Diseño y desarrollo de una aplicación web para compartir piso

Memoria de Proyecto Final de Grado/Máster

Máster de Aplicaciones Multimedia.

Diseño y Desarrollo de SmartContent

Itinerario profesional

Autor: Carlota Lázaro Hernández

Consultor: Sergio Schvarstein Liuboschetz

Profesores: David García Solórzano y Laura Porta Simó

Fecha de entrega: 08/01/2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Diseño y desarrollo de una aplicación web para compartir piso*
Nombre del autor:	*Carlota Lázaro Hernández*
Nombre del consultor/a:	*Sergio Schvarstein Liuboschetz*
Nombre del PRA:	*David García Solórzano* *Laura Porta Simó*
Fecha de entrega (mm/aaaa):	01/2018
Titulación::	*Máster Universitario de Aplicaciones Multimedia: Diseño y Desarrollo de SmartContent*
Área del Trabajo Final:	*Trabajo de Fin de Máster*
Idioma del trabajo:	*Castellano*
Palabras clave	*Aplicación web, Compartir piso, proyecto*

Resumen del Trabajo (máximo 250 palabras):

La temática sobre la que se basa el proyecto es la compartición de pisos. Actualmente, compartir piso es una opción por la que muchos se decantan, desde estudiantes que se mudan de ciudad por motivos de estudio o trabajo, hasta personas de un rango de edad mayor que deben alquilar una o varias habitaciones de su vivienda para poder subsistir.

Según las estadísticas elaboradas por *idealista*, la demanda de las habitaciones en alquiler se ha incrementado un 80% en un sólo año. Concretamente, el incremento ha sido de un 78,1% pasando de las 33.065.748 búsquedas realizadas en los 6 primeros meses del año 2016 a unas 58.900.927 búsquedas durante el mismo periodo del año 2017.

Por ello, muchas personas hacen uso de diversos aplicativos para facilitar la tarea de o bien búsqueda de un piso, o hacer uso de ellos para alquilar su vivienda.

El presente trabajo tiene como objetivo la conceptualización de la aplicación, el diseño de su arquitectura y su implementación a través de herramientas web con el fin de diseñar y desarrollar una aplicación web en la que los usuarios puedan acceder a través de una cuenta de usuario y gestionen información sobre el alquiler y compartición de un piso/habitación.

Abstract (in English, 250 words or less):

The topic of this project consists of sharing apartment. Nowadays, share an apartment is an option which some people opt for, from students who move from their original cities due to their studies or work, to elderly people who have to rent one or some rooms for surviving.

According to the statistics carried out by *idealista*, sharing apartment demand has been increased to 80% during the last year. In particular, the growth of the demand has been a 78,1% passing from 33.065.748 searches in the 2016's first six months to 58.900.927 searches in the same period of the 2017s.

For this reason, a great amount of people use some applications for the purpose of looking for an apartment or renting a room of their house.

The aim of the present project has been to do the conceptualization of the application, its architecture design and its implementation through the web tools in order to design and develop a web application that the users can access to it using an account and manage their information about renting.

Agradecimientos

A todas aquellas personas que han hecho posible que este proyecto haya salido a la luz.

A ti, que lo estás leyendo.

Abstract

The topic of this project consists of sharing apartment. Nowadays, share an apartment is an option which some people opt for, from students who move from their original cities due to their studies or work, to elderly people who have to rent one or some rooms for surviving.

According to the statistics carried out by *idealista*, sharing apartment demand has been increased to 80% during the last year. In particular, the growth of the demand has been a 78,1% passing from 33.065.748 searches in the 2016's first six months to 58.900.927 searches in the same period of the 2017s.

For this reason, a great amount of people use some applications for the purpose of looking for an apartment or renting a room of their house.

The aim of the present project has been to do the conceptualization of the application, its architecture design and its implementation through the web tools in order to design and develop a web application that the users can access to it using an account and manage their information about renting.

Resumen

La temática sobre la que se basa el proyecto es la compartición de pisos. Actualmente, compartir piso es una opción por la que muchos se decantan, desde estudiantes que se mudan de ciudad por motivos de estudio o trabajo, hasta personas de un rango de edad mayor que deben alquilar una o varias habitaciones de su vivienda para poder subsistir.

Según las estadísticas elaboradas por *idealista*, la demanda de las habitaciones en alquiler se ha incrementado un 80% en un sólo año. Concretamente, el incremento ha sido de un 78,1% pasando de las 33.065.748 búsquedas realizadas en los 6 primeros meses del año 2016 a unas 58.900.927 búsquedas durante el mismo periodo del año 2017.

Por ello, muchas personas hacen uso de diversos aplicativos para facilitar la tarea de o bien búsqueda de un piso, o hacer uso de ellos para alquilar su vivienda.

El presente trabajo tiene como objetivo la conceptualización de la aplicación, el diseño de su arquitectura y su implementación a través de herramientas web con el fin de diseñar y desarrollar una aplicación web en la que los usuarios puedan acceder a través de una cuenta de usuario y gestionen información sobre el alquiler y compartición de un piso/habitación.

Palabras clave

Aplicación web, Implementación, Diseño, Compartir piso, Alquiler piso, Vivienda, Proyecto

Índice

Figuras y tablas

Índice de figuras

Índice de tablas

Capítulo 1: Introducción

1.Introducción

En la actualidad la temática de compartir piso está en auge. La demanda de alquiler de un piso o habitación ha experimentado un crecimiento en los últimos años ya que el rango de edad que comparte y/o alquila no se encuentra solamente entre la juventud, sino en todas las edades, desde divorciados, jóvenes con sus primeros trabajos o personas solteras con empleo.

Una de las posibles causas de este fenómeno es que alquilar es o se encuentra cerca de ser un lujo para la mayoría de los ciudadanos, sobre todo en las dos grandes capitales del país, lugar dónde están llegando aquellos jóvenes de provincias que buscan un puesto de trabajo con una perspectiva laboral un poco menos desalentadora. Sin embargo, se encuentran con la situación de que alquilar una vivienda entera se encuentra, en la mayoría de los casos, fuera de sus posibilidades ya que, en Madrid, un piso de 80 metros cuesta unos 1.029 euros al mes y en Barcelona, 1.192 euros mensuales.

2. Descripción/Definición

El proyecto surge debido al interés que hay en la actualidad sobre la búsqueda de un piso que compartir sin importar la edad de aquel o aquella que lo busca.

A través del estudio y desarrollo del proyecto, se busca el desarrollo de una aplicación web que permita a aquellos usuarios que busquen un alquiler en cualquier ciudad del país obtener resultados en función que aquellas características que se tengan en mente. Además, se pretende que la aplicación sea de uso sencillo e intuitivo para que cualquier usuario pueda acceder y compartir información sobre un alquiler.

El proyecto se comenzará a desarrollar desde cero y será publicado en un servidor desde el cual sea posible visualizarlo.

Para la implementación del proyecto, se cuenta de antemano con conocimientos de desarrollo web siendo PHP el lenguaje por parte del servidor elegido junto con la base de datos de tipo MySQL y ficheros de extensión JSON para el envío de los datos del servidor al cliente. Por otra parte, en lo que respecta a la parte del cliente, las tecnologías que se utilizan son HTML5 como lenguaje de marcado para construir la estructura de la aplicación, CSS3 como lenguaje para conseguir mejorar la estética de la web en lo que al diseño se refiere, BootStrap como framework de diseño web y JavaScript para mostrar el contenido de los JSON además de aplicar algún efecto. Cabe destacar que se utilizará la librería jQuery de JavaScript para facilitar la tarea.

El desarrollo del proyecto con las tecnologías anteriormente nombradas permitirá mejorar y aprender nuevos conceptos y adquirir nuevos conocimientos de cara a las salidas profesionales ya que, en el mercado actual, las empresas demandan perfiles de desarrolladores tanto de backend como de frontend (conocidos como full-stack) que conozcan las tecnologías anteriormente nombradas.

3. Objetivos generales

3.1 Objetivos principales

Los objetivos principales del proyecto son los siguientes:

- Conceptualización del proyecto: definir el proyecto con sus requisitos funcionales y requisitos que podrán ser implementados en un futuro con el fin de mejorar la aplicación web.
- Diseño de la arquitectura web que consta de los siguientes puntos:
    - Diseño de la interfaz de usuario a través de wireframes
    - Diseño e implementación de la lógica de la aplicación
    - Diseño de la arquitectura de la información

Objetivos para el cliente/usuario:

- Interfaz de usuario usable que conste de un menú superior por el que navegar en la aplicación sin problemas.
- Registro y acceso sencillo sin necesidad de solicitar datos personales relevantes.

Objetivos personales del autor del TF:

- Adquirir nuevos conceptos y mayor soltura en el desarrollo web, tanto en el frontend como en el backend.
- Conocer nuevas formas de implementar el protocolo OAuth2 con diversas redes sociales.

4. Metodología y proceso de trabajo

La metodología de trabajo para el proyecto se basó en el desarrollo de un producto nuevo desde cero que siguió un desarrollo libre en cuanto a interfaz y a implementación de código. Por lo tanto, no hubieron versiones del producto desde un primer momento, simplemente se desarrolló y dentro del desarrollo se iban proponiendo cambios y características que mejorasen el producto final.

El proceso de trabajo se ha basado en la planificación que se propuso en las primeras PEC de la asignatura sin embargo, en la entrega de la PEC 4, se introdujeron una serie de correcciones que mejoran la aplicación web.

Principalmente, se llevó a cabo un estudio sobre aquellos ciudadanos que compartían pisos. Dicho estudio se basó en las edades de los usuarios, las ciudades más demandadas del país donde se realizaban los alquileres, los precios de los mismos y las características que más ofrecían aquellos usuarios que arrendaban algún alquiler. Además, se comprobó el funcionamiento de aplicaciones similares a la propuesta con el fin de recabar información de las interfaces, de los pros y de los contras que presentaban dichas aplicaciones.

Una vez analizado el mercado y la situación actual, se procedió a diseñar los wireframes que podrían representar las pantallas de la interfaz de la aplicación tanto para los dispositivos móviles como tablets y ordenadores de sobremesa partiendo siempre del principio de consistencia entre los diseños. Seguidamente, se procedió al diseño relacional de la base de datos donde se tuvo en cuenta que los usuarios podrían loguearse a través del método convencional o a través de redes sociales. Una vez desarrollada la base de datos, se procedió a instalar todas las tecnologías necesarias que serán descritas más adelante con las que se implementó la aplicación web. Finalmente, se desplegó la aplicación en un servidor remoto y se llevaron a cabo los cambios necesarios para que todas las funcionalidades posibles funcionasen correctamente.

5. Planificación

La planificación del trabajo se representa en la Tabla 1:

Nombre	Duración	Inicio	Final
PEC 2	13 días	10/10/2017	23/10/2017
Estado del arte	11 días	10/10/2017	21/10/2017
Definición de objetivos	2 días	18/10/2017	20/10/2017
Definición del alcance	2 días	21/10/2017	23/10/2017
PEC 3	28 días	24/10/2017	20/11/2017
Conceptualización del producto	2 días	24/10/2017	25/10/2017
Diseño de wireframes	3 días	26/10/2017	28/10/2017
Diseño de la arquitectura web	2 días	29/10/2017	30/11/2017
Diseño relacional de la Base de Datos	1 día	31/11/2017	31/11/2017
Diagrama de flujo	1 día	01/11/2017	01/11/2017
Descarga de software y creación de la base de datos	2 días	02/11/2017	03/11/2017
Interfaz del login	1 día	04/11/2017	04/11/2017
Registro y login con Google	4 días	05/11/2017	08/11/2017
Registro y login con Facebook	4 días	09/11/2017	12/11/2017
Registro y login con usuario y contraseña	2 días	13/11/2017	14/11/2017
Interfaz e inserción de un nuevo piso en la base de datos	3 días	15/11/2017	17/11/2017
Búsqueda de errores y testeo	3 días	18/11/2017	20/11/2017
PEC 4	28 días	21/11/2017	18/12/2017
Interfaz e implementación para mostrar todos los pisos	7 días	21/11/2017	27/11/2017
Interfaz e implementación para realizar filtros	8 días	28/11/2017	05/12/2017
Interfaz e implementación del perfil de usuario	7 días	06/12/2017	12/12/2017
Mejoras de código	3 días	13/12/2017	15/12/2017

Testeo y búsqueda de errores	3 días	16/12/2017	18/12/2017
PEC 5	21 días	19/12/2017	08/01/2018
Redacción de documentos	16 días	1912/2017	03/01/2018
Realización de la presentación	5 días	04/01/2018	08/01/2018

Tabla 1: Planificación del proyecto

En cuanto a los hitos del proyecto, la Tabla 2 refleja los mismos:

Nombre	Duración	Inicio	Final
PEC 1	15 días	25/09/2017	09/10/2017
PEC 2	13 días	10/10/2017	23/10/2017
PEC 3	28 días	24/10/2017	20/11/2017
PEC 4	28 días	21/11/2017	18/12/2017
PEC 5	21 días	19/12/2017	08/01/2018

Tabla 2: Hitos del proyecto

En este caso, los hitos del proyecto coinciden con las entregas de las PEC de la asignatura Trabajo de Fin de Máster propuestas. Cabe destacar que la entrega de la PEC 4 ha supuesto una fecha importante ya que a partir de dónde se han llevado a cabo las correcciones finales y la PEC5 debido a que es la entrega final del proyecto. La Tabla 3 muestra la nueva planificación:

Nombre	Duración	Inicio	Final
Creación de las presentaciones	3 días	19/12/2017	21/12/2017
Correcciones del proyecto	8 días	22/12/2017	29/01/2017
Redacción de los documentos	10 días	30/12/2017	08/01/2018

Tabla 3. Nueva planificación

6. Presupuesto

A continuación se presenta una propuesta estimada acerca del presupuesto que habría costado el desarrollo de la aplicación web. En este caso, el desarrollo completo ha sido llevado a cabo por una sola persona, por lo que los costes y los tiempos pudiesen variar en el caso de que se trabajase con un equipo. Destacar que la complicación del desarrollo ha influenciado en el precio del coste de la hora.

Tareas	Horas diarias	Días empleados	Coste de la hora	Precio final
Tareas de diseño				
Diseño de Wireframes	6 Horas	3 Días	10 Euros	180 Euros
Diseño de la arquitectura web	6 Horas	2 Días	10 Euros	120 Euros
Diseño relacional de la base de datos	6 Horas	1 Día	10 Euros	60 Euros
Tareas de implementación				
Implementación del registro/logout de usuarios	6 Horas	11 Días	20 Euros	1320 Euros
Implementación pantalla “Mostrar todos los pisos”	6 Horas	7 Días	10 Euros	420 Euros
Implementación pantalla “Filtros”	6 Horas	8 Días	20 Euros	960 Euros
Implementación pantalla “Perfil del usuario”	6 Horas	7 Días	7 Euros	294 Euros
Implementación pantalla “Añadir nuevo piso”	6 Horas	3 Días	10 Euros	180 Euros
Implementación pantalla “Aviso legal”	6 Horas	1 Día	5 Euros	30 Euros
Implementación pantalla “Funcionamiento”	6 Horas	1 Día	5 Euros	30 Euros
Tareas de gestión de la base de datos				
Creación de la base de datos con restricciones	6 Horas	1 Día	20 Euros	120 Euros
TOTAL				**3714 Euros**

Tabla 4: Propuesta de presupuesto

7. Estructura del resto del documento

La estructura que seguirá el proyecto se basa en 5 capítulos más: Análisis, Diseño, Implementación, Demostración y Conclusiones y Líneas Futuras.

El capítulo de Análisis hará una descripción del estado del arte en el que se estudiará el mercado de la vivienda en España desde el estallido de la burbuja inmobiliaria hasta la actualidad. Además, se estudiarán otros sitios web que se encuentren dentro de la temática de la vivienda. Se hará un estudio de quiénes serán los usuarios potenciales de la aplicación y se definirán los objetivos y las especificaciones de la aplicación web.

El capítulo de Diseño hará una descripción de la conceptualización del producto, se definirá la arquitectura cliente / servidor además de la descripción de la base de datos. Se comentan los lenguajes de programación, las APIs utilizadas para la implementación y el software necesario para el desarrollo. Finalmente, se mostrarán los diseños de wireframes y los diagramas de flujo de la aplicación web.

El capítulo de Implementación especificará de manera detallada como se ha implementado el sitio web en cuanto a código fuente.

El capítulo de Demostración hará una ilustración sobre el funcionamiento de la aplicación.

El capítulo de Conclusiones y Líneas futuras presentará las conclusiones obtenidas de todo el proyecto así como la exposición de ideas que mejorarían la aplicación web en un futuro si se decidiera seguir con la implementación del sitio web.

Capítulo 2: Análisis

1. Estado del arte

Con el fin de poder llevar a cabo el proyecto ha sido indispensable llevar a cabo principalmente un estudio de como ha ido avanzando con el paso de los años el mercado de la vivienda en España junto con todas sus características. Desde la burbuja inmobiliaria hasta la decisión actual de alquilar una o varias partes de una vivienda por diversos motivos.

Para conocer mejor la situación, se debe ser consciente acerca de qué factores hacen que un ciudadano elija la opción de alquilar una habitación o vivienda y además, qué características hacen que dicho ciudadano se decante por un piso o por otro por lo que, a través de diversos artículos que han sido redactados por portales que se dedican a la temática de la vivienda junto con los datos proporcionados por el Instituto Nacional de Estadística, ha sido posible la tarea de recabar información para darle mejor forma a la aplicación web.

2. Análisis del mercado

2.1 La burbuja inmobiliaria y la crisis económica en España

Durante los años 1998 y 2007, se gestó en España una burbuja inmobiliaria que hizo que los precios de las viviendas se disparan de forma desorbitada. Se pretendía generar un aumento de creación de viviendas para todos aquellos emancipados de la época que pretendían comprar una vivienda. En el año 2002, el ritmo de construcción se había disparado y junto con el, el precio de las viviendas.

Según los datos de la Sociedad de Tasación, la Figura 1 muestra los datos de la evolución del precio de la vivienda en España:

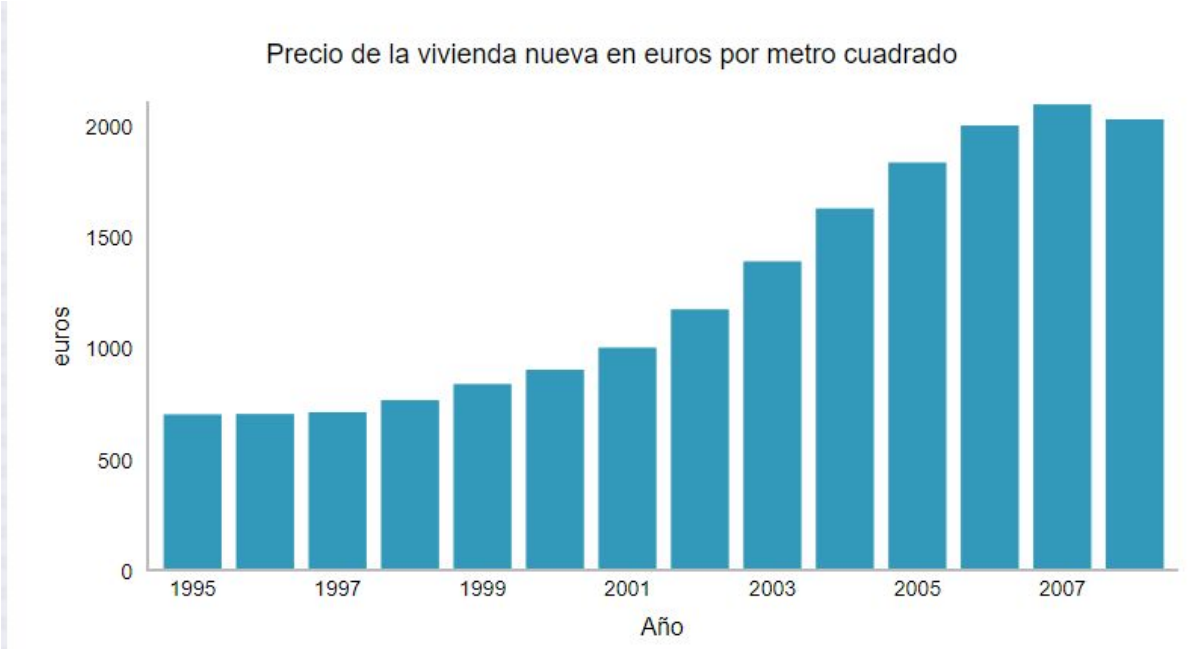


Figura 1: Precio de la vivienda nueva en euros por metro cuadrado

Como se puede apreciar en el gráfico, a lo largo de los años el precio de la vivienda sube, pudiéndose apreciar como en 10 años, de 2006 a 2016, el precio de la vivienda pasó de costar 694,4 euros/m2 a 1999,50 euros/m2, es decir, en diez años, el precio de la vivienda en España ha aumentado 1305,1 euros/m2. Así mismo, en el año 2007 el precio de la vivienda se situó en 2085,50 euros/m2 siendo dicho dato el precio más elevado que ha tenido la vivienda en España.

En el año 2008, la crisis que estalló en Estados Unidos se propagó a nivel mundial. En España, la cifra del paro comenzó a incrementar siendo el año 2012, el año que cierra con la mayor tasa de paro durante la crisis económica Española tal y como muestra la Figura 2:

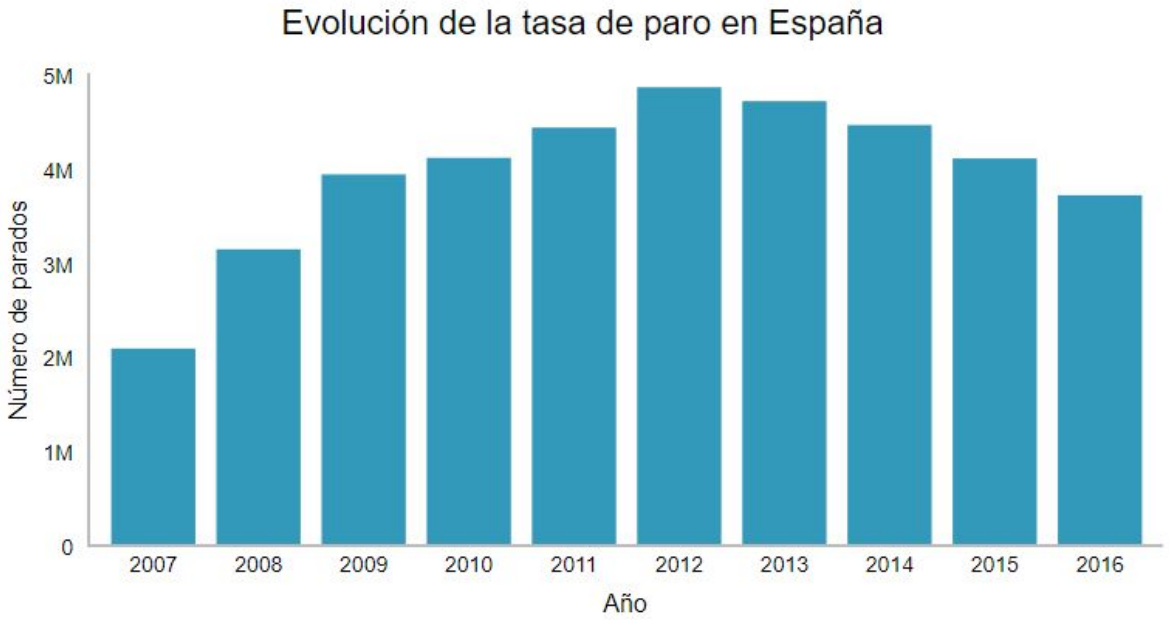


Figura 2: Evolución de la tasa de paro en España

Del mismo modo, se debe destacar el salario medio de los españoles. Aunque cada año el salario ha ido subiendo, la diferencia entre el sueldo medio en el año 2010 y el año 2015 ha sido de 54,9 euros.

Figura 3: Salario medio de los españoles

2.2 ¿Comprar o alquilar?

Aunque la idea de comprar una vivienda sigue estando en la mente de los ciudadanos españoles, una gran parte de la población todavía no puede permitirse la compra de una vivienda a pesar de la bajada del precio y la apertura del crédito por ello, el alquiler de vivienda sigue ganando terreno poco

a poco a la compra de una vivienda. Un 37% de la población prefiere comprar una vivienda mientras que un 30% de la población prefiere alquilar una vivienda. Los datos sociodemográficos extraídos del último informe llevado a cabo por Fotocasa acerca de la relación de los españoles con la vivienda son los mostrados en la Figura 4:

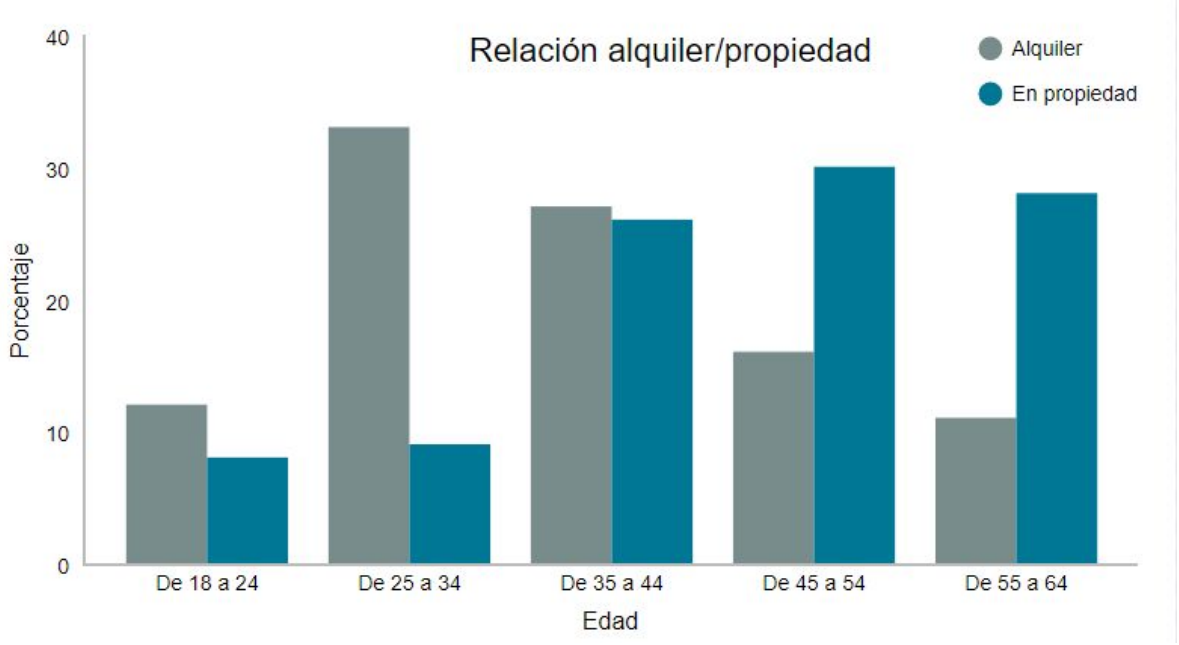


Figura 4: Relación alquiler/propiedad

De los datos se obtiene que el mayor porcentaje de ciudadanos que se encuentran dentro de un contrato de alquiler están dentro del rango de 25 a 34 años; se reafirma la idea de que a mayor rango de edad, la población tiende a comprar vivienda antes de alquilar.

Paralelamente, se encuentran datos relativos a la situación personal y profesional. En cuanto a su situación personal, un 67% de los ciudadanos que alquilan no tienen hijos en contraposición al 44% que vive en propiedad; además, el porcentaje de población que alquila y que se encuentra trabajando es del 84% mientras que el de la población que vive en propiedad es del 78%.

Además, según la intención de llevar a cabo una compra futura, se diferencian 3 tipos de rangos de edad. La generación Millennial corresponde a la población que ha nacido entre los años 1980 y 2000, la generación X corresponde a la población que ha nacido entre los años 1965 y 1979 y finalmente, la generación Baby Boomer que corresponde a la población que ha nacido dentro del rango de los años 1945 y 1964. Según los datos mostrados en la Figura 5, el mayor porcentaje de los Millenials no comprará en un futuro próximo al igual que la generación X sin embargo, en el caso de los Baby Boomers, la cifra de ciudadanos que no comprará casi equivale a la cifra de los que seguro que

comprará en el futuro. En general, el porcentaje de los que no compran sino que vivirán de alquiler independientemente de la generación, es el porcentaje más alto.

