



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA



Escola Tècnica  
Superior d'Enginyeria  
Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica  
Universitat Politècnica de València

# Diseño de una aplicación Android para el intercambio de viviendas

Trabajo Fin de Grado

**Grado en Ingeniería Informática**

**Autor:** Haloui, Abdelwahed

**Tutor:** Sáez Barona, Sergio

2015/16



## *Resumen*

---

Este proyecto se basa en el desarrollo de una aplicación móvil para dispositivos con sistema operativo Android. Se trata de una aplicación para el intercambio de viviendas. La aplicación ofrece la posibilidad de ofertar tu vivienda, especificando las fechas en las que está disponible y sus características, realizar ofertas de intercambio con otras viviendas ofertadas y gestionar las ofertas recibidas. Igualmente, la aplicación permite la valoración de las viviendas por los usuarios.

El proceso de desarrollo de la aplicación se ha llevado a cabo en dos fases. Por un lado, tenemos el proceso de diseño centrado en el usuario y, por otra parte, se ha realizado el desarrollo e implementación de la aplicación. Para la realización de esta aplicación se ha utilizado lenguajes como JAVA, PHP, MySQL, XML, JSON y Android Studio como entorno.

**Palabras clave:** Android, Aplicación móvil, intercambio de viviendas, PHP, Java, MySQL, XML, JSON, Android Studio.

## *Abstract*

---

This project is based on the development of a mobile application for devices with Android operating system. This is an application for home exchange. It offers the possibility of offering your home, specifying the dates on which it is available and its features, make exchange offers with other homes offered and manage the offers received. Similarly, the application allows the valuation of homes by users.

The process of application development has been carried out in two phases. On the one hand, we have the process of user-centered design, and on the other hand, has made the development and implementation of the application. For the realization of this application it has been used languages like JAVA, PHP, MySQL, XML, JSON and Android Studio like setting.

**Key Words:** Android, Mobile Application, Home Exchange, PHP, Java, MySQL, XML, JSON, Android Studio.



# Tabla de Contenidos

1.	INTRODUCCIÓN .....	10
1.1	MOTIVACIÓN .....	10
1.2	OBJETIVOS.....	10
1.3	ESTRUCTURA DE LA MEMORIA .....	11
2.	ESPECIFICACIÓN DE REQUISITOS .....	12
2.1	INTRODUCCIÓN .....	12
2.1.1	Propósito .....	12
2.1.2	Ámbito.....	12
2.1.3	Definiciones y acrónimos .....	12
2.1.4	Referencias.....	13
2.2	DESCRIPCIÓN GENERAL.....	13
2.2.1	Perspectivas del producto.....	13
2.2.2	Funcionalidad del producto .....	14
2.2.3	Características de los usuarios .....	14
2.2.4	Suposiciones y Dependencias .....	14
2.3	REQUISITOS ESPECÍFICOS .....	14
2.3.1	Requisitos Funcionales .....	14
2.3.2	Requisitos de interfaces externas .....	17
2.3.3	Requisitos no funcionales .....	18
3.	FASE DE ANÁLISIS .....	19
3.1	DIAGRAMA DE CLASES O ENTIDAD-RELACIÓN .....	19
3.2	CASOS DE USO .....	19
3.2.1	Fichas de casos de uso para Usuarios .....	21
3.2.2	Fichas de casos de uso para Administrador .....	26
4.	FASE DE DISEÑO .....	29
4.1	INTRODUCCIÓN .....	29
4.2	CAPA DE PRESENTACIÓN .....	29
4.3	CAPA DE PERSISTENCIA.....	45
4.4	CAPA DE LÓGICA .....	46
5.	DETALLES DE IMPLEMENTACIÓN .....	49
5.1	INTRODUCCIÓN .....	49
5.2	APLICACIÓN ANDROID.....	49
5.3	INTERFACES CON XML.....	52
5.4	PHP Y SERVIDOR WEB .....	53

6.	PRUEBAS.....	57
7.	CONCLUSIONES .....	65
7.1	TÉCNICAS .....	65
7.2	PERSONALES .....	65
7.3	POSIBLES AMPLIACIONES DEL PROYECTO .....	66
8.	BIBLIOGRAFÍA.....	67

## Índice de Figuras

Figura 1.- Diagrama de Clases.....	19
Figura 2.- Diagrama de Casos de Uso para Usuarios.....	20
Figura 3.- Diagrama de Casos de Uso para el Administrador.....	21
Figura 4.- Interfaz de la aplicación web “ninjamock” .....	29
Figura 5.- Pantalla inicial de la aplicación.....	30
Figura 6.- Pantalla para crear usuario.....	31
Figura 7.- Pantalla para añadir vivienda.....	32
Figura 8.- Pantalla para iniciar sesión.....	33
Figura 9.- Pantalla donde se buscan los anuncios.....	34
Figura 10.- Pantalla con anuncios publicados.....	35
Figura 11.- Pantalla con detalles de un anuncio.....	36
Figura 12.- Pantalla de Perfil de un usuario.....	37
Figura 13.- Pantalla de Solicitud de Reserva.....	38
Figura 14.- Manú de Navegación.....	39
Figura 15.- Pantalla de Perfil.....	40
Figura 16.- Pantalla con las Casas Visitadas.....	41
Figura 17.- Pantalla con las Casas Reservadas.....	42
Figura 18.- Pantalla para Dejar Opinión.....	43
Figura 19.- Pantalla Sobre Nosotros.....	44
Figura 20.- Esquema de la Base de Datos.....	45
Figura 21.- Diagrama de Flujo. Mapa de la aplicación.....	48
Figura 22.- Ejemplo de Relación Programa-Servidor .....	48
Figura 23.- Carpeta HomeAcc.....	50
Figura 24.- Ejemplo de método de APIManager.....	51
Figura 25.- Carpeta res.....	52
Figura 26.- Parte del código de la clase Opinion desde donde se accede a la API.....	54
Figura 27.- Método InsertarOpinion de la clase APIManager.....	55
Figura 28.- Archivo php para la inserción de opiniones.....	56
Figura 29.- Pantalla de apertura de la aplicación.....	57
Figura 30.- Pantallas de registro de usuario.....	57

## Diseño de una aplicación Android para el intercambio de viviendas

Figura 31.- Pantalla de registro de vivienda.....	58
Figura 32.- Pantalla de Inicio de Sesión.....	58
Figura 33.- Menú navegador.....	59
Figura 34.- Pantalla Inicial y de Buscador de Viviendas.....	59
Figura 35.- Pantalla de Anuncio.....	60
Figura 36.- Pantalla de Perfil de usuario.....	61
Figura 37.- Pantalla de Reservas de Viviendas.....	61
Figura 38.- Pantalla de Viviendas Visitadas.....	62
Figura 39.- Pantalla de Dejar Opinión.....	62
Figura 40.- Pantalla de Editar Perfil.....	63
Figura 41.- Pantalla de Editar Anuncio.....	63
Figura 42.- Pantalla de Ver Notificaciones.....	63



# 1. INTRODUCCIÓN

En este primer apartado de la memoria se empezará describiendo la motivación para la realización de este proyecto en particular, así como los objetivos del trabajo, donde se especificará la razón por la que se realiza este proyecto y el resultado al que se quiere llegar, y ya para terminar, la estructura de la memoria, apartado donde se explicará cómo está organizado este proyecto.

## 1.1 MOTIVACIÓN

Tras alcanzar estas alturas del Grado en Ingeniería Informática, me doy cuenta de que uno de los campos tecnológicos que más me interesan, el de desarrollo de aplicaciones para dispositivos móviles, es el área que menos aprendí y con el que menor contacto tuve.

Las asignaturas cursadas me ayudaron a tener una base para desarrollarme en este campo, por lo que he decidido dedicar el proyecto más importante de la carrera a estas tecnologías y de una manera un poco autodidacta aprender a programar para entornos móviles, en este caso Android.

