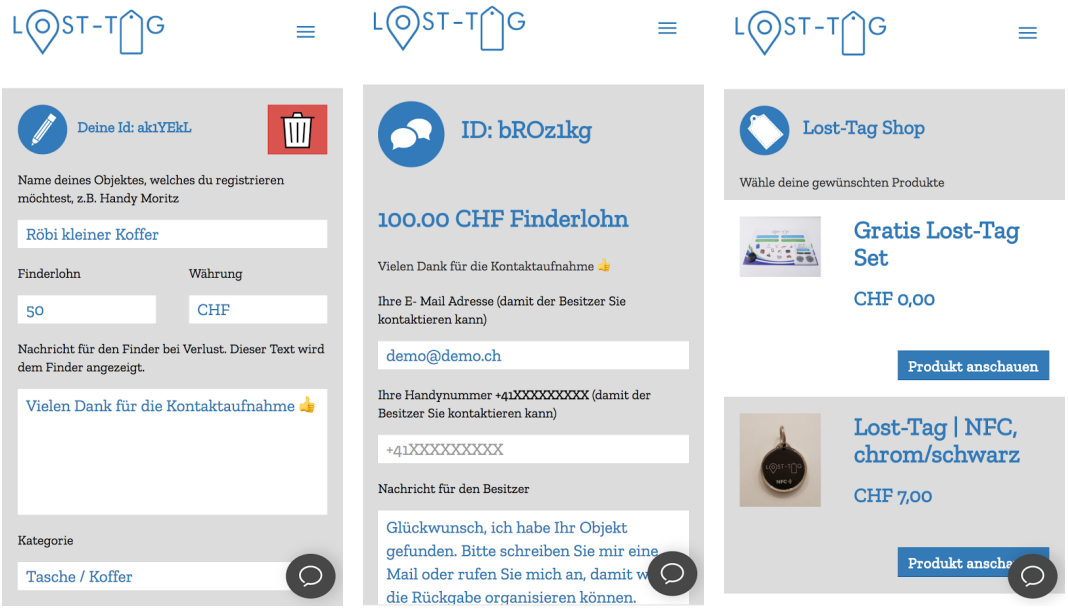
Si alguien encuentra un objeto, puede introducir o el ID de la etiqueta o el código QR para que la aplicación le brinde los datos necesarios para contactar con el dueño del objeto (b), además de obtener la información de si recibirá recompensa o no.

Por último, la aplicación tiene una tienda con todos los objetos encontrados que no han sido reclamados por su propietario (c).



(a) Registro en la aplicación (b) Etiquetar objetos (c) Modificar etiquetas

Figura 5.7: Lost-Tag: Sección principal



(a) Formulario objeto perdido (b) Formulario objeto encontrado (c) Tienda

Figura 5.8: Lost-Tag: Otras secciones

5.2.12. Qué nos interesa

Es una idea muy original el hecho de etiquetar objetos en la aplicación para que, en caso de pérdida, se puedan encontrar más fácilmente.

A pesar de su originalidad, observamos varios problemas en la aplicación. Uno de ellos es que, antes de la pérdida, es necesario haberse registrado junto con las etiquetas y, en general, el usuario no suele estar predispuesto a etiquetar todos los objetos que lleva encima. Nunca se piensa que vayamos a perder nuestras pertenencias hasta que, finalmente, ocurre. Por lo que, a menos que el usuario sea previsor, este mecanismo no tendría demasiado sentido.

Otro problema encontrado en esta aplicación consiste en que la persona que encuentra un objeto, nada más escanear la etiqueta, puede saber si recibirá una recompensa por devolver ese objeto o no. Esto puede llevar a esa persona a sopesar si realmente le merece la pena devolver el objeto en cuestión. Situación que queremos evitar a toda costa para el correcto funcionamiento de nuestra aplicación.

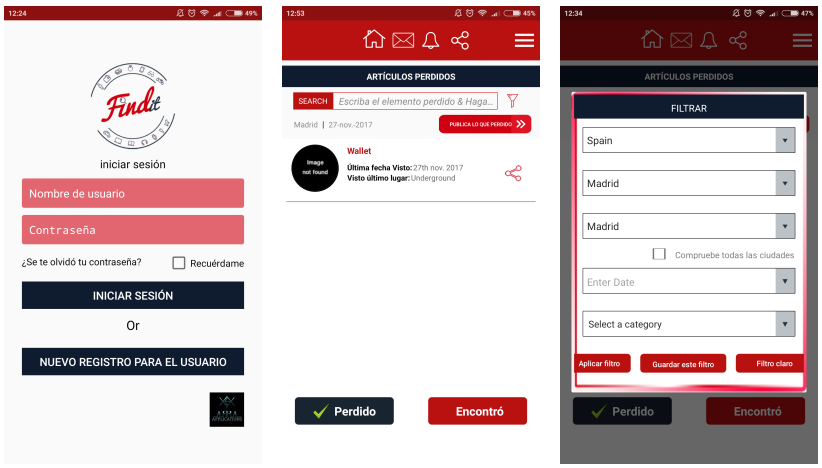
Por último, la tienda de objetos no reclamados también es una posible sección para nuestra aplicación, para vender los objetos encontrados de un anuncio que caduca y nunca llega a su propietario. Esto motivaría a los usuarios que encuentran un objeto a registrarlo en la aplicación.

5.2.13. *Find it - Lost and Found*

Find it - Lost and Found [11] es una aplicación con funcionalidades semejantes a todas las aplicaciones anteriores ya analizadas. En este caso, al igual que ocurría con la aplicación *Lost or found* que analizamos en la sección 5.2.4, "*Lost or Found*" se pueden localizar personas o mascotas que se hayan perdido, además de objetos.

5.2.14. Interfaz

Como observamos en la figura 5.9, nada más abrir la aplicación, nos encontramos con una pantalla de login (a), similar a la que pretendemos crear para nuestra aplicación. En ella podemos entrar si ya tenemos cuenta o registrarnos en caso contrario. Una vez hemos entrado y rellenado los campos de información que nos pide la aplicación (país, ciudad, idioma, etc), accedemos a una pantalla principal algo confusa (b), en la cual tenemos un listado de nuestros anuncios publicados (si hemos publicado alguno), y un buscador de objetos junto con un icono para filtrar la búsqueda por lugar, fecha y categoría (c).

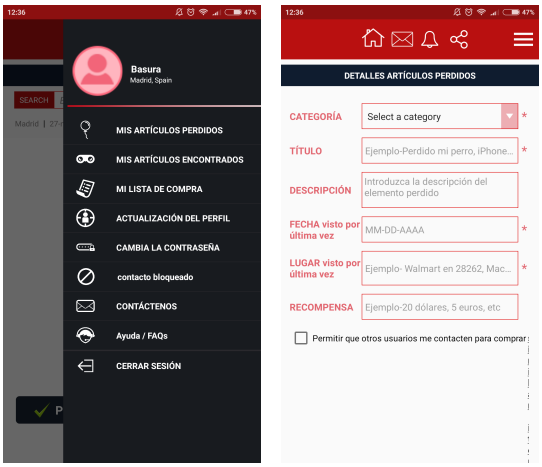


(a) Pantalla de login (b) Pantalla principal (c) Filtro de búsqueda

Figura 5.9: *Find it - Lost and Found*: Pantallas principales

Además de lo anterior, en la figura 5.10 observamos que la aplicación cuenta con un menú que se despliega desplazando el dedo hacia la izquierda de la pantalla (a). Desde este menú se puede acceder a múltiples opciones, como una lista de nuestros objetos perdidos o encontrados, preferencias de nuestro perfil o secciones de configuración.

A la hora de crear un anuncio de objeto perdido (b), se nos pide rellenar un formulario con distintos campos como la categoría, el título, la descripción, la fecha y el lugar donde se vio por última vez y, además, un campo opcional de recompensa que la persona esté dispuesta a dar por conseguir su objeto de vuelta. Por último, nos brinda la posibilidad de añadir imágenes a nuestro anuncio.



(a) Menú opciones (b) Anuncio de pérdida

Figura 5.10: *Find it - Lost and Found*: Otras secciones

5.2.15. Qué nos interesa

De esta aplicación hemos concluido que tiene algunas funcionalidades interesantes, como filtrar la búsqueda del objeto perdido o encontrado por varios criterios, o el menú desplegable, que se hace muy cómodo para dispositivos con pantallas no demasiado grandes como teléfonos móviles, haciéndolo accesible e intuitivo al mismo tiempo.

La idea de mostrar los anuncios publicados por el usuario en la pantalla principal es interesante. De esta manera, el usuario tiene un acceso rápido y puede comprobar si se ha dado algún tipo de match con otro anuncio que pueda contener su objeto, simplemente pulsando un botón.

Como contrapartida, la pantalla principal de la aplicación no guía demasiado al usuario y es poco intuitiva, además de no hacer distinción en los campos del formulario si se ha perdido un objeto, una mascota o una persona, brindando la posibilidad de dar distintos tipos de información en un caso u otro.

5.3. Resumen análisis herramientas

Tras realizar el análisis de las herramientas mencionadas en la sección anterior, hemos elaborado un resumen comparando las principales características de cada una de las aplicaciones analizadas, tratando de obtener una visión global de todas ellas que nos ayude a extraer las conclusiones finales del estudio efectuado. El resumen se puede observar en el cuadro 5.1.

	MissingX	Lost Or Found	Find My Lost	Lost-Tag	Find it - Lost and Found
Varias posibilidades para ubicar el lugar	X		X		
Posibilidad de ofrecer recompensa			X	X	X
Listado de objetos de la aplicación		X	X		X
Tienda de objetos no devueltos				X	
Chat entre usuarios	X		X		
Barra de navegación					X

Cuadro 5.1: Tabla resumen características

En la tabla observamos que muy pocas de las herramientas analizadas brindan funcionalidades importantes en aplicaciones de este tipo, como tener varias posibilidades para ubicar el lugar o un chat entre los usuarios.

Otras ofrecen funcionalidades como la posibilidad de recompensar por la devolución de un objeto, o una lista donde el usuario pueda buscar el objeto perdido o encontrado, aunque no en todas existe la opción de filtrar por alguna característica. Esto acaba resultando en una lista interminable de objetos que sirve de poca utilidad para el usuario.

Por último, es importante destacar que algunas funcionalidades como la tienda de objetos no devueltos o la barra de navegación tan solo las implementa una aplicación.

5.4. Conclusiones

Tras realizar el análisis de todas las aplicaciones y resumir sus características principales en la tabla de la sección anterior, hemos sacado las siguientes conclusiones:

- Como comentamos previamente en la sección 5.2.3, "Qué nos interesa" es importante establecer varias posibilidades para seleccionar el lugar donde ha ocurrido la pérdida o hallazgo del objeto en cuestión. De esta manera, el usuario puede elegir qué opción le resulta más adecuada para especificar el lugar: señalándola en un mapa, describiendo la dirección concreta, o especificando la línea y la estación en caso de que se trate de un lugar de transporte.

- A pesar de que varias de las aplicaciones analizadas brindan la opción de ofrecer una recompensa por obtener un objeto de vuelta, nosotros hemos optado por la opción de confiar en el altruismo de nuestros usuarios y no ofrecer esta posibilidad. Si, en un futuro, vemos que esto afecta a la efectividad de la aplicación, nos plantearíamos ofrecerla.

- Como hemos visto en la tabla resumen, es importante ofrecer un listado de objetos de la aplicación, ya que demuestra a los usuarios que no son los únicos que usan la aplicación y que tienen posibilidades de recuperar o devolver su objeto. Además, cabe destacar la importancia de que exista la posibilidad de filtrar en este listado por categoría del objeto o por tipo de anuncio, para poder realizar búsquedas más concretas.

- La idea de integrar en la aplicación una tienda con los objetos que, finalmente, no se han podido devolver a sus legítimos dueños nos ha parecido muy interesante ya que, de esta manera, das salida a los objetos que se encuentran en esta situación. A pesar de esto, consideramos más prioritario que la aplicación consiga su objetivo principal, por lo cual no creemos que esta sección sea esencial y no formará parte de nuestro sistema.

- Al contrario que en el punto anterior, una sección que consideramos esencial, y que no todas las aplicaciones analizadas tienen, es el chat. De esta manera, los usuarios se pueden comunicar entre sí y ponerse de acuerdo estableciendo en qué términos van a resolver la situación, con el objetivo final de devolver el objeto a su dueño original.

- A pesar de que tan solo una de las aplicaciones analizadas contaba con una barra de navegación, esta opción nos gustó más que el menú inferior, puesto que se deja más espacio libre para el resto de la información en la pantalla del móvil, siendo esto una ventaja debido a sus reducidas dimensiones. Además, la barra de navegación nos ofrece más espacio al ser un menú vertical, por lo que podemos establecer un

mayor número de secciones más concretas para facilitar la navegabilidad y hacer más intuitiva la aplicación.

- Como punto final, cabe destacar que no hemos encontrado ninguna aplicación de este tipo que implemente funcionalidades relacionadas con *open data*, por lo que será el gran factor que diferencie nuestra aplicación de todas las demás que tratan este tipo de temas.

Capítulo 6

Open Data SNCF y *Google APIs*

6.1. Introducción

En este capítulo se explica detalladamente todo el trabajo realizado con ayuda del open data escogido finalmente para la aplicación, como se mencionó en la sección 4.7, "Uso de datos abiertos en nuestra aplicación", además del proceso llevado a cabo para usar la *Geocoding API de Google*, esto se explicará más adelante en 6.6.2, "*Geocoding API* de *Google*".

6.2. Elección de los conjuntos de datos abiertos

Tal y como se explicó en el capítulo 4, "*Open Data*" los datos abiertos son una parte esencial de este proyecto, por ello escoger correctamente el conjunto de datos a utilizar en nuestra aplicación era un paso fundamental. Siendo conscientes de esto hicimos una búsqueda exhaustiva de conjuntos de datos abiertos sobre objetos perdidos en nuestro país, pero no pudimos encontrar ningún catálogo que tuviera unas dimensiones y una frecuencia de actualización suficientes para nuestro proyecto.

A pesar de nuestra búsqueda infructuosa decidimos realizar una solicitud al ayuntamiento de Madrid para crear un conjunto de datos referente a objetos perdidos, pero tan solo conseguimos que valorasen y publicaran nuestra propuesta en la página web del ayuntamiento de Madrid, que se puede consultar en el siguiente enlace:

`https://datos.madrid.es/portal/site/egob/menuitem.3efdb29b813ad8241e830cc2a8a409a0/?vgnextoid=3af2a7ede3e01610VgnVCM1000001d4a900aRCRD&vgnextchannel=102612b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default`

6.3. Conjuntos de datos abiertos *SNCF*

Al no tener a nuestro alcance un catálogo de datos sobre objetos perdidos de nuestro país tuvimos que buscar proveedores de *open data* de objetos perdidos de otros países. Al final, decidimos utilizar los datos de la "*Société nationale des chemins de fer français*",

SNCF [36], o la "*sociedad nacional de ferrocarriles franceses*", ya que dispone de un total de 211 conjuntos de datos distintos que se actualizan unas tres veces al día.

Esta cantidad de datos en constante actualización nos sirven para demostrar el potencial que tienen las iniciativas de datos abiertos y la gran ayuda que nos pueden brindar a la hora de realizar aplicaciones como la nuestra, por lo que investigamos de qué manera podíamos utilizar la *SNCF API* [4] disponible para conseguir los datos que necesitábamos en nuestra aplicación.

En nuestro caso concreto nos interesaban dos conjuntos de datos de la página, por un lado *Déclarations de pertes* [5], que registra las declaraciones de pérdida de objetos y por otro *Objets trouvés* [28], referente a objetos encontrados.

Ambos conjuntos de datos tienen información sobre los objetos perdidos o encontrados en los ferrocarriles de toda Francia. El formato de estos dos conjuntos de datos pueden verse en la figura 6.1.