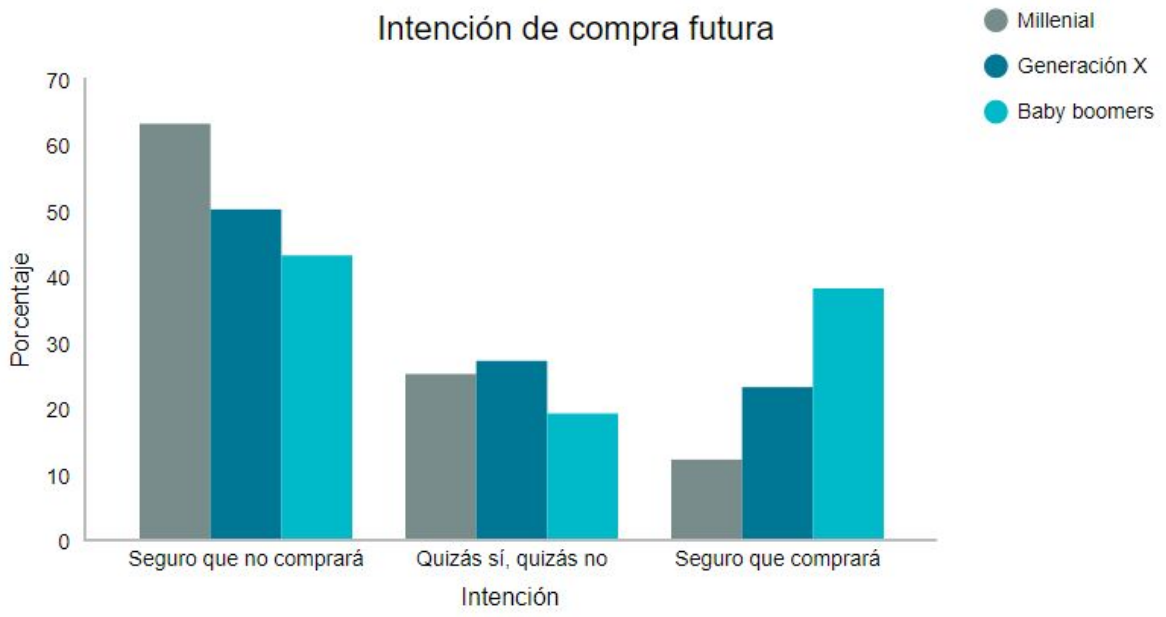


Figura 5: Intención de compra futura

2.3 Compartir piso, una nueva forma de subsistir

Una manera de ahorrar a la hora de pagar el alquiler es la de compartir piso. Aunque compartir piso no presta las mismas libertades y privacidad de una vivienda propia, una gran parte de la población se ha decantado por esta opción. Según el último informe anual de pisos compartidos desarrollado por la web pisos.com y los informes desarrollados por la web fotocasa, los datos obtenidos son los siguientes son los siguientes:

De entre las diez ciudades que más porcentaje de demanda de alquiler de habitación hay, se encuentran las principales capitales de provincia. Madrid y Barcelona obtienen los porcentajes mayores, un 25.05 y un 21.03 respectivamente; representando las 10 capitales el 77.90% del total de toda España. En la Figura 6 se muestran las capitales por orden de menor a mayor porcentaje:

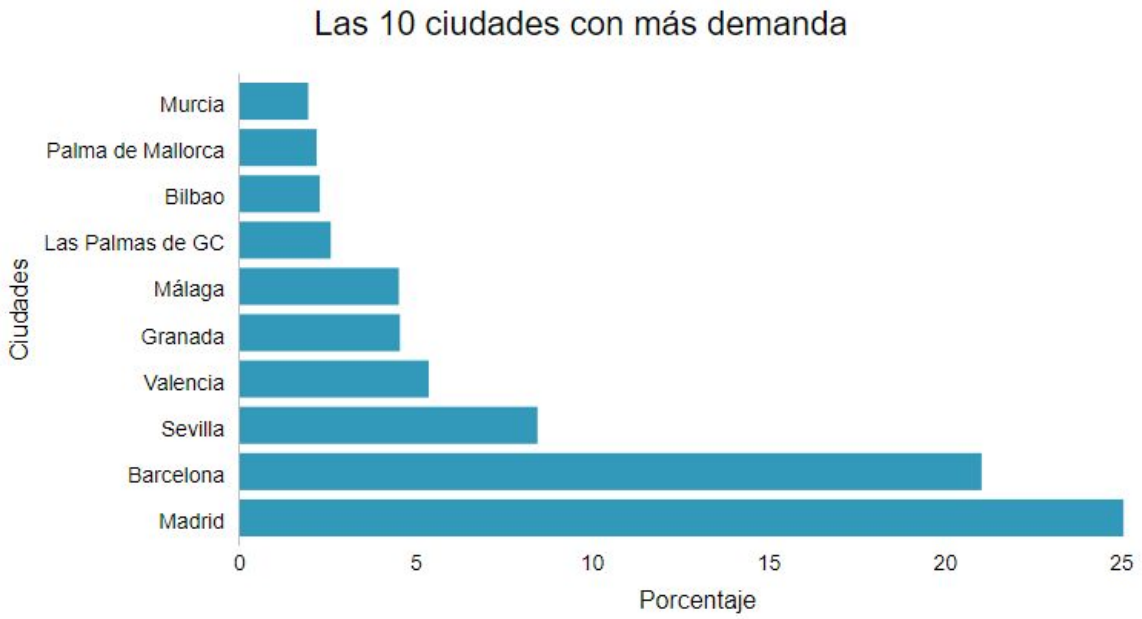


Figura 6: Ciudades con mayor demanda de habitación en alquiler

En lo que respecta al perfil de la población que demanda los pisos compartidos, la edad varía en función de la zona geográfica, el grueso del porcentaje lo obtienen aquellos ciudadanos que se encuentran dentro del rango de 25 a 35 años siendo la edad media los 30. La Figura 7 muestra un desglose del porcentaje en función de la edad:

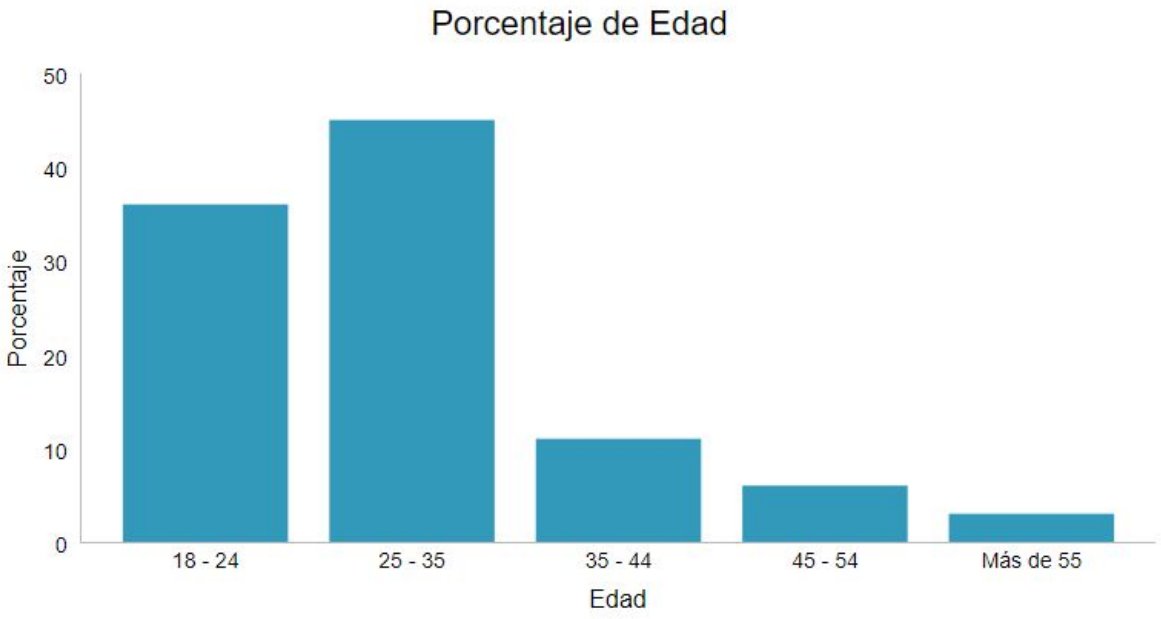


Figura 7: Perfil del ciudadano por edad

Además, el porcentaje de mujeres es del 60% en contraposición con el 40% de los hombres, el 100% de ambos sexos no tienen hijos y la clase social media y media-alta es la que prevalece a la hora de compartir.

La figura del inquilino que es buscado es de sexo indiferente sin embargo, la preferencia de mujeres es mayor que la de hombres (un 19,44% más), son no fumadores y algunos aceptan parejas y mascotas. La Figura 8 representa la figura del inquilino estadísticamente:

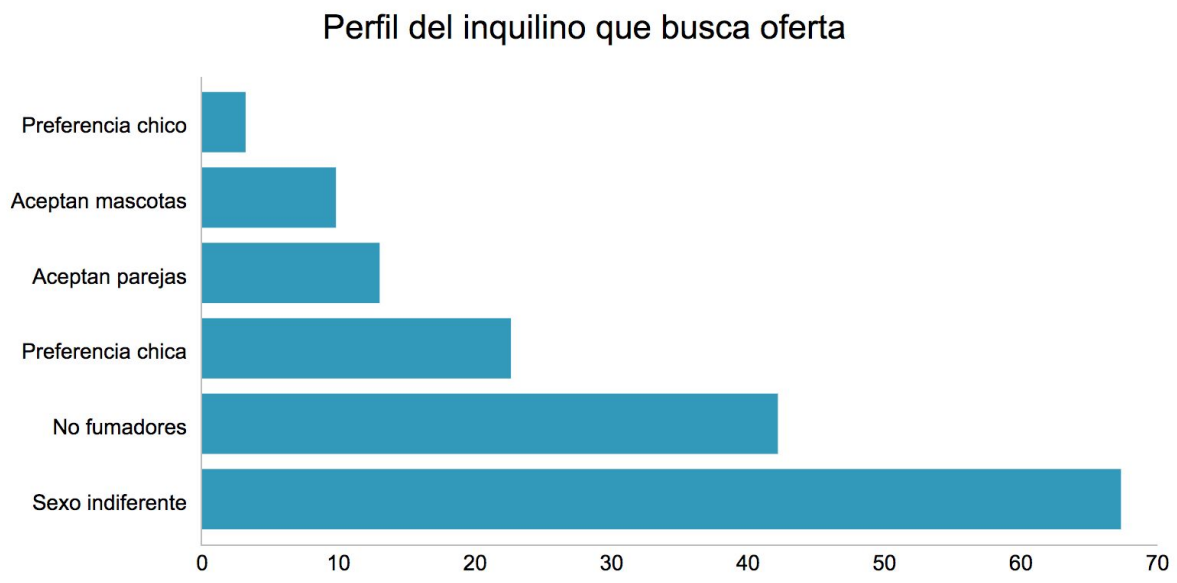


Figura 8: Perfil de inquilino que es buscado

Finalmente, en lo que respecta al equipamiento de los pisos compartidos y las habitaciones en alquiler, las 5 herramientas que más se ofrecen en los pisos son la lavadora (un 93,02%), la televisión (un 81,57%), tendedero (un 79,76%), internet (un 78,95%) y la plancha (un 67,93%). En cuanto a las habitaciones, los tres tipos más destacados son habitaciones amuebladas (un 95,49%), habitaciones exteriores (un 47,31%) y habitaciones con calefacción (un 43,24%).

2.4 Otros sitios web similares:

Una aplicación web muy utilizada por los usuarios que desean ofrecer alojamiento o realizar una búsqueda a nivel mundial es *Airbnb*. *Airbnb* es una plataforma web que se dedica a la oferta de alojamiento a particulares y turísticos. El sitio web ofrece la los usuarios búsquedas filtradas por numerosos campos distintos además de ofrecer diversos idiomas y monedas extranjeras entre muchas características. Su interfaz se caracteriza por ser limpia a la vez que intuitiva que ofrece registro y acceso a aquellos usuarios que deseen poseer una cuenta en la aplicación web. La Figura 9 muestra la interfaz de *Airbnb*:

Figura 9: Interfaz de Airbnb al acceder

Otra aplicación web que permite elegir entre compartir, alquilar y comprar es *idealista*. *Idealista* es una compañía que lleva 17 años ofreciendo un portal web en el que poder gestionar cualquier a de las tareas nombradas anteriormente. Además, presenta una serie de oficinas físicas por si el usuario lo necesitase. En cuanto a la aplicación, ésta presenta una serie de filtros nada más acceder al sitio web que permiten al usuario realizar búsquedas más rápidas. Al ser un portal que lleva tantos años en funcionamiento, su interfaz no es tan llamativa como puede ser la de *Airbnb*, por lo que los usuarios podrían decantarse por la opción anterior. La Figura 10 muestra la interfaz de *Idealista*:

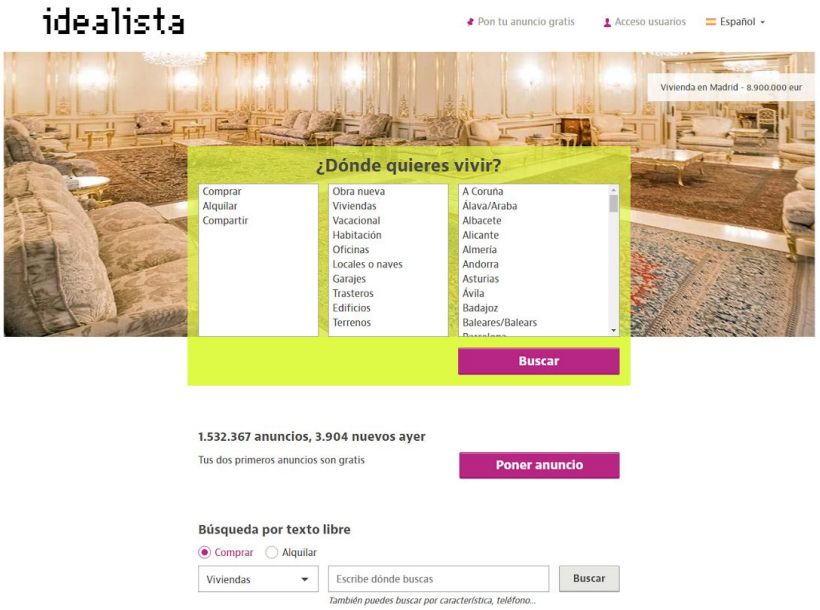


Figura 10: Interfaz de la aplicación de *Idealista*

Otra aplicación web bien conocida por aquellos usuarios que son asiduos a la búsqueda de una vivienda es *Fotocasa*. *Fotocasa* es una aplicación web que permite la compraventa y alquiler de viviendas de segunda y primera mano en España. Al igual que las aplicaciones anteriores, ésta permite la búsqueda filtrada para facilitar la tarea a los usuarios que hagan uso de su aplicación web. Además, les ofrece a los usuarios una sección de ayuda para que aquellos que no tengan del todo claro qué es lo que están buscando mostrándoles ofertas de viviendas de todo tipo. En cuanto a su interfaz, ésta resulta limpia aunque no tan atractiva como *Airbnb*. La Figura 11 muestra su interfaz:

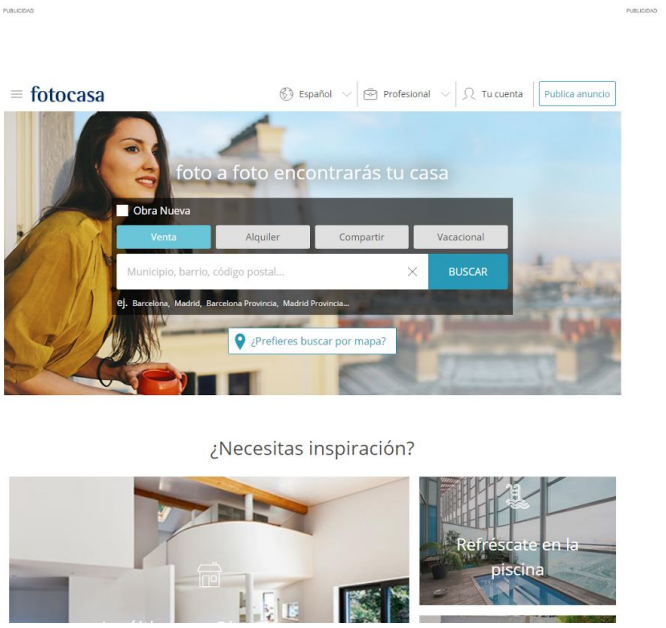


Figura 11: Interfaz de la aplicación de *Fotocasa*

3. Público objetivo y perfiles de usuario

Después de haber llevado a cabo un estudio a través de diferentes estadísticas publicadas por periódicos como *El País*, sitios web especializados en el mercado de la vivienda como el *idealista* y a través de la estadísticas publicadas por el Instituto Nacional de Estadística, más conocido como INE, se han obtenido las siguientes conclusiones:

El público objetivo al que se dirige la aplicación web que por tanto son los usuarios potenciales del sitio web son aquellos que cumplen las siguientes características:

- Ciudadanos mujeres que se encuentren dentro del rango de edad de 25 y 35 años.
- No fumadoras.
- Que alquilan en alguna de las siguientes ciudades:
    - Madrid
    - Barcelona
    - Sevilla
    - Valencia
    - Granada
    - Málaga
- Que no tengan hijos.

Además, los motivos cuya mayor fuerza cobran a la hora de compartir un piso son la posibilidad de ahorrar dinero para poder ser invertido en un futuro en viajes de ocio y, la posibilidad que ofrecen para de irse de casa de los padres.

4. Definición de objetivos/especificaciones del producto

Los objetivos y especificaciones que debe contener la aplicación web se enumeran a continuación:

- La aplicación web debe ser responsive es decir, debe poder adaptarse a cualquier tipo de pantalla y que sea posible utilizarla sin ningún problema.

- La aplicación web debe contar con un menú en la parte superior de la pantalla que permita a los usuarios navegar por el sitio web sin problemas.

- La aplicación web debe contar con un registro de usuarios y un login de los mismos. En cuanto al registro,este puede ser a través de las credenciales usuario y contraseña o, los usuarios podrán registrarse haciendo uso de las redes sociales de Google y Facebook. Se han diseñado estos dos modos de registro debido a que el usuario de esta forma puede sentirse libre de elegir como prefiere acceder a la aplicación web. Además, si se registra por usuario y contraseña simplemente deberá aportar en el registro su nombre y su apellido con el fin de que aporten las credenciales mínimas debido a que por lo general, los usuarios tienden a ser reacios a dar datos personales. En cuanto a las redes sociales, éstas son las que gestionarán las credenciales de los usuarios, por lo tanto se le añade una capa de seguridad a la aplicación ya que no se almacenan contraseñas.

- La aplicación debe contar con una interfaz en la que se muestren los últimos 10 alquileres que han sido añadidos a la base de datos con el fin de mostrar los datos más actualizados.

- La aplicación web debe poder dejar acceder a la información completa sobre un alquiler y además, poder poner en contacto a los usuarios a través de un formulario de contacto.

- La aplicación web debe poder realizar filtros de distinto tipo con el fin de que los usuarios puedan realizar búsquedas de una forma más rápida dadas unas características.

- La aplicación web debe poder añadir un nuevo alquiler a la base de datos y que éste sea mostrado automáticamente en la interfaz.

- La aplicación web debe propiciar al usuario un perfil de usuario con la finalidad de consultar sus datos personales además de gestionar sus publicaciones.

- La aplicación web debe mostrar el aviso legal además del funcionamiento de la aplicación en diferentes partes.

Capítulo 3: Diseño

1. Conceptualización del producto

El producto a desarrollar es una aplicación web. La aplicación web, a través de un registro en el que el usuario pueda elegir entre las redes sociales de Google o Facebook o, registro a través de credenciales de email y contraseña, podrá acceder a los contenidos publicados por otros usuarios o por el mismo sobre la temática de compartición de pisos.

Las publicaciones sobre los alquileres presentan una serie de datos que conformarán las características de dichos alquileres y, a través de las cuales, el usuario pueda elegir entre un alquiler u otro.

Con tal de facilitar la búsqueda a los usuarios, la aplicación web cuenta con una serie de filtros sobre datos de mayor relevancia para el usuario, como pueden ser el precio del aquiler o los metros cuadrados, para que, en función de las características que más les interesen, se les ofrezca una serie de ofertas de entre las cuales podrán elegir la que más les convenza. Además, para aquellos usuarios que en vez de realizar búsquedas sobre alquileres deseen publicar información sobre uno o varios alquileres, tendrán la opción de poder añadir un nuevo alquiler a la base de datos en función de las características que la aplicación ofrece en cuanto a la publicación.

Por otra parte, para que los usuarios puedan moverse por la interfaz sin tener problemas, la aplicación cuenta con un menú en la parte superior de la pantalla que contiene los siguientes enlaces:

- Mostrar los últimos 10 alquileres que han sido añadidos a la aplicación web.
- Realizar búsquedas con filtros.
- Añadir un nuevo alquiler a la aplicación y publicación automática en ella.
- Perfil del usuario para gestionar sus publicaciones.
- Contacto con el arrendador del alquiler.
- Zonas de información:
    - Aviso legal
    - Funcionamiento de la aplicación

Cabe destacar que, se han tenido en cuenta más características que podrían mejorar la aplicación web pero que, por falta de tiempo, no se han podido completar. Las características se definen a continuación:

- Añadir más campos para la publicación de un alquiler que puedan resultar interesantes, por lo que se debería además modificar la base de datos.
- Mejorar la interfaz.

- Hacer pruebas de usabilidad.
- Añadir otras redes sociales para el login de la aplicación.
- Mejorar la accesibilidad de la aplicación.
- Añadir mejoras en la comunicación entre el cliente y el arrendador de un alquiler.
- Añadir más filtros en función de los nuevos campos añadidos.
- Promocionar la aplicación.
- Añadir varios idiomas.
- Añadir posibles pagos en la aplicación.

2. Arquitectura Cliente / Servidor

La arquitectura cliente servidor es un modelo de diseño en el que un cliente realiza peticiones a un servidor y éste último le envía las respuestas a sus peticiones.

Los elementos principales en este tipo de arquitectura son el cliente y el servidor por lo que se podría decir que un cliente es un consumidor de recursos y un servidor es un proveedor de servicios además, el cliente es quien inicia la conexión entre ambos. Además, existe otro elemento denominado middleware que es aquella parte del sistema encargado de transportar los mensajes entre el cliente y el servidor y que permite independizar a los clientes y a los servidores por lo que el middleware facilita el desarrollo de aplicaciones debido a que es el que resuelve la parte de transporte de mensajes y facilita la interconexión de sistemas heterogéneos sin utilizar tecnologías propias de cada uno de ellos.

La Figura 12 representa la arquitectura cliente / servidor haciendo uso de un servidor web Apache con PHP. Dicha arquitectura será explicada a continuación:

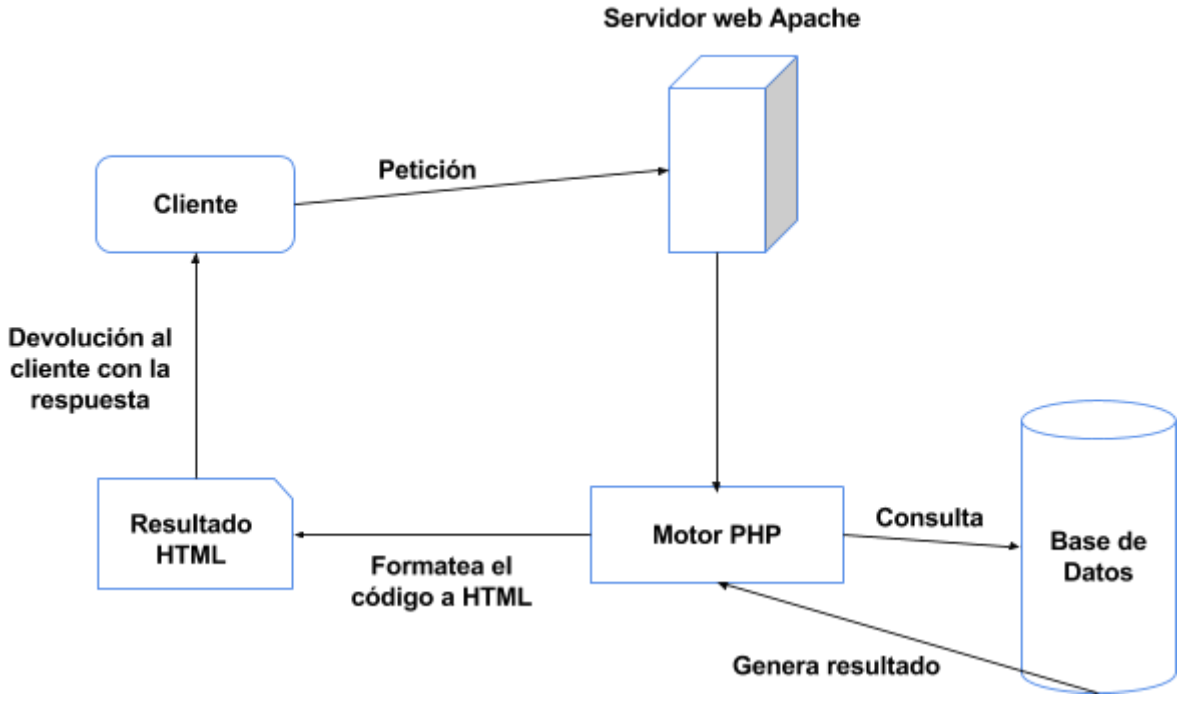


Figura 12. Arquitectura cliente / servidor

- El servidor estando arrancado esperará a que el cliente realice alguna petición.
- Cuando el cliente realiza una petición al servidor apache el cual, a través del motor de PHP realiza una consulta a la base de datos.
- La base de datos le responde con los datos solicitados de la consulta.
- Una vez que el motor de PHP obtenga la respuesta, enviará dicha respuesta a un código formateado en HTML para que el cliente pueda procesarlo.
- Una vez que el cliente lo procesa, el navegador lo visualizará para el usuario.

3. Arquitectura de la información (Base de Datos)

Para poder almacenar los datos de la aplicación, es necesario tener instalado un banco de datos donde poder realizar las peticiones que se den en dicha aplicación.

Una base de datos es un almacén de información que permite agrupar los datos en función de unas características dadas. En una base de datos se pueden aplicar las operaciones CRUD (Create, Read, Update, Delete) para la gestión de la misma por parte del usuario, siendo la base de datos MySQL la más utilizada a nivel mundial.

MySQL es uno de los tipos de base de datos relacionales más importante en la actualidad. Sus desarrolladores son MySQL AB, Sun Microsystems y Oracle Corporation y lleva en uso desde el año 1995 cuyo acrónimo viene de My Structured Query Language.

Una de las características más interesante de este tipo de base de datos la posibilidad de recurrir a bases de datos multiusuario a través de la web y en diferentes lenguajes de programación que se adaptan a requisitos y necesidades. Además, permite la consulta de datos e información de forma rápida. Otras características que ofrece MySQL son las siguientes:

- Amplio subconjunto del lenguaje SQL.
- Transacciones y claves ajenas.
- Replicación.
- Seguridad en la conectividad.

MySQL es ampliamente utilizado en aplicaciones web desarrolladas a través de diversos CMS como Wordpress o Drupal y en plataformas AMP encontrándose altamente ligado al desarrollo de aplicaciones con PHP. Cabe destacar que MySQL implementa una serie de funcionalidades propias que la distinguen de otros tipos de bases de datos. Las funcionalidades se enumeran a continuación:

● Agrupar transacciones, reuniendo varias transacciones de distintas conexiones para aumentar el número de transacciones por segundo.

● Posibilita la elección entre múltiples motores de almacenamiento para cada tabla.

Se entiende por diseño relacional de una base de datos el modelado que se basa en la lógica de predicados y la teoría de conjuntos. Para ello, se hace uso de relaciones (tablas) que definen las agrupaciones de datos en lo que se denomina tuplas (registros).

El modelo relacional diseñado se compone de tres tablas:

- usuarios
- usuarios_rs
- pisos_publicados

La tabla usuarios representa los usuarios que se registran en la web. Se trata de una entidad fuerte ya que no depende de ninguna otra tabla para poder existir. Los campos siguientes mostrados en la Tabla 4 la definen:

Campo	Tipo de dato
id (primary key)	**int** auto_increment
nombre	**varchar(255)** not null
email	**varchar(255)** unique
apellidos	**varchar(255)**
contrasena	**varchar(255)**

Tabla 5. Tabla usuarios

El primer campo es un identificador de usuario el cual es clave primaria en la tabla; siendo además de tipo entero y auto incrementándose cada vez que un usuario nuevo se inserta.

Además, posee cuatro campos más, uno de ellos es el nombre del usuario siendo un tipo string que no puede ser nulo. El campo **email** representa el correo electrónico siendo de tipo *string* y añadiendo la características de ser *único*; **apellidos** y **contrasena** se utilizarán en el caso de que el usuario se registre por correo electrónico y contraseña.

La tabla *usuarios_rs* representa las cuentas de red social que posee un usuario, es decir, un usuario de la tabla usuarios puede encontrarse logueado a través de varias redes sociales almacenadas en esta tabla. Por ello, en este caso la tabla *usuarios_rs* se trata de una **entidad débil** que depende la tabla *usuarios*. Ambas tablas se relacionan a través del **identificador**. Los campos de la Tabla 5 son los siguientes:

Campo	Tipo de dato
id (primary key)	**int** auto_increment
id_usuario	**int** not null

id_social	**varchar(255)** not null
servicio	**varchar(255)** not null

Tabla 6. Tabla usuarios_rs

En este caso, el **id** es un identificador para cada usuario con red social para ser distinguido del resto. El **id_usuario** es el campo con el que se establece la relación con la tabla *usuarios*, el campo **id_social** representa un identificador generado por la API de la red social con la que un usuario se loguea y el campo **servicio** representa el tipo de red social con la que se accede.

Finalmente, la tabla *pisos_publicados* contiene la información relativa a los campos que forman la publicación de un alquiler. En este caso, la tabla es una **entidad débil** ya que se relaciona a través del campo **email** con la tabla *usuarios* para poder llevar un control sobre qué usuario realiza una publicación, por ello, el **email** de la tabla *pisos_publicados* se convierte en **clave foránea**. El campo **id_publicacion** representa la **clave primaria** para distinguir cada publicación de un alquiler. El campo **email** es con el que se hace la relación con la tabla *usuarios* para saber qué usuario realiza cada publicación. Los demás campos representan los datos necesarios que deben ser publicados sobre un alquiler.

Campo	Tipo de Dato
id_publicacion (primary key)	**int** auto_increment
email	**varchar(255)** not null
titulo	**varchar(255)** not null
descripcion	**varchar(255)** not null
calle	**varchar(255)** not null
numero	**varchar(255)** not null
piso	**varchar(255)** not null
poblacion	**varchar(255)** not null
precio	**int** not null
metros	**int** not null
lavadora	**tinyint**
secadora	**tinyint**
calefaccion	**tinyint**
television	**tinyint**
internet	**tinyint**

plancha	**tinyint**
piso_centrico	**tinyint**
garaje	**tinyint**
transp_publico	**tinyint**
ascensor	**tinyint**
fumadores	**tinyint**
imagen	**tinyint**

Tabla 7. Tabla pisos_publicados

La Figura 13 representa el diagrama relacional de lo que se ha explicado anteriormente:

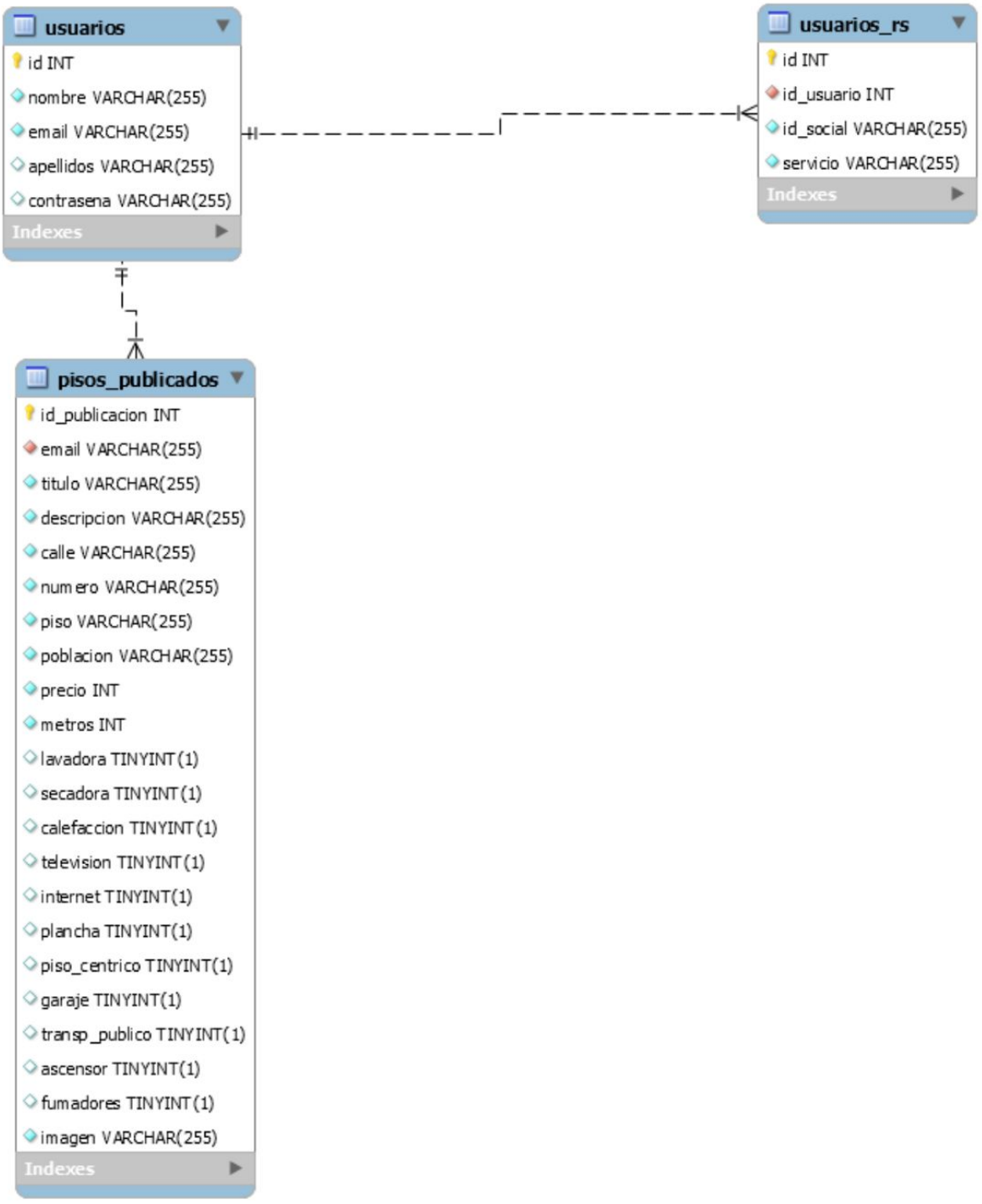


Figura 13. Modelo relacional de la base de datos

4. Lenguajes de programación, APIs utilizadas y software necesario

1.1 Servidor web Apache

Se entiende por servidor web aquel programa software que está diseñado para transferir datos de hipertexto es decir, sitios web con todos sus elementos utilizando para ello, el protocolo HTTP. Los

servidores web se encuentran alojados en un ordenador que posea conexión a internet a la espera de que algún navegador le realice una petición para así, atender la solicitud.