La motivación principal para elegir desarrollar una aplicación de intercambio de viviendas es mi experiencia personal a la hora de realizar viajes. Al recorrer diferentes ciudades y diferentes países me he dado cuenta que una parte importante del presupuesto del viaje estaba dirigida al alojamiento, así que cuando apareció en la lista de Trabajos de Fin de Grado esta idea, no tenía dudas a la hora de elegirla. Una aplicación que te permite ahorrar dinero y al mismo tiempo conocer a nueva gente.

## 1.2 OBJETIVOS

Realizar un trabajo que comprende el proceso de desarrollo de una aplicación móvil para dispositivos Android, un proceso que abarca el diseño, la usabilidad y la funcionalidad acorde con lo que se pide. Cabe destacar que el desarrollo de este proyecto se divide en dos fases: por una parte, tenemos el diseño de la interfaz de usuario, y por otra, la implementación de la aplicación móvil.

La aplicación gestionará un intercambio de viviendas, de manera que dos personas pueden intercambiar sus hogares durante un período determinado sin tener que pagar uno al otro.

De esta manera, los viajeros tendrán un problema menos en el que pensar, ya que según un estudio realizado por la junta de Andalucía se puede observar como el gasto medio de los turistas por alojamiento en España es de 62,46 €, un gasto que muchas personas no se pueden permitir.

Como objetivos personales están el aprender a manejar el lenguaje *PHP*, además de un gestor de base de datos como es el *MySQL*. También están el mejorar el conocimiento del lenguaje Java que, con la rama de Sistemas de Información, lo tengo un poco apartado.

### 1.3 ESTRUCTURA DE LA MEMORIA

Para la realización de esta memoria, voy a empezar en el siguiente capítulo con la especificación de los requisitos, donde se detallarán los objetivos de la aplicación, además de exponer los requisitos según la norma *IEEE 830-1998*.

En el capítulo 3 se especificará la fase de análisis de la aplicación detallada mediante dos diagramas. Para empezar, se especificará el diagrama de clases o entidad-relación. A continuación, vendrán detallados dos casos de uso, uno para los usuarios y el otro para el administrador, seguidos de sus fichas respectivamente.

En el capítulo 4, hablaremos de las fases de diseño, divididas en tres partes. La primera fase de la que se va a hablar es la de presentación, donde aparecerán bocetos y muestras de cómo debería ser la aplicación. Luego se detallará la capa de persistencia donde se analizará cómo será la base de datos y cómo estará estructurada. Para finalizar con este capítulo, hablaremos de la capa lógica, donde se mostrará un mapa de cómo se puede recorrer la aplicación.

En el siguiente capítulo, el 5, se detallará la implementación. En este capítulo explicaré como he ido realizando la aplicación, qué tecnologías se han ido utilizando y cómo.

El capítulo 6 es el de pruebas. En éste, iré mostrando capturas de pantalla de la aplicación para ver su correcto funcionamiento además de una breve descripción de cada una de ellas.

En cuanto al capítulo 7, trata de las conclusiones obtenidas de este proyecto, de cómo me ha ayudado a desarrollarme tanto a nivel personal como profesional.

Ya para finalizar, el capítulo 8 marcará la bibliografía utilizada para la realización de este Trabajo de Fin de Grado.

## 2. ESPECIFICACIÓN DE REQUISITOS

En esta sección detallamos los objetivos de la aplicación, las funciones que debe realizar y las necesidades que debe cubrir, así como las restricciones que contiene. Ésta sigue la estructura y directrices dadas por el estándar *830-1998 - IEEE Recommended Practice for Software Requirements Specifications*.

### 2.1 INTRODUCCIÓN

En este apartado veremos tres aspectos importantes. El primero, propósito, tratará de responder a la pregunta de por qué se ha realizado esta aplicación. El segundo, ámbito, explicará los requisitos básicos que tendrá la aplicación y en cuanto al tercero, definiciones y acrónimos, aparecerán definidos brevemente algunos términos que se van a utilizar a lo largo de esta memoria.

#### 2.1.1 Propósito

El propósito es ofrecer un entorno para todas aquellas personas que deseen intercambiar sus viviendas. Dicho de otra manera, se pretende ofrecer un lugar donde pasar las vacaciones con coste cero en cuanto a alojamiento.

#### 2.1.2 Ámbito

- La aplicación que se va a desarrollar realizará las labores de tablón de anuncios.
- La aplicación deberá adaptarse a cualquier cantidad de anuncios publicados en ella.
- La aplicación no proporcionará ningún método de pago ni se hará responsable de ninguno de los anuncios publicados en esta.
- No se permitirá a ningún usuario registrarse si no ofrece una vivienda para intercambiar.

#### 2.1.3 Definiciones y acrónimos

Esta sección pretende servir como punto de referencia a una persona no vinculada al área informática y familiarizarlo con los términos y conceptos utilizados, además de definir conceptos relacionados con la aplicación:

- **Login:** es el momento de identificación al ingresar en el sistema.
- **Logout:** Es el momento de cerrar la sesión en el sistema.
- **PHP:** PHP Hypertext Pre-processor. Es el lenguaje de programación para la creación de páginas web dinámicas.
- **MySQL:** Es un sistema de gestión de bases de datos.

- **XML:** Extensible Markup Language. Es un lenguaje basado en etiquetas para compartir información de una manera segura, fiable y fácil.
- **Android Studio:** es un entorno de desarrollo integrado para la plataforma Android.
- **Administrador:** Persona que gestiona la aplicación para su buen uso, teniendo el poder de tomar las medidas oportunas en cada situación.

### 2.1.4 Referencias

- 830-1998 - *IEEE Recommended Practice for Software Requirements Specifications*
- *Software Requirements Specification Template*  
([http://courses.cs.washington.edu/courses/cse403/06sp/assgn/requirements\\_spec.doc](http://courses.cs.washington.edu/courses/cse403/06sp/assgn/requirements_spec.doc))

## 2.2 DESCRIPCIÓN GENERAL

Para este apartado se definirán las perspectivas del producto, además de explicar la funcionalidad de éste, definir las características de los usuarios y aclarar brevemente las suposiciones y dependencias.

### 2.2.1 Perspectivas del producto

La aplicación debe proporcionar un lugar donde publicar anuncios de forma pública, aumentando así las posibilidades de encontrar alojamiento en la ciudad deseada, es por ello, que necesita estar conectada a una base de datos externa, en este caso *MySQL*, de donde poder obtener, insertar, eliminar o modificar los datos previamente guardados por los usuarios a través de los formularios pertinentes.

- **Interfaces del sistema:** El sistema debe contener, como mínimo, una interfaz de conexión con la base de datos *MySQL*.
- **Interfaces de usuario:** El usuario requiere que disponga del sistema operativo de Android, cualquier versión de éste es suficiente para el correcto funcionamiento.
- **Interfaces Hardware:** Se requiere al menos un móvil inteligente con el sistema operativo Android.
- **Interfaces Software:** No se requiere de ninguna interfaz software.
- **Interfaces de comunicación:** Al menos se requiere de conexión a internet para poder dar servicio a los usuarios que accedan a ella.

### 2.2.2 Funcionalidad del producto

Esta aplicación pretende gestionar un intercambio de viviendas. Dado que es una aplicación realizada exclusivamente con fines didácticos como trabajo de fin de grado, su funcionalidad, como por ejemplo, la seguridad a la hora de registrarse estará limitada.

La aplicación gestionará los usuarios, así como las viviendas que éstos ponen a disposición, todo esto poniendo especial atención a su adecuada presentación. También se ofrecerá una búsqueda de las viviendas según la provincia a la que se quiere viajar y según la disponibilidad de fechas para la vivienda en cuestión.