	Date	Gare	Code UIC	Nature d'objets	Type d'objets
1	28 mai 2018 14:58	Lyon Perrache	0087722025	Carte d'abonnement	Pièces d'identités et papiers personnels
2	28 mai 2018 14:57	Paris Est	0087113001	Téléphone portable	Appareils électroniques, informatiques,
3	28 mai 2018 14:55			Manteau, veste, blazer, parka, blouson,	Vêtements, chaussures
4	28 mai 2018 14:51			Lunettes en étui	Optique
5	28 mai 2018 14:49			Téléphone portable	Appareils électroniques, informatiques,
6	28 mai 2018 14:49			Livre de poche	Livres, articles de papéterie
7	28 mai 2018 14:47			Ordinateur, ordinateur portable, noteb	Appareils électroniques, informatiques,
8	28 mai 2018 14:46			Tablette tactile protégée (étui, housse)	Appareils électroniques, informatiques,
9	28 mai 2018 14:46			Chaussures (autre que chaussures de sk	Vêtements, chaussures
10	28 mai 2018 14:45	Paris Gare du Nord	0087271007	Autre bagagerie (préciser)	Bagagerie: sacs, valises, cartables
11	28 mai 2018 14:45			Etui à lunettes vide	Optique
12	28 mai 2018 14:42			Manteau, veste, blazer, parka, blouson,	Vêtements, chaussures
13	28 mai 2018 14:37	Lyon Perrache	0087722025	Porte-monnaie, portefeuille	Porte-monnaie / portefeuille, argent, tit
14	28 mai 2018 14:35			Porte-monnaie, portefeuille	Porte-monnaie / portefeuille, argent, tit
15	28 mai 2018 14:33			Porte-monnaie, portefeuille	Porte-monnaie / portefeuille, argent, tit

	Date	Type d'objets	Nature d'objets	Gare	Code UIC	Date et heure de restitution
1	28 mai 2018 15:00	Bagagerie: sacs, valises, cartables	Valise, sac sur roulettes	Rennes	0087471003	
2	28 mai 2018 14:51	Pièces d'identités et papiers personnels	Carte d'abonnement	Lyon Perrache	0087722025	
3	28 mai 2018 14:48	Bagagerie: sacs, valises, cartables	Valise, sac sur roulettes	Rennes	0087471003	
4	28 mai 2018 14:41	Articles de sport, loisirs, camping	Tente	Lyon Perrache	0087722025	
5	28 mai 2018 14:36	Bagagerie: sacs, valises, cartables	Sac à dos	Bordeaux Saint-Jean	0087581009	
6	28 mai 2018 14:36	Vêtements, chaussures	Manteau, veste, blazer, parka, blouson,	Lyon Perrache	0087722025	
7	28 mai 2018 14:34	Appareils électroniques, informatiques,	Téléphone portable	Paris Est	0087113001	28 mai 2018 14:58
8	28 mai 2018 14:32	Vêtements, chaussures	Bonnet, chapeau	Lyon Perrache	0087722025	
9	28 mai 2018 14:31	Pièces d'identités et papiers personnels	Pièce d'identité personnelle et autre	Angers Saint-Laud	0087484006	
10	28 mai 2018 14:31	Bagagerie: sacs, valises, cartables	Valise, sac sur roulettes	Lyon Part Dieu	0087723197	
11	28 mai 2018 14:28	Vêtements, chaussures	Bonnet, chapeau	Bordeaux Saint-Jean	0087581009	
12	28 mai 2018 14:25	Appareils électroniques, informatiques,	Téléphone portable protégé (étui, coqu	Paris Est	0087113001	
13	28 mai 2018 14:25	Optique	Lunettes	Rennes	0087471003	
14	28 mai 2018 14:23	Divers	Autres divers (préciser)	Angers Saint-Laud	0087484006	
15	28 mai 2018 14:21	Vêtements, chaussures	Manteau, veste, blazer, parka, blouson,	Bordeaux Saint-Jean	0087581009	

Figura 6.1: Conjuntos de datos sobre declaraciones de pérdida y objetos encontrados

Una funcionalidad relevante de *Lost2Found* consiste en que una de las posibilidades de las que dispone el usuario a la hora de especificar un lugar donde ha perdido o encontrado un objeto es indicando la línea y la estación del medio de transporte en cuestión. La base de datos de *Lost2Found* contiene la información de líneas y estaciones de ferrocarril disponiles en la web de *SNCF*, esto se puede apreciar en la figura 6.2.

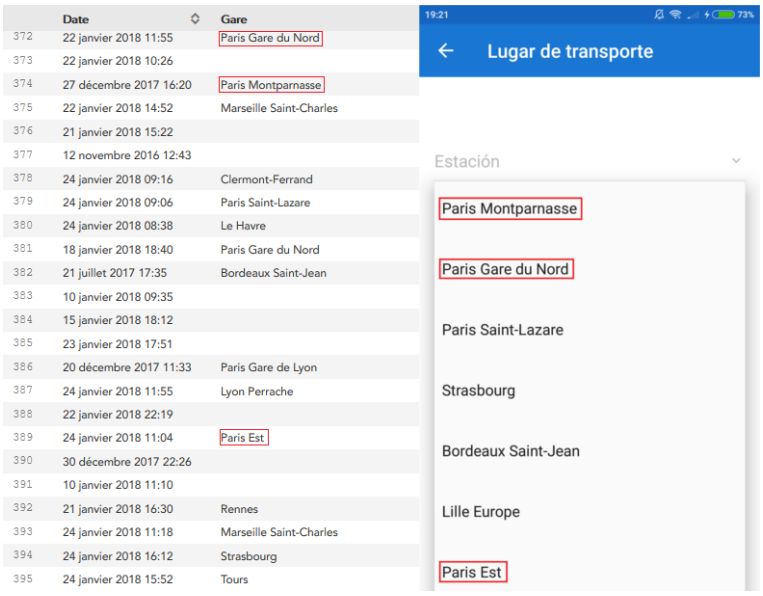


Figura 6.2: Coincidencia de estaciones entre nuestra aplicación y el conjunto de datos abiertos

Gracias a esto es posible realizar un match de un objeto de un anuncio local, es decir, creado por el usuario en nuestra aplicación, con un objeto que esté publicado en el conjunto de datos abiertos. Esto mejora de una manera muy notable la eficiencia de la aplicación ya que no solo se comparan los objetos locales publicados en la aplicación sino que también se estan comparando con conjuntos de datos públicos que contienen miles de objetos: el catálogo de declaraciones de pérdida por ejemplo lleva tan sólo en 2018 públicados más de 65000 objetos.

Debido a la diferencia de idioma ha sido necesario crear una tabla nueva en la base de datos llamada *conversor* que cumple la función de traducir las categorías de objetos del francés al español. Para poder hacer comparaciones entre ellas, la tabla contiene una columna con el nombre de la categoria de nuestra aplicación en español, y otra columna con su correspondiente nombre en francés. De esta manera, a la hora de buscar objetos que hagan match en el open data se traducen dinámicamente los nombres de los mismos, en caso de no coincidir con ninguna de las cuatro categorias en español se considera como un objeto dentro de la categoría “Otros“.

Sin esta tabla no sería posible realizar matches entre los objetos locales y los publicados en los conjuntos de datos abiertos. Su estructura puede verse en la figura 6.3.

	nombreTabla	tipoOpenData	tipoObjetoFrances
Editar Copiar Borrar	Tarjeta bancaria	SNCF	Carte de crédit
Editar Copiar Borrar	Tarjeta transporte	SNCF	Carte d´abonnement
Editar Copiar Borrar	Cartera	SNCF	Porte-monnaie, portefeuille
Editar Copiar Borrar	Telefono	SNCF	Téléphone portable

Figura 6.3: Estructura de la tabla conversor

6.4. Conexión con el proveedor de open data *SNCF*

Para conectarnos al open data y obtener los datos que nos interesan se hace algo semejante a que se explica en la sección 8.8, "Uso de AsyncTask", solo que en esta ocasión se usa el método *GET* en vez de *POST*.

Dentro de la *URL* de la petición *GET* realizada al open data existen distintas opciones que nos ofrece la *API* con las que podemos especificar las distintas preferencias que tengamos a la hora de consultar los datos, como por ejemplo qué columnas nos interesan, en qué orden queremos los datos, cuántos datos queremos, la zona horaria, etc. Todas estas maneras distintas de filtrar la petición *GET* están disponibles en la *SNCF API* [4].

En el caso concreto de nuestra aplicación, hacemos una peticion *GET* al conjunto de datos de objetos encontrados si estamos buscando match con un anuncio de tipo *Pérdida* o al otro conjunto en caso contrario.

En ambos casos el formato de la peticion es semejante: guardamos en un *String* la *URL* de la *API* correspondiente al conjunto al que hagamos la consulta, y le pasamos a ésta la fecha del anuncio original en el formato adecuado, para buscar objetos que hayan sido publicados en fechas cercanas a la del anuncio original.

En el enlace de más abajo por ejemplo se hace una petición al conjunto de objetos encontrados, de manera que se estan pidiendo 500 filas, ordenadas de manera descendente, al poner un guión antes de la palabra reservada *sort*, desde el día 20 de mayo de 2018 al día 23 de mayo de 2018.

`https://data.sncf.com/api/records/1.0/search//?dataset=objets-trouves-restitution&q=date+%3E%3D+2018%2F05%2F20+date+%3C+2018%2F05%2F23&rows=500&sort=-date&timezone=Europe/Madrid`

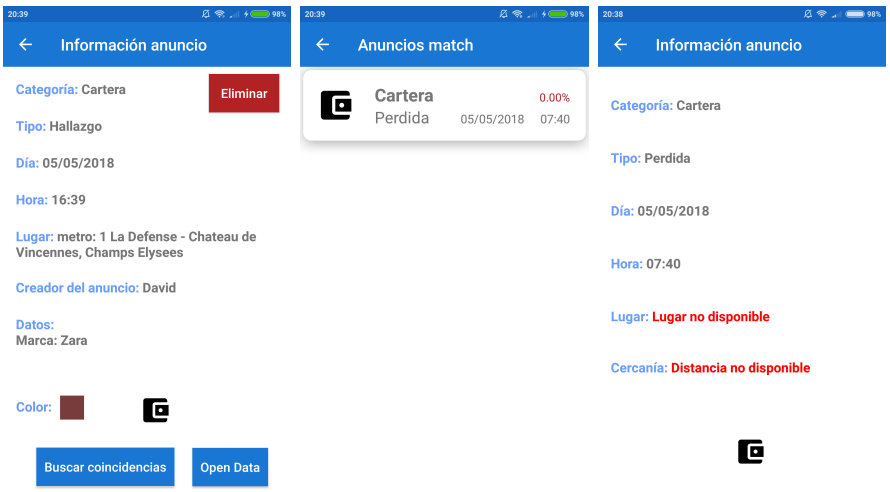
6.5. Inconvenientes surgidos con el proveedor de open data

A pesar de la potencia que nos brinda disponer de tal cantidad de datos, existen inconvenientes que han surgido a la hora de conectar nuestra aplicación con el conjunto de datos abiertos.

Un ejemplo de este tipo de inconvenientes es el hecho de que en el catálogo de declaraciones de pérdida no siempre esté disponible el lugar donde se ha perdido el objeto. Esto influye en el matching realizado entre anuncios ya que al no tener disponible el lugar de pérdida no se puede comparar con el lugar de hallazgo, siendo imposible calcular el porcentaje de distancia entre ambos lugares, tal y como se explica en la sección 8.6, "Búsqueda de correspondencia de anuncios compatibles".

Ocurre exactamente lo mismo con el color del objeto, ya que es un atributo que en nuestra aplicación si consultamos al usuario pero en el catálogo de datos abiertos no está publicado, por lo que tampoco es posible calcular el porcentaje de matching entre ambos colores.

De esta manera se pierden dos factores de gran relevancia a la hora de realizar el matching entre dos anuncios, por lo que en este sentido el matching realizado con los objetos publicados en el conjunto de datos abiertos es más impreciso que el realizado de manera local. En la figura 6.4 se puede ver el mensaje de error mostrado al usuario cuando la distancia de un objeto del conjunto de datos no está disponible.



(a) Anuncio original (b) Resultados match con open data (c) Lugar y distancia no disponibles

Figura 6.4: Lugar no disponible al realizar match con open data

6.6. Uso de las *API* de *Google*

Una de las características relevantes de nuestro sistema es la posibilidad de introducir el lugar de pérdida o hallazgo de un objeto de diversas formas: marcar la posición en un mapa, introducir una dirección o un lugar de transporte. Además, para buscar las posibles correspondencias de anuncios de pérdida y hallazgo (matching), es necesario calcular la distancia entre lugares de anuncios distintos. Para ello, hemos estudiado las *API* que proporciona *Google Maps* como datos abiertos.

6.6.1. *Distance Matrix API* de *Google*

Con el propósito de calcular un porcentaje de distancia que aportase información relevante al matching final entre los dos anuncios, nuestra intención era hacer uso de las *API* de *Google* para obtener la distancia existente entre dos lugares indicados por los usuarios.

Para esta labor nos es de gran utilidad la *Distance Matrix API de Google* [7], ya que proporciona información sobre la distancia y el tiempo de viaje para una ruta con un origen y un destino, en nuestro caso solo necesitamos conocer la distancia existente entre los dos puntos para calcular el porcentaje de distancia adecuado en cada caso.

Tras investigar sobre esta *API* y leer su documentación atentamente observamos la cantidad de opciones que ofrece a la hora de hacer la petición, además de la comodidad y sencillez de su uso.

Tan solo hace falta tener una cuenta de *Google* y solicitar una clave de *API*, esto se hace debido a que hay más opciones disponibles pero son de pago, como filtrar según el tipo de transporte o aumentar el número de peticiones a la *API* por día, ya que existe un límite para los usuarios que la usan de manera gratuita.

Debido a estas limitaciones, nos planteamos utilizar otra solución distinta a la *API*, ya que no podemos medir el número de peticiones que se van a realizar desde nuestra aplicación, al no saber cuántos usuarios van a utilizarla, por lo que buscamos otras alternativas y encontramos una solución usando la *Fórmula del semiverseno* [9], como se comenta en la sección 8.6, "Búsqueda de correspondencia de anuncios compatibles".

Con esta fórmula no tenemos la necesidad de utilizar la *Distance Matrix API* de *Google* ni ningún tipo de limitación de uso, por lo que finalmente la usamos para calcular la distancia y con ella calcular un porcentaje de distancia que contribuya de forma notable al porcentaje de matching final.

Figura 6.5: Google Developers

6.6.2. *Geocoding API* de *Google*

Tras comprobar el correcto funcionamiento de la formula del semiverseno para calcular la distancia entre dos puntos a partir de sus latitudes y longitudes, nos encontramos con el problema que se da si el usuario ha escogido otra alternativa distinta a señalar en un mapa el lugar donde ha perdido o encontrado un objeto, como se explica en 8.4.3, "Pantallas de lugar", ya que entonces no disponemos de la latitud y la longitud para usar esta fórmula y conseguir la distancia.

Para arreglar esta situación necesitamos convertir las direcciones especificadas por parte del usuario, ya sean direcciones concretas o estaciones de transporte, a coordenadas GPS con latitud y longitud, y esto es precisamente lo que hace la *Geocoding API de Google* [13], ya que convierte direcciones específicas en coordenadas geográficas.

Por lo tanto y sirviéndonos de los conocimientos adquiridos al investigar la *Distance Matrix API* de *Google* a pesar de no haberla utilizado finalmente, utilizamos la *Geocoding API* para realizar el paso previo al cálculo de la distancia entre dos puntos, que consiste en conseguir la longitud y la latitud de una dirección especifica.

Para ello, en el caso de que el usuario hubiera especificado la dirección de una manera concreta, es decir, indicando calle, número y código postal, realizamos la petición a la *API* pasándole esta información, si en cambio ha especificado el lugar indicando una línea y estación de transporte, o tan solo una línea como es el caso del tren, entonces utilizamos el nombre de la linea o estación para conseguir las coordenadas geograficas.

De esta manera resolvemos el problema y podemos calcular la distancia entre dos puntos sin importar de qué manera ha especificado el usuario el lugar, y siempre y cuando tengamos las distancias de ambos objetos disponibles, en otro caso se le comunica al usuario que no puede calcularse la distancia, como se comentó en 6.5, "Inconvenientes surgidos con el proveedor de open data".

A pesar de las limitaciones que tienen las *API* de *Google* en su versión gratuita, el número de usuarios que eligen especificar el lugar de una manera distinta a indicarlo en un mapa son una minoría, por lo tanto no supone un problema a la hora de tener un número limitado de peticiones por día que se pueden realizar a esta *API*.

Capítulo 7

Primer prototipo

7.1. Introducción

En este capítulo se detallan las características principales del primer prototipo de la aplicación, la arquitectura inicial del sistema, el diseño preliminar de la interfaz de usuario, los tipos de anuncios existentes, la base de datos, etc.

Las interfaces del prototipo las hemos diseñado mediante el programa *Just In Mind* [20] tratando de seguir estilo de diseño de *Material Design de Google* para *Android*.

También realizamos unos primeros diseños de la estructura de la base de datos, que fuimos refinando poco a poco hasta llegar a la versión final, que se describe en el capítulo 8, "Diseño e implementación de Lost2Found".

7.2. Arquitectura inicial del sistema

A la hora de gestionar el manejo de datos para el prototipo de la aplicación, sopesamos la idea de que el servidor fuera el encargado de descargar toda la información que necesitásemos de los conjuntos de datos abiertos que usáramos. Hicimos esto porque pensábamos que si el cliente era el encargado de gestionar esta gran cantidad de datos podría causar problemas o un tiempo de espera mayor del deseado.