Apache es el servidor web de código abierto más utilizado a nivel mundial debido a la estabilidad y a la seguridad que ofrece. Está desarrollado por Apache Software Foundation comenzando dicho desarrollo en el año 1995 y que implementa el protocolo HTTP. Las características que ofrece Apache se enumeran a continuación:

- Dar soporte a diversos lenguajes como PHP o Python.
- Dar soporte de seguridad SSL y TLS.
- Dar autentificación de datos utilizando un Sistema Gestor de Base de Datos
- (SGBD).

Además, Apache es un servidor web multiplataforma, lo que quiere decir que puede trabajar con diversos sistemas operativos sin perder la estabilidad que ofrece. Una de las formas más sencillas de instalar Apache es utilizando un software denominado XAMPP el cual es un servidor web que contiene otras herramientas como Perl, PHP y MySQL incluyendo los módulos de phpMyAdmin y OpenSSL que lo complementan para el desarrollo de sitios web.

1.2 Lenguaje de programación PHP

PHP (PHP Hypertext Preprocessor) es un lenguaje de programación de código abierto del lado del servidor desarrollado para la implementación de sitios web dinámicos y que puede ser embebido dentro del código HTML. Desarrollado por el PHP Group y diseñado en 1995 por Rasmus Lerdorf es un lenguaje de programación multiparadigma que abarca desde la programación orientada a objetos (OOP) hasta la programación funcional y operativa. El funcionamiento de PHP se basa en que el código es interpretado por un servidor web con módulo de procesador de PHP que genera una página web resultante. Una de las características que presenta es el soporte para gran cantidad de bases de datos como son Oracle, MySQL, PostgreSQL. También ofrece la integración con bibliotecas externas además de otras ventajas como las siguientes:

- Gran extensión de documentación.
- Permite la programación orientada a objetos.
- Es libre y multiplataforma.
- Permite implementar el modelo MVC (Modelo Vista Controlador).

En la Figura 14, se muestra el funcionamiento de un fichero PHP alojado en un servidor y solicitado por un navegador web.

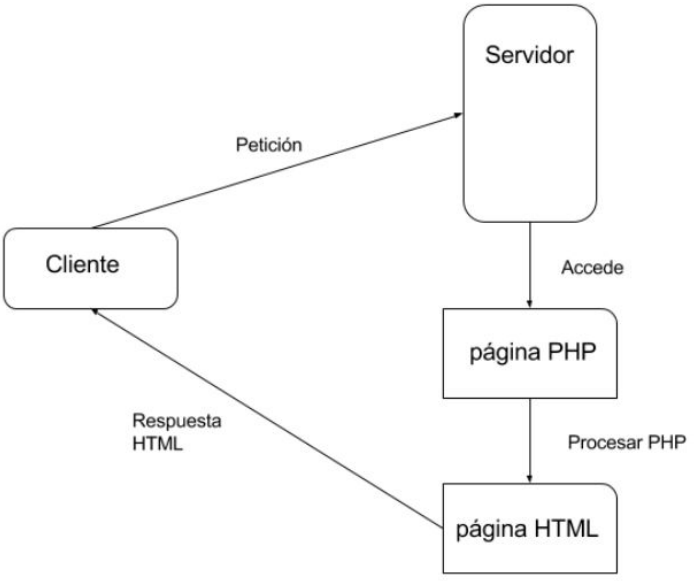


Figura 14: Funcionamiento de PHP

1.3 Software XAMPP

Para poder trabajar en la máquina local con Windows, se ha hecho uso del software XAMPP el cual es una distribución de Apache que permite tener un servidor web el cual gestione la base de datos de tipo MySQL y permite la ejecución de código de PHP y Perl. XAMPP es el acrónimo de Apache MysQL PHP y Perl. Para poder ser utilizado, basta con ejecutar de entre sus opciones *Apache* y *MySQL.*

1.4 MySQL Workbench

MySQL Workbench es una herramienta que permite gestionar la base de datos de tipo MySQL desde diseño de una base de datos, su creación y su mantenimiento, además integra desarrollo de software. Se ha utilizado durante el desarrollo del proyecto ya que permite el manejo de la base de datos desde la máquina local.

1.5 Postman

Postman es un software surgido de una extensión para el navegador de Google Chrome, que permite realizar peticiones a APIs internas o de terceros con la finalidad de llevar a cabo tests para validar el comportamiento de dichas APIs. En este caso, se ha hecho uso de Postman para acceder a los .json generados en la parte del backend y poder así testear la aplicación para su uso en la parte del frontend.

1.6 Editor Atom

Atom es un editor de código fuente multiplataforma (Windows, Linux y Mac) que soporta múltiples lenguajes de programación así como diversos plugin que facilitan el desarrollo de aplicaciones. Es

conocido por ser un editor de textos hackeable ya que fácilmente puede ser configurado por el desarrollador que lo necesite. Para facilitar la escritura de HTML, se ha hecho uso del plugin Emmet el cual provee un kit de desarrollo para los desarrolladores web.

1.7 Git y Github

Git es un controlador de versiones es decir, un sistema que permite registrar cambios sobre un fichero o un conjunto de ficheros de forma que si en un futuro se necesita acceder a una versión anterior, se puede recuperar fácilmente a través de una serie de comandos que provee la herramienta Git Bash.
Por otro lado, GitHub es una plataforma que permite alojar código en un servidor remoto a través del controlador de versiones Git. Para no perder el código fuente implementado durante cada etapa del desarrollo, se ha hecho uso de Git trabajando en local y a su vez, se ha utilizado la plataforma GitHub para alojar el código y poder así disponer siempre del proyecto en el caso de que hubiese algún problema.

1.8 Extensión para Google Chrome: Web Server

La extensión para Chrome Web Server es un servicio HTTP de código abierto. Se trata de una extensión muy utilizada por los desarrolladores web que previene errores como puede ser intentar llamar un fichero de extensión .json en localhost.

1.9 HTML5

HTML5 es el acrónimo de Hypertext Markup Language versión 5 y es la quinta versión del lenguaje de la World Wide Web que se utiliza para definir la estructura y contenido de una página o un documento web. Se basa en utilizar un lenguaje para hacer referencia a otros documentos como ficheros, imágenes, audio o vídeo (entre otros).

1.10 CSS3

CSS u hojas de estilo en cascada es un lenguaje para añadir diseño a un sitio web. CSS3 se encuentra diseñado para enfatizar la separación entre el contenido del documento y la forma de como se presenta, caracterizado por los layouts, los colores y las fuentes de texto. Dicha separación busca mejorar la accesibilidad del documento, proveer más flexibilidad y control en la especificación de características presentacionales, permitir que varios HTML puedan compartir los mismos estilos permitiendo así que no se repita el código el cual, sigue el principio de desarrollo DRY (Don’t Repeat Yourself / Una vez y sólo una).

1.11 BootStrap

BootStrap es un framework de diseño web basado en HTML, CSS y JavaScript. Fue desarrollado por la empresa Twitter como un framework para fomentar la consistencia entre las herramientas internas. Se caracteriza porque presenta un sistema de grid (rejilla) que permite que los sitios y páginas web en las que se utilice sigan el principio de diseño web responsivo (Responsive Web Design)

permitiendo así, que dichas páginas y sitios web puedan ser renderizados desde distintos tamaños de pantalla de dispositivos. Actualmente, Bootstrap se encuentra desarrollando las mejoras de su versión 4.

1.12 Google API

La API de Google permite a los desarrolladores que los usuarios que hagan uso de sus aplicaciones puedan acceder a ellas a través de su cuenta de google con la finalidad de añadir una capa de seguridad a la aplicación. La API de Google permitirá al desarrollador elegir qué datos de los usuarios puede seleccionar para así, mostrarlos en su aplicación sin necesidad de ser almacenados en sus bases de datos. Además, si el usuario lo desea, podrá revocar el acceso de su cuenta de Google a la aplicación dónde se ha registrado desde su cuenta de Gmail.

1.13 Facebook API

La API de Facebook, al igual que la de Google, permite a los desarrolladores que los usuarios que han uso de sus aplicaciones puedan acceder a ellas a través de su cuenta de Facebook sin almacenar ningún tipo de dato personal en sus bases de datos. Para poder hacer uso de la API de Facebook, se necesita tener una cuenta para así poder acceder a la Facebook Graph API.

1.14 Protocolo oAuth2

El protocolo oAuth permite que los usuarios autoricen a terceros a acceder a su información sin que estos tengan que conocer las credenciales del usuario. Es decir, aplicaciones de terceros pueden acceder a contenidos de propiedad del usuario, sin que estas aplicaciones conozcan sus credenciales. Además, el usuario puede revocar los privilegios cuando lo desee.
La principal ventaja que da este protocolo, es la facilidad que presta ante la comunicación entre el usuario y una aplicación tercera generando la confianza al usuario de no tener que entregar sus credenciales en la misma.
Para que el protocolo funcione correctamente, se necesitan al menos dos partes diferenciadas, el cliente y el servidor. El cliente trata de solicitar permiso para acceder a la información y el servidor el que decide si puede y durante cuánto tiempo tiene acceso a los datos.

El cliente debe conocer la dirección URL de dónde se encuentran los servicios de autenticación además de los requisitos para solicitarlos. Por otra parte, el servidor reconoce la información y envía al cliente un token temporal para poder comenzar con el proceso de autenticación.

Seguidamente, el cliente envía dicho token temporal de vuelta al servidor añadiendo otro tipo de información la cual dependerá del servicio en una solicitud segura (usando SSL). Finalmente, el servidor enviará el token definitivo para comenzar a utilizar sus recursos en función de la información intercambiada y el tiempo del que dispone para hacer uso de dichos recursos.

Todo el proceso descrito anteriormente, posibilita la clara identificación del cliente y servidor, por lo que se reduce la posibilidad de captura de la información con otros tipos fines.

El flujo de oAuth se muestra en la Figura 15:

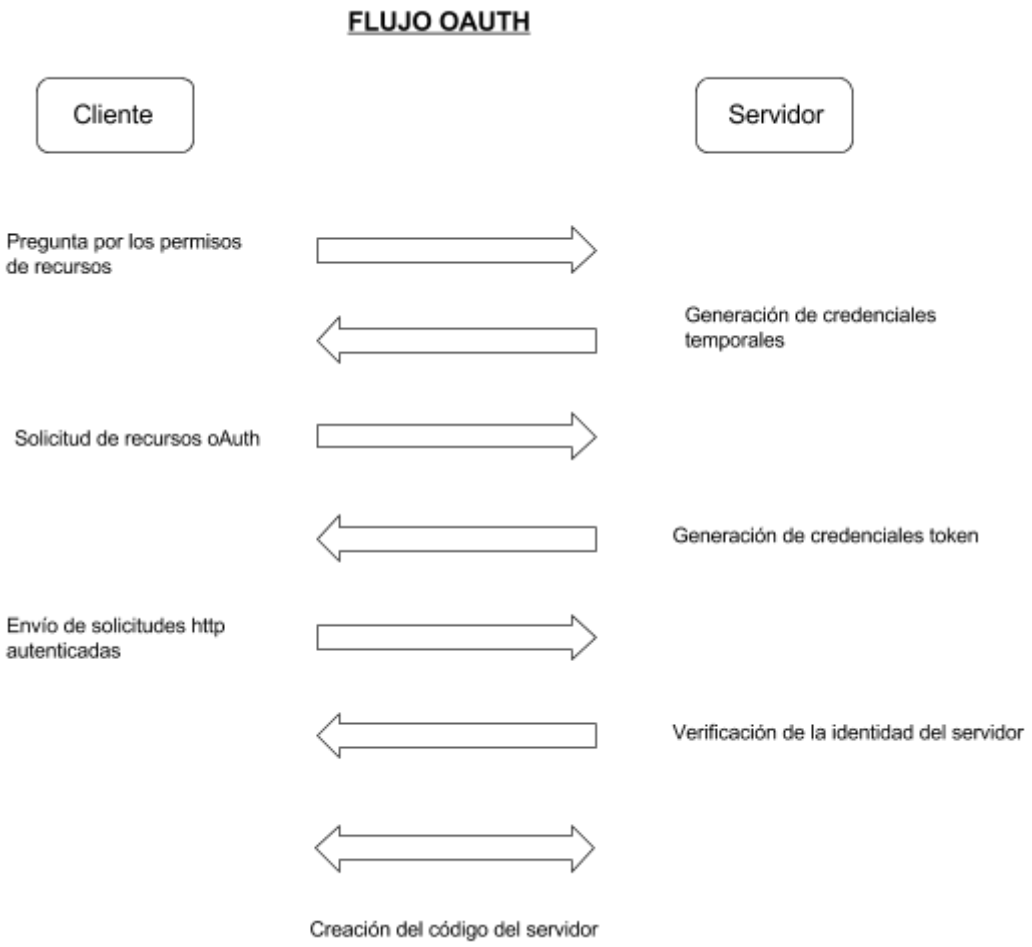


Figura 15: Funcionamiento del protocolo oAuth2

5. Diseño de Wireframes

El esquema de la web (wireframes) representa el esqueleto de la aplicación web. En este caso se han llevado a cabo los siguientes diseños de la web para la aplicación a desarrollar. Todos los diseños mostrados a continuación son consistentes es decir, se hacen uso de elementos propios de un sitio web como menús de navegación que siempre se van a encontrar situados en la parte superior de la pantalla manteniendo los enlaces que presenta situados en el mismo orden dentro de todas las pantallas. Además, los wireframes representan los elementos más importantes del diseño, por lo que una vez se implemente el frontend a través de código, pueden aparecer otros elementos no mostrados pero que no son lo suficientemente importantes como para resaltarlos en este diseño previo.

En la Figura 16 se muestra el diseño del Registro / Log In de la aplicación web:

Figura 16: Diseño de registro / login

En la Figura 17, se muestra la interfaz propuesta para mostrar todos los pisos que se encuentran insertados dentro de la base de datos de la aplicación.

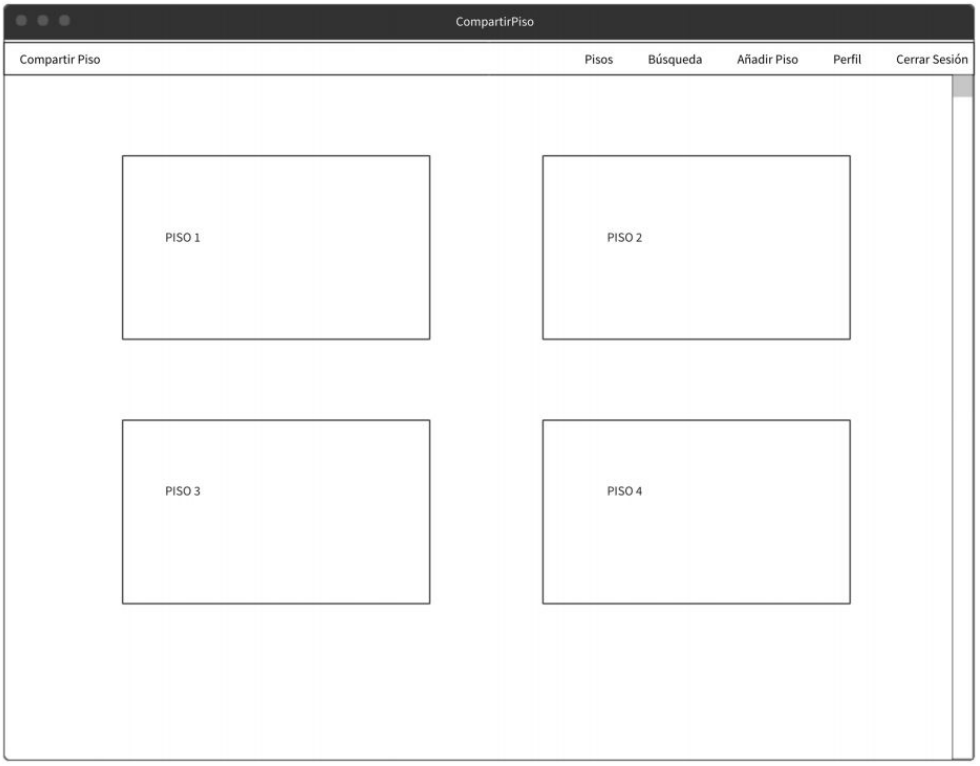


Figura 17: Diseño para mostrar todos los alquileres

En la Figura 18, se muestra la interfaz propuesta para que el usuario pueda llevar a cabo búsquedas de alquileres dentro de la aplicación. Para ello, contará con un pequeño formulario a rellenar con filtros mostrándose los resultados de dichos filtros a la derecha de la pantalla:

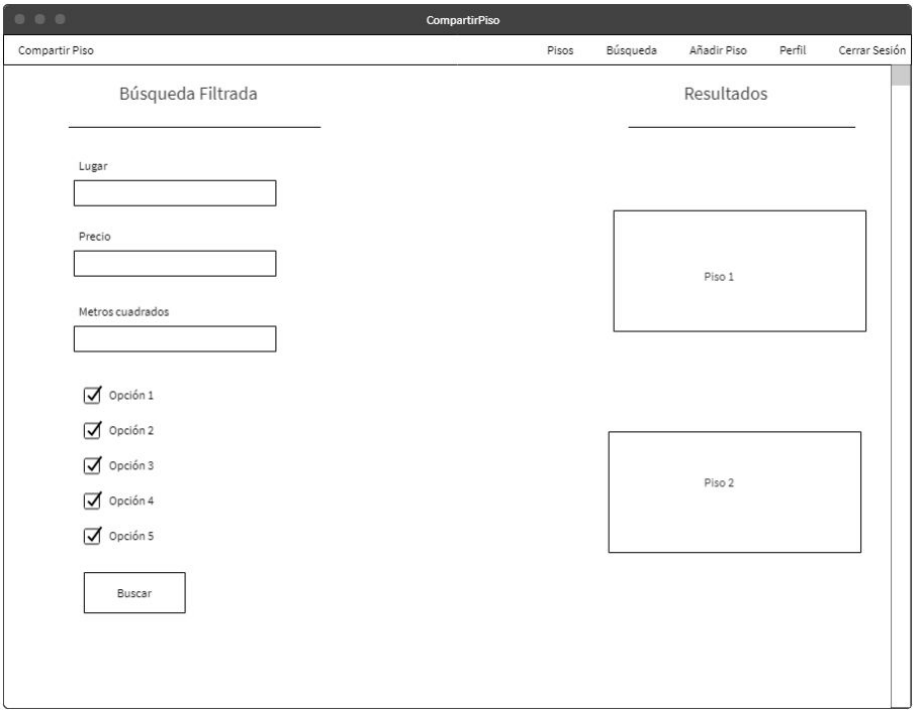


Figura 18: Interfaz para la búsqueda de alquileres

En la Figura 19, se muestra la interfaz diseñada para añadir un nuevo alquiler a la base de datos. En ella, el usuario deberá rellenar los campos propuestos y aceptar.

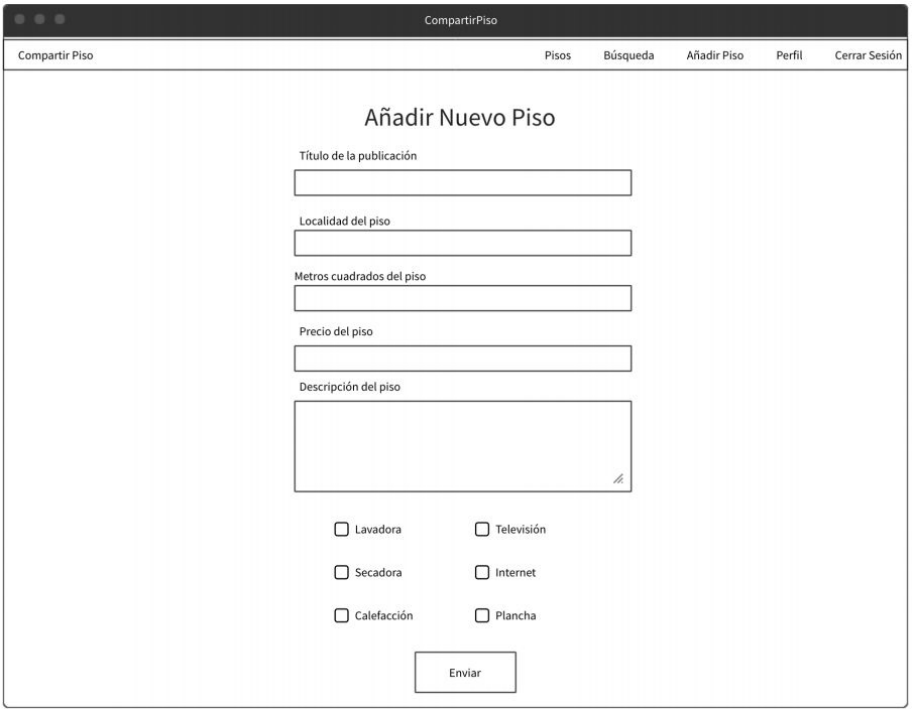


Figura 19: Interfaz para añadir un nuevo alquiler

Finalmente, el último diseño propuesto para la aplicación es el del perfil del usuario. En ella, se mostrarán los campos relativos a los datos personales del usuario que se encuentre logueado en un momento determinado. La Figura 20 representa dicho diseño:

Figura 20: Interfaz para el perfil de usuario

En lo que corresponde a dispositivos más pequeños como Tablet y Móvil, el diseño es el que se muestra a continuación. En la Figura 21 se muestra la interfaz para el Registro / Log In y en la Figura 22 la interfaz para mostrar todos los pisos. En el caso de la Figura 21, cada campo ocupa toda una columna con la finalidad de que el usuario se encuentre cómodo mientras introduce los datos. Por otra parte, el menú superior se ha modificado pasando a mostrarse en forma de acordeón en todas las pantallas.

Figura 21: Registro / Log In

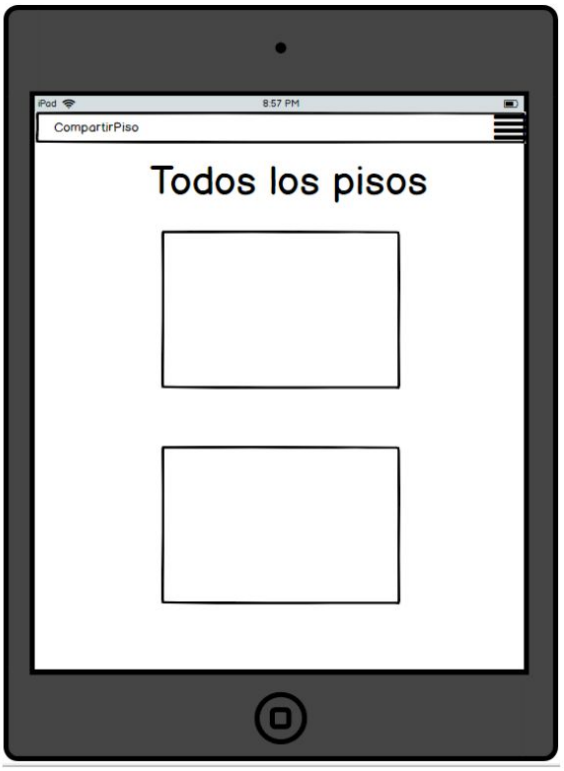


Figura 22: Interfaz para mostrar todos los alquileres

En la Figura 23 se muestra la interfaz para la búsqueda de pisos a través de filtros. En este caso, los filtros se muestran en la parte superior. Por otra parte, en la Figura 24 se muestra la interfaz para añadir un nuevo piso a la base de datos:

Figura 23: Interfaz para búsqueda filtrada

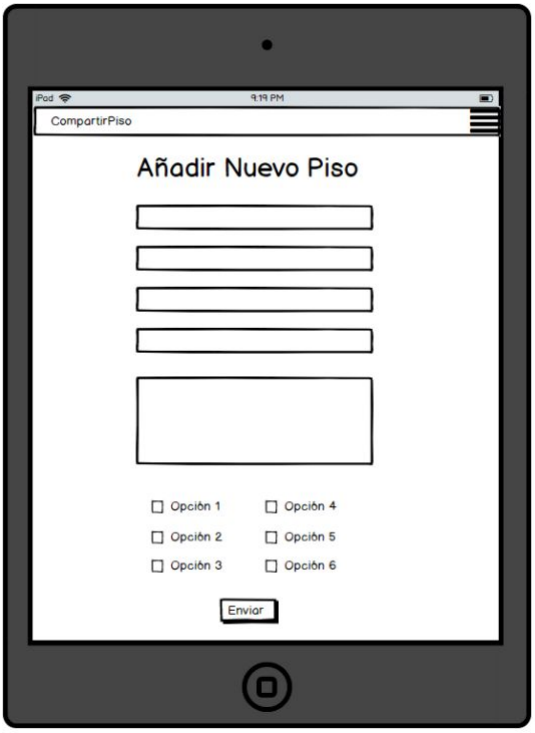


Figura 24: Interfaz para añadir un alquiler

Finalmente, en la Figura 25 se muestra la interfaz del perfil del usuario. Además, dicha interfaz muestra como se despliega el menú en acordeón.

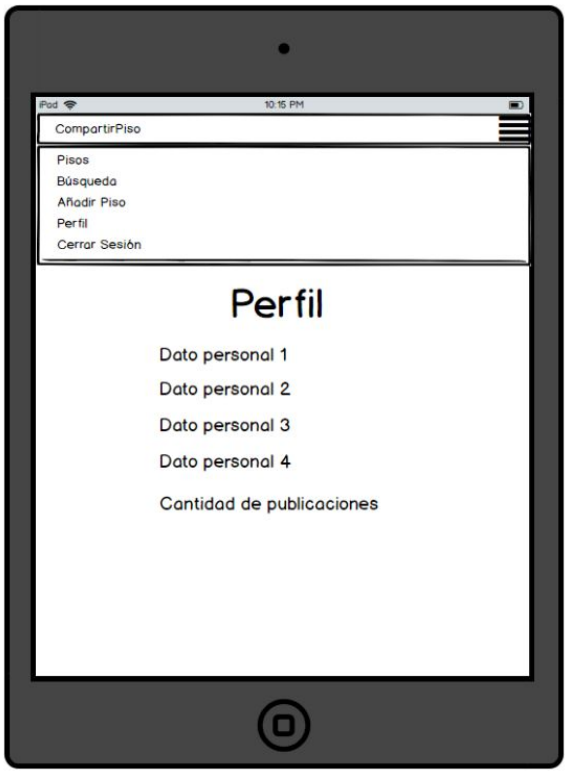


Figura 25: Interfaz del perfil del usuario

6. Diagramas de flujo

El primer diagrama de flujo representa el Registro y Login de los usuarios dentro de la aplicación. Se representa en la Figura 26:

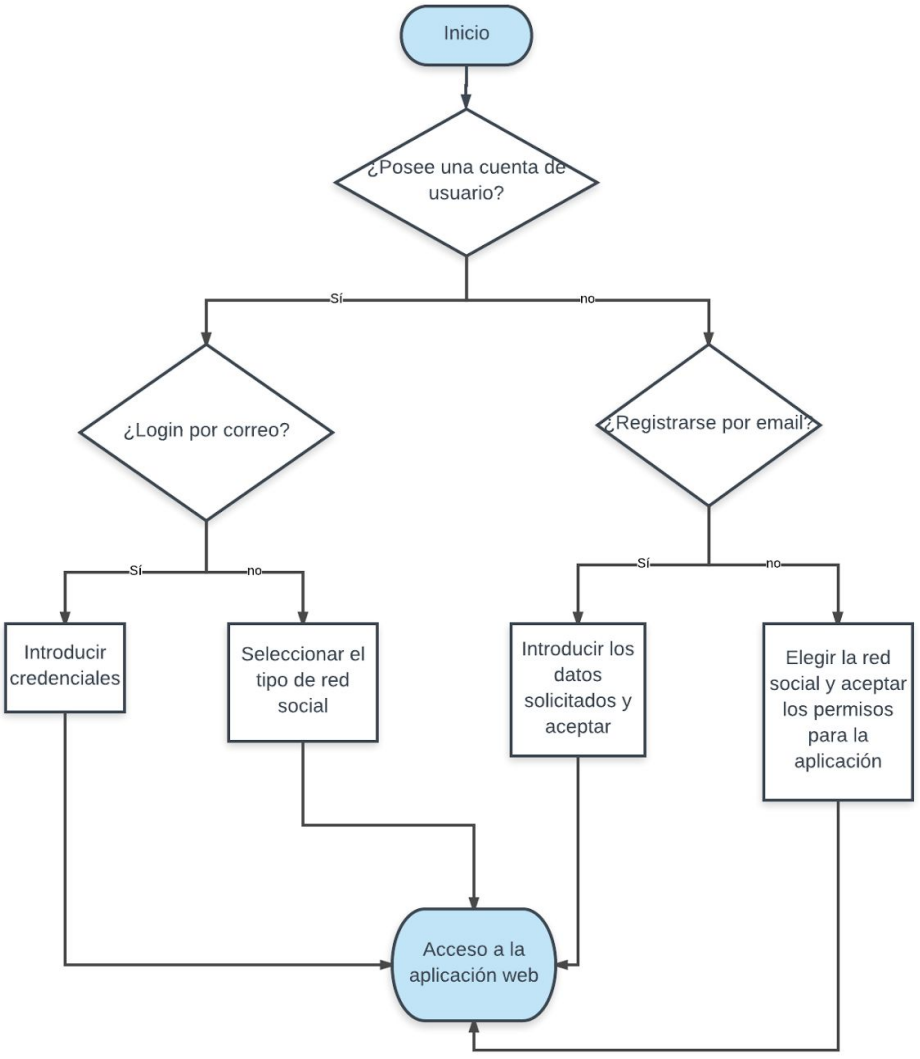


Figura 26: Diagrama de flujo del registro y login

En la Figura 27 se muestra el diagrama de flujo para la opción de “Añadir un nuevo alquiler”.