### 2.2.3 Características de los usuarios

Principalmente los usuarios van a ser de dos tipos. Por una parte, tenemos los usuarios de la aplicación, y por otra, al administrador de la aplicación. A continuación, se detallan las características de ambos usuarios:

<b>Tipo de Usuario</b>	Usuario de la aplicación
<b>Formación</b>	Varios perfiles diferentes
<b>Habilidades</b>	Varios perfiles diferentes
<b>Actividades</b>	Intercambiar su vivienda

<b>Tipo de Usuario</b>	Administrador de la aplicación
<b>Formación</b>	Grado en Ing. Informática
<b>Habilidades</b>	Desarrollo Android
<b>Actividades</b>	Administrar la aplicación

### 2.2.4 Suposiciones y Dependencias

La aplicación estará escrita en los lenguajes más comunes para este tipo de herramientas: Java, PHP, MySQL, XML y Android Studio como entorno de desarrollo, por lo que será posible acceder a ella con cualquier tipo de móvil, siempre y cuando tenga Android como Sistema Operativo.

## 2.3 REQUISITOS ESPECÍFICOS

En este apartado se explican los requisitos que deberá satisfacer la aplicación. Todos los requisitos que se presentan son fundamentales para conseguir el objetivo principal de la aplicación. Estos requisitos se dividen en Requisitos Funcionales y Requisitos de Interfaces Externos.

### 2.3.1 Requisitos Funcionales

Definiremos las acciones que debe realizar el software, es decir, de qué forma se procesan y aceptan un conjunto de entradas para producir una determinada salida.

**Registrarse en el sistema por parte del usuario:**

<b>Introducción</b>	Para poder tener acceso al sistema, el usuario debe registrarse y conocer de esta manera sus datos.
<b>Entrada</b>	Campos como nombre, apellidos, DNI, contraseña, Correo electrónico, teléfono e imagen.
<b>Salida</b>	Dar de alta una vivienda.

**Dar de alta una vivienda**

<b>Introducción</b>	Registrar la vivienda que el usuario quiere intercambiar.
<b>Entrada</b>	Campos como número de habitaciones, descripción de la vivienda, dirección y ciudad, disponibilidad, imágenes.
<b>Salida</b>	Creación de la cuenta y acceso a la aplicación.

**Login en el sistema por parte del usuario:**

<b>Introducción</b>	Existe la necesidad de saber quién accede al sistema para el intercambio correcto de datos con los demás usuarios, sin olvidar que uno de los requisitos más importantes es que solo los usuarios registrados pueden interactuar en la aplicación.
<b>Entrada</b>	Nombre del usuario – contraseña.
<b>Salida</b>	Acceso a la aplicación.

**Listado de anuncios**

<b>Introducción</b>	Consultar las viviendas disponibles en la provincia elegida por el usuario.
<b>Entrada</b>	Provincia de destino deseada, fecha de entrada y salida.
<b>Salida</b>	Listado de viviendas disponibles en la provincia elegida.

**Consultar anuncio de una vivienda**

<b>Introducción</b>	Consultar los detalles y características de una vivienda en particular: Fotos de la vivienda, descripción, opiniones.
<b>Entrada</b>	Código de identidad del anuncio.
<b>Salida</b>	Muestra la información referente a un anuncio.

**Contactar con el anunciante**

<b>Introducción</b>	Contactar con el anunciante vía correo electrónico.
<b>Entrada</b>	Correo electrónico del anunciante.
<b>Salida</b>	Escribir un correo electrónico al anunciante.

**Pantalla principal de la aplicación**

<b>Introducción</b>	La primera pantalla que aparece a la hora de entrar a la aplicación, donde se puede registrar o iniciar sesión.
<b>Entrada</b>	Código de identidad del botón pulsado.
<b>Salida</b>	Dependiendo de la elección, se accede a la pantalla de inicio de sesión o la de registro de usuario.

**Ver perfil de un usuario**

<b>Introducción</b>	Visualizar el perfil de un usuario, donde aparecerán datos suyos, tales como nombre, apellidos, número de teléfono, correo electrónico, opiniones sobre viviendas suyas.
<b>Entrada</b>	DNI del usuario.
<b>Salida</b>	El perfil del usuario

**Solicitar reserva**

<b>Introducción</b>	Solicitar reserva de la vivienda de un usuario.
<b>Entrada</b>	DNI del anunciante.
<b>Salida</b>	Solicitar intercambio de una vivienda.

**Notificaciones del usuario**

<b>Introducción</b>	Solicitudes de reserva pendientes de contestar por el usuario
<b>Entrada</b>	DNI del usuario.
<b>Salida</b>	Las diferentes solicitudes pendientes de confirmar o rechazar.

**Editar perfil**

<b>Introducción</b>	Editar datos personales de un usuario.
<b>Entrada</b>	DNI del usuario.
<b>Salida</b>	Pantalla con datos del usuario que se pueden modificar.

**Editar anuncio**

<b>Introducción</b>	Editar datos de un anuncio
<b>Entrada</b>	Código de identidad del anuncio.
<b>Salida</b>	Pantalla con datos del anuncio que se pueden modificar.

### Ver viviendas visitadas

<b>Introducción</b>	Listado de las viviendas que un usuario ha visitado, con opción a dejar una valoración sobre el estado de la vivienda y el trato recibido por el dueño.
<b>Entrada</b>	DNI del usuario y Código de identidad de anuncios de viviendas visitadas.
<b>Salida</b>	Listado de las viviendas visitadas hasta la fecha actual.

### Ver viviendas reservadas

<b>Introducción</b>	Listado de las viviendas que un usuario ha reservado, pero que todavía no ha visitado.
<b>Entrada</b>	DNI del usuario y Código de identidad de anuncios de viviendas reservadas sin visitar.
<b>Salida</b>	Listado de las viviendas reservadas hasta la fecha actual.

### 2.3.2 Requisitos de interfaces externas

La aplicación permite la introducción de datos de usuario y de la vivienda por medio de los formularios de forma remota y por medio de la entrada estándar de teclado del móvil, así como la navegación por la interfaz.

#### **Interfaces de usuario**

Las interfaces deberán ser sencillas y cómodas para que los usuarios puedan navegar por ellas de manera intuitiva, además deberán ser en todo lo posible visualmente atractivas.

#### **Interfaces de hardware**

La aplicación puede utilizarse en todos los dispositivos que dispongan del Sistema Operativo Android. No se necesitará ningún hardware especial y el dispositivo no tendrá ningún requisito mínimo a nivel de hardware para poder acceder a la aplicación.

#### **Interfaces de software**

El cliente únicamente accede a la aplicación mediante un teléfono inteligente con Sistema Operativo Android, y la interfaz es la de la propia aplicación.

#### **Interfaces de comunicación**

La aplicación está diseñada para que se utilice a través de la red, la única restricción es que el dispositivo desde el que se accede a la aplicación debe estar conectado a Internet.

### 2.3.3 Requisitos no funcionales

#### **Requisitos de rendimiento**

Dado que el volumen de usuarios no va a ser importante, puesto que el proyecto ha sido desarrollado únicamente con fines didácticos, el sistema no soportará gran volumen de usuarios accediendo concurrentemente. Sí que debe permitir un buen rendimiento para unos pocos usuarios accediendo de forma local o remota para hacer pruebas.

#### **Seguridad**

Para acceder al sistema se necesita un usuario y contraseña, con ello se controla el acceso a la aplicación. Además, se emplea un cifrado de las contraseñas cuando se almacenan en la base de datos para una mayor seguridad.

La aplicación contará con un sistema de seguridad que no permita el acceso al sistema a usuarios no registrados o sin permisos para utilizar la aplicación. También se controlará que todos los usuarios hayan dado de alta una vivienda, en caso contrario no podrán registrarse en el sistema.

#### **Mantenimiento**

Las tareas de mantenimiento se basan, en su mayoría, en introducir nuevos usuarios y viviendas. En cuanto a las copias de seguridad no serán necesarias, ya que es un proyecto que se da por finalizado y no es previsible que crezca en complejidad, una vez hecho el respaldo oportuno al final de este, no es necesario que se hagan periódicamente.

#### **Disponibilidad**

No se requiere disponibilidad para el proyecto, puesto que está hecho con fines didácticos. Debería estar disponible para su comprobación cuando el tutor lo requiera.