Tal y como se puede apreciar en las figuras 7.6, "Base de datos" de este mismo capítulo, existe una tabla llamada *Open Data (Estación)* en la que se pensaba almacenar la información procedente del conjunto de datos abiertos, como la fecha, la hora, el tipo de objeto y el objeto en cuestión.

Debido a esto, el diagrama de alto nivel de la arquitectura del sistema de la figura 7.1 consta de un cliente, que en este caso se trata de una aplicación desarrollada en java para dispositivos móviles, y un servidor *Ubuntu 14.04* [39] con *Apache* [2] instalado, en el cual se ejecutan los ficheros *PHP* [32] que comunican el servidor con la base de datos *MySQL* [27], también incluye comunicación con sistemas externos como las *APIs* de *SNCF* y de *Geocoding de Google*. Nuestra intención era que el cliente guardara sólamente una copia de aquellos datos que necesita, y no tiene que consultar al servidor cada poco tiempo.

Figura 7.1: Arquitectura inicial del sistema

7.3. Comunicación entre los componentes del sistema

Desde el principio del proyecto teníamos claro qué componentes íbamos a utilizar para desarrollar nuestra aplicación. Lo que tuvimos que investigar fue cómo se iban a comunicar esos componentes entre sí.

Tras la investigación, finalmente nos acabamos decantando por usar un servicio web mediante *PHP* que comunicara nuestra aplicación con el servidor, y que este a su vez tratase con nuestra base de datos *MySQL* a través de un protocolo de intercambio de datos llamado *JSON*, conocido por ser ligero y usarse, normalmente, en navegadores.

De esta manera los archivos PHP se encargarían de realizar las operaciones y las consultas necesarias, comunicándose con la base de datos y recibiendo las respuestas a las peticiones correspondientes en cada caso.

(a) *PHP* (b) *MySQL* (c) *JSON*

Figura 7.2: Componentes del sistema

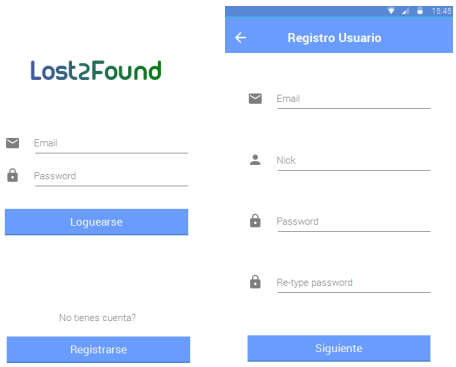
7.4. Diseño preliminar del interfaz de usuario

Tratamos de diseñar las interfaces de nuestra aplicación siguiendo la guía de diseño de *Google Material Design* [24], empezando por las pantallas principales esenciales como la de login o la pantalla principal del usuario.

Para seguir fielmente la guía de diseño nombrada anteriormente, era necesario definir una paleta de colores que usáramos en la aplicación. Nos decantamos por usar *Indigo 500* como color principal e *Indigo 700* como color complementario, además de usar el blanco como color de fondo en las pantallas de la aplicación.

Figura 7.3: Prototipo: Paleta, icono y logo de Lost2Found

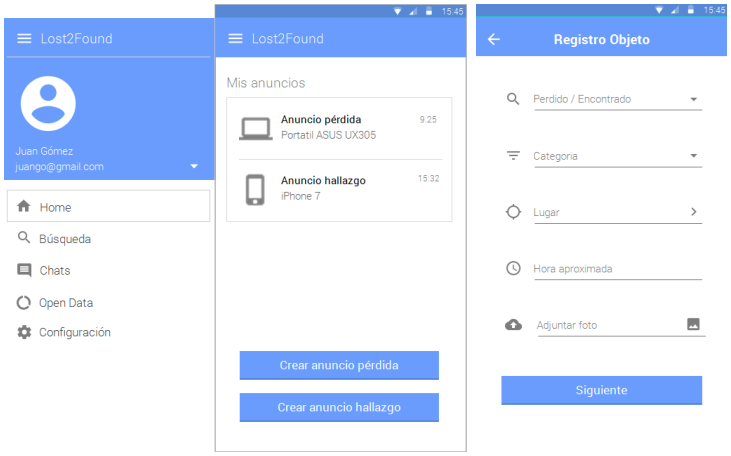
Tras esto, pasamos a diseñar las pantallas principales con *Just In Mind*, la herramienta que hemos mencionado anteriormente. Al ser la pantalla de login la primera que ve el usuario al abrir la aplicación, nuestra intención era que causara una impresión positiva e incitara a seguir usando la aplicación. En la figura 7.4 podemos ver el diseño realizado para la pantalla de login y la pantalla de registro.



(a) Pantalla de login (b) Pantalla de registro

Figura 7.4: Prototipo: Pantallas de login y registro

Además, se realizaron bocetos para la pantalla principal donde el usuario ve los anuncios que ha creado, junto con una barra de navegación desplegable para ir a las otras secciones de la aplicación y para la pantalla de registro de un objeto al crear un nuevo anuncio.



(a) Pantalla principal (b) Pantalla de registro objeto

Figura 7.5: Prototipo: Pantalla principal y de registro objeto

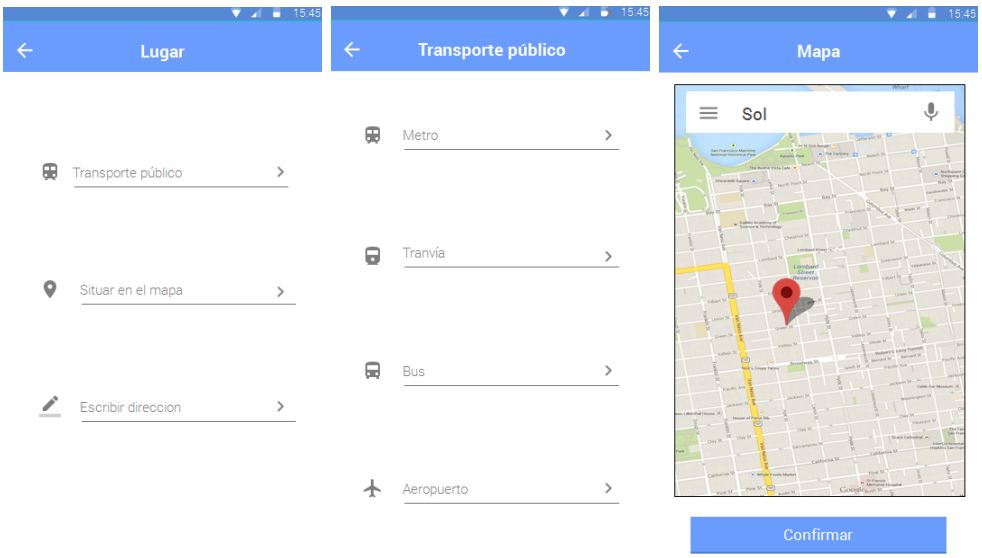
7.5. Tipos de lugar

Una de las conclusiones extraídas del análisis realizado sobre las aplicaciones semejantes en el capítulo 5, "Análisis de aplicaciones similares" fue la de hacer una clara distinción sobre los tipos de lugar a especificar por parte del usuario, como se explicaba en la sección 5.2.3, "Qué nos interesa".

Con esta idea en mente, realizamos bocetos sobre las pantallas, dando al usuario la posibilidad de indicar la manera en que prefería especificar el lugar donde había perdido o encontrado el objeto en cuestión.

Como podemos observar en la figura 7.6, la primera de las pantallas da la opción al usuario de especificar si el lugar se trata de algún tipo de transporte público. En caso de hacer esta elección, se le ofrece también la posibilidad de concretar, más aún, el lugar de transporte, indicando de cuál se trata en concreto, metro, autobús, tranvía, etc, como vemos en la segunda captura.

También dispone de las opciones de escribir la dirección concreta donde ha ocurrido, indicando la calle, número y código postal del lugar o, por último, de especificar la posición GPS del lugar en cuestión en un mapa como el de la tercera captura, 7.6(c).



(a) Tipos de lugar (b) Tipos de lugar de transporte (c) Mapa

Figura 7.6: Tipos de lugar a escoger por el usuario

7.6. Base de datos

A la hora de escoger el gestor para la base de datos de la aplicación elegimos Oracle [27], debido a que teníamos conocimientos previos del mismo gracias a varias asignaturas cursadas en la carrera. Ya habíamos trabajado con él previamente y cumplía con nuestras necesidades y requisitos, ofreciéndonos sencillez y facilidad de uso.

Para el primer diseño de la base de datos realizamos un diagrama entidad relación en el que plasmamos los requisitos a cumplir. En primer lugar, indicamos cada una de las entidades de la base de datos, como los usuarios, anuncios, chats, mensajes, lugares, objetos, etc, así como cada uno de sus atributos correspondientes. Después establecimos, tanto las relaciones entre las entidades, como las cardinalidades de cada una de ellas.

También consideramos cinco categorías de objetos distintas en la aplicación, ya que según [38] eran los tipos de objetos que más se perdían en el transporte público, estas categorías son:

- Tarjetas bancarias: Cuenta con atributos concretos como el banco y los datos del propietario.
- Tarjetas de transporte: En este caso solamente se almacenan los datos del propietario.

- Carteras: Sus atributos concretos son la marca y la documentación guardada en la cartera.

- Teléfonos: Tiene como atributos concretos la marca del teléfono y el modelo, así como alguna tara existente en el mismo.

- Otros: Esta categoría se creó con la intención de cubrir todos aquellos objetos que no entran en las cuatro indicadas anteriormente, por ello sus atributos concretos son el nombre y una descripción del objeto en cuestión.

Entre estas categorías de objetos existen atributos comunes, como la marca en las carteras y móviles o los datos del propietario en las tarjetas, así como atributos concretos, que el usuario no está obligado a especificar, a pesar de que recomendamos dar toda la información posible, ya que de esta manera el algoritmo encargado de encontrar la correspondencia entre objetos funcionará mejor, como se explica en la sección 7.7, "Algoritmo de matching" de este mismo capítulo.

Estas categorías, así como las distinciones entre dos tipos de anuncios (hallazgo y pérdida) y dos tipos de lugar (de transporte y concreto), se pueden observar en la figura 7.7 que contiene uno de los primeros diagramas que diseñamos para la base de datos. También comprobamos como establecimos, de manera errónea, dos relaciones entre las entidades de usuario y chat, que deberían ir entre chat y mensaje.

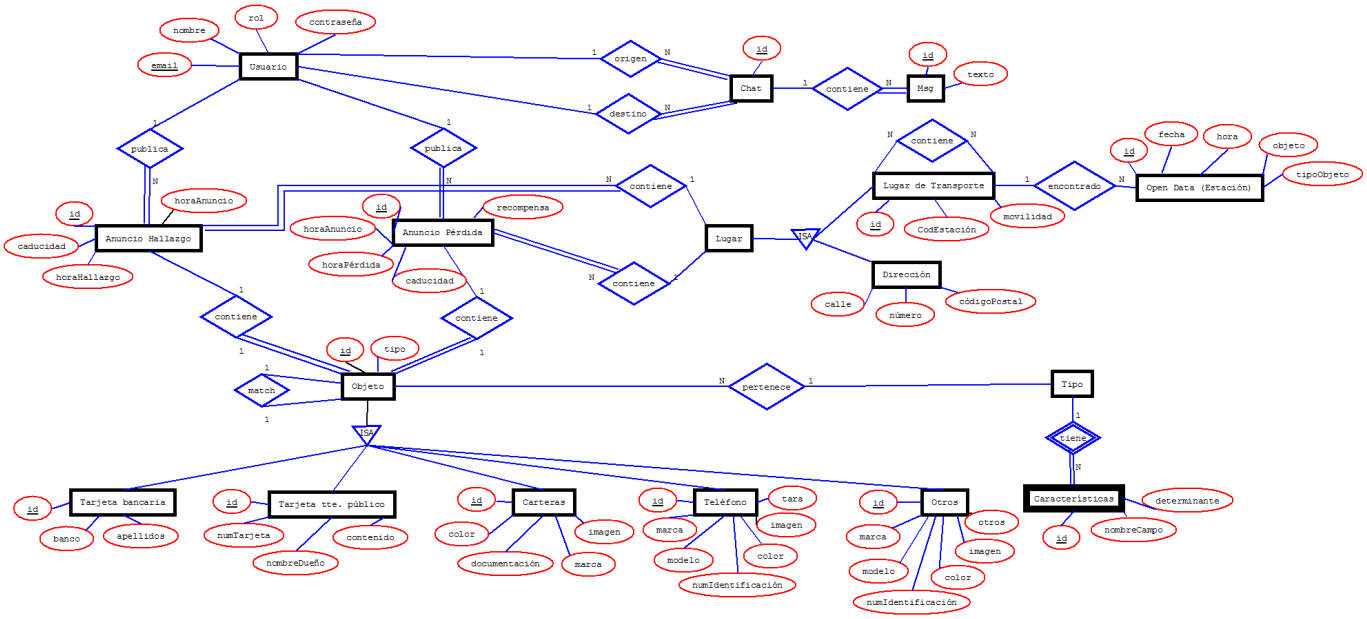


Figura 7.7: Primer diseño diagrama entidad relación

Entre las entidades más importantes destacan los usuarios, con email como clave primaria, nombre, contraseña y rol, los anuncios, que tenían atributos como el tipo, la hora, un id con auto incremento, etc, y los objetos, que podían ser de distinto tipo, y cada uno tenía unos atributos propios. Este diagrama fue evolucionando a medida que refinábamos nuestras necesidades, desapareciendo e incluyendo entidades, atributos y relaciones.

En la figura 7.8 se detallan los cambios realizados en el diagrama entidad relación respecto al primer diseño, tras refinar y repensar las entidades y las relaciones entre sí. Entre los cambios más significativos cabe destacar que ya no se distingue la entidad anuncio según el tipo del mismo, ya que los atributos específicos de cada uno se marcan como nullables según convenga. De esta manera, solo necesitamos una entidad de anuncio. También cabe destacar la corrección de las relaciones entre las entidades usuario, chat y mensaje.

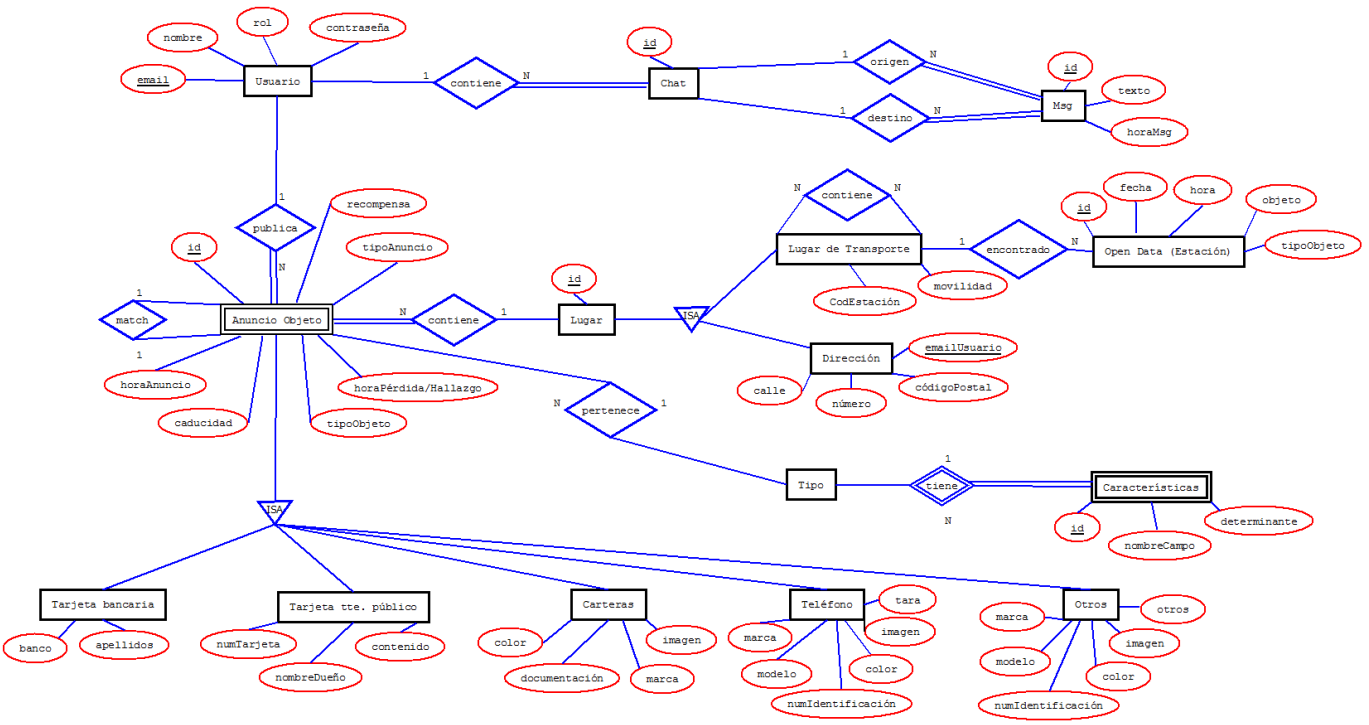


Figura 7.8: Segundo diseño diagrama entidad relación

Además de estos cambios que hemos observado en el diseño, se realizaron muchos otros hasta llegar a la versión definitiva del diseño de la base de datos, la que se detallará en el capítulo 8, "Diseño e implementación de Lost2Found".