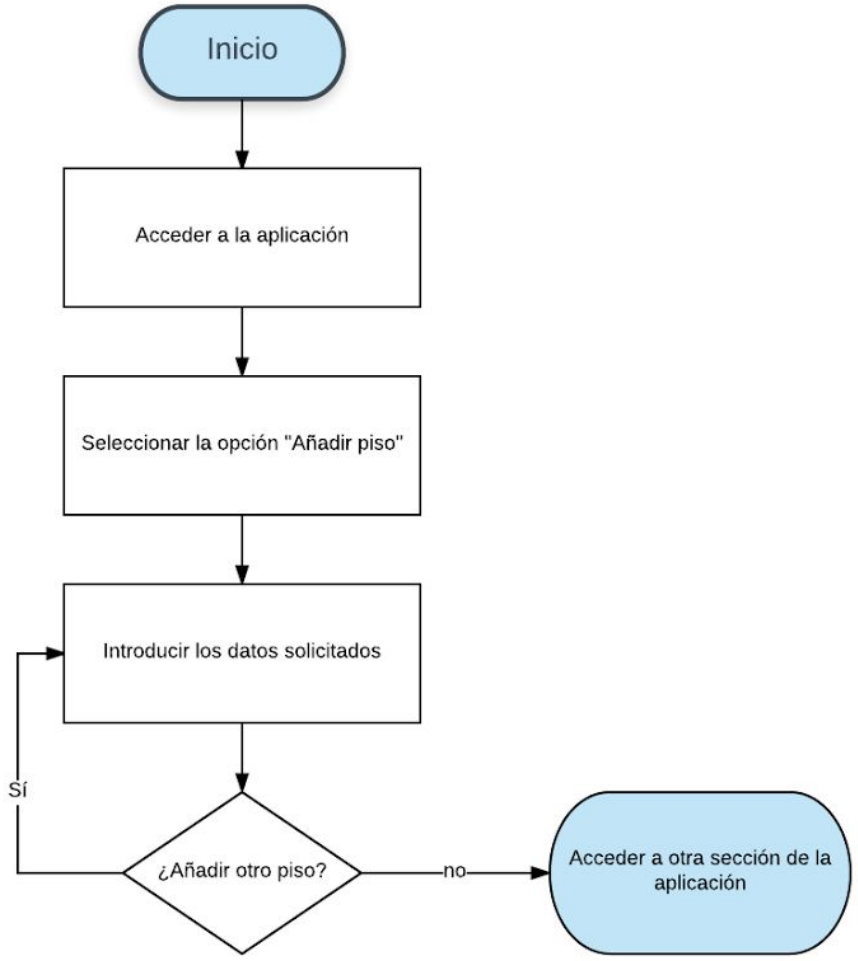


Figura 27: Diagrama para añadir un nuevo alquiler

La Figura 28 representa el diagrama de flujo para realizar filtros de pisos en la aplicación:

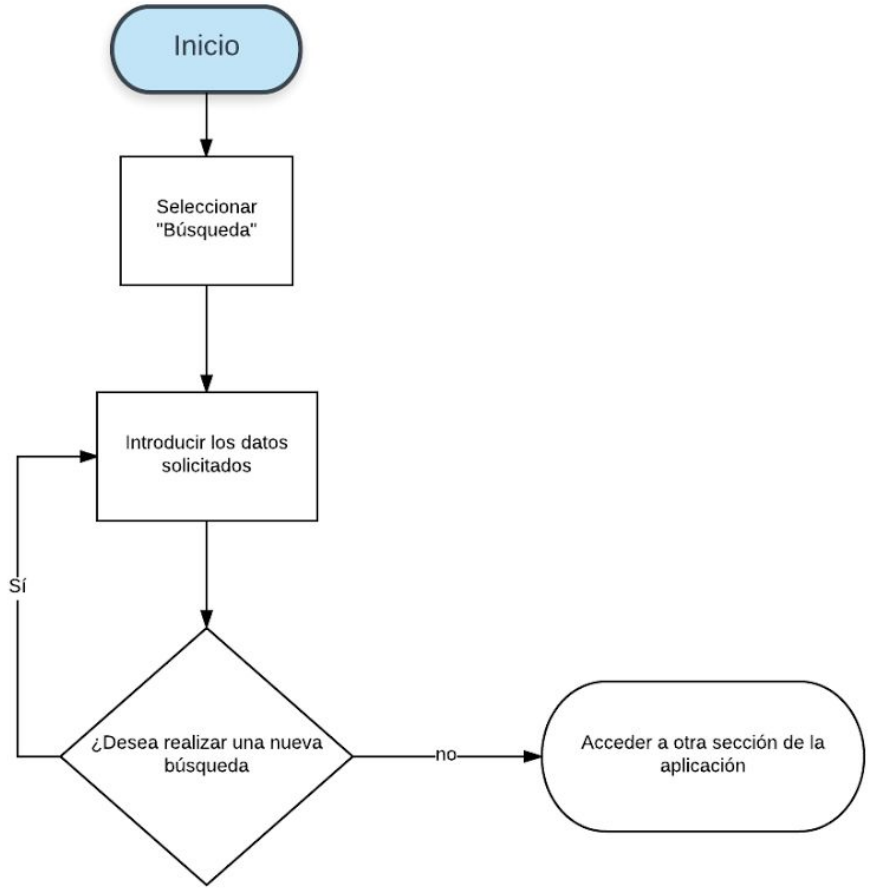


Figura 28: Diagrama de flujo para la búsqueda de un alquiler

Finalmente, la Figura 29 representa el diagrama para la selección de un alquiler:

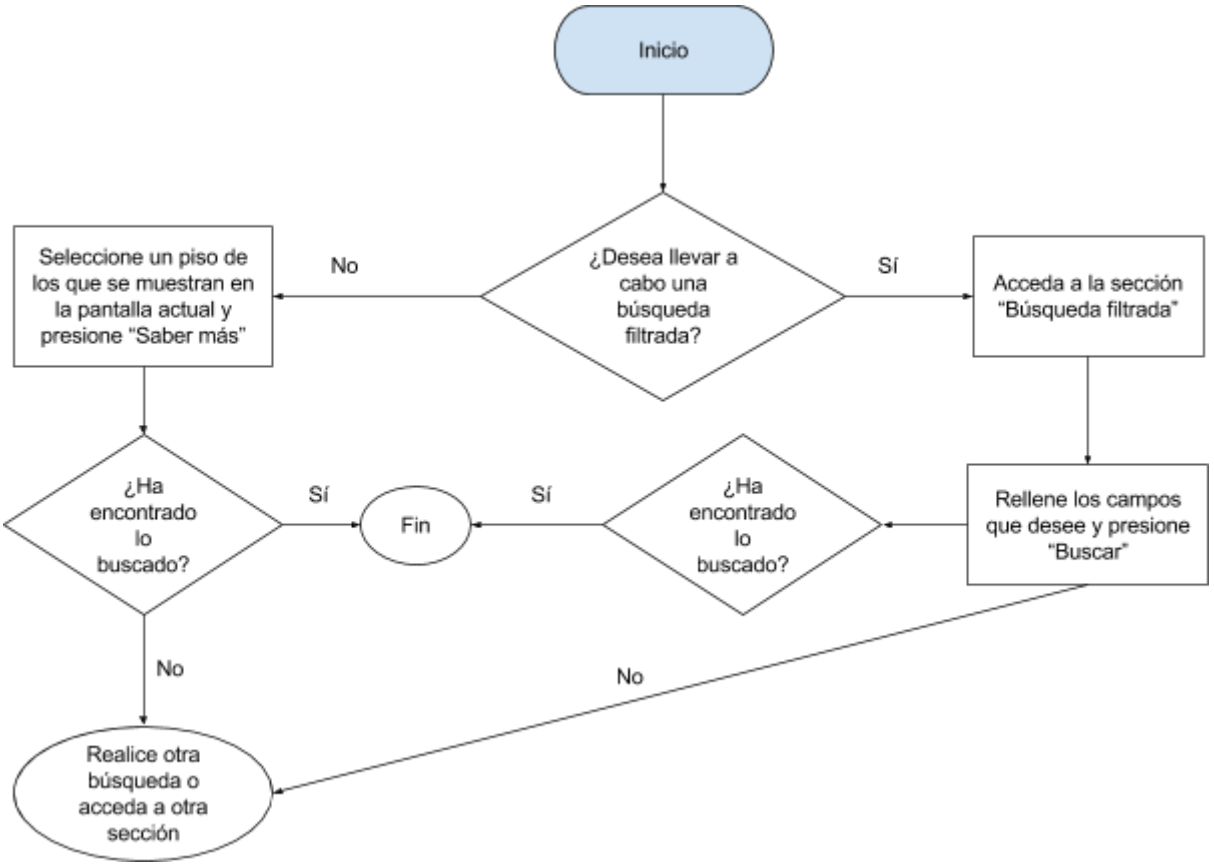


Figura 29: Diagrama para la selección de un alquiler

Capítulo 4: Implementación

1. Definición y creación de la base de datos a través de la herramienta MySQL Workbench

La primera tarea a realizar fue la creación de la base de datos para poder comenzar con el desarrollo del proyecto. Para la creación de la base de datos, se ha utilizado el software denominado *MySQL Workbench*. Primero, se ha definido la base de datos a través de la siguiente sentencia:

```
create database proyecto
```

Y se procedió a crear las tres tablas definidas anteriormente. Para crear la tabla *usuarios*, se ha utilizado la siguiente sentencia:

```
create table proyecto.usuarios(
id int primary key auto_increment,
nombre varchar(255) not null,
email varchar(255) unique,
apellidos varchar(255),
contrasena varchar(255)
```

```
);
```

Para crear la tabla *usuarios_rs*, se ha utilizado la siguiente sentencia:

```
create table proyecto.usuarios_rs(
id int primary key auto_increment,
id_usuario int not null,
id_social varchar(255) not null,
servicio varchar(255) not null
);
```

Como dichas tablas deben estar relacionadas, se añadió la siguiente sentencia para añadir la clave foránea donde además se añadió la propiedad on delete cascade para que, si el usuario es eliminado de la base de datos, todas sus publicaciones también lo sean:

```
alter table proyecto.usuarios_rs add foreign key (id_usuario) references
proyecto.usuarios(id) on delete cascade;
```

Finalmente, se creó la tabla *pisos_publicados* a través de las siguientes sentencias:

```
create table proyecto.pisos_publicados(
id_publicacion int primary key auto_increment,
email varchar(255) not null,
titulo varchar(255) not null,
calle varchar(255) not null,
numero varchar(255) not null,
piso varchar(255) not null,
poblacion varchar(255) not null,
precio int not null,
metros int not null,
lavadora tinyint,
secadora tinyint,
calefaccion tinyint,
television tinyint,
internet tinyint,
plancha tinyint,
piso_centrico tinyint,
garaje tinyint,
transp_publico tinyint,
ascensor tinyint,
fumadores tinyint,
imagen varchar(255) not null
);
```

Y se añadió la misma restricción para la relación entre la tabla *usuarios* y *pisos_publicados*:

```
alter table proyecto.pisos_publicados add foreign key (email) references
```

```
proyecto.usuarios(email) on delete cascade;
```

En este caso, aunque el campo email en la tabla usuarios no sea clave primaria, tiene asignada la restricción unique por lo que es posible realizar la consulta exitosamente. En el caso de que se quiera hacer consultas sobre qué contiene la tabla, la sentencia SELECT permite realizar la acción:

```
select * from proyecto.usuarios;
select * from proyecto.usuarios_rs;
select * from proyecto.pisos_publicados;
```

2. Registro de usuarios

2.1 Registro de usuario a través de la API de Google

Para poder utilizar la API de Google, es necesario previamente tener una cuenta de Gmail para poder hacer uso de la parte de desarrolladores que proveen. Una vez se tenga la cuenta, se accede a dicha parte y se busca la opción de Google+ API dónde se debe crear un nuevo proyecto, escoger la opción de aplicación web generando la API finalmente, dos tipos de token, el ID del cliente y el Secreto del cliente. El funcionamiento de la API de Google junto con la aplicación se refleja en la Figura 30:

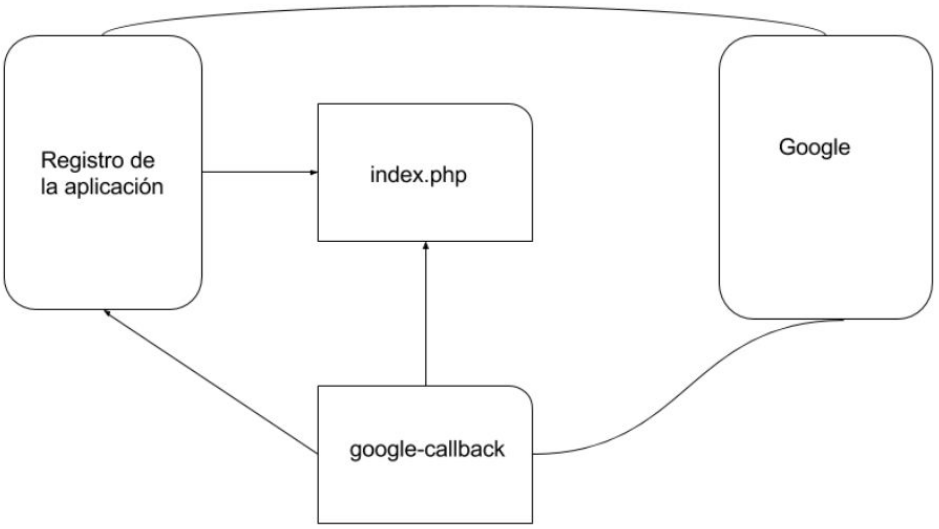


Figura 30: API de Google

Para que la aplicación pueda registrar/loguear a un usuario a través del a cuenta de google, primero la aplicación se debe conectar con **Google** el cual, comprobará si las credenciales definidas en el fichero *callback* coinciden con las que la API proporciona. En el caso de que todo esté correcto, el

callback redirigirá al usuario al *index* de la aplicación, en caso contrario, lo redireccionará de nuevo a la página del *registro* de la aplicación.

En el fichero *config.php* se define el objeto **oAuth** de **Google** y se especifican los **token secretos** e **ID** generados por **Google** para poder conectar con la aplicación.

En lo que respecta a la implementación, el fichero *google-callback.php* contiene el código necesario para comprobar el login de un usuario a través de la cuenta de **Google**. Los pasos para la implementación son los siguientes:

Primero se debe comprobar si en la tabla *usuarios* se encuentra el correo electrónico introducido. En el caso de que no éste se añaden, en caso contrario (es decir, que el correo se encuentre en dicha tabla), se debe comprobar si en la tabla *usuarios_rs* se encuentra insertado el usuario haciendo uso de su ID. En caso afirmativo, el usuario no se añadirá si no simplemente, se logueará como otras veces. En caso negativo, el correo se añade especificando el servicio actual (google). Los procedimientos que se han definido se enumeran en la Tabla 7:

Procedimiento	Funcionalidad
almacenarUsuario($user_data)	Almacena los usuarios en función de una serie de condiciones y da paso a que los usuarios entren en la aplicación web.
comprobarUsuarioRS($user_data, $id_usuarios)	Comprueba si se encuentra el usuario de la red social con la que se accede ya almacenado en la base de datos.
almacenarUsuarioRS($user_data, $id_usuarios)	Almacena aquellos usuarios que hayan utilizado la cuenta de red social por la que intentan acceder al sistema.
getUser($user_data)	Comprueba si existe el email que se pretende utilizar para loguearse en la tabla usuarios de la base de datos.

Tabla 8. Procedimientos utilizados para el registro / login de Google

- Procedimiento: **almacenarUsuario**($user_data): El funcionamiento parte de la condición de si en la tabla *usuarios* se encuentra el email con el que el usuario quiere acceder a la aplicación. En el caso de que no se encuentre, el usuario se registra en la tabla *usuarios* y seguidamente, se añade a la tabla *usuarios_rs*. En caso de que se encuentre en la tabla, se accede a la tabla *usuarios_rs* y se comprueba que esté. En caso afirmativo, el código redirige al usuario al index de la aplicación y en caso contrario, lo añade a la tabla *usuarios_rs* y finalmente, el usuario accede.

- Procedimiento **comprobarUsuarioRS**($user_data,$id_usuarios): El funcionamiento del procedimiento se basa en si el usuario se encuentra en la tabla *usuarios_rs*. Para ello, se hace uso de la sentencia **SELECT FROM WHERE** de *MySQL* a través de los datos que la **API** de **Google** ofrece. En caso de que se encuentre, el procedimiento devolverá un **true** y en caso contrario, devolverá un false.
- Procedimiento **almacenarUsuarioRS**($user_data,$id_usuarios): El funcionamiento del procedimiento se basa en insertar un usuario en la tabla *usuarios_rs* a través de la sentencia **INSERT INTO VALUES** especificando que el servicio en este caso es google.
- Procedimiento **getUser**($user_data): El funcionamiento del procedimiento se basa en comprobar si en la tabla *usuarios* se encuentra el usuario que accede a la aplicación insertado. Para ello, se realiza una consulta **SELECT FROM WHERE** condicionando a través del email. En caso afirmativo, el procedimiento retornará un **true** y en caso contrario, retornará un **false**.

2.2 Registro de usuarios a través de la API de Facebook

Para poder utilizar la **API** de **Facebook**, es necesario previamente tener una cuenta de **Facebook** para poder hacer uso de la parte de desarrolladores que proveen. Una vez se tenga la cuenta, se accede a la sección de **Facebook Developers** el cual provee los token secretos para conectar con la aplicación. El funcionamiento de la API de Facebook junto con la aplicación se refleja en la Figura 31:

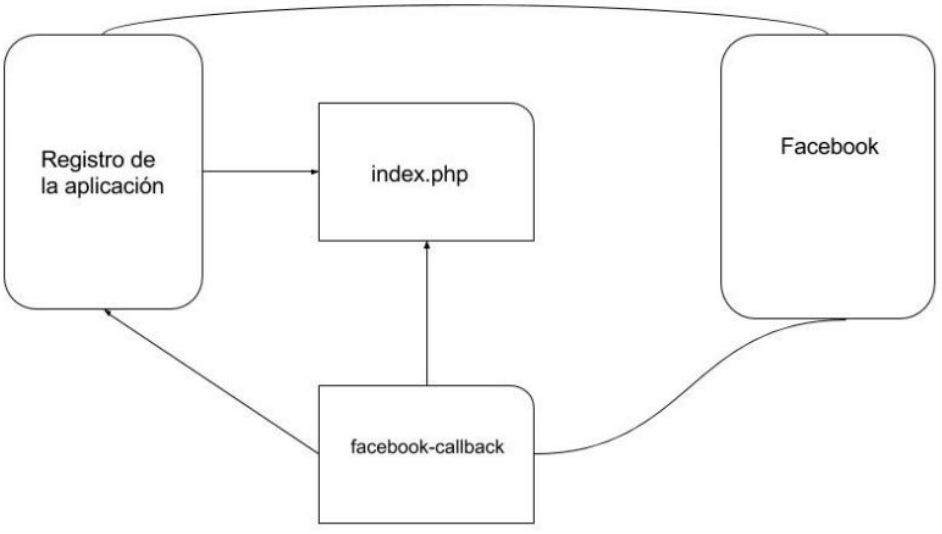


Figura 31: API de Facebook

Para que la aplicación pueda registrar/loguear a un usuario a través del a cuenta de facebook, primero la aplicación se debe conectar con **Facebook** el cual, comprobará si las credenciales definidas en el fichero *callback* coinciden con las que la API proporciona. En el caso de que todo esté

correcto, el *callback* redirigirá al usuario al index de la aplicación, en caso contrario, lo redireccionará de nuevo a la página del registro de la aplicación.

Para que la aplicación pueda registrar/loguear a un usuario a través del la cuenta de facebook, primero la aplicación se debe conectar con Facebook el cual, comprobará si las credenciales definidas en el fichero callback coinciden con las que la API proporciona. En el caso de que todo esté correcto, el callback redirigirá al usuario al index de la aplicación, en caso contrario, lo redireccionará de nuevo a la página del registro de la aplicación.

En el fichero *config.php* se define el objeto **oAuth** de **Facebook** y se especifican los **token secretos** e **ID** generados por Facebook para poder conectar con la aplicación.

En lo que respecta a la implementación, el fichero *facebook-callback.php* contiene el código necesario para comprobar el login de un usuario a través de la cuenta de Facebook. Los pasos para la implementación son los siguientes:

Primero se debe comprobar si en la tabla *usuarios* se encuentra el correo electrónico introducido. En el caso de que no éste se añaden, en caso contrario (es decir, que el correo se encuentre en dicha tabla), se debe comprobar si en la tabla *usuarios_rs* se encuentra insertado el correo. En caso afirmativo, el usuario no se añadirá si no simplemente, se logueará como otras veces. En caso negativo, el correo se añade especificando el servicio actual (facebook). Los procedimientos que se han definido se enumeran en la Tabla 8:

Procedimiento	Funcionalidad
almacenarUsuario($user_data)	Almacena los usuarios en función de una serie de condiciones y da paso a que los usuarios entren en la aplicación web.
comprobarUsuarioRS($user_data, $id_usuarios)	Comprueba si se encuentra el usuario de la red social con la que se accede ya almacenado en la base de datos.
almacenarUsuarioRS($user_data, $id_usuarios)	Almacena aquellos usuarios que hayan utilizado la cuenta de red social por la que intentan acceder al sistema.
getUser($user_data)	Comprueba si existe el email que se pretende utilizar para loguearse en la tabla usuarios de la base de datos.

Tabla 9: Procedimientos utilizados para el registro / login de Facebook

- Procedimiento **almacenarUsuario**($user_data): El funcionamiento parte de la condición de si en la tabla *usuarios* se encuentra el email con el que el usuario quiere acceder a la aplicación. En el caso de que no se encuentre, el usuario se registra en la tabla usuarios y seguidamente, se añade a la tabla *usuarios_rs*. En caso de que se encuentre en la tabla, se accede a la tabla *usuarios_rs* y se comprueba que esté. En caso afirmativo, el código redirige al usuario al index de la aplicación y en caso contrario, lo añade a la tabla *usuarios_rs* y finalmente, el usuario accede.
- Procedimiento **comprobarUsuarioRS**($user_data,$id_usuarios): El funcionamiento del procedimiento se basa en si el usuario se encuentra en la tabla *usuarios_rs*. Para ello, se hace uso de la sentencia **SELECT FROM WHERE** de **MySQL** a través de los datos que la API de Facebook ofrece. En caso de que se encuentre, el procedimiento devolverá un **true** y en caso contrario, devolverá un **false**.
- Procedimiento **almacenarUsuarioRS**($user_data,$id_usuarios): El funcionamiento del procedimiento se basa en insertar un usuario en la tabla *usuarios_rs* a través de la sentencia **INSERT INTO VALUES** especificando que el servicio en este caso es facebook.
- Procedimiento **getUser**($user_data): El funcionamiento del procedimiento se basa en comprobar si en la tabla *usuarios* se encuentra el usuario que accede a la aplicación insertado. Para ello, se realiza una consulta **SELECT FROM WHERE** condicionando a través del email. En caso afirmativo, el procedimiento retornará un **true** y en caso contrario, retornará un **false**.

La Figura 32 muestra el fichero *config.php*:

```php
<?php

session_start(); // Se inicia una sesión

/*Configuración de los Token y URL de acceso para Google */
require_once "GoogleAPI/vendor/autoload.php";
$google_client = new Google_Client(); // Definición del objeto
/*Se setean todos los parámetros para acceder a la API */
$google_client->setClientID("997862673051-00e9j7fuuk5qpglk2fgpc0chgqfbgmt6.apps.googleusercontent.com");
$google_client->setClientSecret("bnHPA0U6DYOEumVHpSbaIICd");
$google_client->setApplicationName("TFM");
$google_client->setRedirectUri("http://localhost/Proyecto/google-callback.php");
$google_client->setAccessType('offline');
$google_client->setScopes(array('https://www.googleapis.com/auth/plus.login', 'https://www.googleapis.com/auth/userinfo.profile',
                        'https://www.googleapis.com/auth/plus.me', 'https://www.googleapis.com/auth/userinfo.email'));


/* Configuración de los Token para Facebook */
require_once "Facebook/autoload.php";

/* Se setean los parámetros para acceder a la API*/
$fb = new Facebook\Facebook([
  'app_id' => '136102443820860',
  'app_secret' => '3502936a8e0948d7f5cdf2b4c61ae83b',
  'default_graph_version' => 'v2.10'
]);

/*Objeto para el acceso */
$fb_client = $fb->getRedirectLoginHelper();

?>
```

Figura 32: Fichero config.php

2.3 Registro por usuario y contraseña

Para registrar a un usuario por correo y contraseña, el usuario debe rellenar un formulario con los campos de Nombre, Apellido, Email y Contraseña. Estos campos serán almacenados a la tabla usuarios.

El el caso de que el usuario no se encuentre registrado, rellenará los campos anteriormente mencionados a través del formulario de registro que se encuentra disponible en el fichero login.php. Una vez rellenados, hará click en el botón enviándose los campos a través del método POST a la base de datos. Para ello, principalmente, se comprobará si el correo ya está en la base de datos. En caso afirmativo, la aplicación web enviará un mensaje de error y, en caso negativo, lo añadirá a la tabla usuarios a través de la sentencia **INSERT INTO VALUES**.

El campo contraseña se encriptará a través de un procedimiento que provee PHP denominado *base64_encode()* en el cual se deberá pasar por parámetros el valor introducido en el input del HTML que corresponde a la contraseña.

En el caso de que el usuario esté registrado y quiera acceder a la aplicación web, deberá rellenar el formulario que se encuentra en el fichero *login.php*. Una vez introducidos los campos, el usuario deberá hacer click en el botón. Seguidamente, se comprobará que el usuario y la contraseña son válidos haciendo uso del procedimiento *base64_decode()* para desencriptar la contraseña y

comprobar que coincide con la introducida en el formulario. En caso afirmativo, el usuario será redirigido a la aplicación y en caso contrario, se mostrará un mensaje de error.

3. Código común para todos los ficheros

Se pretende que todo el sitio web cumpla el principio de consistencia es decir, que ciertos elementos como el menú superior y el footer de cada página web que componen la aplicación web tengan el mismo diseño. Para conseguir esto, principalmente se setearon en la cabecera del HTML de cada página PHP definida, una serie de enlaces mostrados y comentados a continuación en la Figura 33:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0">
  <title>Compartir Piso</title>
  <link rel="stylesheet" href="/Proyecto/css/style.css">
  <script src="https://code.jquery.com/jquery-3.2.1.min.js"
  integrity="sha256-hwg4gsxgFZhOsEEamdOYGBf13FyQuiTwlAQgxVSNgt4=" crossorigin="anonymous"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.3/umd/popper.min.js"
  integrity="sha384-vFJXuSJphROIrBnz7yo7oB41mKfc8JzQZiCq4NCceLEaO4IHwicKwpJf9c9IpFgh" crossorigin="anonymous"></script>
  <link href="https://fonts.googleapis.com/css?family=Raleway" rel="stylesheet">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/normalize/7.0.0/normalize.css">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/css/bootstrap.min.css"
  integrity="sha384-PsH8R72JQ3SOdhVi3uxftmaW6Vc51MKb0q5P2rRUpPvrszuE4W1povHYgTpBfshb" crossorigin="anonymous">
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-beta.2/js/bootstrap.min.js"
  integrity="sha384-alpBpkh1PFOepccYVYDB4do5UnbKysX5WZXm3XxPqe5iKTfUKjNkCk9SaVuEZflJ" crossorigin="anonymous"></script>
  <script type="text/javascript" src="/Proyecto/js/script.js"></script>
  <script src="https://use.fontawesome.com/cf3e71d641.js"></script>
</head>
```

Figura 33: Sección de la etiqueta head de todos los ficheros de CompartirPiso

Primero, se define el tipo de codificación de caracteres para poder mostrar elementos de escritura como son las tildes. Seguidamente, se define la etiqueta meta que permite que el sitio web se adapte a distintos tamaños de pantalla. La siguiente etiqueta corresponde al título del sitio web que irá en la pestaña de cada ventana. El siguiente enlace corresponde al fichero de estilos para modificar el diseño de la interfaz. Los siguientes ficheros son enlaces a distintos **CDNs** para enlazar **jQuery**, **BootStrap**, el fichero normalize para eliminar los estilos puestos por los navegadores, y el tipo de letra **Raleway** obtenida de **Google Fonts** gracias a **FontAwesome**. Finalmente, en este caso se enlaza un fichero script para mostrar cierto contenido que se comentará en los siguientes apartados.

Por otra parte, todas las pantallas (excepto el registro / login) contienen un menú superior a través del cual el usuario podrá navegar en el sitio web. La Figura 34 muestra el menú:

```
<div class="container-fluid" style="margin-bottom: 25px;">

<nav class="navbar navbar-expand-lg navbar-light bg-light">
<a class="navbar-brand" href="index.php"><strong>Compartir piso</strong></a>
  <div class="col-sx-12 col-sm-12">
    <ul class="navbar-nav">
      <li class="nav-item"><a class="nav-link" href="index.php"><strong>Pisos</strong></a></li>
      <li class="nav-item"><a class="nav-link" href="search.php"><strong>Búsqueda</strong></a></li>
      <li class="nav-item"><a class="nav-link" href="new_flat.php"><strong>Añadir Piso</strong></a></li>
      <li class="nav-item"><a class="nav-link" href="profile.php"><strong>Perfil</strong></a></li>
      <li class="nav-item"><a class="nav-link" href="work.php"><strong>¿Cómo funciona?</strong></a></li>
      <li class="nav-item"><a class="nav-link" href="advise.php"><strong>Aviso Legal</strong></a></li>
      <li class="nav-item"><a class="nav-link" href="logout.php"><strong>Cerrar Sesión</strong></a></li>
    </ul>
  </div>
</nav>
```

Figura 34: Menú superior para la navegación

El menú se define a través de un elemento de BootStrap cuyos enlaces se definen a través de elementos ítem de lista. En cada ítem de la lista, se añade un enlace para poder enlazar a cada fichero a través de *href* además, se le aplica un estilo *strong* para que la letra se visualice mejor.

Finalmente, en todas las pantallas además se muestra un footer donde está el nombre de la autora del proyecto. La Figura 35 muestra el código del HTML:

```
</div>
<footer>Proyecto desarollado por Carlota Lázaro Hernández</footer>
</body>
</html>
```

Figura 35: Footer del sitio web CompartirPiso

Por otra parte, se ha definido un fichero que realiza las conexiones con la base de datos y que es usado por todos los ficheros para poder realizar las consultas y mostrar los resultados. La Figura 36 muestra como establecer la conexión con la base de datos. Para ello, se crea una clase denominada DataBase que contiene un atributo denominado *conection* y por el cual, se abre la conexión para *localhost*, el usuario *root*, en este caso no hay contraseña y se especifica que el nombre de la base de datos es *proyecto*.