#### **Fiabilidad**

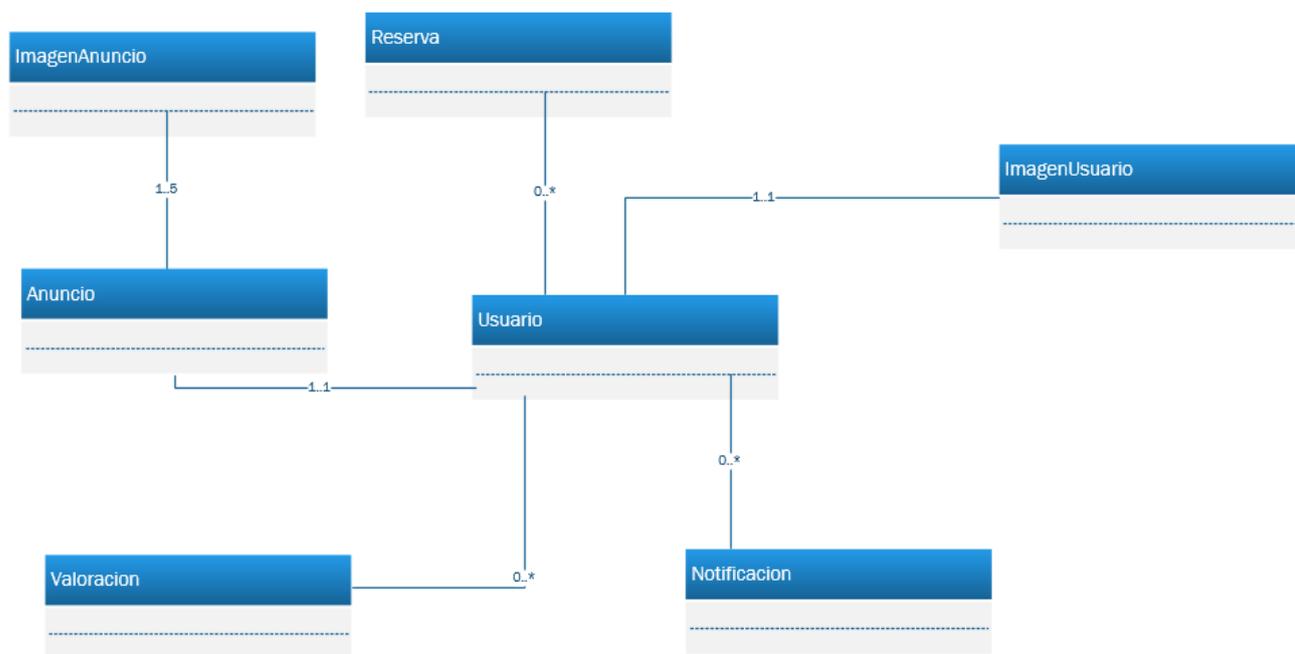
Teniendo en cuenta la envergadura del proyecto y su utilización, los errores deben ser mínimos o incluso inexistentes para el correcto funcionamiento de la aplicación.

## 3. FASE DE ANÁLISIS

En esta sección representaré, por una parte, las entidades relevantes de la aplicación mediante un diagrama entidad-relación, y por otra, los diagramas de casos de uso, tanto del administrador como de los usuarios, y sus fichas correspondientes.

### 3.1 DIAGRAMA DE CLASES O ENTIDAD-RELACIÓN

Como expliqué anteriormente, en esta parte representaré el diagrama entidad-relación correspondiente a mi aplicación. Hay que tener en cuenta que en este diagrama solo representaré la relación entre las tablas y la multiplicidad, dejando los atributos para la sección de diseño.



**Figura 1. Diagrama de Clases**

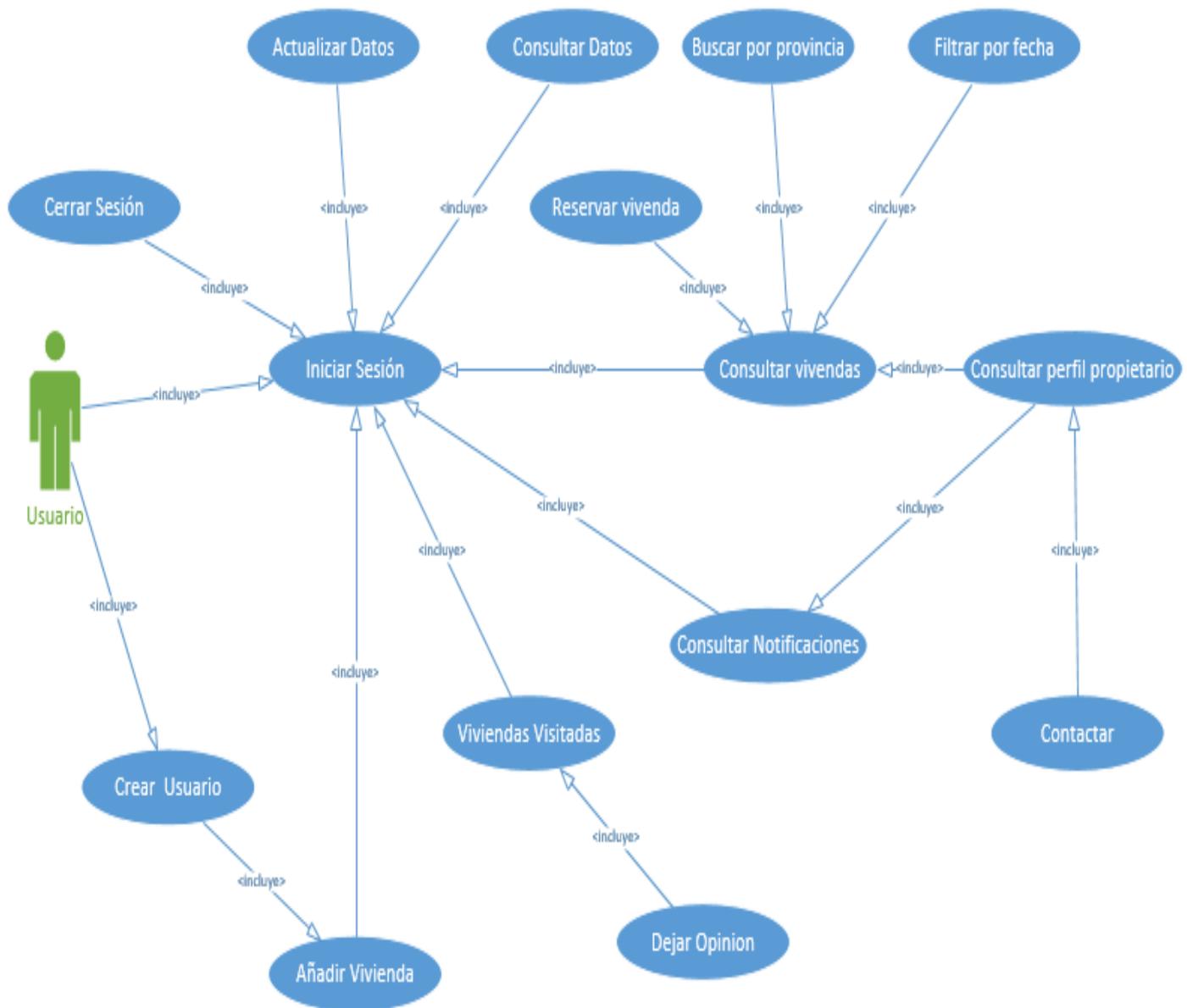
Como podemos observar en el diagrama, un usuario solo puede y está obligado a publicar un anuncio, que, a su vez, dispone de 5 imágenes. Por la otra parte, el usuario dispone de una imagen, que sería la imagen de perfil. El usuario también puede tener notificaciones, que oscilan entre ninguna y muchos. Lo mismo ocurre con las reservas y las valoraciones, un usuario puede no tener valoraciones ni reservas, como puede tener muchas de ambas.

### 3.2 CASOS DE USO

En cuanto a este apartado, veremos los diagramas de casos de uso. Como he mencionado anteriormente, vendrán diferenciados los casos de uso de los usuarios de los del administrador. Tras ver los dos diagramas, pasaremos a observar las fichas de estos casos de uso, que también vendrán diferenciados entre los del usuario y los del administrador.