7.7. Algoritmo de matching

Una de las funcionalidades más importantes de la aplicación es la manera según la cual se realiza la comparación entre anuncios para ver si estos corresponden entre sí, de manera que ambos anuncios se refieran realmente al mismo objeto. A este proceso lo llamamos *matching* y el algoritmo que se ha desarrollado para el cálculo del mismo no es trivial, es decir, no se reduce a comparaciones exactas entre los datos disponibles de los dos anuncios ya que, de esta manera, las coincidencias entre anuncios de la aplicación solo se darían en el caso que las características de los objetos de los anuncios fueran exactamente iguales, excluyendo todos aquellos casos en los que por ejemplo el usuario escribiera mal una característica de un objeto o escogiera otro tono del mismo color elegido por el primer usuario.

Por todo esto, consideramos necesario que el algoritmo de matching que calcula esta correspondencia entre anuncios tuviese en cuenta factores como qué información se tiene disponible de cada uno de los dos objetos y qué semejanzas pueden existir entre ellos, además de considerar los posibles errores de los usuarios al describir características o al tener distintas percepciones de un mismo color, haciendo así una comparación aproximada entre los objetos de ambos anuncios.

Para ello, el algoritmo de matching desarrollado tiene en cuenta una serie de factores que producen un porcentaje de coincidencia o matching entre ambas características. De esta manera, entre ambos objetos existe, por ejemplo, un porcentaje de cercanía de la distancia entre los dos puntos geográficos señalados por los usuarios en cada uno de sus anuncios, así como un porcentaje de coincidencia entre los colores de cada uno de los objetos. El cálculo de estos porcentajes y de qué manera influyen en el cálculo del match final se explicarán más detalladamente en la sección 8.6, "Búsqueda de correspondencia de anuncios compatibles" del capítulo 8, "Diseño e implementación de Lost2Found".

También es importante destacar que existen dos tipos de matching distintos en nuestra aplicación, uno local, que tiene en cuenta las características de los objetos pertenecientes a los anuncios que han sido creados en la aplicación, y que realiza los cálculos necesarios y las comparaciones aproximadas anteriormente explicadas entre ellos. El otro tipo de matching existente en nuestra aplicación es el matching con open data. En este caso se tienen en cuenta aquellos objetos publicados en el conjunto de datos abiertos sobre objetos perdidos o encontrados de *SNCF*, dependiendo de cada caso concreto. Para ello se hace uso de la *API* proporcionada por *SNCF* para realizar una consulta sobre qué objetos se han publicado, con características semejantes a las del anuncio original, en fechas aproximadas, calculando así la correspondencia entre el objeto perteneciente al anuncio original y los publicados en los conjuntos de datos abiertos. Los detalles sobre de qué manera se realizan este tipo de consultas a la *API* de *SNCF* se han explicado en el capítulo 6, "*Open Data SNCF* y *Google APIs*".

En el momento en que nos paramos a pensar sobre toda la carga computacional que iba a conllevar realizar el cálculo del algoritmo de matching, nos pasó algo parecido a lo explicado en la sección 7.2, "Arquitectura inicial del sistema" sobre la obtención de la información de los conjuntos de datos abiertos, ya que en un principio pensábamos que todas las operaciones necesarias para el cálculo del algoritmo se iban a realizar en el servidor, debido a la alta carga que pensábamos que supondría.

A pesar de esto, en la versión final de la aplicación, el cálculo de todos los factores necesarios para obtener el porcentaje de match final entre dos anuncios, explicados en la sección 8.6, "Búsqueda de correspondencia de anuncios compatibles" del siguiente capítulo, se realizan en la aplicación cliente, y no en el servidor.

Además de las operaciones y cálculos realizados en el cliente, también se utiliza un algoritmo de *sort* para reordenar todos los anuncios resultantes, de manera que se muestren primero al usuario los que tengan el porcentaje más alto y, por lo tanto, más posibilidades de tratarse del mismo objeto que contiene su anuncio.

Capítulo 8

Diseño e implementación de Lost2Found

8.1. Introducción

En este capítulo se explicará todo el trabajo referente al diseño, implementación y planificación realizados a lo largo del proyecto. Además, se podrán apreciar los cambios que nos hemos visto obligados a realizar, debido a las dificultades surgidas o a las alteraciones de los requisitos a lo largo del proyecto. Se ha redactado el capítulo en orden cronológico para que así el lector pueda seguir los pasos realizados en el proyecto a medida que va leyendo las distintas secciones.

8.2. Arquitectura del sistema

A pesar de lo dicho en la sección 7.2, "Arquitectura inicial del sistema" del capítulo 7 "Primer prototipo", donde se hablaba sobre que el servidor se encargase de la gestión de las funcionalidades que conllevaran el manejo de grandes cantidades de datos o el cálculo de numerosas operaciones, en la versión final de la aplicación se delegó esta tarea en el cliente, es decir, la aplicación, al comprobar que no suponía ningún tipo de problema ni una espera demasiado larga para el usuario.

Por lo tanto, finalmente se delegó la tarea de la gestión de la información de los conjuntos de datos abiertos al cliente y no al servidor. De esta manera, es la aplicación cliente la que se conecta al open data en cuestión y obtiene los datos necesarios en cada caso.

Como se observa en las figuras de la sección 8.5, "Implementación y cambios en la base de datos" de este capítulo, finalmente la tabla *Open Data (Estación)* no está presente, ya que tras nuestra decisión, no teníamos necesidad alguna de que existiera en el diseño final de la estructura de nuestra base de datos.

Tal y como hemos comentado, en la figura 8.1 observamos el diagrama de alto nivel de la arquitectura final del sistema, en el que se puede observar que ha cambiado la gestión de la información proveniente, tanto de los conjuntos de datos abiertos de *SCNF*, como de la *Geocoding API de Google*, siendo la aplicación cliente la encargada de conectarse y obtener la información correspondiente de estas fuentes. Este proceso se explicó en mayor profundidad en el capítulo 6, "*Open Data SNCF* y *Google APIs*".

Figura 8.1: Arquitectura del sistema

8.3. Conexión entre servidor y aplicación Android

Como se explicó previamente en el capítulo anterior en la sección 7.3, "Comunicación entre los componentes del sistema", a la hora de elegir de qué manera íbamos a realizar la conexión entre el servidor y la aplicación nos decantamos por un servicio web mediante el lenguaje *PHP* [32], usando *JSON* [19] como protocolo de intercambio de datos.

De esta manera hemos creado una clase en *PHP* para cada una de las entidades existentes en nuestra aplicación, en la cual se implementan todas las operaciones necesarias en cada caso para su correcta interacción con nuestra base de datos.

También hemos definido adicionalmente un archivo *PHP* para cada una de las operaciones realizadas, en las cuales se tratan los datos recibidos mediante el protocolo *HTTP* y el método *POST*, guardando estos datos en variables para posteriormente llamar a la clase correspondiente a la entidad en cuestión y realizar la operación procesando esos datos.

En la figura 8.2 podemos observar una de las funciones ejecutadas en los servicios *PHP* del servidor para el correcto funcionamiento del login y el registro de la aplicación, la clase *getUserByEmailJSON.php*, en la que tras procesar y guardar en variables los datos correspondientes, se llama al método *select()* de la clase *UserClass.php*, para realizar la consulta correspondiente a la base de datos.

```php
<?php
    if(isset($_POST["json"])) {
        $json = $_POST["json"];
        $json = urldecode($json);
        $json = str_replace("\\", "",$json);
        $jsonencode = json_decode($json);

        $email = $jsonencode[0]->email;


        require_once("userClass.php");
        $userObject = new User();
        $user = $userObject->select($email);
        echo json_encode($user);
    }
?>
```

Figura 8.2: Clase usuario

En la figura 8.3 vemos la clase *UserClass.php* donde se implementan los métodos de selección, inserción y actualización correspondientes a las operaciones de SELECT, INSERT y UPDATE de nuestra base de datos. También observamos la inclusión de la clase *dbFunctions.php* para la conexión con el servidor, y vemos la función *select()*, que realiza la consulta "*SELECT * FROM usuario WHERE email = ?*" para obtener toda la información sobre un usuario a partir de su email y guardarla posteriormente en un array *rawdata[]*.

```php
<?php
    include('../dbFunctions.php');

    class User {
        // Devuelve un array con el resultado de la consulta.
        function select($email) {
            $connection = connectDB();

            $sql = mysqli_prepare($connection, "SELECT * FROM usuario WHERE email = ?");
            mysqli_stmt_bind_param($sql, "s", $email);

            $query = $sql->execute();

            if(!$query)
                die();
            $result = $sql->store_result();

            $realresult = $sql->bind_result($id, $email, $name, $passHash);

            $rawdata = array();

            $sql->fetch();

            $correct = password_verify($password, $passHash);

            $rawdata['id'] = utf8_encode($id);
            $rawdata['email'] = utf8_encode($email);
            $rawdata['nombre'] = utf8_encode($name);
            $rawdata['correct'] = $correct;

            disconnectDB($connection);
            return $rawdata;
        }

        // Inserta un usuario en la base de datos
        function insert($email, $name, $password) {
            $connection = connectDB();
```

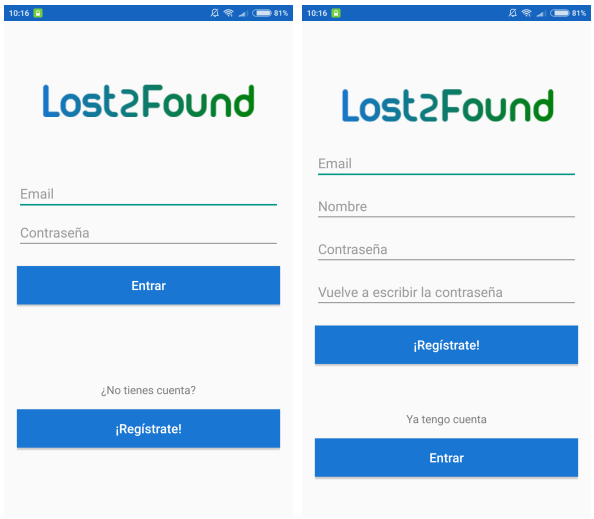
Figura 8.3: Clase *getUserByEmailJSON.php*

8.4. Diseño del interfaz de usuario

8.4.1. Pantallas de login y registro

Para empezar con el desarrollo de la aplicación lo primero fue pasar las pantallas que habíamos diseñado en nuestro primer prototipo con *Just In Mind* a los layouts propios de Android mediante archivos con formato *XML*.

Los primeros layouts realizados fueron los correspondientes a las pantallas mostradas nada más abrir la aplicación, que son la de login y registro, mostradas en la figura 8.4.



(a) Pantalla de login (b) Pantalla de registro

Figura 8.4: Pantallas de login y registro de Lost2Found

La interfaz de estas pantallas está formada por un *ConstraintLayout* que nos permite ajustar el contenido dependiendo del tamaño de la pantalla del móvil, dentro tiene un *LinearLayout* que dispone los elementos uno debajo de otro. Este tipo de layout es muy apropiado para interfaces que contengan formularios, como ocurre en este caso.

En la pantalla de login, dentro del *LinearLayout* tenemos una imagen para el logo de la aplicación, dos campos de texto para que el usuario introduzca su email y contraseña, y un botón para entrar en la aplicación. Más abajo hay un texto y otro botón para acceder a la pantalla de registro en caso de que no tengamos una cuenta. Además, existe un texto oculto debajo del botón para entrar que se muestra en caso de que las credenciales introducidas no sean correctas, comunicándole el mensaje de error al usuario.

En cuanto a la pantalla de registro su interfaz tiene una estructura semejante a la de login, solo que cuenta con más campos de texto con datos a rellenar para poder registrar al usuario en la aplicación.

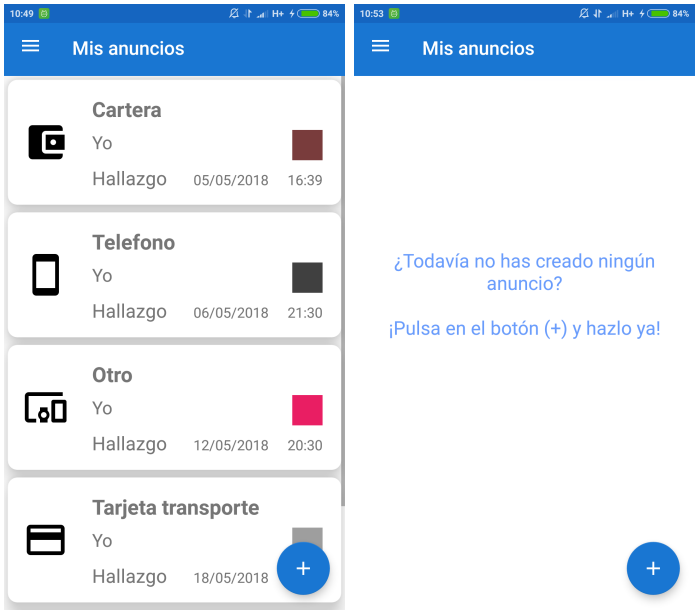
8.4.2. Pantalla principal y de creación de un anuncio

La pantalla principal de la aplicación muestra al usuario sus anuncios, si ha creado alguno. Esto se hace gracias a un *RecyclerView*, componente visual que muestra los elementos de una lista, en este caso anuncios. Para que se muestre correctamente, esta vista tiene que estar dentro de un elemento llamado *DrawerLayout*.

Cada vista de este tipo tiene un adaptador (*Adapter*) asociado que sirve para gestionar los elementos de la lista y enlazarlos con los elementos correspondientes. En la aplicación se hace uso de este tipo de vistas para varias funcionalidades importantes como los anuncios, la búsqueda y el chat.

Además de estos elementos, la pantalla principal también dispone de una barra superior o *toolbar*, siguiendo la guía de diseño de *Material Design de Google* y tratando de esta manera de captar la atención del usuario de forma que las interfaces le resulten atractivas y agradables.

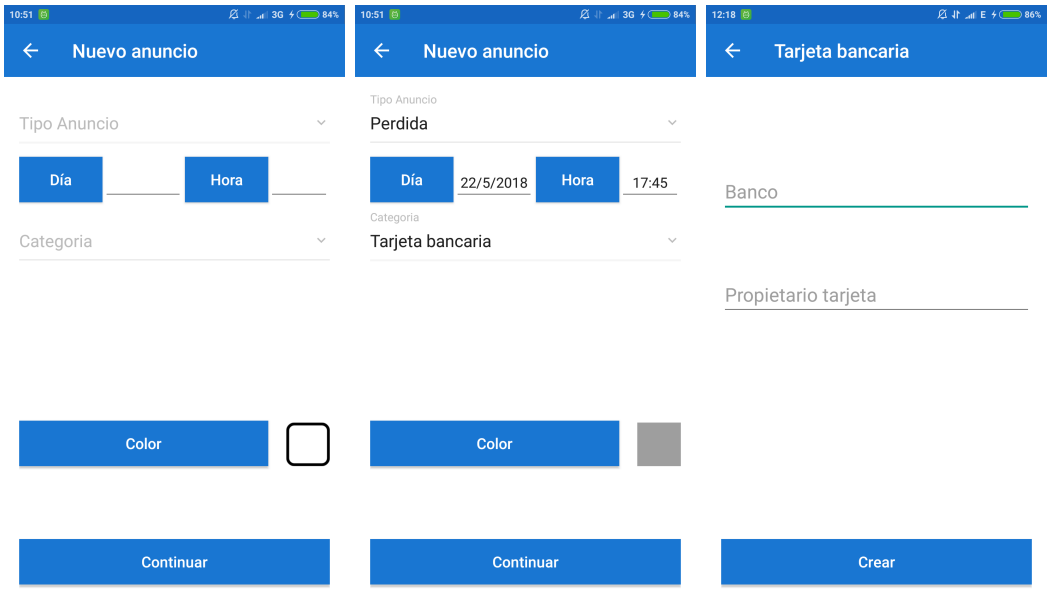
También existe un texto oculto que solamente se muestra en caso de que el usuario no haya creado anuncios todavía. Un botón flotante (*Floating Action Button*) que, al pulsarlo, brinda la posibilidad de crear un nuevo anuncio y una barra de navegación desplegable (*NavigationView*) para poder acceder a las demás secciones de la aplicación. Todos estos elementos se pueden ver en la figura 8.5.