```php
<?php
/*** Conexión con la base de datos
     Clase Database dónde se define un atributo y un constructor
     El constructor contiene los parámetros para acceder a la base de datos
***/
class DataBase {

  private $conection;
  public function __construct() {
    $this->conection = new mysqli('localhost', 'root', '', 'proyecto');
  }

  public function getConection() {
    return $this->conection;
  }
}

 ?>
```

Figura 36: Conexión para la base de datos

4. Página para mostrar los últimos 10 alquileres

Para mostrar los 10 últimos alquileres que han sido añadidos a la base de datos, se ha definido un fichero denominado *index.php*. Principalmente, se ha generado la estructura del HTML de la siguiente manera mostrada en la Figura 37.

```
<div class="$row">
  <div class="col-lg-12">
    <h1 class="title_login">Últimas publicaciones</h1>
  </div>
  <div class="row">
    <div class="col-lg-6">
      <div class="row">
        <div class="offset-lg-2 col-lg-7 offset-md-0" id="columnas-par-1">
        </div>

      </div>
    </div>
    <div class="col-lg-6">
      <div class="row">
        <div class="offset-lg-2 col-lg-7 offset-md-0" id="columnas-impar-1">
        </div>
      </div>
    </div>
  </div>

</div>
```

Figura 37: Estructura del HTML del fichero index.php

El primer row del código envuelve un *breadcrumb*, muy utilizado en los sitios web para orientar al usuario sobre el lugar exacto en el que se encuentra en un momento determinado. En este caso, el usuario se encuentra en la sección de Pisos.

El segundo row hace referencia a la estructura del sitio para mostrar el contenido de la base de datos. Primero se muestra un título que resume la pantalla actual, en este caso se muestran las 10 últimas publicaciones, seguidamente, la pantalla se parte en dos columnas para poder mostrar los alquileres de forma paralela referenciando a la columna de la izquierda como columna par y a la de la derecha como columna impar haciendo uso de un elemento **card** de BootStrap.

El código será “insertado” a través de la lectura de un **JSON** usando **jQuery**. El JSON lo enviará la parte del servidor a través de PHP con los datos resultantes de una consulta a la base de datos. El código se muestra a continuación en la Figura 38:

```php
require_once "db/database.php";
require_once "config.php";

// Se crea una nueva conexión con la base de datos
$db = new DataBase();
$conexion = $db->getConection();

//generamos la consulta
$sql = "SELECT * FROM pisos_publicados ORDER BY id_publicacion DESC limit 10";

if(!$result = mysqli_query($conexion, $sql)) die();
mysqli_set_charset($conexion, "utf8"); //formato de datos utf8
$publicacion = array(); //creamos un array

while($row = mysqli_fetch_array($result)) {
    $id_publicacion = $row['id_publicacion'];
    $email          = $row['email'];
    $titulo         = $row['titulo'];
    $descripcion    = $row['descripcion'];
    $calle          = $row['calle'];
    $numero         = $row['numero'];
    $piso           = $row['piso'];
    $poblacion      = $row['poblacion'];
    $precio         = $row['precio'];
    $metros         = $row['metros'];
    $lavadora       = $row['lavadora'];
    $secadora       = $row['secadora'];
    $calefaccion    = $row['calefaccion'];
    $television     = $row['television'];
    $internet       = $row['internet'];
    $plancha        = $row['plancha'];
    $piso_centrico  = $row['piso_centrico'];
    $garaje         = $row['garaje'];
    $transp_publico = $row['transp_publico'];
    $ascensor       = $row['ascensor'];
    $fumadores      = $row['fumadores'];
    $imagen         = $row['imagen'];

    $publicacion[] = array('id_publicacion'=> $id_publicacion, 'email'=> $email, 'titulo'=> $titulo, 'descripcion'=> $descripcion,
                        'calle'=> $calle, 'piso'=> $piso, 'numero'=> $numero, 'poblacion'=> $poblacion, 'precio'=> $precio,
                        'metros'=> $metros, 'lavadora'=> $lavadora, 'secadora'=> $secadora, 'calefaccion'=> $calefaccion, 'television'=> $television,
                        'internet'=> $internet, 'plancha'=> $plancha, 'piso_centrico'=> $piso_centrico, 'garaje'=> $garaje, 'transp_publico'=> $transp_publico,
                        'ascensor'=> $ascensor, 'fumadores'=> $fumadores, 'imagen'=> $imagen);
}
```

Figura 38: Consulta a la base de datos a través de PHP

Primero, se abre una conexión con la base de datos. Seguidamente, se prepara la consulta a la base de datos definiéndola a través de un SELECT a la tabla *pisos_publicados* y cuya restricción debe ser que se muestren los últimos 10 pisos, es decir, los últimos 10 registros de la tabla. El resultado se envía a un array denominado *publicación.* La Figura 39 muestra el código para generar el JSON denominado *publicaciones.json*:

```php
//desconectamos la base de datos
$close = mysqli_close($conexion)
or die("Ha sucedido un error inexperado en la desconexion de la base de datos");

//Creamos el JSON
$json_string = json_encode($publicacion);

// Volcamos los datos en un fichero
$file = 'publicaciones.json';
file_put_contents($file, $json_string);
?>
```

Figura 39: Generación del JSON por la consulta

Finalmente, el código en JavaScript (jQuery) para mostrarlo en la interfaz es el siguiente localizado en el fichero *script.js* mostrado en la Figura 40:

```
$.get$.getJSON( "Proyecto/publicaciones.json", function( data ) {

  String(data);

  for (var i = 0; i < data.length; i++) {
    if (i % 2 === 0) {
      var imagen = data[i].imagen;
      if ((imagen.indexOf(",") > -1) === true ) {
        var img_sep = imagen.split(",");
        var img1 = "";
        var img2 = "";
        img1 = img_sep[0];
        img2 = img_sep[1];
      $('#columnas-par-1').append("<div class='card' id='columnas-par' style='margin-bottom: 15px;'>");
      $('#columnas-par-1').append("<h4 class='card-title'>" + data[i].titulo + "</h4> ");
      $('#columnas-par-1').append("<img class='card-img-top' src='" + img1 + "' alt='Card image' />");
      $('#columnas-par-1').append("<div class='card-block'>");
      $('#columnas-par-1').append('<ul class="list-group">');
      $('#columnas-par-1').append('<li class="list-group-item"><p><strong>Lugar: </strong>' + data[i].poblacion + ' </p></li>');
      $('#columnas-par-1').append('<li class="list-group-item"><p><strong>Precio: </strong>' + data[i].precio + '</p></li>');
      $('#columnas-par-1').append('<li class="list-group-item"><p><strong>Metros: </strong>' + data[i].metros + '</p></li>');
      $('#columnas-par-1').append('</ul>');
      $('#columnas-par-1').append("<div style='margin-top:15px;'>");
      $('#columnas-par-1').append("<a href='post.php?id=" + data[i].id_publicacion +  "'><button type='button' class='btn btn-primary btn-circle'> Saber más </button></a>");
      $('#columnas-par-1').append("</div>");
      $('#columnas-par-1').append("</div>");
      $('#columnas-par-1').append("</div>");
    }
    else {
      $('#columnas-par-1').append("<div class='card' id='columnas-par' style='margin-bottom: 15px;'>");
      $('#columnas-par-1').append("<h4 class='card-title'>" + data[i].titulo + "</h4> ");
      $('#columnas-par-1').append("<img class='card-img-top' src='" + data[i].imagen + "' alt='Card image' />");
      $('#columnas-par-1').append("<div class='card-block'>");
      $('#columnas-par-1').append('<ul class="list-group">');
      $('#columnas-par-1').append('<li class="list-group-item"><p><strong>Lugar: </strong>' + data[i].poblacion + ' </p></li>');
      $('#columnas-par-1').append('<li class="list-group-item"><p><strong>Precio: </strong>' + data[i].precio + '</p></li>');
      $('#columnas-par-1').append('<li class="list-group-item"><p><strong>Metros: </strong>' + data[i].metros + '</p></li>');
      $('#columnas-par-1').append('</ul>');
      $('#columnas-par-1').append("<div style='margin-top:15px;'>");
      $('#columnas-par-1').append("<a href='post.php?id=" + data[i].id_publicacion +  "'><button type='button' class='btn btn-primary btn-circle'> Saber más </button></a>");
      $('#columnas-par-1').append("</div>");
      $('#columnas-par-1').append("</div>");
      $('#columnas-par-1').append("</div>");
    }
```

Figura 40: Script que muestra la columna de la izquierda en la interfaz

El código anterior muestra la columna de la izquierda para la interfaz. Primero, se utiliza el procedimiento $.getJSON() de jQuery para mostrar el fichero JSON especificado. Seguidamente, el objeto se convierte al tipo String (cadena de texto) para poder mostrar el contenido. Una vez convertido, se recorre el fichero y, se condiciona si hay una o dos imágenes almacenadas para una misma publicación. Si hay dos imágenes almacenadas, se deben coger los path que se encuentran separados por coma en la base de datos, y se muestra la primera imagen. En caso contrario, simplemente se mostrarían los datos a través del uso de data[i].*campo* junto con los selectores de jQuery sobre HTML Para la sección de la derecha, se ha implementado el mismo código pero modificando el selector de jQuery tal y como muestra la Figura 41:

```
    else {
      var imagen = data[i].imagen;
      if ((imagen.indexOf(",") > -1) === true ) {
        var img_sep = imagen.split(",");
        var img1 = "";
        var img2 = "";
        img1 = img_sep[0];
        img2 = img_sep[1];
      $('#columnas-impar-1').append("<div class='card' id='columnas-impar' style='margin-bottom: 15px;'>");
      $('#columnas-impar-1').append("<h4 class='card-title'>" + data[i].titulo + "</h4>");
      $('#columnas-impar-1').append("<img class='card-img-top' src='" +img1 + "' alt='Card image' />");
      $('#columnas-impar-1').append("<div class='card-block'>");
      $('#columnas-impar-1').append('<ul class="list-group">');
      $('#columnas-impar-1').append('<li class="list-group-item"><p><strong>Lugar: </strong>' + data[i].poblacion + ' </p></li>');
      $('#columnas-impar-1').append('<li class="list-group-item"><p><strong>Precio: </strong>' + data[i].precio + '</p></li>');
      $('#columnas-impar-1').append('<li class="list-group-item"><p><strong>Metros: </strong>' + data[i].metros + '</p></li>');
      $('#columnas-impar-1').append('</ul>');
      $('#columnas-impar-1').append("<div style='margin-top:15px;'>");
      $('#columnas-impar-1').append("<a href='post.php?id=" + data[i].id_publicacion +  "'><button type='button' class='btn btn-primary btn-circle'> Saber más </button></a>");
      $('#columnas-impar-1').append("</div>");
      $('#columnas-impar-1').append("</div>");
      $('#columnas-impar-1').append("</div>");
    }
    else {
      $('#columnas-impar-1').append("<div class='card' id='columnas-par' style='margin-bottom: 15px;'>");
      $('#columnas-impar-1').append("<h4 class='card-title'>" + data[i].titulo + "</h4> ");
      $('#columnas-impar-1').append("<img class='card-img-top' src='" + data[i].imagen + "' alt='Card image' />");
      $('#columnas-impar-1').append("<div class='card-block'>");
      $('#columnas-impar-1').append('<ul class="list-group">');
      $('#columnas-impar-1').append('<li class="list-group-item"><p><strong>Lugar: </strong>' + data[i].poblacion + ' </p></li>');
      $('#columnas-impar-1').append('<li class="list-group-item"><p><strong>Precio: </strong>' + data[i].precio + '</p></li>');
      $('#columnas-impar-1').append('<li class="list-group-item"><p><strong>Metros: </strong>' + data[i].metros + '</p></li>');
      $('#columnas-impar-1').append('</ul>');
      $('#columnas-impar-1').append("<div style='margin-top:15px;'>");
      $('#columnas-impar-1').append("<a href='post.php?id=" + data[i].id_publicacion +  "'><button type='button' class='btn btn-primary btn-circle'> Saber más </button></a>");
      $('#columnas-impar-1').append("</div>");
      $('#columnas-impar-1').append("</div>");
      $('#columnas-impar-1').append("</div>");
    }
  }
}
});
```

Figura 41: Script que muestra la columna de la derecha en la interfaz

5. Página para realizar búsquedas filtradas

Con tal de que el usuario pueda llevar a cabo búsquedas filtradas, se ha implementado un fichero denominado *search.php* que realiza la tarea. A continuación se mostrará el código y se comentará como se ha resuelto la tarea. La Figura 42 representa la primera parte del código en la que principalmente se define un formulario con todos los campos posibles para que el usuario rellene a la hora de realizar una búsqueda filtrada. El usuario podrá elegir entre las siguientes opciones por lo que filtrar, siendo posible mezclar tantas características como el usuario desee: filtrar por población, por metro, por precio o por las siguientes opciones:

- Lavadora
- Secadora
- Calefacción
- Internet
- Plancha
- Televisión
- Piso céntrico
- Garaje
- Transporte público
- Ascensor
- Fumadores

```
      <div class="checkbox">
      <strong>Opciones: </strong><br>
      <div class="row">
        <div class="col-lg-6">
          <label><input type="checkbox" name="lavadora"       value="lavadora">       Lavadora           </label> <br>
          <label><input type="checkbox" name="calefaccion"    value="calefaccion">    Calefacción        </label> <br>
          <label><input type="checkbox" name="piso_centrico"  value="piso_centrico">  Piso Céntrico      </label> <br>
          <label><input type="checkbox" name="secadora"       value="secadora">       Secadora           </label> <br>
          <label><input type="checkbox" name="internet"       value="internet">       Internet           </label> <br>
          <label><input type="checkbox" name="garaje"         value="garaje">         Garaje             </label> <br>
        </div>
        <div class="col-lg-6">
          <label><input type="checkbox" name="ascensor"       value="ascensor">       Ascensor           </label> <br>
          <label><input type="checkbox" name="television"     value="television">     Televisión         </label> <br>
          <label><input type="checkbox" name="plancha"        value="plancha">        Plancha            </label> <br>
          <label><input type="checkbox" name="fumadores"      value="fumadores">      Fumadores          </label> <br>
          <label><input type="checkbox" name="transp_publico" value="transp_publico"> Transporte público </label> <br>
        </div>
      </div>
      </div>
      <div class="row">
        <div class="col-lg-12 text-center">
          <button type="submit" class="btn btn-primary btn-circle ">Buscar</button>
        </div>
      </div>
    </form>
  </div>
</div>
```

```
<div class="row">
  <div class="col-lg-5 col-md-12 col-sm-12 col-xs-12" style="margin-botton: 25px;">
    <ol class="breadcrumb">
      <li class="breadcrumb-item active">Está en: Búsqueda </li>
    </ol>
  </div>
</div>


<div class="row">

<div class="col-lg-6">
  <div class="title_login"><h1> Búsqueda Filtrada </h1> <hr style="margin-left: 55px; margin-right: 55px;"> </div>
  <div class="col-lg-6 offset-3 col-sm-12 offset-0 col-xs-12 offset-0" style="margin-left: 15px;">
    <form class="" enctype="multipart/form-data" method="POST" action="search.php">
      <div class="form-group">
      <label class="col-form-label" for="poblacion"><strong>Población del piso</strong></label>
      <input type="text" class="form-control" name="poblacion" id="poblacion" placeholder="Población">
      </div>

      <div class="form-group">
      <label class="col-form-label" for="precio"><strong>Precio del piso</strong></label>
      <input type="text" class="form-control" name="precio_min" id="precio_min" placeholder="Precio mínimo" >
      <input type="text" class="form-control" name="precio_max" id="precio_max" placeholder="Precio máximo" >
      </div>

      <div class="form-group">
      <label class="col-form-label" for="metros"><strong>Metros cuadrados del piso</strong></label>
      <input type="text" class="form-control" name="metros_min" id="metros_min" placeholder="Metros mínimos">
      <input type="text" class="form-control" name="metros_max" id="metros_max" placeholder="Metros máximos">
      </div>
```

Figura 42: Formulario del fichero search.php

Para llevar a cabo los filtros, desde el punto de vista del código se ha realizado lo siguiente:

- Se han obtenido los datos que han sido introducidos en los campos de los formularios y se han almacenado en variables.
- Seguidamente, se ha declarado una variable denominada $filtro en la que se irán concatenando aquellas variables que no estén vacías.
- Finalmente, se ha llevado a cabo la consulta a la base de datos.

A continuación, se muestra un ejemplo en la Figura 43 a través de código de lo comentado anteriormente con el campo para filtrar por precio:

```
/* Filtro para cuando sólo se ha introducido un precio máximo */
if (empty($f_precio_min) && (!empty($f_precio_max))) {
  if($filtro == "") {
    $filtro .= "WHERE precio <= '".$f_precio_max."'";
  }
  else {
    $filtro .= " AND precio <= '".$f_precio_max."' ";
  }
}

/* Filtro para cuando sólo se ha introducido un precio mínimo */
if (!empty($f_precio_min) && (empty($f_precio_max))) {
  if($filtro == "") {
    $filtro .= "WHERE precio >= '".$f_precio_min."'";
  }
  else {
    $filtro .= " AND precio >= '".$f_precio_min."' ";
  }
}

/* Filtro para cuando se ha introducido un rango de precios */
if (!empty($f_precio_min) && (!empty($f_precio_max))) { // Cuando se rellenan ambos valores
  if($filtro == "")
    $filtro .= "WHERE precio BETWEEN '".$f_precio_min."' AND '".$f_precio_max."'";
  else
    $filtro .= " AND precio BETWEEN '".$f_precio_min."' AND '".$f_precio_max."'";
}
```

Figura 43: Filtro por precio

La Figura 44 muestra tanto la consulta, como el almacenamiento de los datos a través del fichero JSON denominado *search_post.json* para que pueda ser renderizado en la interfaz el resultado de la consulta:

```php
$sql = "SELECT * FROM pisos_publicados $filtro";

if(!$result = mysqli_query($conexion, $sql)) die();

mysqli_set_charset($conexion, "utf8"); //formato de datos utf8
$search = array(); //creamos un array

while($row = mysqli_fetch_array($result)) {
  $id_publicacion = $row['id_publicacion'];
  $email          = $row['email'];
  $titulo         = $row['titulo'];
  $descripcion    = $row['descripcion'];
  $calle          = $row['calle'];
  $numero         = $row['numero'];
  $piso           = $row['piso'];
  $poblacion      = $row['poblacion'];
  $precio         = $row['precio'];
  $metros         = $row['metros'];
  $lavadora       = $row['lavadora'];
  $secadora       = $row['secadora'];
  $calefaccion    = $row['calefaccion'];
  $television     = $row['television'];
  $internet       = $row['internet'];
  $plancha        = $row['plancha'];
  $piso_centrico  = $row['piso_centrico'];
  $garaje         = $row['garaje'];
  $transp_publico = $row['transp_publico'];
  $ascensor       = $row['ascensor'];
  $fumadores      = $row['fumadores'];
  $imagen         = $row['imagen'];

  $search[] = array('id_publicacion'=> $id_publicacion, 'email'=> $email, 'titulo'=> $titulo, 'descripcion'=> $descripcion,
                    'calle'=> $calle, 'piso'=> $piso, 'numero'=> $numero, 'poblacion'=> $poblacion, 'precio'=> $precio,
                    'metros'=> $metros, 'lavadora'=> $lavadora, 'secadora'=> $secadora, 'calefaccion'=> $calefaccion, 'television'=> $television,
                    'internet'=> $internet, 'plancha'=> $plancha, 'piso_centrico'=> $piso_centrico, 'garaje'=> $garaje, 'transp_publico'=> $transp_publico,
                    'ascensor'=> $ascensor, 'fumadores'=> $fumadores, 'imagen'=> $imagen);
}
$close = mysqli_close($conexion) or die("Ha sucedido un error inexperado en la desconexion de la base de datos");
//Creamos el JSON
$json_string = json_encode($search);
// Volcamos los datos en un fichero
$file = 'search_post.json';
file_put_contents($file, $json_string);
```

Figura 44: Consulta a la base de datos y envío a un fichero JSON.

Destacar que además de la sentencia **SELECT FROM**, se han utilizado **WHERE** para añadir condición, **AND** para condicionar de que se deben usar los campos que no se encuentran vacíos y **BETWEEN** para llevar a cabo un rango (siendo en este caso los precios mostrados anteriormente). Finalmente, la Figura 45 muestra el **JavaScript** utilizado y escrito en el fichero *script-search.js* que recorre el **JSON** enviado desde el lado del servidor y muestra el contenido en la interfaz. Al igual que en otras pantallas como la de mostrar todos los pisos, se lee el fichero especificado en el procedimiento **$.getJSON** y se recorre a través de la sentencia iterativa *for* condicionando si hay una o dos imágenes y mostrando siempre la primera imagen a través del selector de jQuery sobre HTML:

```
$.getJSON( "/Proyecto/search_post.json", function( data ) {

  String(data);
  for (var i = 0; i < data.length; i++) {
    if (i % 2 == 0){
      var imagen = data[i].imagen;
      if ((imagen.indexOf(",") > -1) == true ) {
        var img_sep = imagen.split(",");
        var img1 = "";
        var img2 = "";
        img1 = img_sep[0];
        img2 = img_sep[1];
        $('#search-post-1').append("<div class='card' id='columnas-par' style='margin-bottom: 15px;'>");
        $('#search-post-1').append("<h4 class='card-title'>" + data[i].titulo + "</h4> ");
        $('#search-post-1').append("<img class='card-img-top' src='" + img1 + "' alt='Card image' />");
        $('#search-post-1').append("<div class='card-block'>");
        $('#search-post-1').append('<ul class="list-group">');
        $('#search-post-1').append('<li class="list-group-item"><p><strong>Lugar: </strong>' + data[i].poblacion + ' </p></li>');
        $('#search-post-1').append('<li class="list-group-item"><p><strong>Precio: </strong>' + data[i].precio + '</p></li>');
        $('#search-post-1').append('<li class="list-group-item"><p><strong>Metros: </strong>' + data[i].metros + '</p></li>');
        $('#search-post-1').append('</ul>');
        $('#search-post-1').append("<div style='margin-top:15px;'>");
        $('#search-post-1').append("<a href='post.php?id=" + data[i].id_publicacion +  "'><button type='button' class='btn btn-primary btn-circle'> Saber más </button></a>");
        $('#search-post-1').append("</div>");
        $('#search-post-1').append("</div>");
        $('#search-post-1').append("</div>");
      }
      else {
        $('#search-post-1').append("<div class='card' id='columnas-par' style='margin-bottom: 15px;'>");
        $('#search-post-1').append("<h4 class='card-title'>" + data[i].titulo + "</h4> ");
        $('#search-post-1').append("<img class='card-img-top' src='" + data[i].imagen + "' alt='Card image' />");
        $('#search-post-1').append("<div class='card-block'>");
        $('#search-post-1').append('<ul class="list-group">');
        $('#search-post-1').append('<li class="list-group-item"><p><strong>Lugar: </strong>' + data[i].poblacion + ' </p></li>');
        $('#search-post-1').append('<li class="list-group-item"><p><strong>Precio: </strong>' + data[i].precio + '</p></li>');
        $('#search-post-1').append('<li class="list-group-item"><p><strong>Metros: </strong>' + data[i].metros + '</p></li>');
        $('#search-post-1').append('</ul>');
        $('#search-post-1').append("<div style='margin-top:15px;'>");
        $('#search-post-1').append("<a href='post.php?id=" + data[i].id_publicacion +  "'><button type='button' class='btn btn-primary btn-circle'> Saber más </button></a>");
        $('#search-post-1').append("</div>");
        $('#search-post-1').append("</div>");
        $('#search-post-1').append("</div>");
      }
    }
```

Figura 45: Lectura y muestra del JSON de los filtros para la columna de la izquierda

```
    else {
      var imagen = data[i].imagen;
      if ((imagen.indexOf(",") > -1) == true ) {
        var img_sep = imagen.split(",");
        var img1 = "";
        var img2 = "";
        img1 = img_sep[0];
        img2 = img_sep[1];

        $('#search-post-2').append("<div class='card' id='columnas-par' style='margin-bottom: 15px;'>");
        $('#search-post-2').append("<h4 class='card-title'>" + data[i].titulo + "</h4> ");
        $('#search-post-2').append("<img class='card-img-top' src='" + img1 + "' alt='Card image' />");
        $('#search-post-2').append("<div class='card-block'>");
        $('#search-post-2').append('<ul class="list-group">');
        $('#search-post-2').append('<li class="list-group-item"><p><strong>Lugar: </strong>' + data[i].poblacion + ' </p></li>');
        $('#search-post-2').append('<li class="list-group-item"><p><strong>Precio: </strong>' + data[i].precio + '</p></li>');
        $('#search-post-2').append('<li class="list-group-item"><p><strong>Metros: </strong>' + data[i].metros + '</p></li>');
        $('#search-post-2').append('</ul>');
        $('#search-post-2').append("<div style='margin-top:15px;'>");
        $('#search-post-2').append("<a href='post.php?id=" + data[i].id_publicacion +  "'><button type='button' class='btn btn-primary btn-circle'> Saber más </button></a>");
        $('#search-post-2').append("</div>");
        $('#search-post-2').append("</div>");
        $('#search-post-2').append("</div>");
      }
      else {
        $('#search-post-2').append("<div class='card' id='columnas-par' style='margin-bottom: 15px;'>");
        $('#search-post-2').append("<h4 class='card-title'>" + data[i].titulo + "</h4> ");
        $('#search-post-2').append("<img class='card-img-top' src='" + data[i].imagen + "' alt='Card image' />");
        $('#search-post-2').append("<div class='card-block'>");
        $('#search-post-2').append('<ul class="list-group">');
        $('#search-post-2').append('<li class="list-group-item"><p><strong>Lugar: </strong>' + data[i].poblacion + ' </p></li>');
        $('#search-post-2').append('<li class="list-group-item"><p><strong>Precio: </strong>' + data[i].precio + '</p></li>');
        $('#search-post-2').append('<li class="list-group-item"><p><strong>Metros: </strong>' + data[i].metros + '</p></li>');
        $('#search-post-2').append('</ul>');
        $('#search-post-2').append("<div style='margin-top:15px;'>");
        $('#search-post-2').append("<a href='post.php?id=" + data[i].id_publicacion +  "'><button type='button' class='btn btn-primary btn-circle'> Saber más </button></a>");
        $('#search-post-2').append("</div>");
        $('#search-post-2').append("</div>");
        $('#search-post-2').append("</div>");
      }
    }
  }
});
```

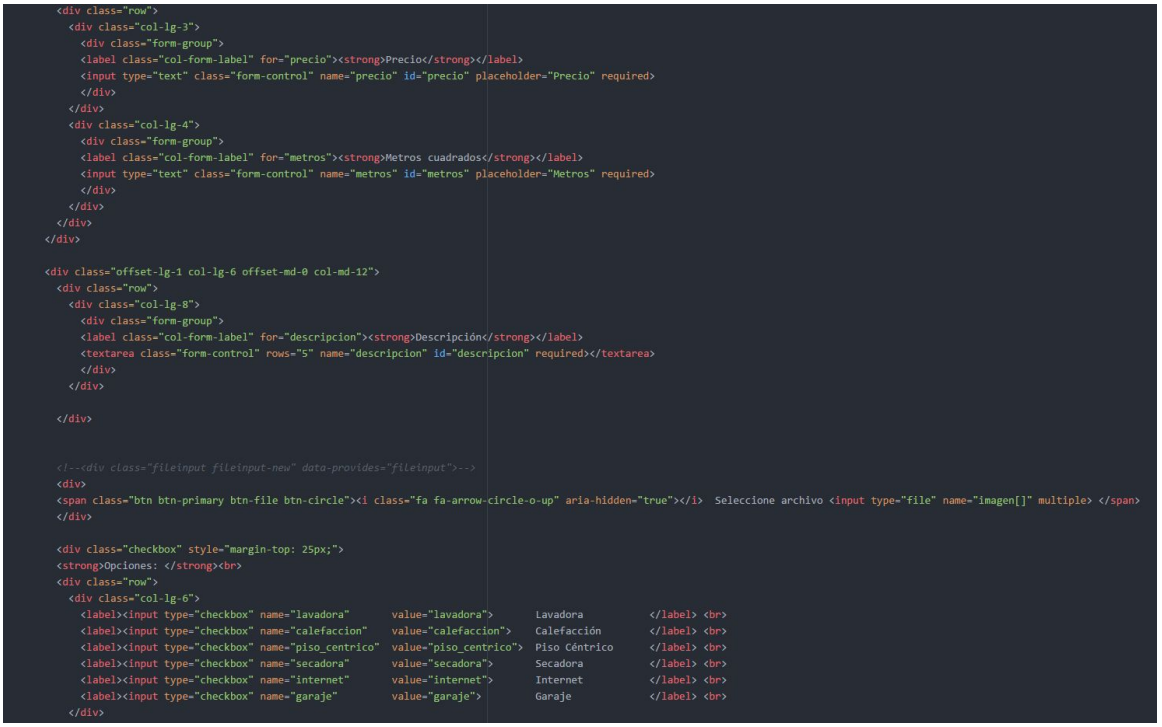
Figura 46: Lectura y muestra del JSON de los filtros para la columna de la derecha

6. Página para añadir un nuevo alquiler

Para añadir un nuevo alquiler a la aplicación web y por tanto, a la base de datos, se ha definido un nuevo fichero denominado *new_file.php*. En él, primero se define un formulario de tipo *POST* y *multipart* para poder llevar a cabo la recogida de los datos de los campos y el envío de los mismos a la base de datos. El formulario se muestra en las FIguras 47, 48 y 49:

```
<div class="row">
  <div class="col-lg-12"><h1 style="text-align:center;"> Publicar un nuevo alquiler </h1></div>
</div>

  <form enctype="multipart/form-data" method="POST" action="new_flat.php" style="margin-top:25px;">
    <div class="row">
      <div class="offset-lg-1 col-lg-4 offset-md-0 col-md-12">
        <div class="form-group">
        <label class="col-form-label" for="titulo"><strong>Título de la publicación</strong></label>
        <input type="text" class="form-control" name="titulo" id="titulo" placeholder="Título" required>
        </div>

        <div class="row">
          <div class="col-lg-5">
            <div class="form-group">
            <label class="col-form-label" for="calle"><strong>Calle</strong></label>
            <input type="text" class="form-control" name="calle" id="calle" placeholder="Calle" required>
            </div>
          </div>

          <div class="col-lg-2">
            <div class="form-group">
            <label class="col-form-label" for="localidad"><strong>Número</strong></label>
            <input type="text" class="form-control" name="numero" id="numero" placeholder="Nº" required>
            </div>
          </div>

          <div class="col-lg-2">
            <div class="form-group">
            <label class="col-form-label" for="localidad"><strong>Letra</strong></label>
            <input type="text" class="form-control" name="piso" id="piso" placeholder="Letra" required>
            </div>
          </div>

        </div>

        <div class="form-group">
        <label class="col-form-label" for="localidad"><strong>Población</strong></label>
        <input type="text" class="form-control" name="poblacion" id="poblacion" placeholder="Población" required>
        </div>
```

Figura 47: Formulario para insertar un nuevo alquiler 1