## Diseño de una aplicación Android para el intercambio de viviendas

A continuación, el diagrama de casos de uso de los usuarios:



**Figura 2. Diagrama de Casos de Uso para Usuarios**

En cuanto al diagrama de usos del administrador quedaría de la siguiente manera:

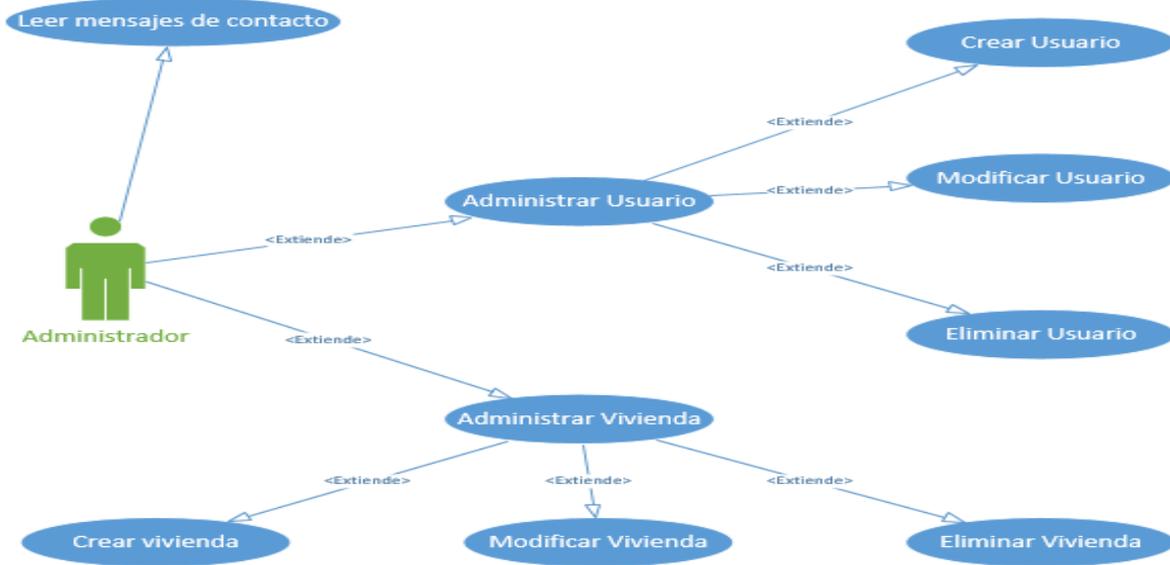


Figura 3. Diagrama de Casos de Uso para el Administrador

En los siguientes apartados, podremos observar las fichas de los casos de uso de los usuarios, primero, seguidas de las del administrador.

### 3.2.1 Fichas de casos de uso para Usuarios

Crear Usuario		
Descripción	Registrarse en la aplicación	
Precondición	No estar registrado previamente	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Entrar en la aplicación
	2	Pulsar botón "Registrarse"
	3	Rellenar todos los campos del formulario y pulsar botón para dar de alta una vivienda.
Post condición	Dar de alta una vivienda	
Excepciones	Rellenar todos los campos del formulario.	

Añadir Vivienda		
Descripción	Dar de alta una vivienda.	
Precondición	Haber creado usuario	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Crear Usuario
	2	Rellenar campos y aceptar.
Post condición	Dar de alta una vivienda y poder de esta manera acceder a la aplicación.	
Excepciones	Hay que rellenar todos los campos del formulario y subir por lo menos 3 imágenes de la vivienda.	

Iniciar sesión		
Descripción	Iniciar sesión con los datos del usuario para acceder a la aplicación.	
Precondición	Haberse registrado en la aplicación.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Entrar a la aplicación.
	2	Pulsar botón para iniciar sesión.
	3	Rellenar campos solicitados y aceptar.
Post condición	Acceder al sistema con los datos registrados.	
Excepciones	Si no está registrado el sistema le avisará y no le dejará entrar.	

Consultar viviendas disponibles		
Descripción	El usuario consulta las viviendas disponibles en cierta provincia entre fechas seleccionadas.	
Precondición	Tener una sesión abierta.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Iniciar Sesión.
	2	Pulsar botón “Viviendas”
Post condición	Pulsando una vivienda determinada se tiene acceso a datos de ésta.	
Excepciones	-	

Actualizar Datos		
Descripción	El usuario accede para actualizar sus datos.	
Precondición	Tener una sesión abierta.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Iniciar Sesión.
	2	Pulsar botón de “Editar Perfil”.
	3	Cambiar los datos deseados y guardar.
Post condición	Volver a la pantalla Principal del usuario.	
Excepciones	No se puede dejar un campo del formulario vacío.	

Consultar Datos del perfil		
Descripción	Pantalla desde la cual el usuario puede acceder a revisar sus datos.	
Precondición	Tener una sesión abierta.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Iniciar Sesión.
	2	Pulsar botón de Perfil
	3	Acceder a la pantalla de Datos del usuario
Post condición	Tener la opción de redirigirse a otras pantallas, tales como Ver Notificaciones, Viviendas Visitadas, Editar Perfil, Editar Anuncio	
Excepciones	-	

Viviendas Visitadas		
Descripción	Pantalla donde aparecen todas las viviendas que el usuario haya visitado	
Precondición	Haber iniciado la sesión.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Iniciar Sesión.
	2	Acceder a Consultar Datos del perfil.
	3	Pulsar botón de “Viviendas Visitadas”
Post condición	Se puede acceder a una vivienda para volver a ver el anuncio, o bien dejar una valoración.	
Excepciones	Solo aparecen las viviendas con fecha de reserva fin posterior a la fecha actual y que no se haya cancelado la reserva.	

Dejar Opinión		
Descripción	Dejar una valoración sobre el estado de la casa y el trato del propietario.	
Precondición	Haber visitado la vivienda de la que se quiere dejar la valoración.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Iniciar Sesión.
	2	Dirigirse a Consultar de Datos del perfil
	3	Ir al apartado de Viviendas Visitadas.
	4	Pulsar en el botón de “Dejar Opinión”
Post condición	Tras dejar opinión, se vuelve automáticamente al menú de Perfil, inhabilitándose la posibilidad de dejar otra opinión a esa reserva en particular.	
Excepciones	No se podrá dejar dos opiniones a la misma reserva.	

Consultar Notificaciones		
Descripción	En este apartado se podrán consultar las solicitudes de reserva que un usuario tiene.	
Precondición	Tener la sesión iniciada.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Iniciar Sesión
	2	Dirigirse a la pantalla de Perfil
	3	Avanzar a Notificaciones
Post condición	Desde esta pantalla se puede visitar el perfil del usuario que ha realizado la reserva y/o aceptar la solicitud.	
Excepciones	-	

Reservar Vivienda		
Descripción	Solicitar reserva en una determinada vivienda.	
Precondición	Estar en el anuncio de la vivienda.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Iniciar Sesión
	2	Dirigirse al anuncio de una vivienda.
	3	Pulsar botón de “Solicitar Reserva”.
Post condición	Solicitar reserva al propietario del anuncio.	
Excepciones	No tener reserva para el periodo solicitado.	

Buscar por provincia		
Descripción	Consultar Viviendas según situadas en la provincia deseada.	
Precondición	Estar en el apartado de Consultar Viviendas.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Iniciar Sesión.
	2	Dirigirse a Consultar Viviendas.
	3	Seleccionar provincia.
Post condición	Aparecerán solo los anuncios de las viviendas localizadas en la provincia seleccionada.	
Excepciones	Si no hay anuncios en la provincia seleccionada aparecerá un mensaje indicándolo.	

Filtrar por fechas		
Descripción	Filtrar los anuncios de las viviendas según disponibilidad.	
Precondición	Haber seleccionado previamente una provincia de destino.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Iniciar Sesión.
	2	Dirigirse a Consultar Viviendas.
	3	Seleccionar provincia.
	4	Elegir fecha de entrada y salida.
Post condición	Aparecerán los anuncios de las viviendas disponibles durante el período indicado en la provincia seleccionada.	
Excepciones	Si no hay viviendas disponibles durante ese período aparecerá un mensaje indicándolo.	