(a) Pantalla principal con anuncios (b) Pantalla principal sin anuncios

Figura 8.5: Pantalla principal de Lost2Found

Tras especificar el lugar de una de las tres maneras posibles como se explicará en 8.4.3, "Pantallas de lugar", se pasa a la pantalla de creación de un anuncio, que consiste en un elemento llamado *ConstraintLayout* que contiene una barra superior y varios elementos organizados dentro. Entre ellos tenemos dos desplegables (*MaterialBetterSpinner*), uno para que el usuario elija el tipo del anuncio y otro para que escoja la categoría del objeto en cuestión. Tras completar toda la información y pulsar en continuar, accedemos a la pantalla de los datos concretos del objeto en cuestión. Como ejemplo, en la figura 8.6 se muestran los datos solicitados si el objeto específico es una tarjeta bancaria.



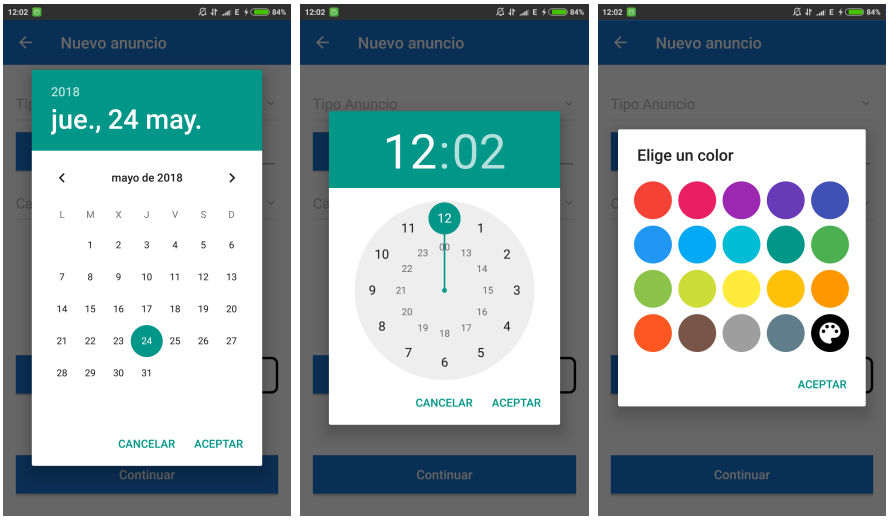
(a) Pantalla de creacion anuncio

(b) Pantalla rellenada

(c) Pantalla especifica tarjeta bancaria

Figura 8.6: Pantalla de creación de un anuncio

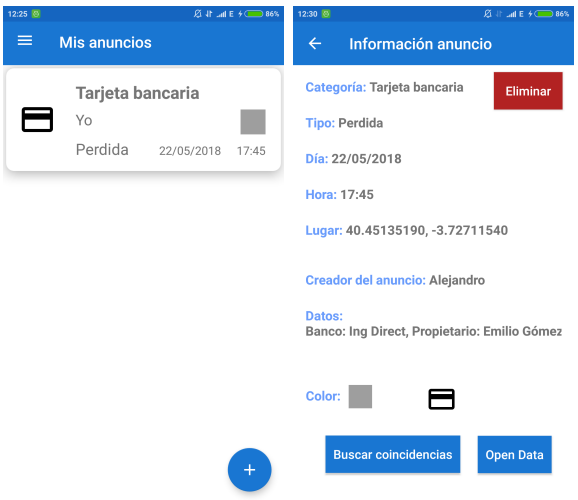
La pantalla de creación de anuncio, también contiene tres botones que al pulsarlos abren un diálogo con el correspondiente desplegable (*Picker*) en cada caso. Existe uno para indicar la fecha en el *DatePicker*, otro similar para elegir la hora en un *HourPicker* y el último para seleccionar el color del objeto en el *ColorPicker*, estos elementos se pueden ver en la figura 8.7.



(a) Date picker (b) Hour picker (c) Color picker

Figura 8.7: Pickers Lost2Found

Una vez se pulsa en el botón confirmar se nos muestra la pantalla principal de la aplicación con nuestro nuevo anuncio creado, mostrado en la lista de anuncios. Si en la ventana de anuncios se pulsa sobre un anuncio existente, se muestra la pantalla de información concreta con todos los datos disponibles en la aplicación sobre el anuncio en cuestión. Además, tenemos la posibilidad de eliminar un anuncio, por ejemplo si el usuario ha conseguido recuperar su objeto, o bien buscar un 'match' con otro anuncio.



(a) Pantalla principal con nuevo anuncio (b) Información concreta anuncio

Figura 8.8: Pantalla principal con nuevo anuncio

8.4.3. Pantallas de lugar

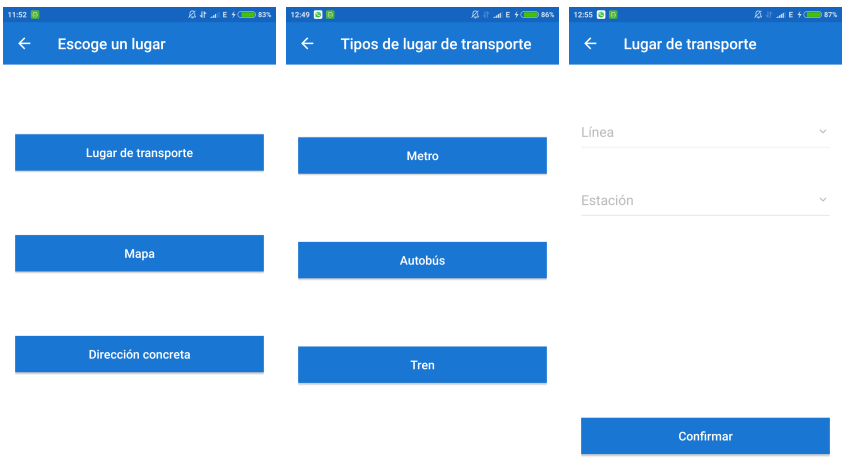
Como se mencionó anteriormente en la sección 5.2.3 "Qué nos interesa", es de vital importancia ofrecer al usuario varias posibilidades a la hora de indicar el lugar donde se ha perdido o encontrado el objeto, y tuvimos esto en cuenta a la hora de diseñar las pantallas de la aplicación para esta funcionalidad en concreto.

También hubo conclusiones erróneas extraídas del análisis realizado en los capítulos anteriores, como la de la sección 5.2.6, "Qué nos interesa" ya que finalmente no cumplimos varias decisiones que tomamos tras el análisis.

Nada más pulsar en el botón flotante de la pantalla principal, se nos brinda la posibilidad de crear un nuevo anuncio, como se explicó en la sección 8.4.2, "Pantalla principal y de creación de un anuncio" más adelante.

Tras esto se le solicita al usuario que indique la información sobre el lugar, para ello se le dan tres opciones distintas. Si el usuario elige la primera de las opciones, que se trata de un lugar de transporte, se le vuelve a solicitar que especifique de qué tipo de transporte se trata, dándole nuevamente tres opciones, como vemos en la segunda captura de la figura 8.9.

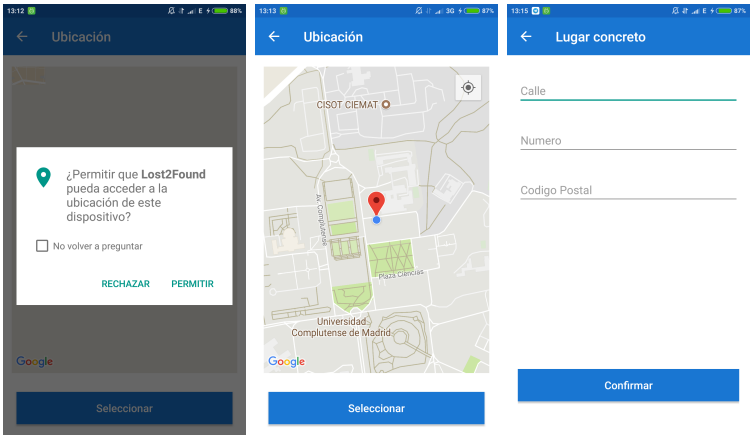
Después de hacer su elección, el usuario deberá indicar en los desplegables en qué línea y estación perdió o encontró el objeto, la información sobre las líneas y estaciones que le ofrece la aplicación al usuario vienen directamente del open data que escogimos en la sección 4.7, "Uso de datos abiertos en nuestra aplicación". La estructura de este open data se explicó detalladamente en el capítulo 6, "*Open Data SNCF* y *Google APIs*".



(a) Tipos de lugar (b) Tipos de lugar de transporte (c) Lugar de transporte especifico

Figura 8.9: Especificar lugar de transporte

En caso de que el usuario escoja la segunda opción de lugar (mapa), se le ofrece la posibilidad de situar el punto exacto donde localizó o perdió el objeto en un mapa como el de la segunda captura de la figura 8.10. Si el dispositivo no tiene activado el GPS, se muestra el mensaje que aparece en 8.10(a).



(a) Solicitud de localización (b) Mapa (c) Especificar lugar concreto

Figura 8.10: Especificar lugar, mapa y dirección concreta

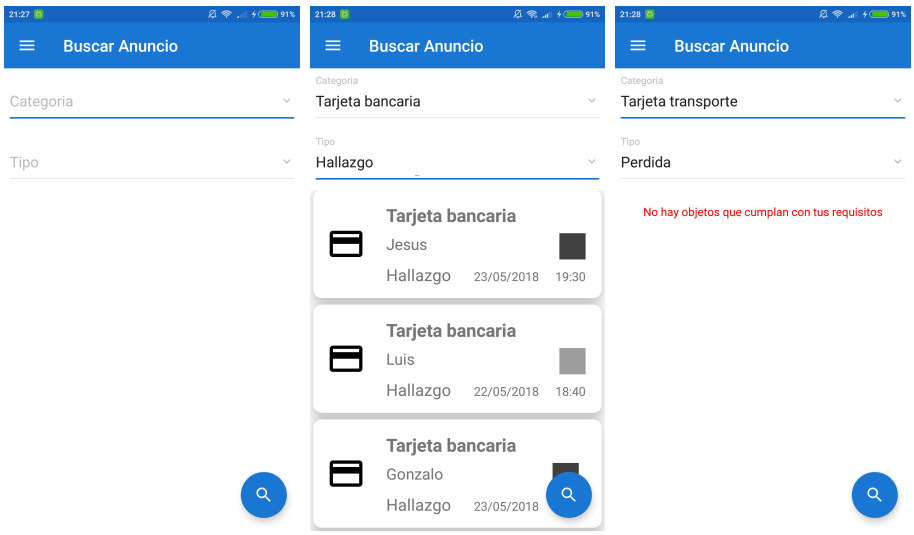
En este caso se guarda en la base de datos la latitud y longitud del punto señalado para su posterior procesamiento a la hora de realizar el match con otro anuncio. También se le ofrece al usuario la posibilidad de localizar su posición pulsando en el botón situado en la parte superior derecha del mapa, en caso de haber concedido los permisos de localización pertinentes. Por último si el usuario escoge la tercera opción de lugar, tendrá que especificar los datos concretos sobre la dirección en cuestión, calle, número y código postal.

8.4.4. Pantallas de búsqueda

El usuario también tiene la posibilidad de buscar entre los anuncios de la aplicación para ver si está publicado alguno que pueda estar relacionado con su objeto, para ello el usuario pulsa en la sección de búsqueda de la barra de navegación desplegable.

Una vez en ella, escoge la categoría y el tipo de anuncio que desea buscar y pulsa en el botón flotante con la lupa. Automáticamente se le muestran al usuario los anuncios de la aplicación que cumplen con los filtros introducidos, y puede pinchar en ellos para obtener más información sobre los mismos.

En caso de que el usuario introduzca filtros que no cumpla ningún objeto de la aplicación se le mostrara un mensaje informándole de la situación, como se ve en la tercera captura de la figura 8.11.



(a) Pantalla de búsqueda (b) Resultados de una búsqueda (c) Mensaje de que no hay resultados

Figura 8.11: Funcionamiento pantalla de búsqueda

8.4.5. Pantallas de match y de chat

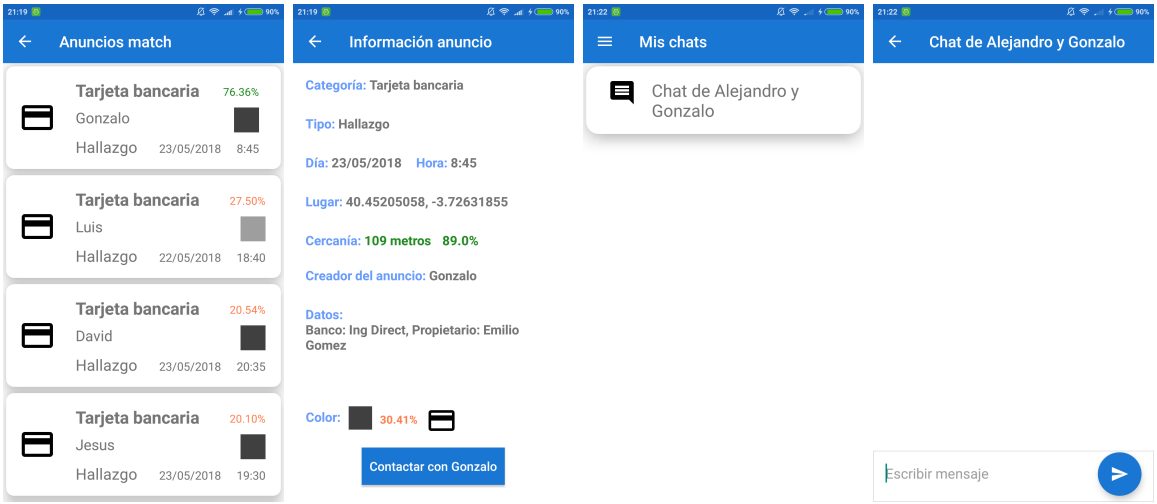
Una vez el usuario ha creado su anuncio, puede buscar 'matches' posibles con otros anuncios que contengan objetos semejantes al que ha encontrado o perdido. Para ello, como hemos visto en la sección anterior, en la pantalla de información concreta de un anuncio dispone de dos botones en la parte inferior de la pantalla, uno para el match local, que busca coincidencias con objetos de anuncios que estén en la aplicación, y otro para el match con open data, que busca entre los objetos publicados en el open data.

En cualquiera de los dos casos, nuestro algoritmo se encarga de calcular el porcentaje de matching entre los pares de anuncios existentes, este proceso se explica detalladamente más adelante en la sección 8.6, "Búsqueda de correspondencia de anuncios compatibles".

Una vez calculados los porcentajes se muestra al usuario una lista con los anuncios coincidentes, ordenada de mayor a menor porcentaje de matching resultante, para que vea primero los anuncios que le puedan resultar más útiles. En caso de no encontrar ningún anuncio con el que se haga matching se le comunica al usuario con un mensaje informativo y se le anima a volver a intentarlo más adelante.

Tras haber localizado el usuario qué anuncio cree que corresponde con su objeto pincha en el mismo y se le ofrece toda la información disponible sobre ese anuncio, además de los porcentajes de matching de la distancia, el porcentaje de matching del color y la distancia existente en metros o kilómetros del lugar del anuncio publicado por el usuario respecto al lugar registrado del anuncio con el que se ha hecho matching.

Para recuperar su objeto el usuario tan solo tiene que pinchar en el botón de contactar y automáticamente se crea un chat entre el usuario y el propietario del anuncio en cuestión.



(a) Lista de anuncios compatibles

(b) Información concreta de un anuncio de match

(c) Nuevo chat creado entre los dos usuarios

(d) Pantalla de chat para comunicarse con el otro usuario

Figura 8.12: Pantalla de creación de un anuncio

8.5. Implementación y cambios en la base de datos

A lo largo del desarrollo del proyecto ha surgido la obligación de realizar modificaciones en la estructura de la base de datos debido a que no se habían tenido en cuenta ciertos aspectos en los diagramas entidad relación o a necesidades durante la programación.

Algunas modificaciones realizadas en la base de datos ya se comentaron en el capítulo anterior, concretamente en la sección "Base de datos", pero aquí se explicará más en detalle la evolución sufrida en la estructura de la base de datos y la última versión de la misma:

- Han desaparecido atributos de algunas entidades al no hacer uso de ellos y no necesitarlos, como el rol del usuario (al no hacer distinciones entre usuarios), el booleano leído en msg, la caducidad y recompensa de un anuncio, el IMEI de un teléfono, etc.

- Las relaciones existentes entre las entidades usuario, chat y msg han cambiado de manera que ahora msg se relaciona directamente con usuario, además de con chat, de esta manera un msg tiene un id propio, un id de usuario y un id de chat, lo que lo hace fácilmente identificable y único.