```
    <div class="row">
      <div class="col-lg-3">
        <div class="form-group">
        <label class="col-form-label" for="precio"><strong>Precio</strong></label>
        <input type="text" class="form-control" name="precio" id="precio" placeholder="Precio" required>
        </div>
      </div>
      <div class="col-lg-4">
        <div class="form-group">
        <label class="col-form-label" for="metros"><strong>Metros cuadrados</strong></label>
        <input type="text" class="form-control" name="metros" id="metros" placeholder="Metros" required>
        </div>
      </div>
    </div>
  </div>

  <div class="offset-lg-1 col-lg-6 offset-md-0 col-md-12">
    <div class="row">
      <div class="col-lg-8">
        <div class="form-group">
        <label class="col-form-label" for="descripcion"><strong>Descripción</strong></label>
        <textarea class="form-control" rows="5" name="descripcion" id="descripcion" required></textarea>
        </div>
      </div>

    </div>


    <!--<div class="fileinput fileinput-new" data-provides="fileinput">-->
    <div>
    <span class="btn btn-primary btn-file btn-circle"><i class="fa fa-arrow-circle-o-up" aria-hidden="true"></i>  Seleccione archivo <input type="file" name="imagen[]" multiple> </span>
    </div>

    <div class="checkbox" style="margin-top: 25px;">
    <strong>Opciones: </strong><br>
    <div class="row">
      <div class="col-lg-6">
        <label><input type="checkbox" name="lavadora"       value="lavadora">       Lavadora           </label> <br>
        <label><input type="checkbox" name="calefaccion"    value="calefaccion">    Calefacción        </label> <br>
        <label><input type="checkbox" name="piso_centrico"  value="piso_centrico">  Piso Céntrico      </label> <br>
        <label><input type="checkbox" name="secadora"       value="secadora">       Secadora           </label> <br>
        <label><input type="checkbox" name="internet"       value="internet">       Internet           </label> <br>
        <label><input type="checkbox" name="garaje"         value="garaje">         Garaje             </label> <br>
      </div>
```

Figura 48: Formulario para insertar un nuevo alquiler 2

```
      <div class="checkbox" style="margin-top: 25px;">
      <strong>Opciones: </strong><br>
      <div class="row">
        <div class="col-lg-6">
          <label><input type="checkbox" name="lavadora"       value="lavadora">       Lavadora           </label> <br>
          <label><input type="checkbox" name="calefaccion"    value="calefaccion">    Calefacción        </label> <br>
          <label><input type="checkbox" name="piso_centrico"  value="piso_centrico">  Piso Céntrico      </label> <br>
          <label><input type="checkbox" name="secadora"       value="secadora">       Secadora           </label> <br>
          <label><input type="checkbox" name="internet"       value="internet">       Internet           </label> <br>
          <label><input type="checkbox" name="garaje"         value="garaje">         Garaje             </label> <br>
        </div>
        <div class="col-lg-6">
          <label><input type="checkbox" name="ascensor"       value="ascensor">       Ascensor           </label> <br>
          <label><input type="checkbox" name="television"     value="television">     Televisión         </label> <br>
          <label><input type="checkbox" name="plancha"        value="plancha">        Plancha            </label> <br>
          <label><input type="checkbox" name="fumadores"      value="fumadores">      Fumadores          </label> <br>
          <label><input type="checkbox" name="transp_publico" value="transp_publico"> Transporte público </label> <br>
        </div>
      </div>
      </div>
    </div>


  <div class="col-lg-12 text-center" style="margin-top:25px;">
    <button type="submit" class="btn btn-primary btn-circle">Enviar</button>
  </div>

  </div>
</form>
```

Figura 49: Formulario para insertar un nuevo alquiler 3

Seguidamente, se procedió a escribir el código que realiza la inserción en la base de datos. Para ello, primero se abrió una nueva conexión a la base de datos, Se seteó el email de Facebook en el caso

que el alquiler fuese a ser añadido en la aplicación web por esta red social, y se pasó a obtener los path de las imágenes para que puedan ser añadidas al campo *imagen* de la tabla *pisos_publicados*. Las imágenes serán almacenadas en el servidor y en la base de datos irán los path para acceder a ellas, por ello, el path de una imagen será del estilo DBImages/*imagen.extensión* dónde DBImages será un directorio situado en el servidor para las imágenes. A través del procedimiento *move_upload_file()* la imagen se subirá al servidor remoto. Luego, antes de llevar a cabo la inserción por SQL se condicionó que, si hay dos imágenes a subir, entre ambas debe existir un delimitador (siendo en este caso una coma “,”) para poder diferenciar los path almacenándose en una variable denominada *$all_path*. En el caso de que sólo haya una imagen, el path se setea en la variable denominada *$all_path.* La Figura 50 muestra el código:

```php
if(isset($_SESSION['user_data']['email']))
  $_SESSION['email'] = $_SESSION['user_data']['email'];

$nombreimg = "";
$archivo = "";
$ruta = "";
$all_path = "";
$path_1 = "";
$path_2 = "";

if (isset($_FILES['imagen'])){

  $cantidad= count($_FILES["imagen"]["tmp_name"]);
  for ($i=0; $i<$cantidad; $i++) {

    if($i == 0) {
      $path_1 = "DBImages/" . $_FILES["imagen"]["name"][$i];
      move_uploaded_file($_FILES["imagen"]["tmp_name"][$i], 'DBImages/'.$_FILES["imagen"]["name"][$i]);
    }

    else {
      $path_2 = "DBImages/" . $_FILES["imagen"]["name"][$i];
      move_uploaded_file($_FILES["imagen"]["tmp_name"][$i], 'DBImages/'.$_FILES["imagen"]["name"][$i]);
    }

}
}

if ($path_2 == "")
  $all_path = $path_1;
else
  $all_path = $path_1 . "," . $path_2;
```

Figura 50: Código para el path de las imágenes

Finalmente, se construyó una consulta a la base de datos de tipo *INSERT INTO* especificando el nombre de todos los campos en el orden en el que se definió en la base de datos. Se utilizó el procedimiento *bind_param()* para evitar ataques de *inyección de SQL*. Para terminar, se ejecutó la consulta pudiendo así, enviar la información sin problemas. La Figura 51 muestra la consulta:

```
$query = "INSERT INTO pisos_publicados (email, titulo, descripcion, calle, numero, piso, poblacion, precio, metros, lavadora, secadora, calefaccion, television, internet, plancha,
                                        piso_centrico, garaje, transp_publico, ascensor, fumadores, imagen) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";
$aux = $conexion->prepare($query);
$aux->bind_param("sssssssiiiiiiiiiiiis",$_SESSION['email'],$_POST['titulo'],$_POST['descripcion'],$_POST['calle'],$_POST['numero'],$_POST['piso'],$_POST['poblacion'],$_POST['precio'],
                $_POST['metros'],$_POST['lavadora'],$_POST['secadora'],$_POST['calefaccion'],$_POST['television'],$_POST['internet'],$_POST['plancha'],$_POST['piso_centrico'],
                $_POST['garaje'],$_POST['transp_publico'],$_POST['ascensor'],$_POST['fumadores'],$all_path);
$aux->execute();
```

Figura 51: Consulta para insertar un alquiler a la base de datos

7. Página para mostrar el perfil del usuario

En cuanto a la página del perfil de usuario, ésta muestra información personal del usuario y las publicaciones que ha realizado junto con un botón para eliminarlas en el caso de que haya arrendado el piso. Para ello, se ha dividido la pantalla en dos secciones en la cual, a la izquierda se muestran los datos personales y a la derecha se muestra una tabla en la que cada fila muestra una publicación junto con un botón para que sea eliminada de la aplicación y de la base de datos.

En lo que respecta a la implementación, para mostrar los datos personales primero se debe considerar el tipo de login que ha hecho, si es por Google, si es por Facebook o si es por credenciales. En función de cada uno, se ha implementado el código dónde se comprueba qué tipo de sesión es a través de aquellos atributos especificados por las APIs de cada red social o, si la sesión es por credenciales.

La API de Google proporciona el nombre del usuario a través del parámetro *givenName*, la API de facebook a través del parámetro [‘user_data’][‘first_name’] y, en el caso de que sea por credenciales, se realiza una búsqueda en la base de datos dado que lo que se tiene para abrir la sesión es el email del usuario. La consulta funcionaría seleccionando el nombre sobre la tabla usuarios donde el email coincidiría con el email de la sesión actual. La Figura 52 muestra el código:

```
<div class="row">
  <div class="col-lg-4">
    <div class="title_login"><h2> Tus datos personales </h2> <hr> </div>
  <div>

  <ul class="list-group">
    <li class="list-group-item list-group-item-secondary"> <strong style="color: #C51162"> Nombre: </strong>
    <?php
    $nombre = "";
    if (isset($_SESSION['givenName']))
      $nombre = $_SESSION['givenName'];
    else {
      if (isset($_SESSION['name'])) {
        $nombre = $_SESSION['user_data']['first_name'];
      }
      else {
        $db = new DataBase();
        $conexion = $db->getConection();

        $email_ = $_SESSION['email'];

        $query = "SELECT nombre FROM usuarios WHERE email = '".$email_."'";
        $resultado = mysqli_query($conexion,$query); // Se ejecuta la consulta

        while($row = mysqli_fetch_assoc($resultado))  // Se busca el nombre y se almacena
          $nombre =  $row['nombre'];
        }
      }
      echo $nombre;
    ?>
    </li>
```

Figura 52: Obtención del nombre del usuario en el *Perfil*

Para mostrar el apellido del usuario, se ha llevado a cabo la misma lógica que para obtener el nombre sin embargo, los identificadores de los parámetros varían. Para mostrar desde la API de Google el apellido se debe utilizar el parámetro *familyName*, para mostrarlo desde la API de Facebook se utiliza el parámetro [‘user_data’[‘last_name’] y para el apellido desde el login por credenciales, se utiliza una consulta a la base de datos seleccionando el apellido de la tabla usuarios donde el email coincida con el email de la sesión actual. La Figura 53 muestra el código:

```
<li class="list-group-item list-group-item-secondary"> <strong style="color: #C51162"> Apellido: </strong>
<?php
$apellido = "";
if (isset($_SESSION['familyName']))
  $apellido = $_SESSION['familyName'];
else {
  if (isset($_SESSION['lastName'])) {
    $apellido = $_SESSION['user_data']['last_name'];
  }
  else {
    $db = new DataBase();
    $conexion = $db->getConection();

    $email_ = $_SESSION['email'];

    $query = "SELECT apellidos FROM usuarios WHERE email = '".$email_."'";
    $resultado = mysqli_query($conexion,$query); // Se ejecuta la consulta

    while($row = mysqli_fetch_assoc($resultado))  // Se busca el apellido y se almacena
      $apellido =  $row['apellidos'];
    }
  }
  echo $apellido;
  ?>
  </li>
```

Figura 53: Obtención del apellido del usuario en el *Perfil*

Finalmente, para mostrar el email del usuario, al estar seteado en la variable de sesión actual, simplemente con mostrarlo basta. La Figura 54 muestra el código:

```
<li class="list-group-item list-group-item-secondary"> <strong style="color: #C51162"> Email: </strong>
<?php
$email = "";
if (isset($_SESSION['email']))
  $email = $_SESSION['email'];
else {
  if (isset($_SESSION['user_data']['email'])) {
    $email = $_SESSION['user_data']['email'];
  }
  else {
    $email = $_SESSION['email'];
  }
}
echo $email;
?>
</li>
```

Figura 54: Obtención del email del usuario en el *Perfil*

Por otra parte, para mostrar la tabla en la que se encuentran las publicaciones realizadas por el usuario y que están presentes en la base de datos, primero se ha definido una tabla a través de HTML como muestra la Figura 55:

```
<div class="col-sm-12 col-xs-12 col-lg-6">
  <div class="title_login"><h2> Tus publicaciones </h2> <hr> </div>
  <table class="table table-hover">
  <thead class="thead-light">
    <tr>
      <th style="color: #C51162"> Título    </th>
      <th style="color: #C51162"> Población </th>
      <th style="color: #C51162"> Precio    </th>
      <th style="color: #C51162"> Metros    </th>
      <th style="color: #C51162"> Editar    </th>
    </tr>
  </thead>
```

Figura 55: Cabecera de la tabla de publicaciones en el *Perfil*

El cuerpo de la tabla lo construirá una consulta a la base de datos de tal forma que, obteniendo el email seteado en la variable de sesión, se haga una consulta seleccionando los campos de título, población, precio y metros a la base de datos sobre la tabla *pisos_publicados* dónde el email de la publicación debe coincidir con el email de la sesión actual. Además, junto con cada publicación se mostrará un botón para eliminar dicha publicación si el usuario lo considera. La figura 56 muestra el código fuente:

```
<tbody class="tbody">
<?php
$db = new DataBase();
$conexion = $db->getConection();

$email = "";
if (isset($_SESSION['email']))
  $email = $_SESSION['email'];
else {
  if (isset($_SESSION['user_data']['email'])) {
    $email = $_SESSION['user_data']['email'];
  }
  else {
    $email = $_SESSION['email'];
  }
}

$sql = "SELECT * FROM pisos_publicados WHERE email = '".$email."'";
if(!$result = mysqli_query($conexion, $sql)) die();

mysqli_set_charset($conexion, "utf8");
while($row = mysqli_fetch_array($result)) { // Se muestran las publicaciones del usuario actual en su perfil
  echo "<tr>";
  echo "<td>" . $row['titulo']    . "</td>";
  echo "<td>" . $row['poblacion'] . "</td>";
  echo "<td>" . $row['precio']    . " euros</td>";
  echo "<td>" . $row['metros']    . " m/2 </td>";
  echo "<td> <button type='button' class='btn btn-primary btn-circle' id='" . $row['id_publicacion'] . "'> Eliminar </button> </td>";
  echo "</tr>";
}
?>
</tbody>
</table>
```

Figura 56: Contenido de la tabla de publicaciones en el *Perfil*

Como se puede apreciar, el botón contiene un id que coincide con el campo **id_publicacion** de la tabla *pisos_publicados*. El identificador sirve para que, una vez que se haga click en él, se haga una consulta a la base de datos realizando un *delete* sobre el identificador pasado. El código se muestra a continuación en la Figura 57:

```php
<?php

require_once "db/database.php";
require_once "config.php";

$db = new DataBase();
$conexion = $db->getConection();

$post_id = $_POST['id'];

$query = mysqli_query($conexion, "DELETE FROM pisos_publicados WHERE id_publicacion = $post_id");


mysqli_close($conexion);

?>
```

Figura 57: Consulta a la base de datos para eliminar una publicaciones en el *Perfil*

8. Página para mostrar toda la información de un alquiler

Para mostrar toda la información de un alquiler en el sitio web, se ha implementado un fichero denominado *post.php* que realiza la tarea. Primero, se definió la estructura del HTML para mostrar la información que se encuentra almacenada en la base de datos. La Figura 58 muestra el código del HTML:

```
<div class="row">
  <div class="col-lg-5 col-md-12 col-sm-12 col-xs-12">
    <ol class="breadcrumb">
      <li class="breadcrumb-item">Está en:<a href="index.php"> Pisos </a></li>
      <li class="breadcrumb-item active">Información del piso </li>
    </ol>
  </div>
</div>

</div>

<div class="row">
<div class="col-lg-7">
  <h3 style="text-align: center;">Información de la publicación</h3>
  <div class="row" style="margin-top:25px;">
    <div class="col-lg-12">
      <div class="row">
        <div class="offset-lg-1 col-lg-5" id="post-impar-1">
        </div>
        <div class="offset-lg-1 col-lg-5" id="post-impar-2">
        </div>
      </div>
    </div>
  </div>

    <div class="row" style="margin-top:25px; margin-bottom:25px;">
      <div class="offset-lg-2 col-lg-8 offset-md-0" id="header-list">
      </div>
    </div>
    <div class="row">
      <div class="offset-lg-2 col-lg-4 offset-md-0" id="post-par-1">
      </div>
      <div class="col-lg-4" id="post-par-2">
      </div>
    </div>

  </div>
```

Figura 58: Interfaz para mostrar la información de un alquiler

Primero, se ha definido la estructura para mostrar la o las imágenes que el usuario haya insertado cuando ha añadido la o las imágenes. Seguidamente, se ha definido otra estructura para poder mostrar la información en sí. Dicha información se mostrará a través de una lista que se dividirá en dos en la pantalla. Por otra parte, se ha definido un formulario mostrado en la Figura 59 que permite al usuario ponerse en contacto con el arrendador del alquiler.

```
<div class="col-lg-5">
  <h3 style="text-align: center;">¿Te interesa?</h3>
  <div class="offset-lg-3 col-lg-8 offset-md-0">
    <p style="text-align: justify; margin-top:35px;"> Si lo deseas, puedes enviar un mensaje al arrendador del piso. Escríbele un mensaje rellenando el formulario y te llegará la respuesta al correo con el que has ac
  </div>

  <div class="row">
    <div class="offset-lg-3 col-lg-8 offset-md-0" style="margin-top:35px;">
      <form method="post">
        <div class="form-group">
          <label for="formGroupExampleInput">Asunto:</label>
          <input type="text" class="form-control" name="asunto">
        </div>
        <div class="form-group">
          <label for="formGroupExampleInput2">Mensaje:</label>
          <textarea class="form-control" rows="5" name="mensaje"></textarea>
        </div>
        <div class="text-center">
          <input type="submit" value="Enviar" class="btn btn-primary">
        </div>

      </form>
    </div>

  </div>
</div>
```

Figura 59: Formulario para ponerse en contacto con el arrendador

Para que la información se pueda visualizar, primero se debe llevar a cabo el código que realiza las consultas a la base de datos. Para ello, se ha obtenido el email seteado a través de la variable de sesión **$_SESSION** condicionando como se ha explicado en otras ocasiones que el email sea a través de credenciales, Google o Facebook. Seguidamente, para que la página abra el post correspondiente, se ha obtenido de la URI el id pasado por parámetros y sobre él se han realizado las consultas a la base de datos a través de un **SELECT *** de la tabla *pisos_publicados* dónde el **id_publicacion** coincidiese con el id que se ha recuperado de la URI. El contenido se ha enviado a un fichero JSON denominado *post.json* del cual se leeran los datos a través de JavaScript y se mostrarán en la interfaz. Por otra parte, para que el envío del mensaje fuese posible, se ha definido la función de PHP denominada *mail()* en la cual se han pasado los parámetros necesarios, Las Figura 60, 61 y 62 muestran el código comentado:

```php
require_once "db/database.php";
require_once "config.php";

if (isset($_POST['asunto']))
  $asunto   = $_POST['asunto'];

$email_destino = "";
$email_saliente = "";

if (isset($_SESSION['email']))
  $email_saliente = $_SESSION['email'];
else {
  if (isset($_SESSION['user_data']['email']))
    $email_saliente = $_SESSION['user_data']['email'];
  else
    $email_saliente = $_SESSION['email'];
}

if (isset($_POST['mensaje']))
  $mensaje  = "Mail para contestar: " . $email_saliente . " " . $_POST['mensaje'];

/*Obtenemos el ID pasado por parámetros dividiendo la cadena y convirtiendo el tipo de dat
de string a int */
$url           = $_SERVER["REQUEST_URI"];
$subcadena_url = explode("=",$url);
$id_pub        = $subcadena_url[1];
$id_pub_int    = (int)($id_pub);


$db = new DataBase();
$conexion = $db->getConection();

$query = "SELECT email FROM pisos_publicados WHERE id_publicacion = '".$id_pub_int."'";
$resultado = mysqli_query($conexion,$query); // Se ejecuta la consulta
mysqli_set_charset($conexion, "utf8"); //formato de datos utf8

while($row = mysqli_fetch_assoc($resultado))  // Se busca el email y se almacena
  $email_destino =  $row['email'];

$header = "From: " . $email_saliente . "" . "\r\n";

mail($email_destino,$asunto,$mensaje,$header);
```

Figura 60: Código para la información del alquiler 1

```php
mail($email_destino,$asunto,$mensaje,$header);

$close = mysqli_close($conexion) or die("Ha sucedido un error inexperado en la desconexion de la base de datos");

/* Mostrar datos */

$db       = new DataBase();
$conexion = $db->getConection();

$sql    = "SELECT titulo, descripcion, calle, numero, piso, poblacion, precio, metros, lavadora, secadora, calefaccion, television, internet, plancha, piso_centrico, garaje, transp_publico, ascensor, fumadores, image
$result = mysqli_query($conexion,$sql); // Se ejecuta la consulta
mysqli_set_charset($conexion,"utf8");

while($row = mysqli_fetch_assoc($result)) {
  $titulo         = $row['titulo'];
  $descripcion    = $row['descripcion'];
  $calle          = $row['calle'];
  $numero         = $row['numero'];
  $piso           = $row['piso'];
  $poblacion      = $row['poblacion'];
  $precio         = $row['precio'];
  $metros         = $row['metros'];
  $lavadora       = $row['lavadora'];
  $secadora       = $row['secadora'];
  $calefaccion    = $row['calefaccion'];
  $television     = $row['television'];
  $internet       = $row['internet'];
  $plancha        = $row['plancha'];
  $piso_centrico  = $row['piso_centrico'];
  $garaje         = $row['garaje'];
  $transp_publico = $row['transp_publico'];
  $ascensor       = $row['ascensor'];
  $fumadores      = $row['fumadores'];
  $imagen         = $row['imagen'];


  $post[] = array('titulo'=> $titulo, 'descripcion'=> $descripcion, 'calle'=> $calle, 'piso'=> $piso, 'numero'=> $numero, 'poblacion'=> $poblacion, 'precio'=> $precio, 'metros'=> $metros,
                'lavadora'=> $lavadora, 'secadora'=> $secadora, 'calefaccion'=> $calefaccion, 'television'=> $television, 'internet'=> $internet,
                'plancha'=> $plancha, 'piso_centrico'=> $piso_centrico, 'garaje'=> $garaje, 'transp_publico'=> $transp_publico, 'ascensor'=> $ascensor, 'fumadores'=> $fumadores, 'imagen' => $imagen);
}
$close = mysqli_close($conexion)
or die("Ha sucedido un error inexperado en la desconexion de la base de datos");
```

Figura 61: Código para la información del alquiler 2

```php
//Creamos el JSON
$json_string = json_encode($post);

// Volcamos los datos en un fichero
$file = 'post.json';
file_put_contents($file, $json_string);

?>
```

Figura 62: Código para la información del alquiler 3

Finalmente, para poder mostrar la información en cuanto a los datos que se encontraban en la base de datos, se ha implementado el código mostrado en las Figura 63, 64, 65:

```
$.getJSON( "/Proyecto/post.json", function( data ) {

  String(data);

  for (var i = 0; i < data.length; i++) {
      $('#post-par-1').append("<ul class='list-group'>");
      $('#header-list').append("<li class='list-group-item title-post'> <strong> <p> Datos del post </p> </strong></li>");
      $('#post-par-1').append("<li class='list-group-item'> <strong> Título: </strong>" + data[i].titulo + "</li>");
      $('#post-par-1').append("<li class='list-group-item'> <strong> Descripción: </strong>" + data[i].descripcion + "</li>");
      $('#post-par-1').append("<li class='list-group-item'> <strong> Calle: </strong>" + data[i].calle + "</li>");
      $('#post-par-1').append("<li class='list-group-item'> <strong> Piso: </strong>" + data[i].piso + "</li>");
      $('#post-par-1').append("<li class='list-group-item'> <strong> Número: </strong>" + data[i].numero + "</li>");
      $('#post-par-1').append("<li class='list-group-item'> <strong> Población: </strong>" + data[i].poblacion + "</li>");
      $('#post-par-1').append("<li class='list-group-item'> <strong> Precio: </strong>" + data[i].precio + " euros/mes </li>");
      $('#post-par-1').append("<li class='list-group-item'> <strong> Metros cuadrados: </strong>" + data[i].metros + "</li>");

      if (data[i].lavadora != null)
        $('#post-par-1').append("<li class='list-group-item'> <strong> Lavadora: </strong> Sí </li>");
      else
        $('#post-par-1').append("<li class='list-group-item'> <strong> Lavadora: </strong> No </li>");

      if (data[i].secadora != null)
        $('#post-par-1').append("<li class='list-group-item'> <strong> Secadora: </strong> Sí </li>");
      else
        $('#post-par-1').append("<li class='list-group-item'> <strong> Secadora: </strong> No </li>");

      if (data[i].calefaccion != null)
        $('#post-par-2').append("<li class='list-group-item'> <strong> Calefacción: </strong> Sí </li>");
      else
        $('#post-par-2').append("<li class='list-group-item'> <strong> Calefacción: </strong> No </li>");

      if (data[i].television != null)
        $('#post-par-2').append("<li class='list-group-item'> <strong> Televisión: </strong> Sí </li>");
      else
        $('#post-par-2').append("<li class='list-group-item'> <strong> Televisión: </strong> No </li>");

      if (data[i].internet != null)
        $('#post-par-2').append("<li class='list-group-item'> <strong> Internet: </strong> Sí </li>");
      else
        $('#post-par-2').append("<li class='list-group-item'> <strong> Internet: </strong> No </li>");

      if (data[i].plancha != null)
        $('#post-par-2').append("<li class='list-group-item'> <strong> Plancha: </strong> Sí </li>");
      else
        $('#post-par-2').append("<li class='list-group-item'> <strong> Plancha: </strong> No </li>");
```

Figura 63: Código para la mostrar la información del alquiler 1

```
if (data[i].piso_centrico != null)
  $('#post-par-2').append("<li class='list-group-item'> <strong> Piso Céntrico: </strong> Sí </li>");
else
  $('#post-par-2').append("<li class='list-group-item'> <strong> Piso Céntrico: </strong> No </li>");

if (data[i].garaje != null)
  $('#post-par-2').append("<li class='list-group-item'> <strong> Garaje: </strong> Sí </li>");
else
  $('#post-par-2').append("<li class='list-group-item'> <strong> Garaje: </strong> No </li>");

if (data[i].transp_publico != null)
  $('#post-par-2').append("<li class='list-group-item'> <strong> Transporte público a 10 minutos: </strong> Sí </li>");
else
  $('#post-par-2').append("<li class='list-group-item'> <strong> Transporte público a 10 minutos: </strong> No </li>");

if (data[i].ascensor != null)
  $('#post-par-2').append("<li class='list-group-item'> <strong> Ascensor: </strong> Sí </li>");
else
  $('#post-par-2').append("<li class='list-group-item'> <strong> Ascensor: </strong> No </li>");

if (data[i].fumadores != null)
  $('#post-par-2').append("<li class='list-group-item'> <strong> Hay fumadores: </strong> Sí </li>");
else
  $('#post-par-2').append("<li class='list-group-item'> <strong> Hay fumadores: </strong> No </li>");


$('#post-par-1').append("</ul>");
$('#post-par-1').append("</ul>");

var imagen = data[i].imagen;
if ((imagen.indexOf(",") > -1) == true ) {
  console.log("ENT");
  var img_sep = imagen.split(",");
  var img1 = "";
  var img2 = "";
  img1 = img_sep[0];
  img2 = img_sep[1];

$('#post-impar-1').append("<div class='card'>");
$('#post-impar-1').append("<img class='card-img-top' src='" + img1 + "'/>")
$('#post-impar-1').append("</div>");
$('#post-impar-2').append("<div class='card'>");
$('#post-impar-2').append("<img class='card-img-top' src='" + img2 + "'/>")
$('#post-impar-2').append("</div>");
```

Figura 64: Código para la mostrar la información del alquiler 2

```
      $('#post-par-1').append("</ul>");
      $('#post-par-1').append("</ul>");

      var imagen = data[i].imagen;
      if ((imagen.indexOf(",") > -1) == true ) {
        console.log("ENT");
        var img_sep = imagen.split(",");
        var img1 = "";
        var img2 = "";
        img1 = img_sep[0];
        img2 = img_sep[1];

      $('#post-impar-1').append("<div class='card'>");
      $('#post-impar-1').append("<img class='card-img-top' src='" + img1 + "'/>")
      $('#post-impar-1').append("</div>");
      $('#post-impar-2').append("<div class='card'>");
      $('#post-impar-2').append("<img class='card-img-top' src='" + img2 + "'/>")
      $('#post-impar-2').append("</div>");
    }
    else {
      $('#post-impar-1').append("<div class='card'>");
      $('#post-impar-1').append("<img class='card-img-top' src='" + data[i].imagen + "'/>")
      $('#post-impar-1').append("</div>");
    }

  }

});
```

Figura 65: Código para la mostrar la información del alquiler 3

Para llevar a cabo la tarea, a través del procedimiento *$.getJSON()* se ha leído el JSON que contiene la información sobre la publicación que había sido obtenida a través de la consulta a la base de datos. El JSON obtenido se ha transformado en un String (cadena de texto) para que pueda ser leído sin ningún problema. Seguidamente, se procedió a recorrer dicho JSON empleando para ello la sentencia iterativa *for*. Para mostrar el contenido, se ha hecho uso del selector de jQuery sobre el HTML y se ha empleado la función *append()* que setea en el HTML. Para mostrar los checkbox se ha condicionado si el contenido se encuentra nulo o no. En caso de nulo, se mostrará en la pantalla *no* y en caso afirmativo, se mostrará en la pantalla *sí*. Para las imágenes, se ha condicionado si se ha insertado una o dos imágenes. En cualquier caso, se ha cogido el valor devuelto por la base de datos (que si se recuerda, previamente se comentó que los path de las imágenes se encuentran separados por una ",", por lo que la cadena se debe dividir) y se ha dividido almacenándose en dos variables que son mostradas en la interfaz por dos etiquetas *img*. En caso de que no haya una coma, simplemente se mostrará el valor del campo que es el path de la imagen en el servidor.

9. Página para mostrar como funciona la aplicación web

Con tal de que la aplicación web pueda ser utilizada por todos aquellos usuarios que accedan, se ha implementado un enlace que muestra el funcionamiento a través de texto de todas las opciones que presta la aplicación web además del propósito de la aplicación. Para ello, se ha dividido la pantalla en dos secciones a través de las columnas de bootstrap dónde la columna de la derecha corresponde al texto explicando el funcionamiento y la columna de la izquierda explicando el propósito de la aplicación web.

Dado que simplemente la página trata de mostrar texto, no se ha hecho ninguna implementación en PHP ya que simplemente, a través de texto, etiquetas HTML y CSS se ha conseguido llevar a cabo la tarea.

La Figura 66 muestra el código que implementa lo comentado al principio:

```
<div class="row">
  <div class="col-lg-6 col-md-12">
    <div class="row">
      <div class="col-lg-12">
        <h1 style="text-align:center; margin-top:35px;">Funcionamiento de la aplicación</h1>
      </div>
      <div class="offset-lg-1 col-lg-10" style="margin-top:25px;">
        <p style="text-align:justify">Una vez que se ha accedido mediante la opción que el usuario considere (Correo electrónico y contraseña, acceso por google o acceso por facebook) entrará en la pantalla inicial do
        <p style="text-align:justify">Si el usuario desea llevar a cabo una búsqueda mediante filtros, el usuario deberá acceder a la sección de Búsqueda del menú superior y una vez ahí, escoger el filtro que consider
        <p style="text-align:justify"><i class="fa fa-angle-double-right" aria-hidden="true"></i> Acceder a la información de un piso a través del botón de Saber más que acompaña a cada publicación.</p>
        <p style="text-align:justify"><i class="fa fa-angle-double-right" aria-hidden="true"></i> Llevar a cabo otra búsqueda haciendo uso del formulario de nuevo.</p>
        <p style="text-align:justify"><i class="fa fa-angle-double-right" aria-hidden="true"></i> Acceder a otra parte de la aplicación a través del menú superior. </p>
        <p style="text-align:justify">Si el usuario, una vez encontrándose en la pantalla que muestra los últimos 10 pisos desea acceder a la información de un piso en cuestión, deberá hacer click en el botón Saber má
        <p style="text-align:justify">Si el usuario desea realizar una nueva publicación de un piso, deberá acceder al menú superior y seleccionar la opción de Añadir piso donde deberá rellenar el formulario y hacer c
        <p style="text-align:justify">Si el usuario desea consultar o eliminar las publicaciones que ha hecho o consultar sus datos personales, deberá acceder a la sección del menú Perfil. En caso contrario, deberá se
        <p style="text-align:justify">En el caso de que el usuario desee ponerse en contacto con el arrendador del piso, deberá acceder a la información del post al hacer click en Saber más de una publicación y rellen
      </div>
    </div>