Consultar Anuncio		
Descripción	Consultar el anuncio de una vivienda.	
Precondición	El anuncio debe de estar dado de alta.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Iniciar Sesión.
	2	Dirigirse a Consultar Viviendas.
	3	Seleccionar anuncio.
Post condición	Dirigirse al anuncio de la vivienda seleccionada.	
Excepciones	-	

Consultar Perfil Propietario		
Descripción	Consultar el perfil del propietario de un anuncio.	
Precondición	Propietario dado de alta previamente.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Iniciar Sesión.
	2	Dirigirse a Consultar Viviendas.
	3	Seleccionar anuncio.
	4	Pulsar en "Ver Perfil"
Post condición	Dirigirse al perfil del perfil del propietario de un anuncio.	
Excepciones	-	

Contactar		
Descripción	Contactar con el propietario de un determinado anuncio.	
Precondición	Propietario dado de alta previamente.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Iniciar Sesión.
	2	Dirigirse a Consultar Viviendas.
	3	Seleccionar anuncio.
	4	Pulsar en “Ver Perfil”
	5	Pulsar en “Contactar”
Post condición	Dirigirse a Gmail, desde donde se puede enviar un email al usuario.	
Excepciones	-	

Cerrar Sesión		
Descripción	El usuario se desconecta de la sesión.	
Precondición	Tener una sesión abierta.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	El usuario está conectado en el sistema con su sesión.
	2	Pulsar el botón de cierre de sesión.
Post condición	Volver a la pantalla inicial de la aplicación.	
Excepciones	-	

### 3.2.2 Fichas de casos de uso para Administrador

Leer Mensajes de Contacto		
Descripción	Leer mensajes de contacto enviado por usuarios al encontrar un problema con la aplicación.	
Precondición	El mensaje ha sido generado por el usuario en la pestaña de contacto	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Acceder al correo electrónico creado para este propósito.
	2	Leer los mensajes recibidos.
Post condición	Responder al mensaje.	
Excepciones	-	

Crear Usuario		
Descripción	Crear un usuario que no ha podido realizarlo por su cuenta por culpa de algún problema.	
Precondición	El usuario no se ha registrado previamente.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Acceder al servidor web con el que está conectado la aplicación.
	2	Crear un usuario.
Post condición	Creación de un usuario.	
Excepciones	Todos los campos deben ser rellenados.	

Modificar Usuario		
Descripción	Modificar datos de un usuario registrado	
Precondición	Estar registrado.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Acceder al servidor web con el que está conectado la aplicación.
	2	Modificar datos del usuario en cuestión.
Post condición	Actualizar datos de un usuario.	
Excepciones	No dejar ningún campo sin rellenar	

Eliminar usuario		
Descripción	Eliminar un usuario por no cumplir con las condiciones.	
Precondición	Estar registrado.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Acceder al servidor web con el que está conectado la aplicación.
	2	Eliminar el usuario y su anuncio.
Post condición	Eliminación del usuario.	
Excepciones		

Crear una vivienda		
Descripción	Dar de alta una vivienda a un determinado usuario.	
Precondición	El usuario tiene que tener una cuenta.	
Secuencia Normal	<b>Paso</b>	<b>Acción</b>
	1	Acceder al servidor web con el que está conectado la aplicación.
	2	Crear una vivienda
Post condición	Creación de una vivienda.	
Excepciones	-	

Modificar una vivienda		
<b>Descripción</b>	Modificar una vivienda dada de alta.	
<b>Precondición</b>	Vivienda dada de alta previamente.	
<b>Secuencia Normal</b>	<b>Paso</b>	<b>Acción</b>
	1	Acceder al servidor web con el que está conectado la aplicación.
	2	Actualizar datos de una vivienda.
<b>Post condición</b>	Actualización de los datos de una determinada vivienda.	
<b>Excepciones</b>	-	

Eliminar Vivienda		
<b>Descripción</b>	Eliminar una vivienda que no cumple con las condiciones.	
<b>Precondición</b>	Vivienda dada de alta previamente.	
<b>Secuencia Normal</b>	<b>Paso</b>	<b>Acción</b>
	1	Acceder al servidor web con el que está conectado la aplicación.
	2	Eliminar la vivienda.
<b>Post condición</b>	Eliminación de una vivienda.	
<b>Excepciones</b>	-	

## 4. FASE DE DISEÑO

### 4.1 INTRODUCCIÓN

En esta sección observaremos la fase de diseño de la aplicación. Empezaremos con la capa de presentación donde vamos a tener bocetos y muestras de cómo sería la aplicación. Para la realización de esta parte se ha utilizado la aplicación web **“ninjamock”**.

Acto seguido se describirá la capa de persistencia donde se detalla el diseño de la base de datos que se va a utilizar mediante un diagrama de clases.

Para terminar, se mostrará la capa lógica, que mediante un diagrama de flujo se representará cómo se navegaría por la aplicación.

### 4.2 CAPA DE PRESENTACIÓN

Como he mencionado anteriormente, se va a utilizar la aplicación web **“ninjamock”** para la realización del diseño de la interfaz gráfica. Es una aplicación que permite realizar prototipos de manera sencilla y rápida.

La interfaz de la aplicación **“ninjamock”** es la que se muestra en la siguiente captura:



**Figura 4. Interfaz de la aplicación web “ninjamock”**

Los prototipos que se van a mostrar en este apartado tienen la finalidad de representar los aspectos interactivos de la aplicación con un cierto nivel de precisión. Y así poder evaluar su funcionalidad.

**Al Abrir la aplicación**

En el momento de abrir la aplicación aparecerá una pantalla como la que se va a mostrar a continuación.



**Figura 5. Pantalla inicial de la aplicación.**

Como se puede observar en la imagen, esta pantalla contendrá dos botones. El primero, el de "Log in" nos dirigirá a la pantalla de inicio de sesión, mientras que el otro, nos dirigirá a la pantalla para registrarse en la aplicación.

### Pantalla para Crear Cuenta

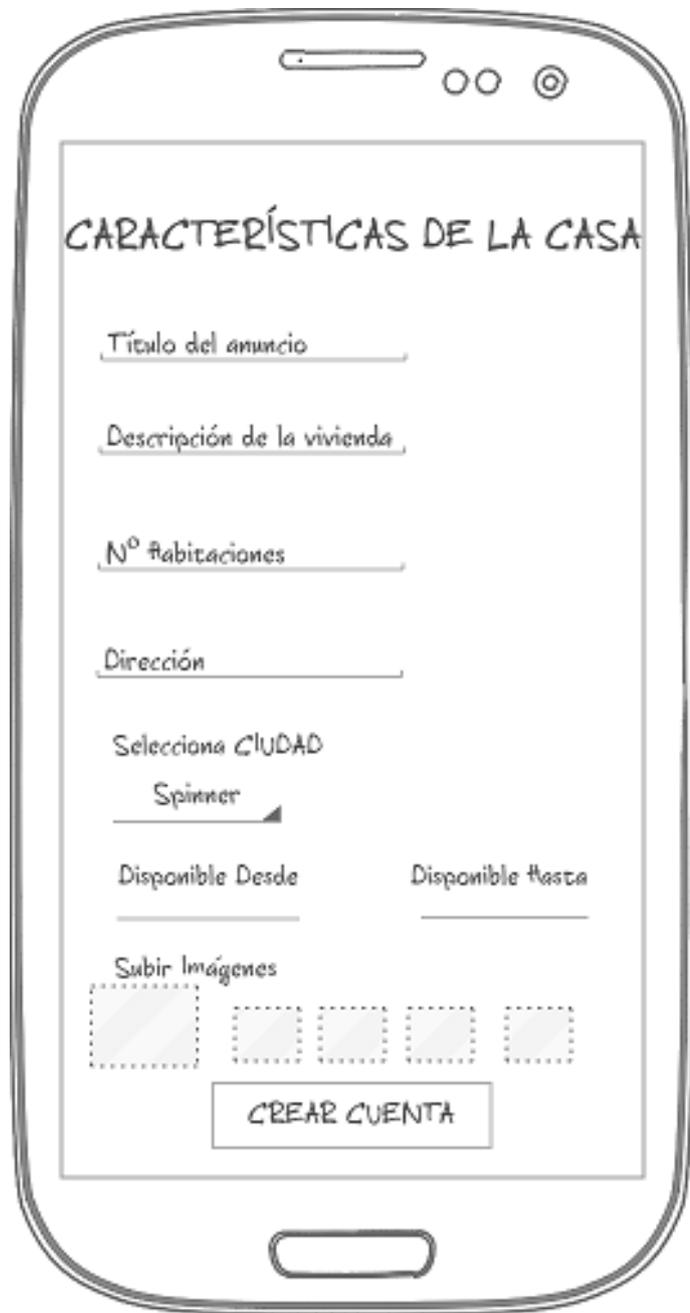
Si es la primera vez que entramos a la aplicación habrá que registrarse como usuario y añadir la vivienda que se quiere intercambiar para poder tener acceso.