- La cardinalidad de la relación existente entre las entidades usuario y chat ha cambiado también debido a que apreciamos que un chat podía tener como máximo dos usuarios, en el caso de nuestra aplicación.

Todos estos cambios se pueden apreciar en las figuras 8.13 y 8.14 donde vemos una versión modificada del modelo entidad relación y su correspondiente modelo relacional.

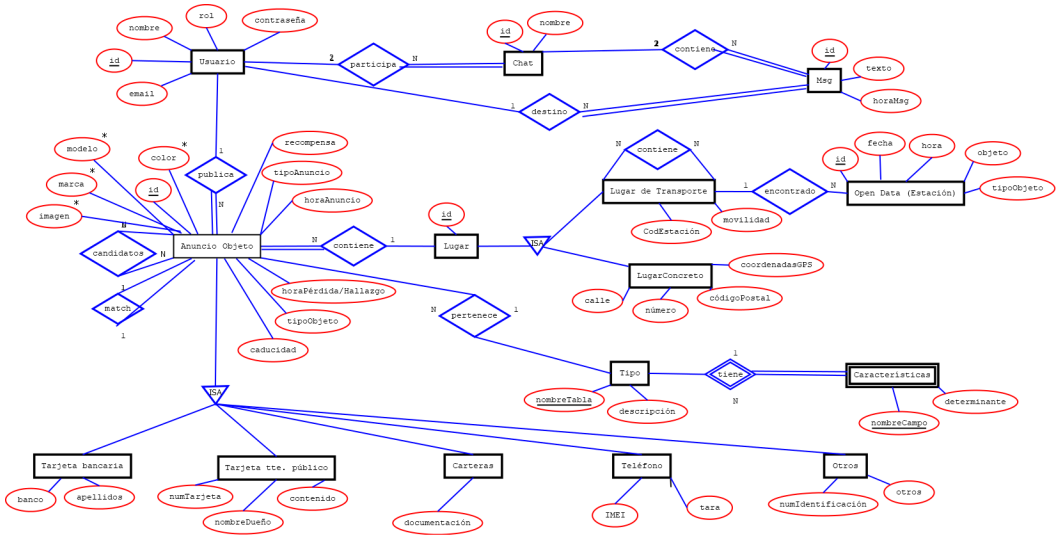


Figura 8.13: Versión modificada del diagrama entidad relación

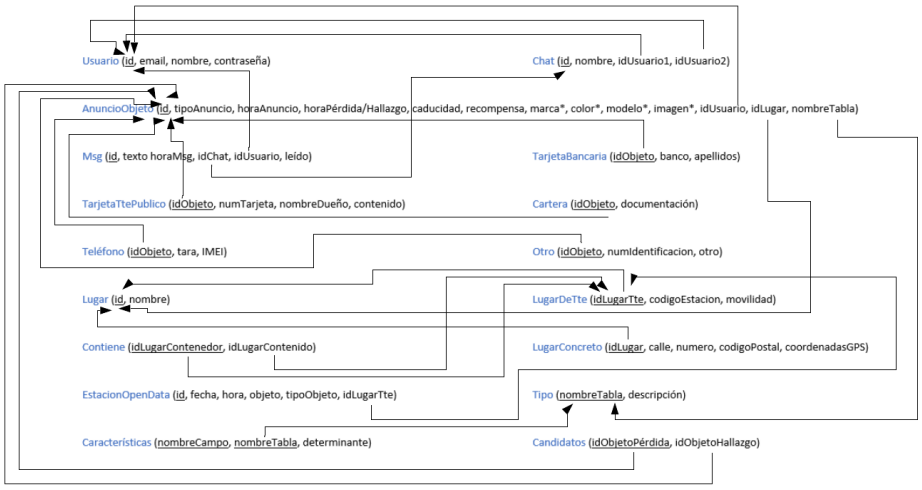


Figura 8.14: Modelo relacional correspondiente al diagrama entidad relación

También hubo entidades que creamos al pasar el modelo entidad relación al modelo relacional que finalmente no han formado parte de la estructura de la base de datos, debido a que el propio diagrama entidad relación sufrió cambios y desaparecieron algunas entidades y relaciones, provocando cambios por consiguiente en el modelo relacional.

Es el caso de entidades como *Estación (open data), contiene, candidatos, matching y característica*. A pesar de la desaparición de estas entidades también hemos tenido que incluir algunas otras a lo largo del desarrollo del código de la aplicación, como la tabla *lugar mapa* o la tabla *conversor*, necesaria a la hora de tratar con los datos del *open data* debido a la inevitable traducción del francés al español, este aspecto se detalló en profundidad en el capítulo "*Open Data SNCF* y *Google APIs*".

Algunos atributos han pasado de una entidad a otra, como en el caso de las coordenadas GPS de la entidad *lugar concreto*, que ahora son parte de la nueva entidad *lugar mapa*, dividiéndose en latitud y longitud.

Esta evolución y todos los cambios mencionados se pueden apreciar en las figuras 8.15 y 8.16, siendo la última la versión final de la base de datos de la aplicación.

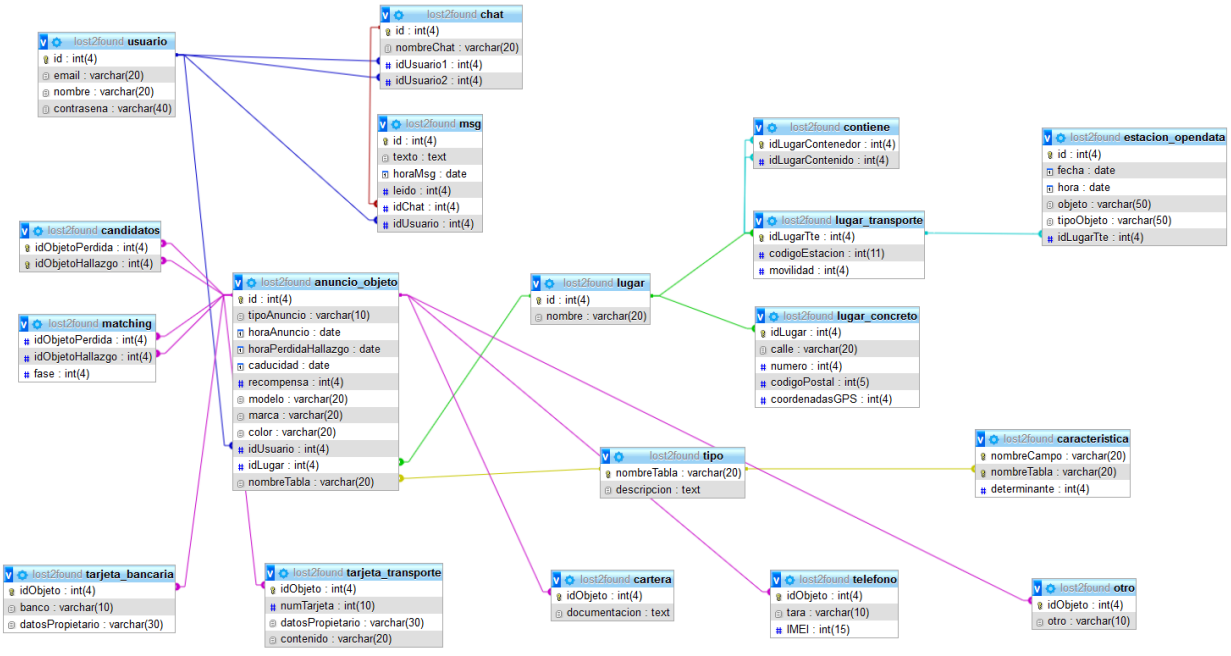


Figura 8.15: Versión antigua de la base de datos de la aplicación

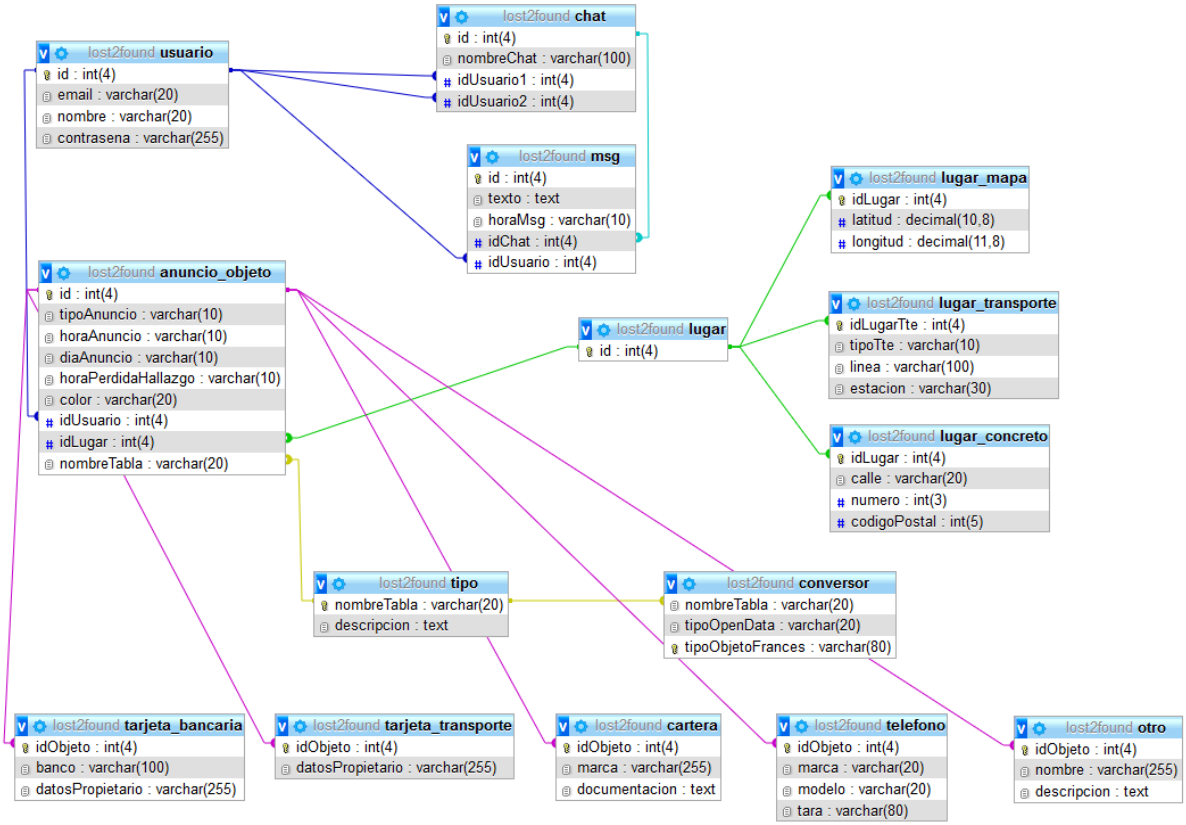


Figura 8.16: Versión final de la base de datos de la aplicación

8.6. Búsqueda de correspondencia de anuncios compatibles

Tal y como se comentó en la sección 7.7 del capítulo anterior, el "Algoritmo de matching" constituye una parte esencial de la aplicación desarrollada en este proyecto, ya que sin él no se podría conseguir su objetivo final, que es devolver los objetos perdidos a sus legítimos dueños.

Además del funcionamiento del algoritmo explicado en la sección mencionada, existen dos tipos de criterios distintos que el algoritmo tiene en cuenta a la hora de su ejecución, a su vez, para cada uno de estos criterios existen también distintos factores:

- Criterios determinantes o excluyentes: Son aquellos que según su coincidencia concluyen o descartan la correspondencia entre dos anuncios, entre ellos se encuentran los siguientes:

- La coincidencia de atributos concretos del objeto en cuestión: Al tener cada uno de los objetos de cada categoría atributos específicos se hace un proceso de comparación entre ellos para ver si coinciden de una manera significativa o distan demasiado para poder tratarse del mismo objeto, son excluyentes atributos como la marca o el nombre del propietario de una tarjeta, por lo cual en caso de no existir semejanza de estos atributos entre dos anuncios no se realiza un emparejamiento y no se incluye el anuncio correspondiente en la lista de match mostrada al buscar coincidencias.
- Que el usuario dueño del anuncio original sea distinto al del anuncio con el que se busca correspondencia: Obviamente entre el listado de los anuncios de match mostrado al usuario no se incluyen aquellos cuyo usuario es el mismo que el del anuncio original, ya que esta situación desde el punto de vista de nuestra aplicación no tiene sentido.
- Que el tipo de anuncio sea el opuesto al original: Si el anuncio original publicado es de *Pérdida* no tiene sentido que se busquen correspondencias con otros anuncios que tengan este mismo tipo, sino con aquellos que tengan el tipo opuesto, *Hallazgo* en este caso y viceversa.

- Criterios aproximados: Son aquellos que en ningún caso descartan correspondencias entre anuncios por sus características, pero contribuyen notablemente en el incremento o decremento del porcentaje de match final, pudiendo de esta manera destacar entre otros pares de anuncios con menor porcentaje, son:

  - El porcentaje de coincidencia de color entre los dos anuncios: Este porcentaje se calcula según la distancia euclidiana [8] existente entre los dos colores, para ello se extrae de cada color sus componentes RGB y se realizan cálculos con el fin de comparar unas componentes con otras para ver la distancia existente entre ambas. Una vez realizado esto y obtenida la distancia, se calcula un porcentaje en base a una máxima distancia posible entre dos colores totalmente opuestos.
  - El porcentaje de distancia entre los lugares de pérdida y hallazgo de los dos anuncios: Para calcular este porcentaje primero se han de conseguir las coordenadas GPS de los dos lugares a comparar, en caso de que el usuario no haya elegido el mapa para localizar el lugar donde perdió o encontró el objeto, para ello se hace uso de la *Geocoding API de Google* [14] que convierte direcciones concretas de cualquier tipo en coordenadas indicadas por una latitud y una longitud, esta API se explicará en profundidad en el capítulo 6, "*Open Data SNCF* y *Google APIs*". Una vez conseguidas las coordenadas necesarias se hace uso de la fórmula del semiverseno [10], que se suele usar para cálculos sobre navegación astronómica y que en nuestro caso nos sirve para hallar la distancia de círculo máximo entre dos puntos de una circunferencia, conociendo su latitud y longitud, de esta manera obtenemos la distancia existente entre dos puntos indicados

por sus coordenadas GPS y por consiguiente entre los dos lugares de los anuncios emparejados. Para calcular el porcentaje a partir de la distancia se hace uso de una fórmula propia que devuelve porcentajes altos cuanto más pequeña es la distancia entre los dos lugares registrados y viceversa. Es importante destacar que en el caso de tratarse de lugares de transporte, lo que se calcula no es la distancia física real entre ambos lugares, sino una distancia virtual, ya que en el caso de que los objetos de ambos anuncios se encuentren en la misma línea, la distancia resultante será 0 metros, a pesar de que los objetos no se encuentren en la misma estación de la línea.

- El número de días que han pasado desde que se publicó la perdida del objeto hasta que se publicó el hallazgo: Se tiene en cuenta este factor ya que hay más posibilidades de que se trate del mismo objeto si han pasado pocos días desde que se denunció su perdida hasta que se publicó su hallazgo.

Además de los criterios y factores explicados anteriormente, es importante resaltar que tal y como se explicó en 7.7, "Algoritmo de matching" existen dos tipos distintos de match, uno local que busca correspondencias entre anuncios creados en la aplicación y otro con open data que tiene en cuenta los objetos publicados en los conjuntos de datos abiertos de *SCNF*.

Cabe destacar que la búsqueda realizada en este último tipo de match se ve forzosamente limitada por los datos que nos proporciona el open data, este tipo de problemas se comentan detalladamente en la sección 6.5, "Inconvenientes surgidos con el proveedor de open data" del capítulo 6, "*Open Data SNCF* y *Google APIs*".

8.7. Diseño e implementación de la estructura de clases

A la hora de ajustar el diseño de la base de datos a nuestro proyecto en Android hemos construido una estructura de clases siguiendo el sistema de mapeo relacional de objetos [23].

De esta manera tenemos una correspondencia existente entre las tablas de nuestra base de datos y los atributos del código orientado a objetos.