  </div>
  <div class="col-lg-6 col-md-12">
    <div class="row">
      <div class="col-lg-12">
        <h1 style="text-align:center; margin-top:35px;">Propósito de la aplicación</h1>
      </div>
      <div class="offset-lg-1 col-lg-10" style="margin-top:25px;">
      El propósito de la aplicación web es la de poder compartir información sobre alquileres de habitación en pisos compartidos al igual que la búsqueda de dicha información. Para ello, los usuarios a través de una c
      </div>
    </div>
  </div>
</div>
```

Figura 66: HTML de la página de *¿Como funciona?*

10. Página para mostrar el Aviso Legal

Dado que la aplicación web almacena ciertos datos de los usuarios, es obligatorio que cuente con una sección que muestre el aviso legal construido bajo la Ley Orgánica de Protección de Datos. Para su implementación, se han definido dos columnas en HTML a través del sistema de rejillas de BootStrap dónde en la columna de la izquierda se muestra el cumplimiento de la normativa y, en la parte de la derecha, el ejercicio de derechos ARCO.
La Figura 67 muestra el código HTML definido para implementar lo definido anteriormente:

```
<div class="row">
  <div class="col-lg-12">
    <h1 style="text-align:center; margin-top:15px;">Aviso Legal</h1>
  </div>
</div>

<div class="row">
  <div class="col-lg-6">
    <h2 style="text-align:center; margin-top:35px;">Cumplimiento de la normativa</h2>
    <div class="row">
      <div class="offset-lg-2 col-lg-8" style="margin-top:35px;">
        <p style="text-align:justify; margin-top:15px;"><i>CompartirPiso</i> cumple con las directrices de la Ley Orgánica 15/1999 de 13 de diciembre de Protección de Datos de Carácter Personal, el Real Decreto 1720/2
        <p style="text-align:justify; margin-top:15px;">En cumplimiento de lo establecido en la LOPD, le informamos que los datos suministrados, así como aquellos datos derivados de su navegación, podrán ser almacenad
        <p style="text-align:justify; margin-top:15px;">Adicionalmente, el USUARIO consiente el tratamiento de sus datos con la finalidad de informarles, por cualquier medio, incluido el correo electrónico, de product
        <p style="text-align:justify; margin-top:15px;">En caso de no autorizar el tratamiento de sus datos con la finalidad señalada anteriormente, el USUARIO podrá ejercer su derecho de oposición al tratamiento de s
      </div>
    </div>
  </div>

  <div class="col-lg-6">
    <h3 style="text-align:center; margin-top:35px;">Ejercicio de Derechos ARCO: Acceso, Rectificación, Cancelación y Oposición</h3>
    <div class="row">
      <div class="offset-lg-1 col-lg-10" style="margin-top:35px;">
        <p style="text-align:justify; margin-top:15px;">Aquellas personas físicas que hayan facilitado sus datos a través de la web CompartirPiso, podrán dirigirse al titular de la misma con el fin de poder ejercitar
        <p style="text-align:justify; margin-top:15px;">El interesado podrá ejercitar sus derechos ARCO , a través del correo electrónico: compartirpisoproyecto@gmail.com </p>
      </div>
    </div>
  </div>

</div>
```

Figura 67: HTML de la página *Aviso Legal*

11. Página de para el logout

El logout, aunque no se visualice en la interfaz como un página y por lo tanto no haya código HTML definido, se manifiesta haciendo click en el enlace de *logout* del menú superior y que redirecciona directamente al registro/login de la aplicación. La implementación de este fichero se basa en destruir la sesión actual construida en el Login. El código se muestra en la Figura 68:

```php
<?php
  require_once "config.php";
  unset($_SESSION['access_token']);
  $google_client->revokeToken();
  session_destroy();                    //Eliminación de la sesión
  header('Location: login.php');  // Se redirecciona al login de la aplicación
  exit();
?>
```

Figura 68: Estructura del código de *Cerrar Sesión*

Capítulo 5: Demostración

1. Demostración para la página de Registro / Login

A continuación se muestra la interfaz en la Figura 69 y una descripción acerca de la página para el registro y el login de los usuarios en la aplicación web de CompartirPiso.

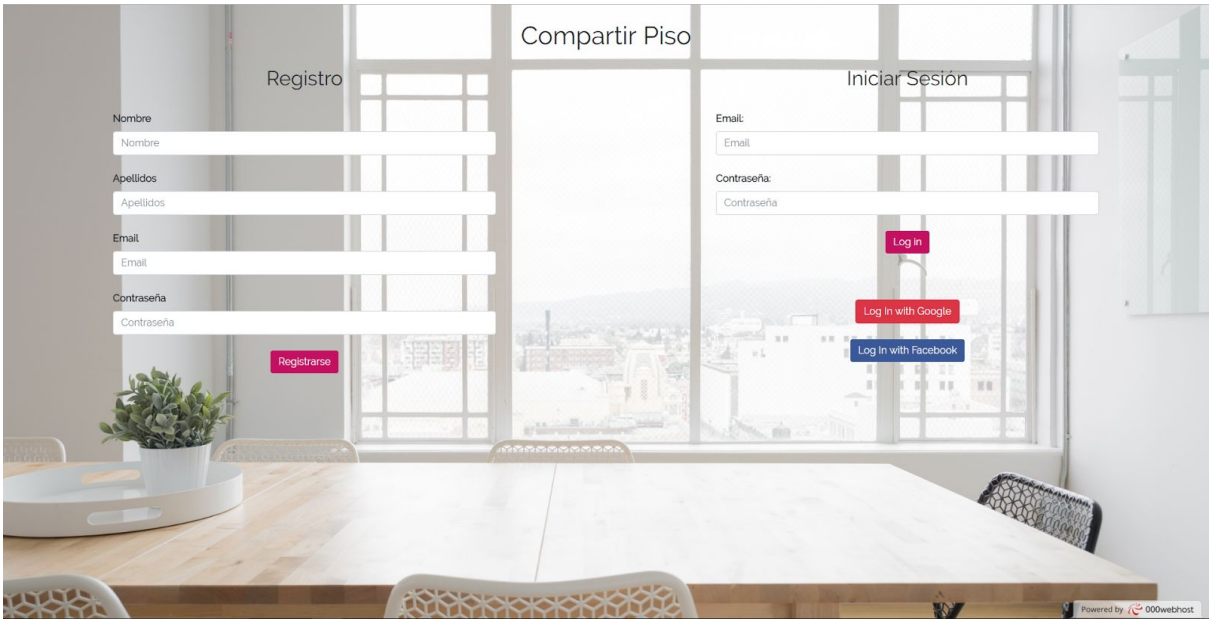


Figura 69: Interfaz para el registro / login

1.1 Registro

Para el registro de los usuarios, éstos deberán completar el formulario que se muestra y seguidamente, hacer click en el botón de *Registrarse*. En el caso de que se quiera acceder por una red social, bastará hacer click en el botón de la que se decida y darle permisos a la aplicación para acceder a los datos y redirigirse a *CompartirPiso*. En el caso del servidor remoto, sólo se encuentra disponible la opción para realizar el Login por Google, tal y como se muestra en las Figuras 70, 71 y 72.

1.2 Inicio de Sesión (Login)

Para iniciar sesión en la aplicación web, se deberá decidir por qué método entrar. En el caso de credenciales, se deberá rellenar el formulario y hacer click en el botón de *Log In*. En el caso de que se quiera acceder por una red social, bastará hacer click en el botón de la que se decida y automáticamente entrará. En el caso del servidor remoto, dado que las opciones de Facebook y login por credenciales están desactivadas, sólo se podrá acceder por Google.

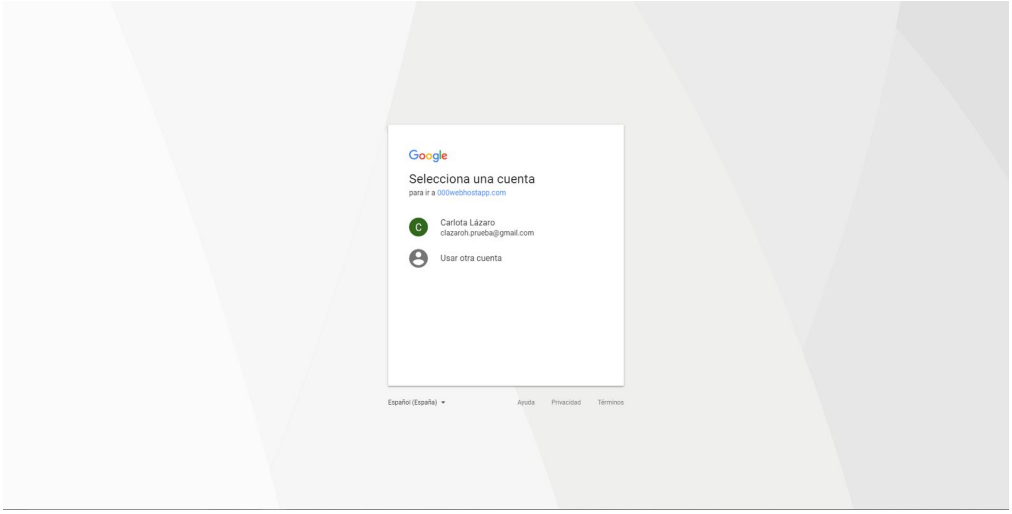


Figura 70: Acceso por Google 1

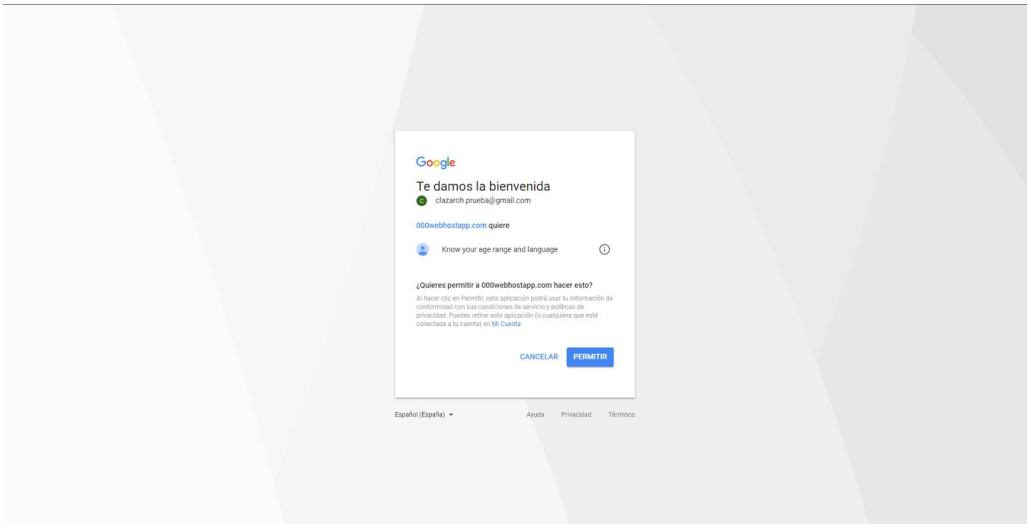


Figura 71: Acceso por Google 2

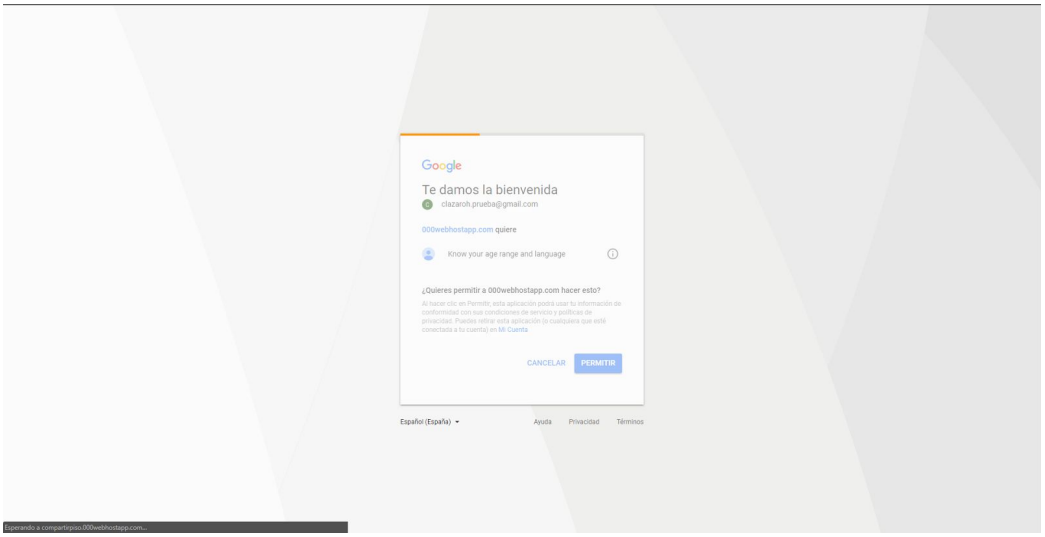


Figura 72: Acceso por Google 3

2. Demostración para la página de Pisos

A continuación se muestra la interfaz en la Figura 73 y una descripción acerca de la página para visualizar los 10 últimos alquileres de la aplicación web *CompartirPiso*.

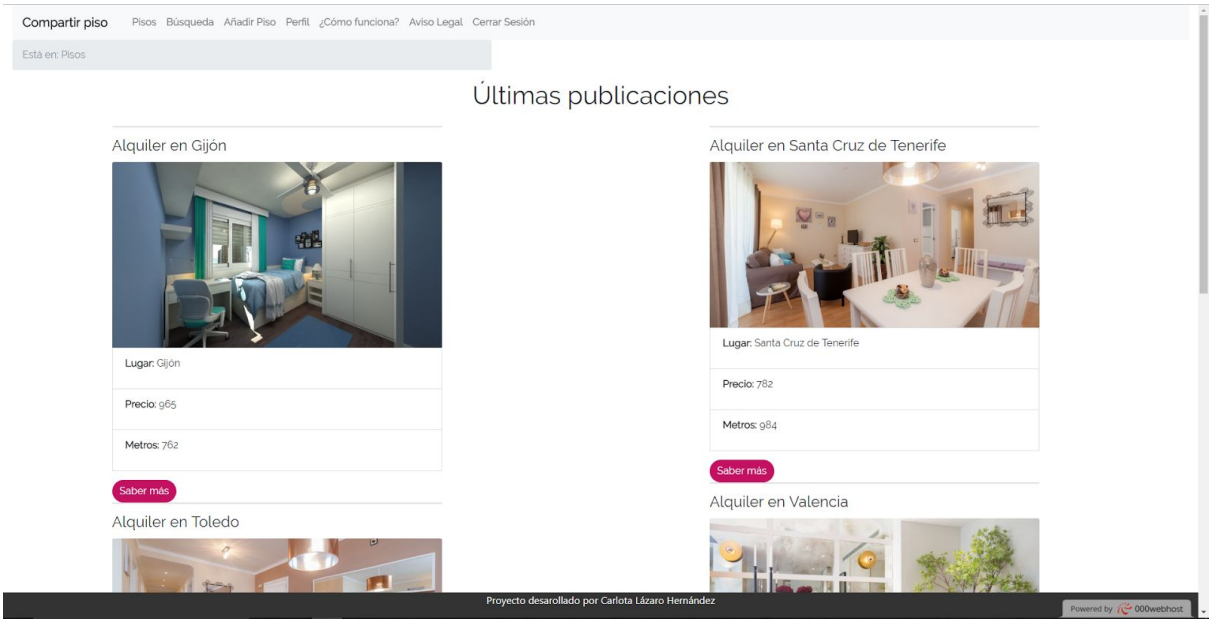


Figura 73: Interfaz para visualizar todos los alquileres

Una vez que el usuario haya iniciado sesión en la aplicación web, ésta le redirigirá a la página que muestra todos los pisos. Para acceder a dicha página desde el menú superior, se puede hacer click tanto en el enlace de *Compartir piso* como en el de *Pisos*.

1.1 ¿Como funciona?

La página le mostrará automáticamente al usuario los 10 últimos alquileres que han sido añadidos a la aplicación y una pequeña descripción del alquiler (el lugar, su precio y los metros cuadrados). En el caso de que al usuario le interese alguno de los alquileres que se muestran en la pantalla, para conocer más información sobre dicho alquiler deberá presionar el botón “Saber más” el cual le redirigirá automáticamente a una nueva página que mostrará toda la información relativa a dicho alquiler.

3. Demostración para la página de añadir de un nuevo alquiler

A continuación se muestra la interfaz en la Figura 74 y una descripción acerca de la página para añadir un nuevo alquier la aplicación web *CompartirPiso*.

Figura 74: Interfaz para añadir un nuevo alquiler

En el caso de que el usuario sea el arrendador de un alquiler, éste podrá añadirlo a la aplicación web *Compartir piso* para su publicación.

1.1 ¿Como funciona?

Una vez que el usuario haya iniciado sesión, éste deberá hacer click en la opción “Añadir Piso” y, deberá rellenar los campos propuestos tal y como se muestra en la Figura 74. La Figura 75 muestra como se debe seleccionar las imágenes para que aparezcan. Una vez que haya realizado la tarea, deberá hacer click en el botón “Enviar”.

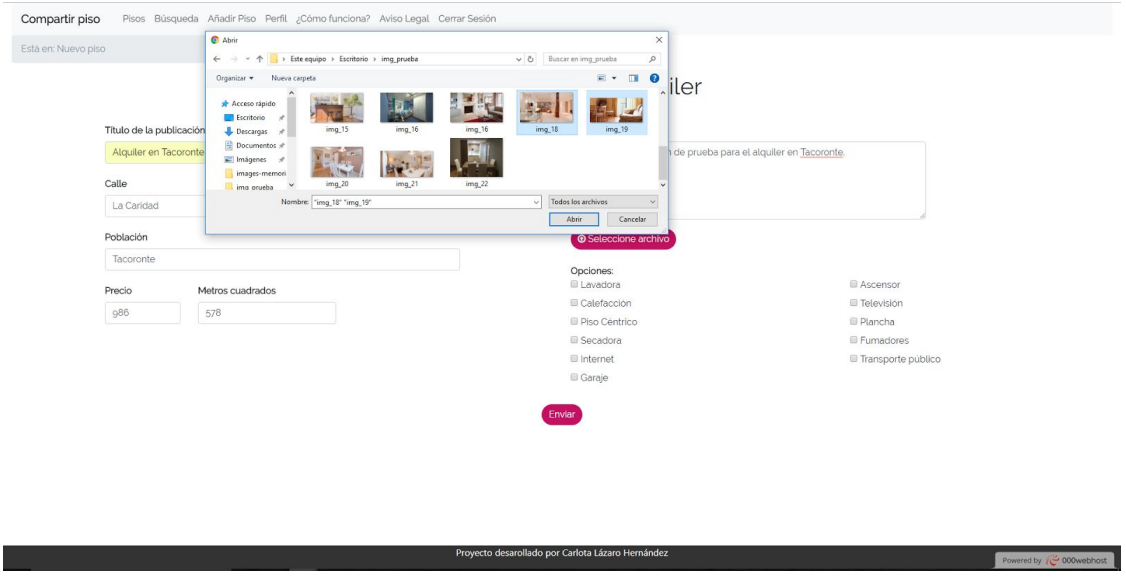


Figura 75: Selección de imágenes

4. Demostración de la página que muestra la información de un piso

A continuación se muestra la interfaz en la Figura 76 y una descripción acerca de la página que muestra toda la información de un alquiler.

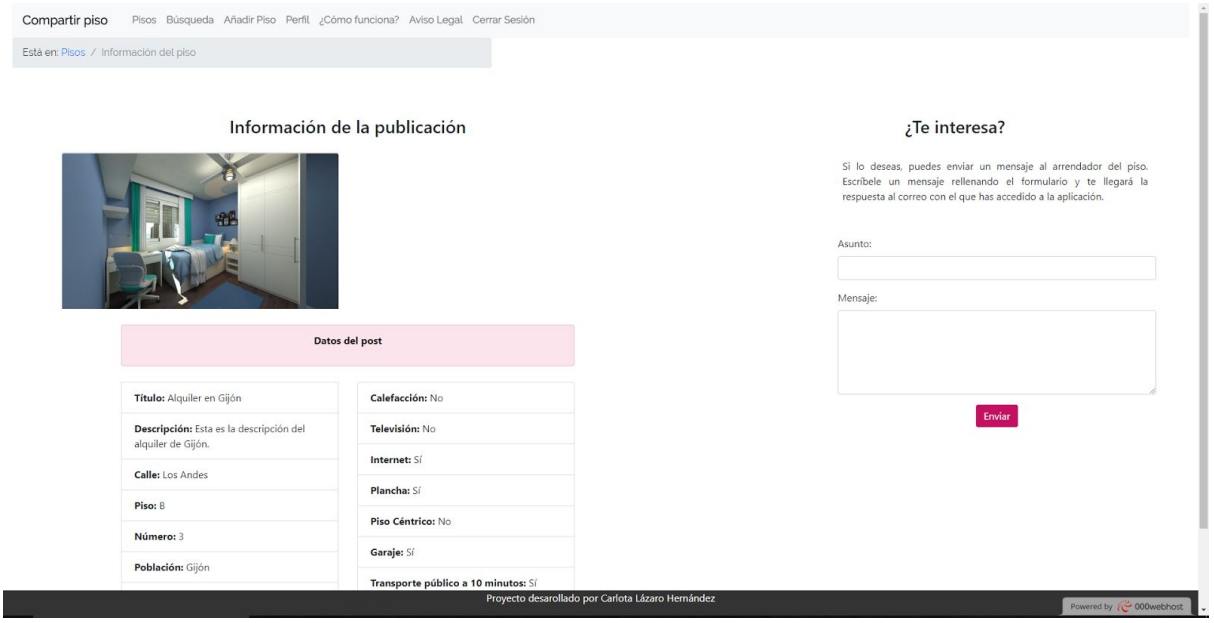


Figura 76: Interfaz que muestra toda la información de un piso

1.1 ¿Como funciona?

Como se ha comentado anteriormente, para acceder a la página se debe hacer previamente click en “Saber más” de la publicación seleccionada. La página mostrará la información de la imagen o imágenes y debajo, la información completa. Además, si el usuario lo desea puede enviarle un mensaje al arrendador solicitando ponerse en contacto.

5. Demostración página del perfil de usuario

A continuación se muestra la interfaz en la Figura 77 y una descripción acerca de la página del perfil del usuario:

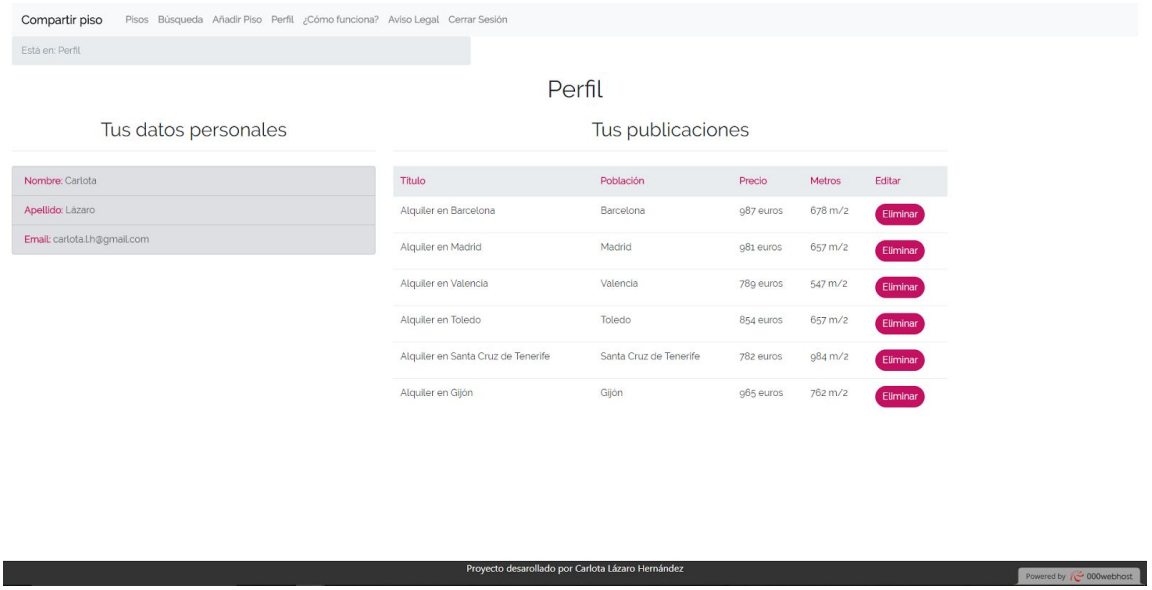


Figura 77: Interfaz del perfil de usuario

1.1 ¿Como funciona?

Para acceder a la sección del perfil del usuario, se deberá hacer click en *Perfil* de la sección del menú. La interfaz muestra a la izquierda los datos personales del usuario actual y, a la derecha las publicaciones que ha llevado a cabo. En el caso de querer eliminar una de las publicaciones, éste podrá hacer click en el botón “Eliminar” de la publicación que desee.

6. Demostración de la página Búsqueda

A continuación se muestra la interfaz en la Figura 78 y una descripción acerca de la página para realizar búsquedas filtradas en la aplicación.

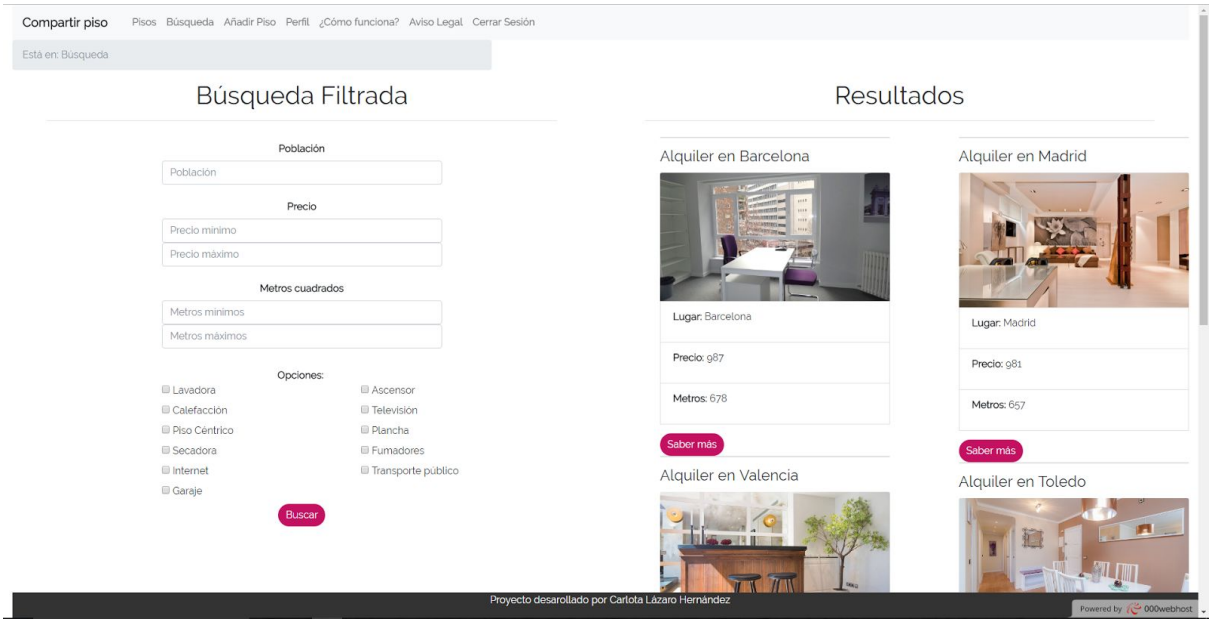


Figura 78: Interfaz al acceder a la sección Búsqueda

Una vez que se ha accedido a la sección de búsquedas, el usuario deberá elegir qué filtros desea aplicar para realizar búsquedas. En este caso, se va a mostrar en la Figura 79 un ejemplo una vez que se ha introducido en el campo *Población* Barcelona.

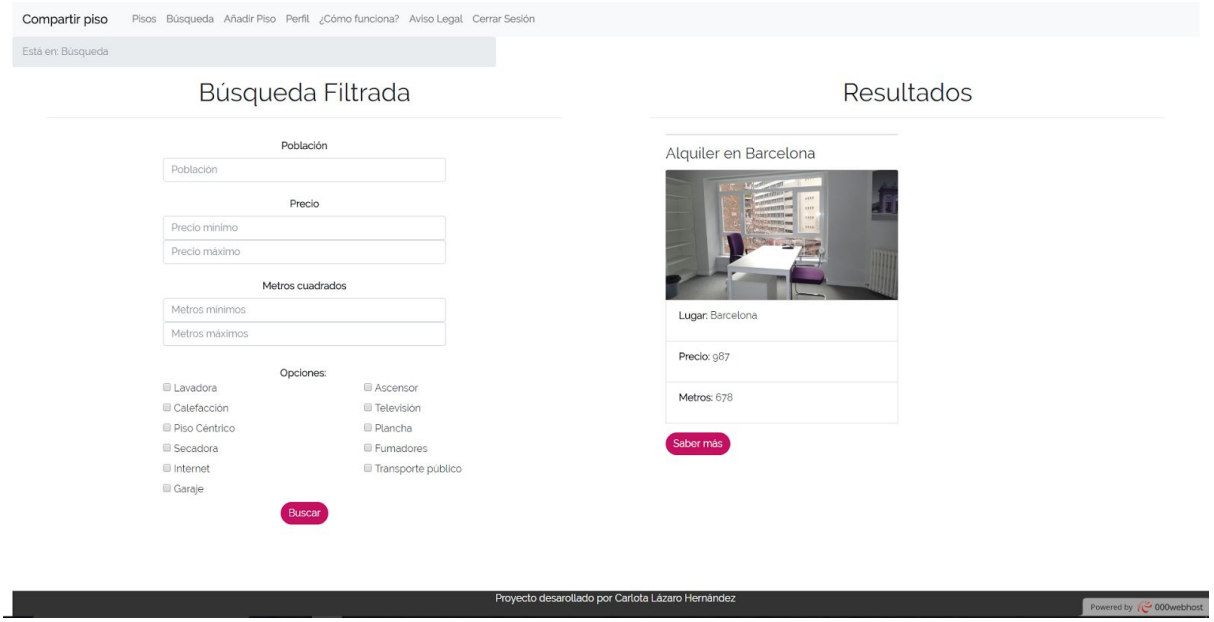


Figura 79: Interfaz al realizar el filtro

Como se puede comprobar, el resultado devuelto es el único alquiler que hay en la base de datos cuya población sea Barcelona.

En el caso de que el usuario quiera llevar a cabo combinaciones de varios filtros, éste podrá combinarlos sin problemas.

7. Demostración para las páginas de Aviso Legal y Funcionamiento

A continuación, se muestran las interfaces para el Aviso Legal y el Funcionamiento de la aplicación en las Figuras 79 y 80 respectivamente. Ambas, simplemente pretenden ofrecerle al usuario información, por lo que no poseen ningún funcionamiento en especial. Simplemente bastará con acceder a la sección deseada a través del menú superior.

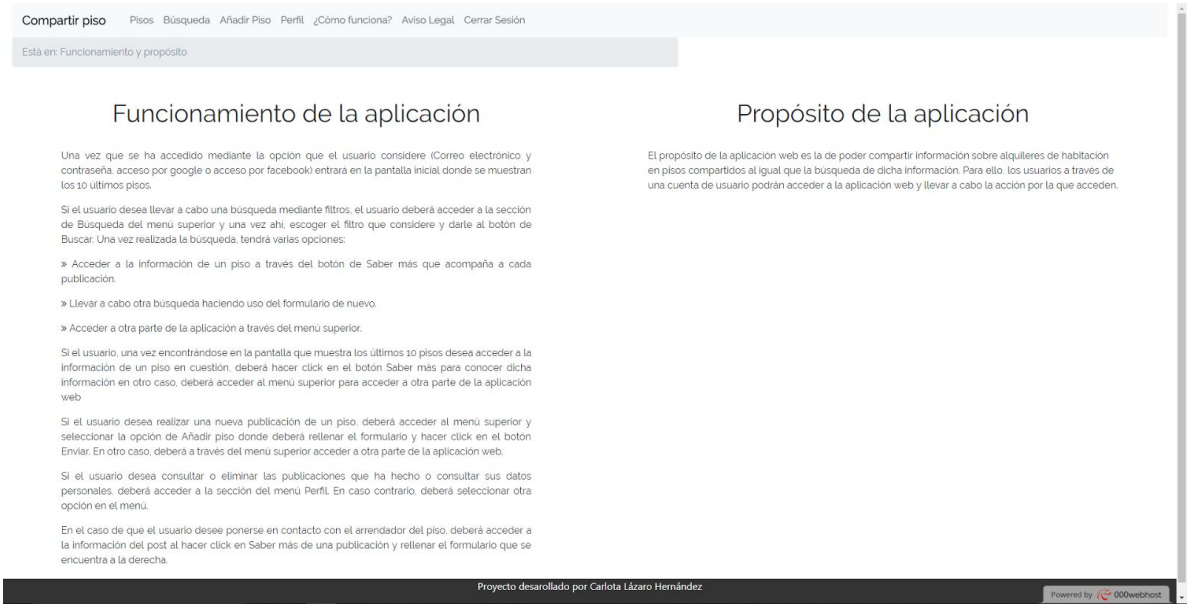


Figura 79: Interfaz del Aviso Legal

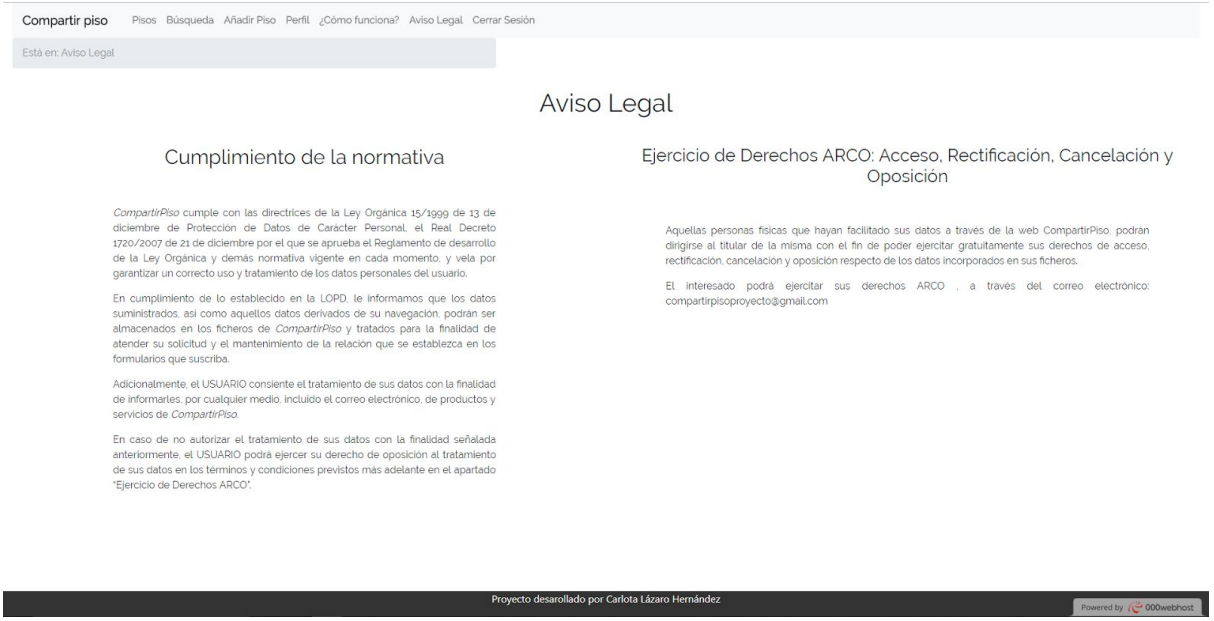


Figura 80: Interfaz del Funcionamiento y propósito

Capítulo 6: Conclusiones y líneas de futuro

1. Conclusiones

Las conclusiones que han sido obtenidas una vez finalizado el proyecto se exponen a continuación.

El proyecto, al tratarse del desarrollo de una aplicación desde cero y sobre un tema que puede abarcar un amplio rango de elementos a incluir, y siendo simplemente desarrollado por una sola persona, ha llegado a completarse lo planeado sin embargo, este tipo de desarrollo requieren una mayor inversión de tiempo en cuanto a meses y si es posible, a un equipo de trabajo para acelerar el trabajo. El desarrollo en local acerca de lo que se ha conseguido llevar a cabo no ha tenido demasiados problemas sin embargo, al subir la aplicación al servidor remoto se presentaron dos problemas principales sin solución:

- Inicio de Sesión por credenciales: La aplicación no redirecciona al sitio web debido a problemas con cabeceras de HTTP y HTTPS.
- Inicio de sesión por Facebook: al incluir la URL del servidor remoto en la sección de Facebook para desarrolladores, no admite dicha URL por ser “maliciosa o abusiva”.

Por ello, en el servidor remoto sólo se ha dejado activo el inicio de sesión a través de Google.

Los objetivos planeados para cada etapa del proyecto se han logrado llevar a cabo sin demasiados problemas sin embargo, la aplicación podría prestar otras funcionalidades más avanzadas si se dispusiese de más tiempo. Finalmente, cuando ha llegado el momento de la corrección final por parte del tutor, éste ha propuesto algunos cambios al proyecto para mejorarlo que han podido ser completados.

Se podría decir que la metodología prevista ha sido la adecuada ya que se partía de conocimientos sobre el desarrollo web y que además, el desarrollo de la aplicación web ha contribuido a la adquisición de otros nuevo conocimientos útiles para el futuro. Finalmente, el único cambio para que la aplicación obtuviese mejores resultados sería añadir más tiempo para implementar prestaciones más avanzadas.

2. Líneas de futuro

En un futuro, si se prosiguiera con el desarrollo de la aplicación web, se podrían tener en cuenta las siguientes ideas que mejorarían sus características actuales:

Se podrían añadir nuevos campos para las características de un alquiler que contribuyan a describir mejor el alquiler que se pretende publicar. Además, la interfaz en lo que a diseño respecta, también podría ser mejorada.

En el caso de que se dispusiese de un presupuesto real, se podrían llevar a cabo pruebas de usabilidad como evaluaciones heurísticas o aplicar el método de test con usuarios que ayudaría a encontrar fallos en la interfaz. También se podría mejorar la accesibilidad de la aplicación web para no llevar a cabo discriminaciones.

En lo que respecta al registro / login de la aplicación, se podrían añadir otras redes sociales con las que acceder a la aplicación para ofrecerle al usuario un mayor número de opciones con las que entrar a la aplicación dado que en función de la edad, conocimientos informáticos o intereses, los usuarios tienden a tener cuentas en redes sociales diferentes.

En lo que respecta al acceso por credenciales de correo electrónico y contraseña, se debería ir comprobando cada cierto tiempo que el cifrado que se utiliza para cifrar la contraseña no haya sido criptoanalizado con el fin de conseguir una mayor seguridad en la aplicación web.

Otra posible mejora sería la comunicación entre el cliente y el arrendador a través de un chat de mensajes sin tener que utilizar directamente sus correos electrónicos.

En la sección de filtros, se deberían añadir otros nuevos debido a las posibles modificaciones que se hagan en la sección para añadir un nuevo alquiler.

La aplicación, con tal de que llegue a un mayor número de usuarios, podría contemplar la posibilidad de selección de idiomas comenzando primero, por aquellos que más se hablen a nivel mundial como la lengua inglesa.

Bibliografía

Abocados a compartir vivienda [en línea] [fecha de consulta: 1 de octubre de 2017] Disponible en: https://elpais.com/economia/2017/07/14/actualidad/1500024628_901080.html

Alquiler de habitaciones como negocio [en línea] [fecha de consulta: 9 de octubre de 2017] Disponible en: http://www.emprender-facil.com/es/alquiler-de-habitaciones-como-negocio/

API de Google para PHP [en línea] [fecha de consulta: 05 de noviembre del 2017] Disponible en: https://github.com/google/google-api-php-client

API de Facebook para PHP [en línea] [fecha de consulta: 09 de noviembre del 2017] Disponible en: https://github.com/facebook/php-graph-sdk

API original de MySQL [en línea] [fecha de consulta: 05 de noviembre del 2017] Disponible en: http://php.net/manual/es/book.mysql.php

¿Comprar o alquilar? Esta es la mejor opción en la actualidad [en línea] [fecha de consulta: 9 de octubre de 2017] Disponible en: http://www.abc.es/economia/inmobiliario/abci-comprar-o-alquilar-esta-mejor-opcion-actualidad-201706030103_noticia.html

Crisis inmobiliaria en Londres: Cuando compartir piso se convierte en un lujo [en línea] [fecha de consulta: 1 de octubre de 2017] Disponible en: http://queaprendemoshoy.com/crisis-inmobiliaria-en-londres-cuando-compartir-piso-se-convierte-en-un-lujo/

Crisis y recuperación. Sectores protagonistas [en línea] [fecha de consulta: 9 de octubre de 2017] Disponible en: http://blog.iese.edu/martinezabascal/2017/01/11/crisis-y-recuperacion-sectores-protagonistas/

Descargar XAMPP [en línea] [fecha de consulta: 02 de noviembre del 2017] Disponible en: https://www.apachefriends.org/es/download.html

Download MySQL Workbench [en línea] [fecha de consulta: 02 de noviembre del 2017] Disponible en: https://dev.mysql.com/downloads/workbench/

El alquiler de viviendas levanta el vuelo [en línea] [fecha de consulta: 10 de octubre de 2017] Disponible en: https://elpais.com/economia/2017/05/26/actualidad/1495792973_914486.html

El alquiler de viviendas, en estado de 'boom' [en línea] [fecha de consulta: 9 de octubre de 2017] Disponible en: http://www.elmundo.es/economia/2017/07/17/59689b94e2704e64078b45c2.html

El alquiler supera los precios de 2007 en seis capitales [en línea] [fecha de consulta: 10 de octubre de 2017] Disponible en: https://elpais.com/economia/2017/06/26/vivienda/1498470700_960514.html

El porcentaje de los españoles que apuesta por el alquiler se acerca al de los partidarios de comprar [en línea] [fecha de consulta: 9 de octubre de 2017] Disponible en: http://www.elmundo.es/economia/vivienda/2017/07/17/596c7e1c468aeb56218b4634.html

El 23.5% de los españoles alquila pisos ¿por obligación o por convicción? [en línea] [fecha de consulta: 9 de octubre de 2017] Disponible en: https://elpais.com/economia/2016/06/23/vivienda/1466670608_178770.html

España es el país europeo donde más gente vive en pisos [en línea] [fecha de consulta: 9 de octubre de 2017] Disponible en: http://www.lavanguardia.com/vida/20151123/30338621049/espana-es-el-pais-europeo-donde-mas-gente-vive-en-pisos.html

Estoy harta de compartir piso pero no me queda otra [en línea] [fecha de consulta: 4 de octubre de 2017]. Disponible en: https://elpais.com/economia/2010/09/02/actualidad/1283412786_850215.html

ESTUDIO FOTOCASA: "Los españoles y su relación con la vivienda en 2015" [en línea] [fecha de consulta: 12 de octubre de 2017] Disponible en: http://blogprofesional.fotocasa.es/estudio-fotocasa-los-espanoles-relacion-la-vivienda-2015/

Informe inmobiliario 2016: previsión y evolución de la vivienda para 2016 y 2017 [en línea] [fecha de consulta: 9 de octubre de 2017] Disponible en: https://blog.bankinter.com/economia/-/noticia/2016/1/10/informe-inmobiliario-2016

Informe: El sector inmobiliario en España (1o parte) [en línea] [fecha de consulta: 9 de octubre de 2017] Disponible en: http://www.megabolsa.com/2017/05/11/informe-el-sector-inmobiliario-en-espana-1a-parte/

La demanda de habitaciones en alquiler se dispara un 80% en solo un año [en línea] [fecha de consulta: 5 de octubre de 2017] Disponible en: https://www.idealista.com/news/inmobiliario/vivienda/2017/08/07/747525-el-encarecimiento-del-alquiler-dispara-la-demanda-de-pisos-compartidos-un-80-en-solo

La mitad de los españoles que comparte piso tienen entre 30 y 50 años [en línea] [fecha de consulta: 9 de octubre de 2017] Disponible en: https://elpais.com/economia/2016/02/17/vivienda/1455708137_795324.html

Los 'erasmus' calientan los alquileres [en línea] [fecha de consulta: 9 de octubre de 2017] Disponible en: https://elpais.com/economia/2016/09/09/actualidad/1473414324_286359.html

Los precios del alquiler se coloca por encima de la era de la burbuja [en línea] [fecha de consulta: 10 de octubre de 2017] Disponible en: https://elpais.com/economia/2017/07/01/actualidad/1498935162_354554.html
Perfil de las personas que comparten vivienda [en línea] [fecha de consulta: 16 de octubre de 2017] Disponible en: http://prensa.fotocasa.es/wp-content/uploads/2017/09/Presentaci%C3%B3n-Perfil-han-buscado-piso-compartido.pdf

Precio de la vivienda en España [en línea] [fecha de consulta: 12 de octubre de 2017] Disponible en: https://es.wikipedia.org/wiki/Precio_de_la_vivienda_en_Espa%C3%B1a

¿Por qué compartir piso? [en línea] [fecha de consulta: 1 de octubre de 2017] Disponible en: http://noticias.universia.es/vida-universitaria/reportaje/2010/10/19/748929/1/compartir-piso-que-debes-saber/compartir-piso.html

¿Qué es la arquitectura web? [en línea] [fecha de consulta: 16 de octubre de 2017] Disponible en: http://www.daniloaz.com/es/que-es-la-arquitectura-web/

¿Qué es PHP? [en línea] [fecha de consulta: 29 de octubre del 2017] Disponible en: http://php.net/manual/es/intro-whatis.php

Anexos

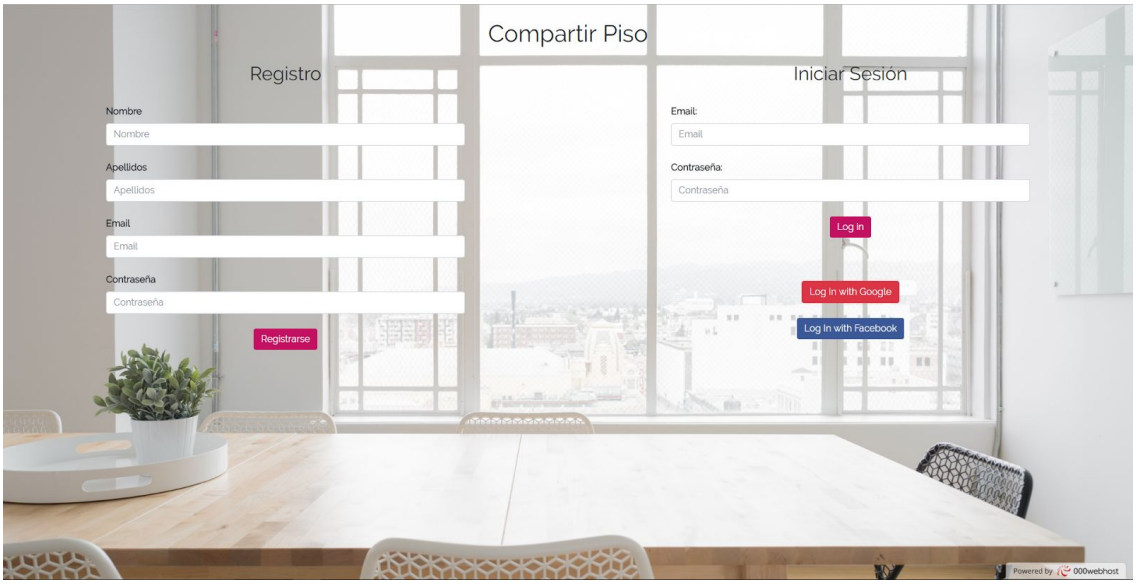
Anexo A: Entregables del proyecto

1. **Directorio GoogleAPI:** Contiene todos los ficheros necesarios para que el registro de usuarios por Google funcione.
2. **Directorio Facebook:** Contiene todos los ficheros necesarios para que el registro de usuarios por Facebook funcione.
3. **Directorio DBImages:** Contiene las imágenes que los usuarios han subido a la aplicación web a través del formulario para añadir un nuevo alquiler.
4. **Directorio db:** Contiene el fichero que realiza la conexión con la base de datos.
5. **Directorio images:** Contiene la imagen que aparece en el login de la aplicación web.
6. **Directorio css:** Contiene los ficheros de estilos que modifican el diseño del sitio web.
7. **Directorio js:** Contiene los ficheros JavaScript para poder mostrar el contenido de los JSON en la interfaz.
8. **Fichero advise.php:** Contiene el código para mostrar el Aviso Legal.
9. **Fichero api_delete.php:** Contiene el código para eliminar publicaciones desde el perfil del usuario.
10. **Fichero config.php:** Contiene el código para llevar a cabo los registros por Google y Facebook.
11. **Fichero facebook-callback:** Contiene el código para la implementación del callback de Facebook.
12. **Fichero google-callback:** Contiene el código para la implementación del callback de Google.
13. **Fichero index.php:** Contiene el código que implementa la interfaz de los últimos 10 alquileres publicados en la aplicación web.
14. **Fichero login.php:** Contiene el código que implementa la interfaz del registro y login de la aplicación web.
15. **Fichero logout.php:** Contiene el código que implementa cerrar sesión en la aplicación web.
16. **Fichero new_flat.php:** Contiene el código que implementa la sección para añadir un nuevo alquiler a la aplicación.
17. **Fichero post.json:** Contiene el resultado de una consulta a la base de datos sobre un post en formato JSON.
18. **Fichero post.php:** Contiene el código que implementa la interfaz para mostrar un alquiler al ser seleccionado.
19. **Fichero profile.php:** Contiene el código que implementa el perfil de usuario.
20. **Fichero publicaciones.json:** Contiene el resultado de una consulta a la base de datos sobre las últimas 10 publicaciones en la base de datos en formato JSON.

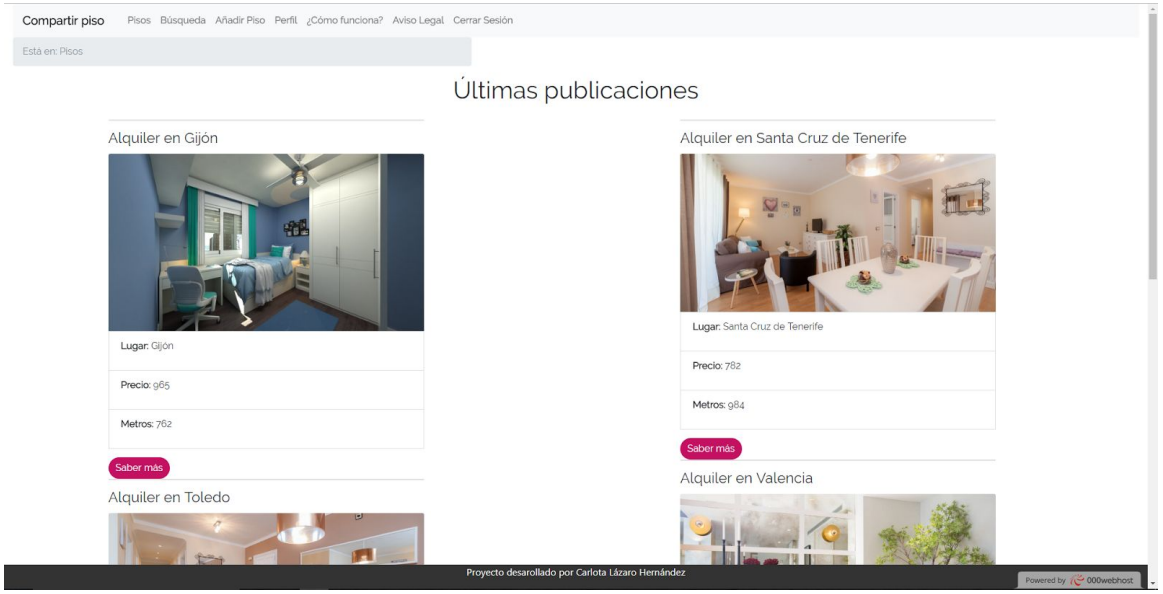
21. **Fichero search_post.json:** Contiene el resultado de una consulta a la base de datos a través de filtros en formato JSON.
22. **Fichero search.php:** Contiene el código que implementa la interfaz para realizar búsquedas filtradas en la base de datos.
23. **Fichero work.php:** Contiene el código que muestra la información sobre como funciona la aplicación y su propósito.
24. **Fichero script.sql:** Fichero que contiene el script de creación de las tablas de la base de datos junto a sus restricciones.

Anexo B: Capturas de pantalla

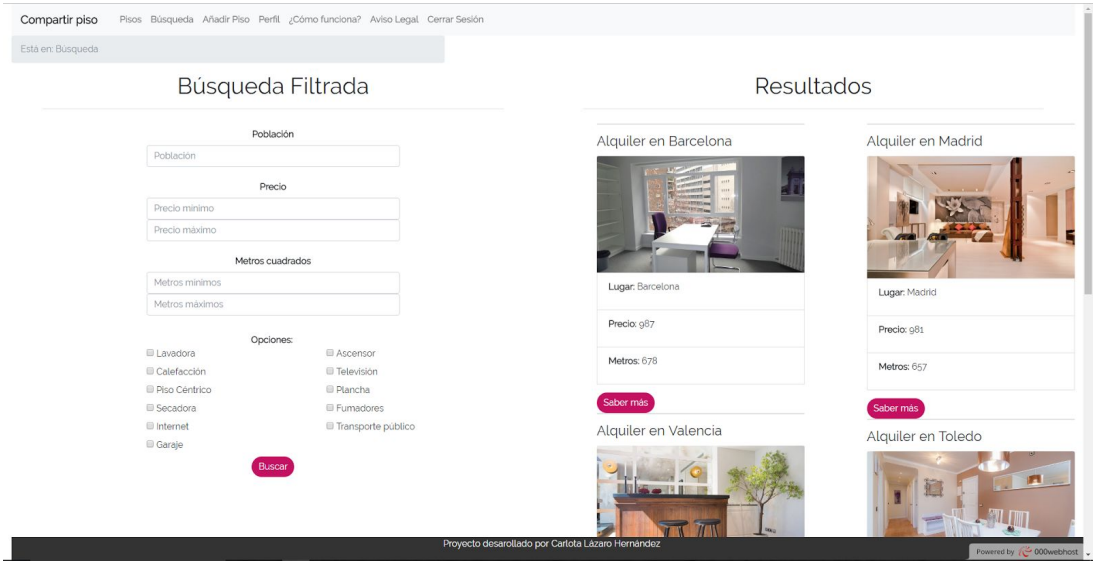
1. Interfaz del registro / login



2. Interfaz para mostrar los 10 últimos alquileres

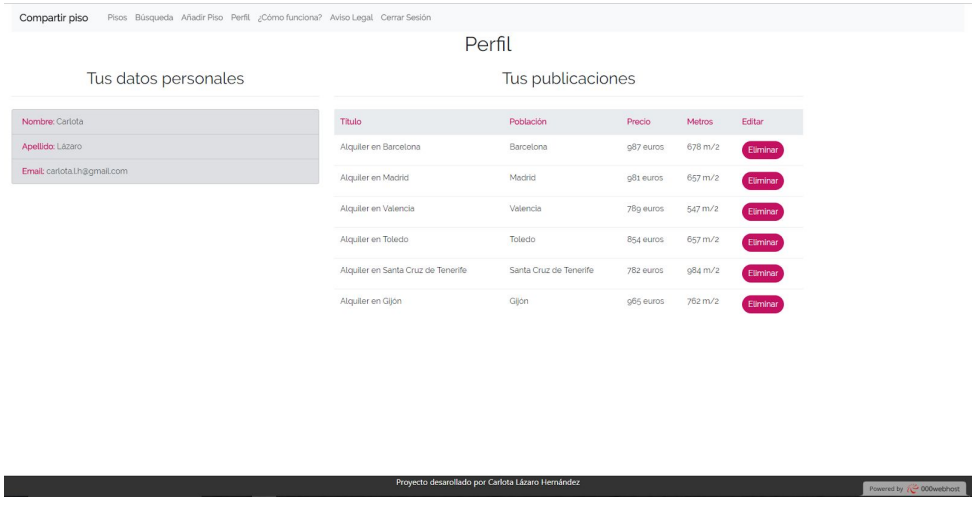


3. Interfaz para búsquedas

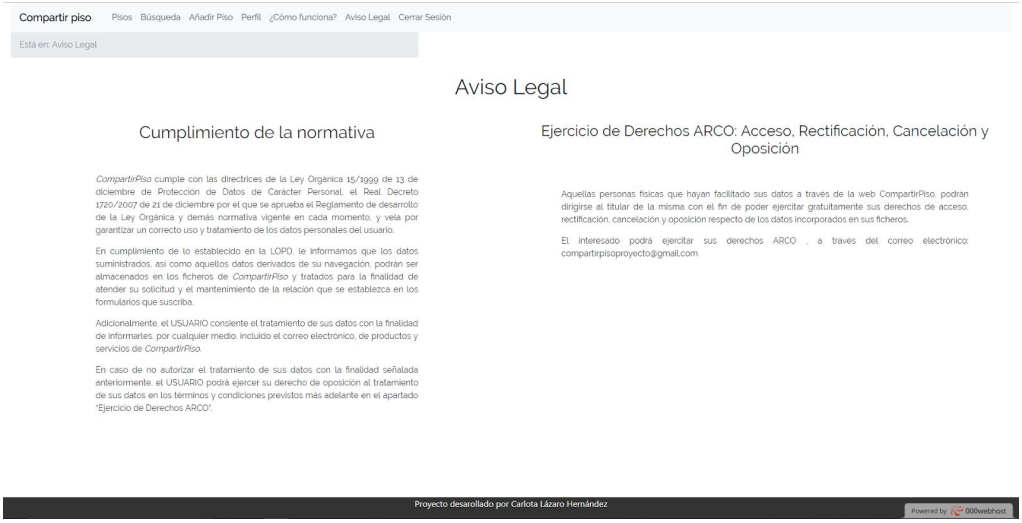


4. Interfaz para añadir un nuevo alquiler

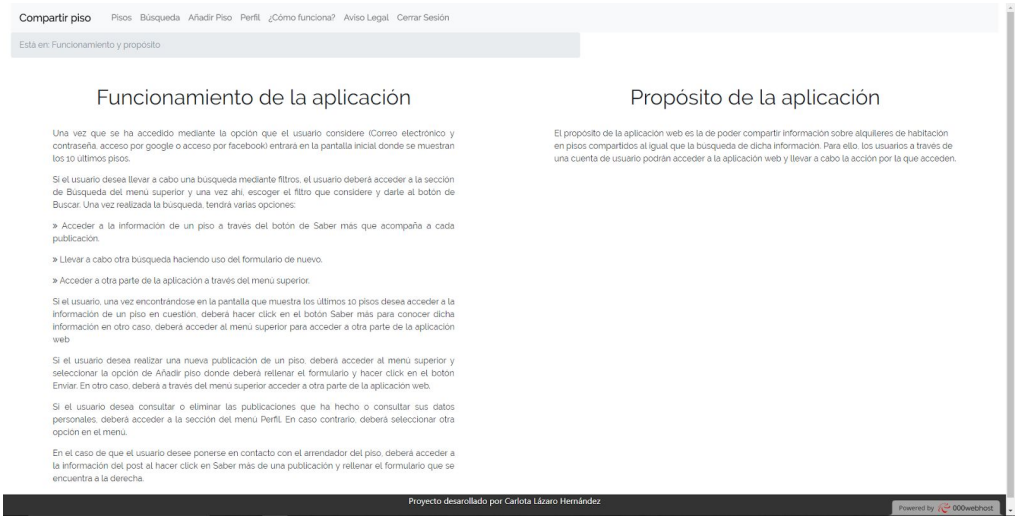
5. Interfaz para el perfil del usuario



6. Interfaz para el aviso legal



7. Interfaz para el funcionamiento y propósito



CARLOTA LÁZARO HERNÁNDEZ

CONTACTO

LINKEDIN: CARLOTA LÁZARO HERNÁNDEZ
EMAIL: CARLOTA.L.H@GMAIL.COM

IDIOMAS

ESPAÑOL: NATIVA
INGLÉS: FIRST CERTIFICATE IN ENGLISH

PERFIL

Soy una persona muy positiva y constante en el trabajo. Me considero una persona con muchas ganas de aprender sobre diferentes áreas como el diseño gráfico, el diseño web, el desarrollo web con diferentes tecnologías y la seguridad en sistemas informáticos.

EXPERIENCIA

IGEKO S.C.
DESARROLLO, DEPURACIÓN Y CORRECCIÓN DE ERRORES DE UN SOFTWARE DE SINCRONIZACIÓN DE FICHEROS CON SERVICIOS DE ALMACENAMIENTO EN LA NUBE.

EDUCACIÓN

GRADO EN INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE LA LAGUNA - 2011 A 2016

MÁSTER DE APLICACIONES MULTIMEDIA
UNIVERSITAT OBERTA DE CATALUNYA - 2016 A 2018

HABILIDADES

- C++
- JavaScript
- PHP
- NodeJS
- HTML5
- CSS3
- Git / GitHub
- JSON
- Materialize
- Algoritmos criptográficos