The image shows a wireframe of a mobile application screen for creating an account. The screen is titled "Crear Cuenta" and contains several input fields: "Sube Imagen de perfil" with a dashed box icon, "Nombre", "Apellidos", "Edad", "Nombre de Usuario", "Contraseña", "E-Mail", "Teléfono", and "¿Quién Soy?". A button labeled "Añadir Vivienda" is positioned at the bottom of the form area.

**Figura 6. Pantalla para crear usuario.**

En esta pantalla se creará el usuario, rellenando todos los campos requeridos, para el correcto funcionamiento del sistema. Cuando se haya rellenado todos los campos se pulsa en el botón de Añadir Vivienda para registrar la vivienda.



**Figura 7. Pantalla para añadir vivienda.**

Una vez rellenados los datos del usuario, se pasa a esta pantalla donde se apuntarán las características de la vivienda y subir imágenes de ésta. También se indicará un período de intervalo que en cualquier momento se podrá actualizar.

**Pantalla de Login**

La imagen que se va a mostrar a continuación, mostrará cómo será la pantalla de inicio de sesión.



**Figura 8. Pantalla para iniciar sesión.**

A través de esta pantalla accedemos al sistema. Para ello tenemos que rellenar los campos de “nombre de usuario” y “contraseña”; si los datos introducidos son correctos se accederá al sistema.

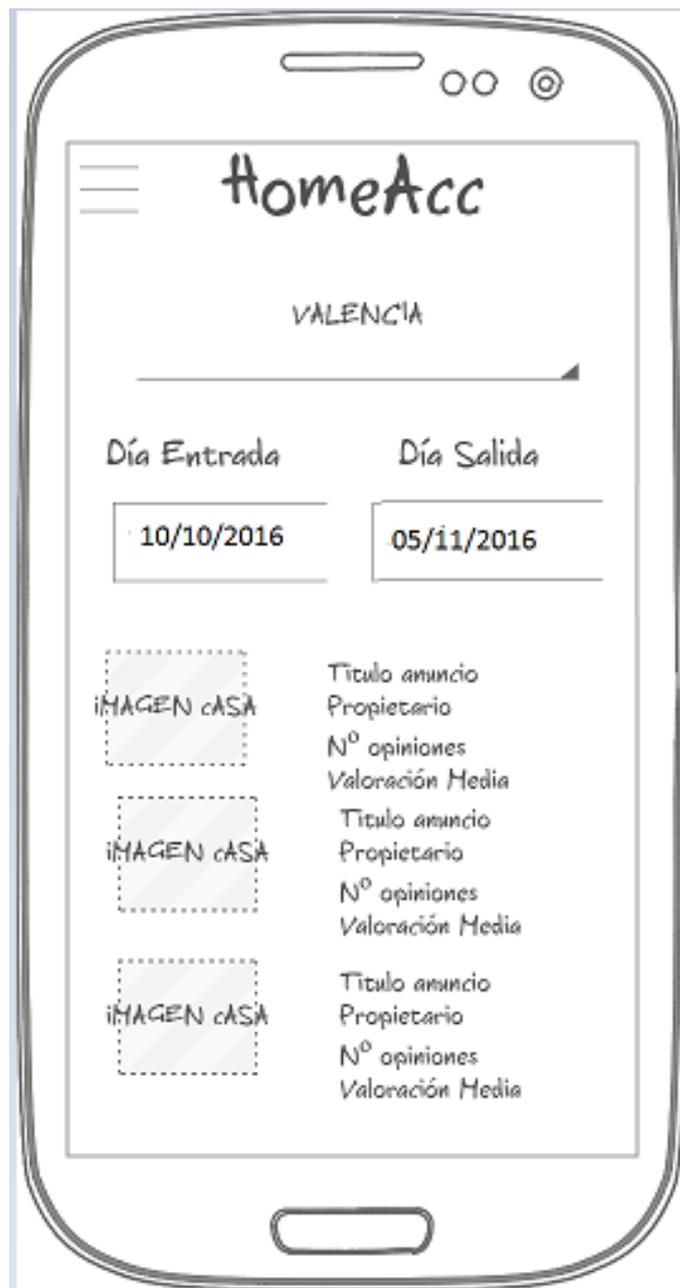
### **Buscar Vivienda**

Una vez que se accede al sistema aparecerá la pantalla donde se buscarán los anuncios publicados y disponibles.



**Figura 9. Pantalla donde se buscan los anuncios.**

En esta pantalla se podrán buscar los anuncios disponibles en la ciudad deseada. Para ello habrá que elegir la ciudad de destino y las fechas de salida y entrada, para comprobar de esta manera que la vivienda está disponible para ese período elegido.

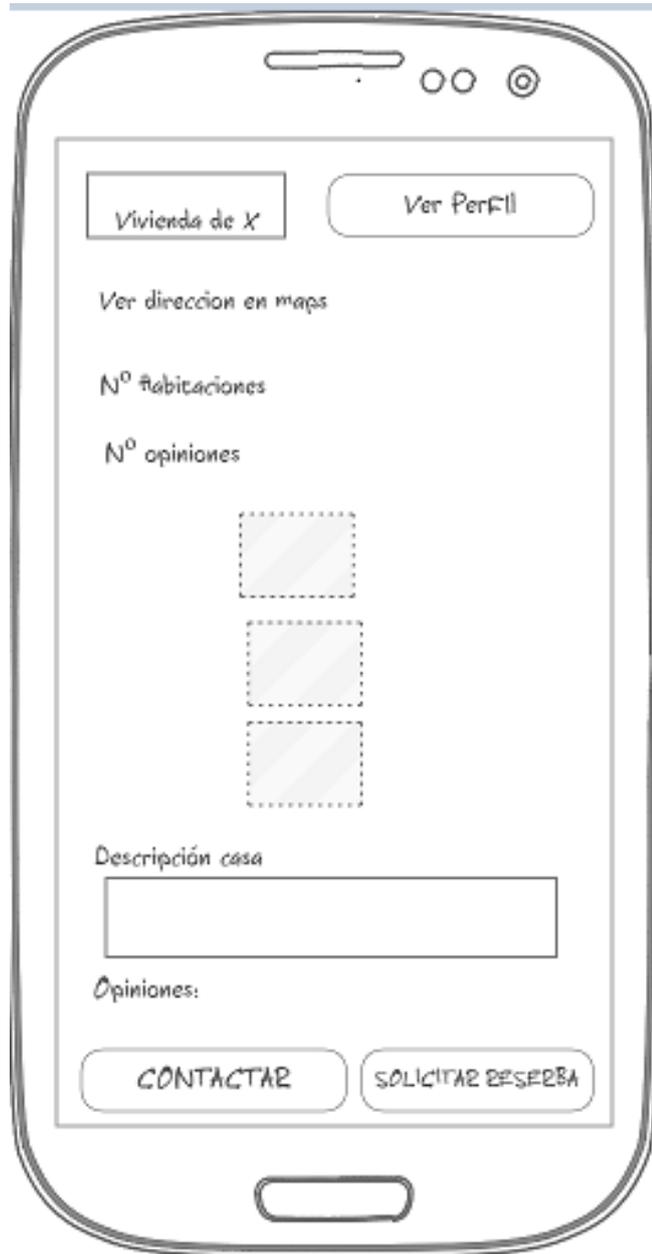


**Figura 10. Pantalla con anuncios publicados.**

La estructura de los anuncios correspondientes a su búsqueda tendrá un aspecto parecido al de la imagen anterior.