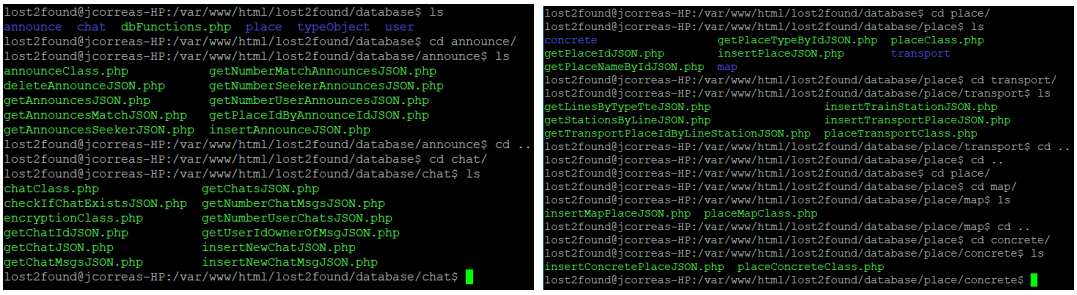
En cuanto a la interfaz de usuario, la estructura de paquetes y clases de nuestro proyecto sigue un patrón común, de tal manera que para cada vista de la aplicación existe un paquete con el nombre en cuestión que contiene todas las clases y el código necesario para la ejecución y el correcto funcionamiento de esa vista.

Entre los servicios *PHP* que brinda nuestro servidor se encuentran:

- *announceClass.php*: Contiene todos los métodos referentes al uso y modificación de la entidad anuncio, así como los encargados de realizar las consultas del match de la aplicación.
- *chatClass.php*: En este caso contiene los métodos encargados de gestionar la entidad chat, así como los mensajes de cada uno de ellos.
- *placeClass.php*: Es la clase principal de la entidad lugar de la aplicación, contiene los métodos de inserción y obtención de lugares de la aplicación.
- *placeTransportClass.php*: Se trata de uno de los tres subtipos de la entidad lugar existentes en la aplicación, contiene los métodos encargados de la trata de lugares de transporte.
- *placeMapClass.php*: Es el segundo de los subtipos de lugar, es la encargada de insertar y obtener los datos obtenidos cuando un usuario pulsa un lugar geográfico en el mapa.
- *placeConcreteClass.php*: El último de los tres subtipos de lugar, al igual que *placeMapClass.php* gestiona la inserción y obtención cuando un usuario indica un lugar de este tipo.
- *typeObjectClass.php*: Incluye todas las funciones para insertar u obtener datos sobre un objeto de cualquier categoría existente en la aplicación.
- *userClass.php*: Contiene los métodos de actualización, inserción y obtención de datos referentes a la entidad usuario.f

En la figura 8.17 podemos observar la mayoría de los archivos *PHP* contenidos en el servidor, en la primera captura observamos que la estructura contiene cinco grandes directorios que representan cada una de las entidades de nuestra aplicación.

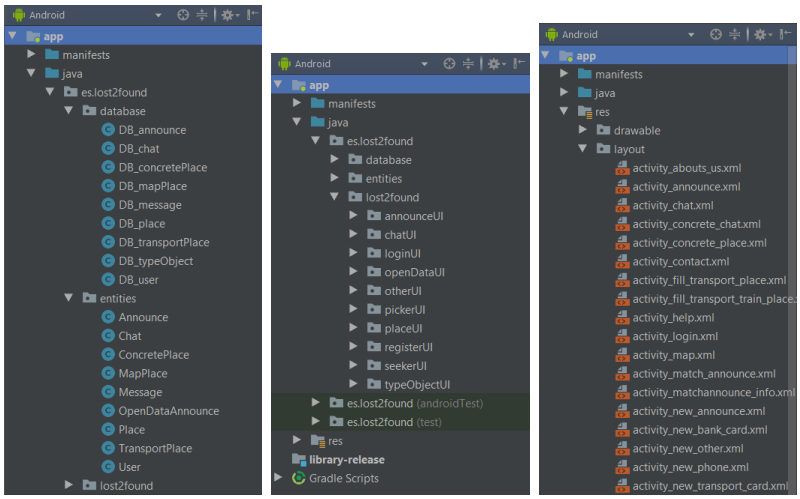
Dentro de cada uno de ellos existen distintos archivos *PHP* encargados de realizar servicios muy concretos, estos realizan llamadas a las funciones contenidas en las clase principal de cada entidad, como por ejemplo *announceClass.php*.



(a) Servicios *PHP* del servidor (b) Servicios *PHP* del servidor

Figura 8.17: Servicios *PHP* del servidor

En la figura 8.18 podemos apreciar la estructura de directorios de nuestro proyecto Android, está dividida en 3 grandes directorios, el primero contiene el archivo *manifest.xml* de la aplicación, en el cual están las actividades de la misma así como números y nombres de versión. Después tenemos el directorio *java* que contiene todas las clases *.java* de la aplicación, como vemos en la primera y segunda captura de la figura 8.18. Esta se subdivide a su vez en otros tres grandes directorios, *database* por un lado, que contiene todas las clases encargadas de tratar con el servidor o con las *API* de los sistemas externos, *entities* por otro lado, donde están las clases que representan las entidades que existen en nuestra aplicación, y por último *lost2found*, el cual se subdivide a su vez en distintos directorios que contienen toda la implementación de la interfaz de usuario de la aplicación. Por último en la tercera captura observamos el directorio *res/layout* que contiene todas las vistas de la aplicación en formato *XML*.



(a) *database* y *entities* desplegados (b) *lost2found* desplegado (c) *res/layout* desplegado

Figura 8.18: Directorios del proyecto Android

8.8. Uso de AsyncTask

A la hora de realizar las conexiones con la base de datos desde el código *Java* mediante *PHP*, es necesario el uso de actividades asíncronas. Esto se debe a que en Android no se permite realizar conexiones con la base de datos en el thread principal, por lo que hemos implementado este tipo de actividades como clases internas privadas dentro de cada una de las clases *Java* en las cuales hemos necesitado realizar una conexión con la base de datos. Este tipo de actividades extienden de *AsyncTask* y tienen tres métodos a implementar, como podemos ver en el ejemplo de la figura 8.19.

```
private class LoginDB extends AsyncTask<String, Void, User> {

    private ProgressDialog dialog = new ProgressDialog(LoginActivity.this);

    @Override
    protected void onPreExecute() {
        this.dialog.setMessage("Entrando, espera");
        this.dialog.show();
    }

    @Override
    protected User doInBackground(String... strings) {
        return DB_user.findUserByEmail(strings[0], strings[1]);
    }

    @Override
    protected void onPostExecute(User result) {
        processLogin(result);
        this.dialog.dismiss();
    }
}
```

Figura 8.19: Implementación de AsyncTask

El primer método *OnPreExecute* se ejecuta antes de realizar la tarea implementada en la actividad, como indica su nombre, en nuestro caso lo usamos para mostrarle al usuario un mensaje en forma de diálogo informándole de que se esta realizando una tarea y que debe esperar.

En segundo lugar, *doInBackground* es donde se llama al método que realiza la conexión con la base de datos, en este caso *findUserByEmail* de la clase *DB_user*. En este tipo de clases se implementan las operaciones que tienen acceso a la base de datos y guardan correspondencia con las ya implementadas en *PHP* en la parte del servidor.

En cada una de ellas existe un atributo privado de tipo *String* que contiene la ruta a la carpeta donde se encuentran los ficheros correspondientes a cada clase, a su vez en cada método de la clase se especifica el nombre del fichero a ejecutar en cada caso, y se crea un objeto JSON al cual se le pasan los parámetros necesarios para realizar la

consulta correspondiente en la base de datos, estableciendo una comunicación mediante el método *POST*, tras esto se lee la respuesta dada por el servidor y se realiza la operación correspondiente en cada caso.

Por último, el método *onPostExecute* es el encargado de realizar las acciones que correspondan con la respuesta dada por el método *doInBackground*, en nuestro caso concreto también lo usamos para ocultar el diálogo que le mostrábamos previamente al usuario, una vez ha terminado de realizarse la tarea correspondiente.

Capítulo 9

Disponibilidad de la aplicación

9.1. Introducción

En este capítulo se explicará la estructura de directorios del código fuente de la aplicación así como la disponibilidad de la versión final de la aplicación Lost2Found desarrollada a lo largo de este proyecto.

9.2. Estructura de directorios

Todo el código fuente de la aplicación esta disponible de forma pública en un repositorio de *GitHub* [15] al que se accede mediante el siguiente enlace:

`https://github.com/DavidZam/Lost2Found`

Entre todo el árbol de directorios y ficheros disponibles caben destacar:

- *Backend servidor*: Contiene una copia de todos los archivos *PHP* alojados en el servidor, de esta manera al publicarlos en el repositorio se encuentran disponibles para cualquier persona que desee consultarlos.

- *app*: Es el directorio principal de la aplicación, contiene la carpeta *src/main* que contiene a su vez por un lado todas las clases *.java* del proyecto en la carpeta *java/es/lost2found* así como todas las vistas existentes de la aplicación en formato *.xml* en la carpeta *res/layout*.

- *LICENSE.md*: Este fichero contiene la licencia del código, que en este caso es *Apache v2.0* [3].

- *README.md*: Por último existe un fichero *readme* que contiene información sobre dónde se puede descargar la aplicación, así como varias capturas de la misma.

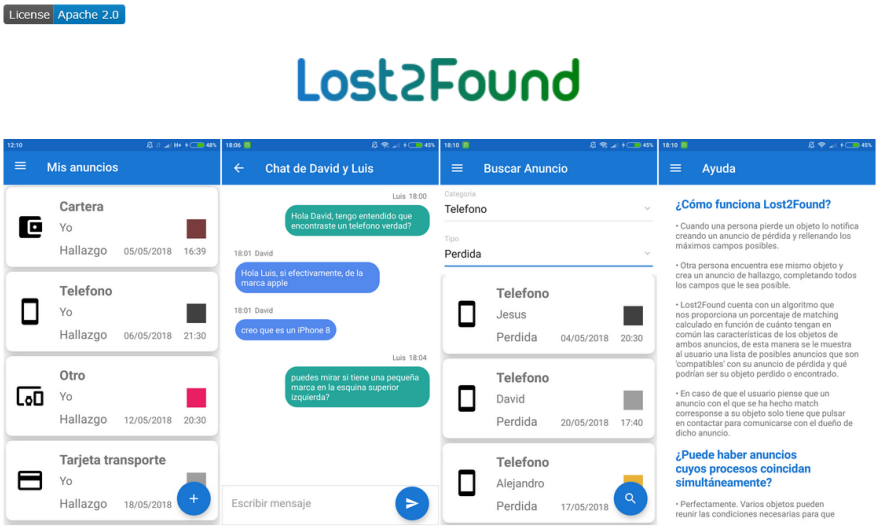


Figura 9.1: *Lost2Found* en el repositorio de *Github*

9.3. Disponibilidad

La aplicación está disponible en *Google Play Store* [17], la hemos subido a la plataforma para facilitar la descarga e instalación por parte de los usuarios, ya que era uno de nuestros objetivos del proyecto. Para ello fue necesario crear una cuenta de desarrollador de *Google* y usar la *Google Play Console* [16] para publicar la aplicación en la tienda de aplicaciones de *Google*. Se puede encontrar buscándola por su nombre, *Lost2Found*, en *Google Play Store* o bien en el siguiente enlace:

`https://play.google.com/store/apps/details?id=es.lost2found`.

Figura 9.2: *Lost2Found* en *Google Play Store*

Capítulo 10

Conclusiones y trabajo futuro

10.1. Conclusiones

Tras finalizar el proyecto, consideramos que nuestro objetivo principal, el cual consistía en desarrollar una aplicación *Android* a modo de ejemplo empírico del potencial y de las ventajas que nos brinda el open data, se ha cumplido.

Además, uno de nuestros objetivos personales con este proyecto consistía en desarrollar una aplicación funcional y ponerla a disposición de los usuarios publicándola en la *Google Play Store*, lo cual hemos logrado.

Cabe destacar la dificultad extra surgida por la falta de conocimientos que teníamos sobre la programación en *Android*, ya que no habíamos programado en este entorno previamente.

Aun así, el aprendizaje de este lenguaje ha sido una grata experiencia con la ayuda de los tutoriales de *Android* [1] que han formado una parte esencial de nuestra instrucción, así como el desarrollo de una aplicación para esta plataforma desde cero.

También es relevante todo lo que hemos aprendido sobre planificación de trabajo, definición de requisitos y todo aquello referente a la gestión de proyectos, ya que al enfrentarnos con un problema real nos hemos visto obligados a tomar las medidas necesarias para resolverlo de la mejor manera posible, consiguiendo así la preparación necesaria para los problemas que se nos presenten en el futuro.

Otro de los aspectos en los cuales hemos adquirido conocimientos es todo lo referente al diseño de la aplicación, hacer interfaces que le resulten atractivas, sencillas y amigables al usuario no es una tarea fácil, y requiere tener en cuenta factores como la disposición de los elementos en las pantallas o el uso de los colores en las mismas.

Uno de los grandes retos del proyecto ha sido llegar a una versión final de la estructura de la base de datos, ya que pasamos meses cambiando constantemente el diseño con intención de conseguir reflejar todos los requisitos del proyecto de manera correcta, hasta que alcanzamos uno cuya implementación encajaba con las necesidades de la aplicación, a pesar de esto hemos tenido que realizar modificaciones de carácter leve en el diseño en algunos puntos del proyecto.

Es relevante mencionar que hemos trabajado a lo largo de todo el proyecto con un repositorio de *GitHub*, lo cual, a pesar de haberlo hecho previamente en alguna ocasión a lo largo de la carrera, nos sirve para conseguir experiencia con este tipo de herramientas cuyo uso hoy en día está en auge en las empresas y que seguramente tengamos que usar en nuestro futuro laboral.

Por último, la tarea de corregir algunos fallos que comprobamos tras tener publicada la aplicación en la *Google Play Store* y divulgar actualizaciones con las correcciones de los mismos nos sitúa en el papel de un desarrollador profesional encargado de asegurar que su aplicación funcione de la manera más eficiente y correcta posible, sirviéndonos de gran experiencia para el futuro.

10.2. Trabajo futuro

A pesar de que el proyecto ha durado un curso entero, lo cual son aproximadamente 9 meses de trabajo, no hemos podido elaborar todos los aspectos del proyecto que nos hubieran gustado, por lo cual quedan como trabajo futuro.

Uno de estos aspectos es la existencia de una versión web de la aplicación, ya que de esta manera se proporcionaría una vía alternativa si el usuario por ejemplo ha perdido su teléfono móvil y no puede acceder a la aplicación para publicar la pérdida o hallazgo de un objeto.

Al principio del proyecto sopesamos esta posibilidad, pero acabamos descartándola debido a que preferíamos desarrollar una versión móvil por su mayor disponibilidad frente a la versión web, y no disponíamos del tiempo suficiente para elaborar las dos versiones.

Otro de los puntos pendientes consiste en ampliar las categorías de objetos incluidas en la aplicación, ya que de esta manera se daría lugar a una mayor precisión que se reflejaría en un mayor acierto por parte de la aplicación.

La razón por la cual incluimos las cinco categorías existentes en la aplicación procede de la información publicada por un medio de comunicación [38] sobre los objetos más frecuentemente perdidos, estos son los cinco tipos de objetos que más se pierden estadísticamente, por lo tanto de esta manera cubríamos la mayor parte de los objetos perdidos o encontrados solamente con unas cuantas categorías.

A pesar de esto, incluir otras categorías de objetos como ordenadores o ropa contribuirían a una mayor precisión del matching, ya que cualquier tipo de objeto que no encaje en las categorías disponibles de la aplicación se considera dentro de la categoría *Otro*, teniendo quizá demasiada variedad dentro de esta última categoría, lo cual puede causar comparaciones entre ordenadores y gafas o prendas de vestir.

También se podrían ampliar los tipos de lugar de transporte de la aplicación, incluyendo por ejemplo aeropuertos al tratarse de un tipo de lugares donde se pierden una gran cantidad de objetos todos los días, de esta manera se estaría cubriendo este tipo de zonas con mayor precisión, si por ejemplo el usuario pudiera especificar en qué terminal o puerta de embarque ha perdido el objeto en cuestión.

En el caso de que se crease un conjunto de datos abiertos relativo a objetos perdidos en nuestro país con un ritmo de actualización y unas dimensiones similares al que hemos utilizado de Francia se podría implementar en nuestra aplicación, consiguiendo así una mayor efectividad al disponer de más objetos con los que poder hacer match.

Por último, también sería deseable que la aplicación estuviera disponible para otros sistemas operativos como *iOS*, cubriendo de esta manera al mayor número posible de usuarios.

Capítulo 11

Conclusions and future work

11.1. Conclusions

After completing the project, we consider that our main objective, which was to develop an application *Android* as an empirical example of the potential and the advantages that open data offers, has been fulfilled.

In addition, one of our personal goals with this project was to develop a functional application and make it available to users by publishing it in the *Google Play Store*, which we have achieved.

Note the extra difficulty arising from the lack of knowledge we had about the programming in *Android*, since we had not programmed in this language previously.

Even so, learning this language has been a pleasant experience with the help of the *Android* [1] tutorials that have formed an essential part of our instruction, as well as the development of an application for this platform from scratch.

It is also relevant everything we have learned about work planning, definition of requirements and everything related to project management, since when faced with a real problem we have been forced to take the necessary measures to solve it in the best possible way , thus obtaining tables for problems that may arise in the future.

Another aspect in which we have acquired knowledge is everything related to the design of the application, making interfaces that are attractive, simple and user friendly is not an easy task, and requires taking into account factors such as the disposition of the elements on the screens or the use of colors in them.

One of the great challenges of the project has been to arrive at a final version of the database structure, since we spent months constantly changing the design with the intention of being able to reflect all the requirements of the project correctly, until we reached one whose implementation fitted with the needs of the application, despite this we had to make minor modifications to the design in some points of the project.

It is important to mention that we have worked throughout the project with a repository of *GitHub*, which, despite having done it previously in some occasion throughout the race, helps us to get experience with this type of tools whose use today is booming in companies and that surely we have to use in our future work.

Finally, the task of correcting some bugs that we check after having published the application in the *Google Play Store* and disclosing updates with the corrections of them places us in the role of a professional developer in charge of ensuring that your application works in the most efficient and correct way possible, making use of great experience for the future.

11.2. Future work

Although the project has lasted a whole course, which is approximately 9 months of work, we have not been able to elaborate all the aspects of the project that we would have liked, for which they remain as future work.

One of these aspects is the existence of a web version of the application, as this would provide an alternative way if the user for example has lost his mobile phone and can not access the application to publish the loss or finding of a object. At the beginning of the project we weighed this possibility, but we ended up discarding it because we preferred to develop a mobile version due to its greater availability compared to the web version, and we did not have enough time to elaborate the two versions.

Another pending point is to expand the categories of objects included in the application, as this would lead to greater precision that would be reflected in greater success on the part of the application. The reason why we included the five existing categories in the application was that according to a study carried out they were the type of objects that were lost statistically, therefore in this way we covered most of the lost or found objects with only a few categories.

In spite of this, including other categories of objects such as computers or clothes would contribute to a greater accuracy of matching, since any type of object that does not fit in the available categories of the application is considered within the category *Other*, perhaps having too much variety within this latter category, which can cause comparisons between computers and glasses or clothing. It could also expand the types of place of transport of the application, including for example airports as it is a type of places where a large number of objects are lost every day, this way it would be covering this type of areas with greater precision , if for example the user could specify in which terminal or boarding gate has lost the object in question.

In the case that an open data set was created relative to lost objects in our country with a rate of update and dimensions similar to the one we have used in France, it could be implemented in our application, thus achieving greater effectiveness by having more objects with which to be able to match.

Finally, it would also be desirable for the application to be available for other operating systems such as *iOS*, thus covering as many users as possible.

Capítulo 12

Aportaciones individuales

Al empezar con el proyecto, mi compañera y yo acordamos que ambos íbamos a realizar las tareas del mismo por igual, por lo que las aportaciones individuales de ambos son iguales.

12.1. Carolina Rivero Fernández

Antes de detallar cuáles han sido mis aportaciones, creo necesario resaltar que este trabajo ha sido planteado desde cero, siendo un año de aprendizaje constante por parte de los dos.

No teníamos muy claro por dónde comenzar exactamente y, gracias a nuestro director, decidimos centrar la investigación en las distintas redes de colaboración existentes, la gran cantidad de portales de open data efectivos en todo el mundo, las tecnologías que querríamos usar e incluso, algunas de ellas, aprender sobre la marcha con miras de futuro tanto profesional como personal.

Mientras mi compañero se dedicaba a las redes de colaboración, yo decidí indagar en open data, resultando algo exasperante la búsqueda de un portal apto para nuestro proyecto por la escasez de los mismos en español, en cantidad de datos y en actualizaciones.

Finalmente, nos decantamos por el portal de datos abiertos de Francia que, aunque todos sus datos estaban en francés, la riqueza y cantidad de los mismos era inmensa.

Después, ambos centramos la investigación en aplicaciones parecidas a la nuestra, haciendo un estudio de las mismas para ver qué aportaciones nuevas y originales tendría *Lost2Found*.

Encontramos cinco aplicaciones en *Google Play Store* y nos las dividimos, siendo las mías *Lost or Found*, *Find My Lost* y *Lost-Tag*. En todo momento nos teníamos informados mutuamente para comparar nuestro trabajo, corregirlo y mejorarlo si era necesario.

Para continuar, se realizaron los diseños de algunas pantallas con ayuda de *Just In Mind* y, en algunos casos, con lápiz y papel, de las cuales hice la búsqueda, la creación de nuevo objeto y los diferentes flujos entre estas pantallas principales.

Mientras, estábamos también con el diagrama entidad relación y, en mi caso, aportando sugerencias, ideas y realizando cambios al primer modelo realizado por mi compañero.

Además, hice una primera tabla con ejemplos y resultados para representar el algoritmo de match que íbamos a implementar. Acabado esto, creamos el archivo *SQL* para, finalmente, importarlo a *phpMyAdmin*.

Antes de ponernos a programar, yo me encargué de buscar y realizar los diferentes procesos que tendría nuestra aplicación y, junto a David, valorar qué herramientas nos serían útiles para su realización. Además, cada uno por su cuenta comenzó a familiarizarse con *Android*, su entorno, lenguaje y funcionamiento, haciendo uso de cursos online y *YouTube*.

Hubo problemas con el servidor dado por nuestro director y, mientras mi compañero intentaba configurarlo adecuadamente, yo realicé una versión del login y del registro de la aplicación usando *XAMPP*, la cual se tuvo que modificar para hacerla funcionar con el servidor mencionado antes.

Viendo que la conexión iba bien, nos dividimos las tareas de programación y, durante los meses de febrero, marzo, abril y hasta mediados de mayo, tanto mi compañero como yo nos hemos dedicado de lleno a la realización del código. He desarrollado consultas de inserción y actualización. Además, fui la encargada de diseñar y programar el algoritmo de match.

Para empezar, me puse como ejemplos varios objetos perdidos y encontrados, algunos coincidentes y otros no, y realicé una tabla con todos los atributos de cada uno, poniéndoles distintos valores entre 0 % y 100 % según correspondía. Siempre tuve en cuenta que los usuarios no estaban atados a rellenar los campos de algunos atributos del objeto que insertaban en la aplicación.

Con todo esto delante, y gracias a ello, llegué a la conclusión de que los atributos, dando igual el objeto al que perteneciesen, se podían dividir en dos grupos: determinantes/excluyentes y contribuyentes al porcentaje final. A partir de ahí, comencé a elaborar todo lo necesario en papel y, cuando tuve las ideas claras para enlazar las múltiples consultas resultantes, las programé.

Por otra parte, y a la vez que programábamos, hemos realizado algunas variaciones en la base de datos por necesidades, problemas y cambios que efectuamos.

Mi compañero y yo teníamos un objetivo, entre muchos, subir nuestra aplicación a *Google Play Store*. Así que, terminada la aplicación, cumplimos este propósito.

Para terminar esta sección, a lo largo de este año he participado en la redacción de esta memoria y realizado los cambios oportunos sugeridos po nuestro director para la mejora de la misma.

12.2. David Zamora Rey

Tras reunirnos en varias ocasiones para concretar nuestros objetivos y los requisitos del proyecto, acordamos realizar una primera investigación sobre los conceptos fundamentales que queríamos implementar en nuestra aplicación, por lo que participé en el estudio buscando artículos que tratasen sobre redes de cooperación u open data, leyéndolos y asimilando su contenido.

A la par, busqué junto a mi compañera las aplicaciones que se encontraban en *Google Play Store* que tuviesen algo que ver con objetos perdidos y participé en el análisis de la competencia que realizamos con el fin de extraer características interesantes y de distinguirnos de las aplicaciones ya existentes en este ámbito.

Una vez realizado el análisis nos repartimos las cinco aplicaciones analizadas y redactamos el estudio elaborado entre los dos, yo me dediqué a las aplicaciones *MissingX* y *Find it - Lost and Found*, tras acabar comprobamos el trabajo escrito por el otro para asegurarnos de que era correcto.

Después de documentarnos adecuadamente sobre los conceptos de redes de cooperación y open data, comenzamos a diseñar las pantallas más esenciales de nuestra aplicación, para ello usamos la herramienta *Just In Mind*, y al igual que hicimos con el análisis de la competencia, nos repartimos las pantallas a diseñar, realizando yo las referentes al login, registro y pantalla principal del usuario.

Además, para el diseño de estas pantallas nos preocupamos de respetar las pautas marcadas por la guía de diseño de *Google Material Design*, con la intención de que le resultasen atractivas e intuitivas al usuario, así como de escoger una paleta de colores adecuada.

Al mismo tiempo que realizábamos el diseño de las interfaces nos ocupamos de representar adecuadamente los requisitos del proyecto en los diagramas entidad relación y relacionales de la base de datos, para esto, realicé un primer diagrama entidad relación y a partir de este fuimos realizando cambios, hasta llegar al correspondiente diseño final, yo fui el encargado de realizar el diagrama relacional correspondiente y después junto a mi compañera volcamos los diseños en un archivo *SQL*.

Mientras conseguíamos el diseño final de la estructura de la base de datos y algunas interfaces de la aplicación, realizamos un curso gratuito online de *Android* para ir adquiriendo los conocimientos que nos iban a hacer falta en un futuro, además de realizar los tutoriales de la guía para desarrolladores de la página de *Android developers*, lo cual nos fue de gran ayuda cuando llegó la hora de programar la aplicación a mediados del proyecto.

Después de esto, nos enfrentamos con el problema de conectarnos al servidor que nos prestó nuestro director del proyecto para empezar a gestionar nuestra base de datos, para

ello, yo me encargué de configurar los aspectos del servidor *Ubuntu* que nos iban a hacer falta, como *MySQL* o *phpMyAdmin*, y la correcta configuración de los mismos.

Una vez configurado el servidor mi compañera y yo nos pusimos a implementar los archivos *PHP* encargados de realizar la correcta conexión con la base de datos, así como de realizar las consultas a la misma.

Para comprobar el correcto funcionamiento de la conexión al servidor y su configuración una vez habíamos terminado de implementar los archivos del mismo, comenzamos a programar en *Android* las primeras funciones *Java* y a implementar, fijándonos en nuestros primeros diseños realizados, los primeros layouts de la aplicación en *XML*.

Respecto a la realización de estas tareas, nos repartimos el trabajo entre mi compañera y yo y, tras una semana de tiempo aproximadamente, conseguimos que el login y el registro de la aplicación funcionaran de manera correcta.

Al mismo tiempo que realizábamos esta tarea yo me ocupe de diseñar un logo y un icono adecuados para la aplicación y para la paleta de colores que habíamos escogido previamente.

En este punto del proyecto disponíamos de una versión estática de la aplicación con las primeras pantallas diseñadas y funcionando, las semanas siguientes seguimos implementando funcionalidades de la aplicación y realizando cambios en el diseño de la base de datos, debidos a las necesidades que surgieron mientras implementábamos el código.

Así, durante los meses de febrero, marzo, abril y mayo nos dedicamos completamente a la implementación del código de la aplicación realizando cambios y subiéndolos a un repositorio de GitHub del proyecto que teníamos mi compañera y yo.

Cuando terminamos de implementar la aplicación, decidimos subirla a *Google Play Store*, lo cual era uno de nuestros objetivos del proyecto. Para ello, yo me registré como desarrollador de *Google* con un correo y pagamos la cuota de 25 dólares al año.

Una vez hecho esto, generé el archivo *APK* firmado digitalmente con una clave mediante el IDE *Android Studio*, y con el archivo cree una nueva versión de la aplicación y su ficha correspondiente en *Google Play Store*.

También ha sido necesario tras la publicación de la aplicación lanzar algunas actualizaciones con correcciones de bugs existentes en la primera versión publicada, al encargarme yo de la tarea, tuve que repetir el proceso de generar el *APK* y crear una nueva versión de la aplicación, para después subirla a producción y ponerla a disposición de los usuarios.

Por último, durante todo este proceso descrito en los párrafos anteriores participé en la redacción de la memoria y de las correcciones que nos propuso nuestro director durante el proyecto.

Bibliografía

[1] *Android Developers.* (2008). URL: https://developer.android.com/guide/?hl=es-419.

[2] *Apache, Apache Software Foundation.* (1995). URL: https://www.apache.org/.

[3] *Apache License, Apache Software Foundation.* (2004). URL: https://www.apache.org/licenses/LICENSE-2.0.html.

[4] *API, SNCF.* (2013). URL: https://data.sncf.com/api/v1/documentation.

[5] *Déclarations de pertes, SNCF.* (2015). URL: https://data.sncf.com/explore/dataset/objets-trouves-gares/?sort=date.

[6] *data.gov, U.S government.* (2009). URL: https://www.data.gov/.

[7] *Distance Matrix API, Google.* (2016). URL: https://developers.google.com/maps/documentation/distance-matrix/start?hl=es-419.

[8] *Distancia euclidiana, Wikipedia.* (2018). URL: https://en.wikipedia.org/wiki/Color_difference.

[9] *Fórmula del semiverseno, StackOverFlow.* (2017). URL: https://stackoverflow.com/questions/1502590/calculate-distance-between-two-points-in-google-maps-v3.

[10] *Fórmula del semiverseno, Wikipedia.* (2017). URL: https://es.wikipedia.org/wiki/FÃşrmula_del_haversine.

[11] *Find It - Lost and Found.* (2013). URL: https://play.google.com/store/apps/details?id=alcanzar.com.findit.

[12] *Formatos disponibles de open data.* (2009). URL: http://datos.gob.es/es/dashboard.

[13] *Geocoding API, Google.* (2016). URL: https://developers.google.com/maps/documentation/geocoding/intro?hl=es-419.

[14] *Geocoding API, Google.* (2017). URL: https://developers.google.com/maps/documentation/geocoding/intro?hl=es-419.

[15] *GitHub, Chris Wanstrath.* (2008). URL: https://github.com/.

[16] *Google Play Developer Console, Google.* (2016). URL: https://developer.android.com/distribute/console/.

[17] *Google Play Store, Google.* (2008). URL: https://play.google.com/store?hl=es.

[18] *Inmuebles de alquiler Aragón.* (2011). URL: https://play.google.com/store/apps/details?id=es.plaa.android.

[19] *JSON, Douglas Crockford.* (2001). URL: https://json.org/json-es.html.

[20] *Just In Mind.* (2015). URL: https://www.justinmind.com/.

[21] *Lost Or Found.* (2013). URL: https://play.google.com/store/apps/details?id=com.teckmapx.lrf.activity.

[22] *Lost-Tag.* (2012). URL: https://play.google.com/store/apps/details?id=ch.aaap.losttag.

[23] *Mapeo relacional de objetos, Wikipedia.* (2017). URL: https://es.wikipedia.org/wiki/Mapeo_objeto-relacional.

[24] *Material Design, Google.* (2014). URL: https://www.material.io/.

[25] *MissingX.* (2012). URL: https://play.google.com/store/apps/details?id=com.missingx.

[26] *Moovit.* (2014). URL: https://play.google.com/store/apps/details?id=com.tranzmate.

[27] *MySQL, Oracle Corporation.* (1995). URL: https://www.mysql.com/.

[28] *Objets trouvés, SNCF.* (2015). URL: https://data.sncf.com/explore/dataset/objets-trouves-restitution/?sort=date.

[29] *Página web del ayuntamiento de Madrid.* (2006). URL: http://datos.madrid.es/.

[30] *Patea la Palma.* (2015). URL: https://github.com/pacomf/SenderosLaPalma.

[31] *Pharmaraba.* (2010). URL: https://play.google.com/store/apps/details?id=com.linsms.pharmaraba.

[32] *PHP.* (2018). URL: https://es.wikipedia.org/wiki/PHP.

[33] *Real Time Rome.* (2009). URL: http://senseable.mit.edu/wikicity/d.

[34] *Reutilización de la información del sector público, gobierno de España.* (2009). URL: https://administracionelectronica.gob.es/pae_Home/pae_Estrategias/pae_Gobierno_Abierto_Inicio/pae_Reutilizacion_de_la_informacion_en_el_sector_publico.html.

[35] *SmartAppCity.* (2015). URL: https://www.esmartcity.es/comunicaciones/smartappcity-app-para-ciudades-inteligentes.

[36] *Société nationale des chemins de fer français, SNCF.* (2010). URL: https://www.sncf.com/fr.

[37] *StackOverFlow.* (2008). URL: https://stackoverflow.com/.

[38] *Telemadrid, Noticias Madrid.* (2017). URL: http://www.telemadrid.es/noticias/madrid/noticia/metro-crea-una-oficina-de-gestion-de-objetos-perdidos-en-plaza-castilla.

[39] *Ubuntu, Canonical Ltd.* (2004). URL: https://www.ubuntu.com/.

[40] *Wikipedia.* (2001). URL: https://es.wikipedia.org/.