### **Pantalla de Detalles del Anuncio**

En la siguiente interfaz observaremos como debería resultar la pantalla con los detalles de un determinado anuncio. Para acceder a esta pantalla se pulsa en el anuncio deseado en la pantalla anterior.

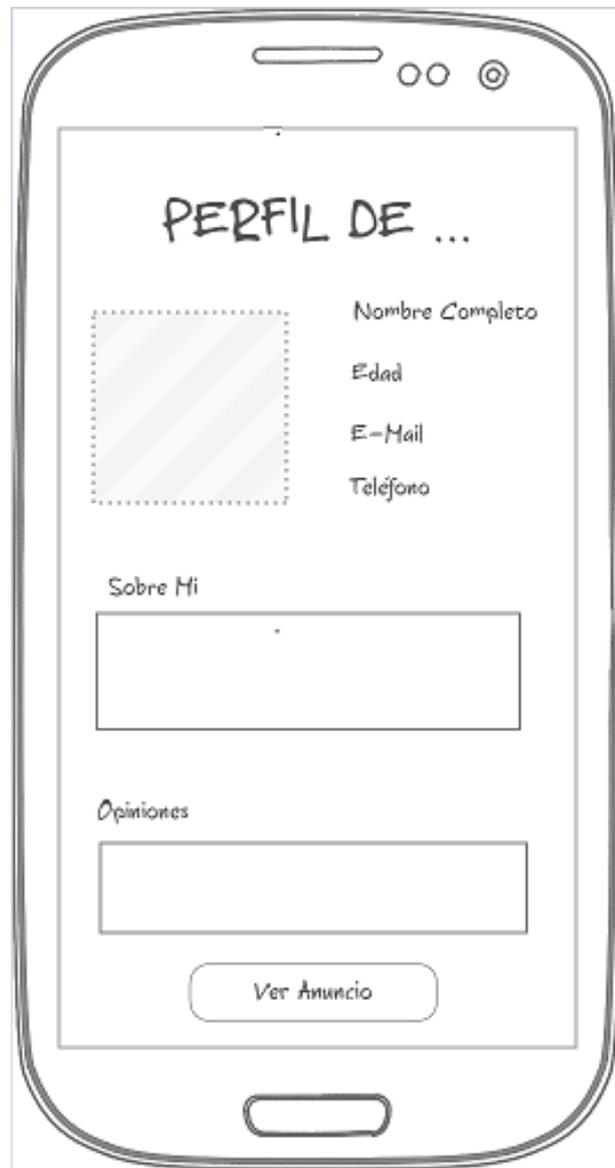


**Figura 11. Pantalla con detalles de un anuncio.**

Con esta pantalla el usuario podrá obtener la información necesaria de un determinado anuncio, incluidas imágenes de la vivienda. Aquí podrá contactar con el usuario, solicitar reserva, o bien, si necesita conocer más detalles del anfitrión podrá acceder a detalles del usuario pulsando "Ver Perfil".

**Pantalla de Perfil de un Usuario**

Con esta imagen se representa la interfaz que tendrá la pantalla de Perfil de un usuario.



**Figura 12. Pantalla de Perfil de un usuario.**

Como se puede observar, en esta pantalla aparecerá información acerca de un usuario, incluidos las opiniones que otros usuarios dejaron sobre él tras realizar un intercambio.

### **Pantalla para Solicitar Reserva**

Esta imagen representa la ventana de solicitud de reserva, donde un usuario solicitará un intercambio de viviendas con otro.



**Figura 13. Pantalla de Solicitud de Reserva.**

Tras elegir los días de entrada y salida, tenemos dos opciones. Por una parte, se puede enviar la solicitud, o por otra parte se cancela y se vuelve a la pantalla principal.

### Menú Navegador

Este menú puede aparecer desde cualquier actividad, saliendo por la parte izquierda de nuestras pantallas.



**Figura 14. Manú de Navegación.**

Con este menú, además de mostrar nuestra imagen de Perfil, nombre de usuario y correo electrónico, podemos cerrar la sesión del sistema, o bien, acceder a varias actividades como Perfil, Buscar Vivienda, Notificaciones o Sobre Nosotros:

### **Pantalla de Perfil**

Con esta pantalla podemos acceder a diferentes aspectos de nuestro perfil, para comprobar informaciones o modificar datos registrados por nosotros.

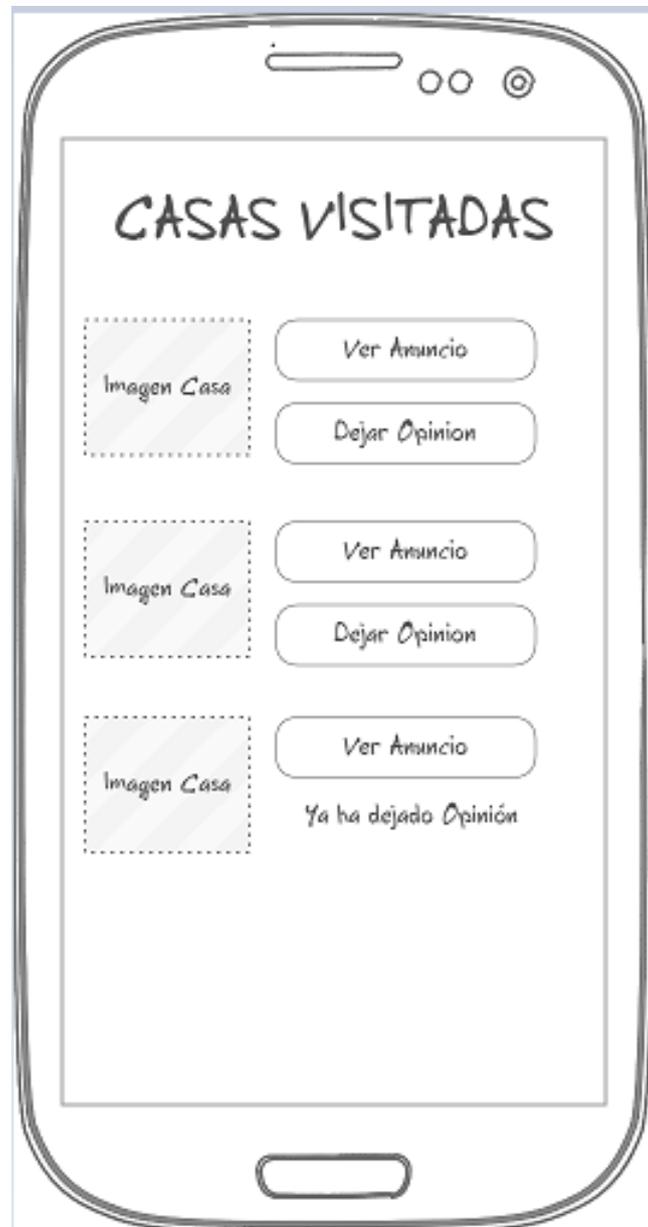


**Figura 15. Pantalla de Perfil**

En este diseño no mostraremos las pantallas de editar perfil y editar anuncio porque serán iguales que las pantallas a la hora de registrarse con la diferencia de venir con los datos ya completados anteriormente y que se pueden modificar.

### **Pantalla con las Casas Visitadas**

En esta pantalla se mostrarán las viviendas que el usuario ha visitado, y que haya finalizado su estancia en esa vivienda.

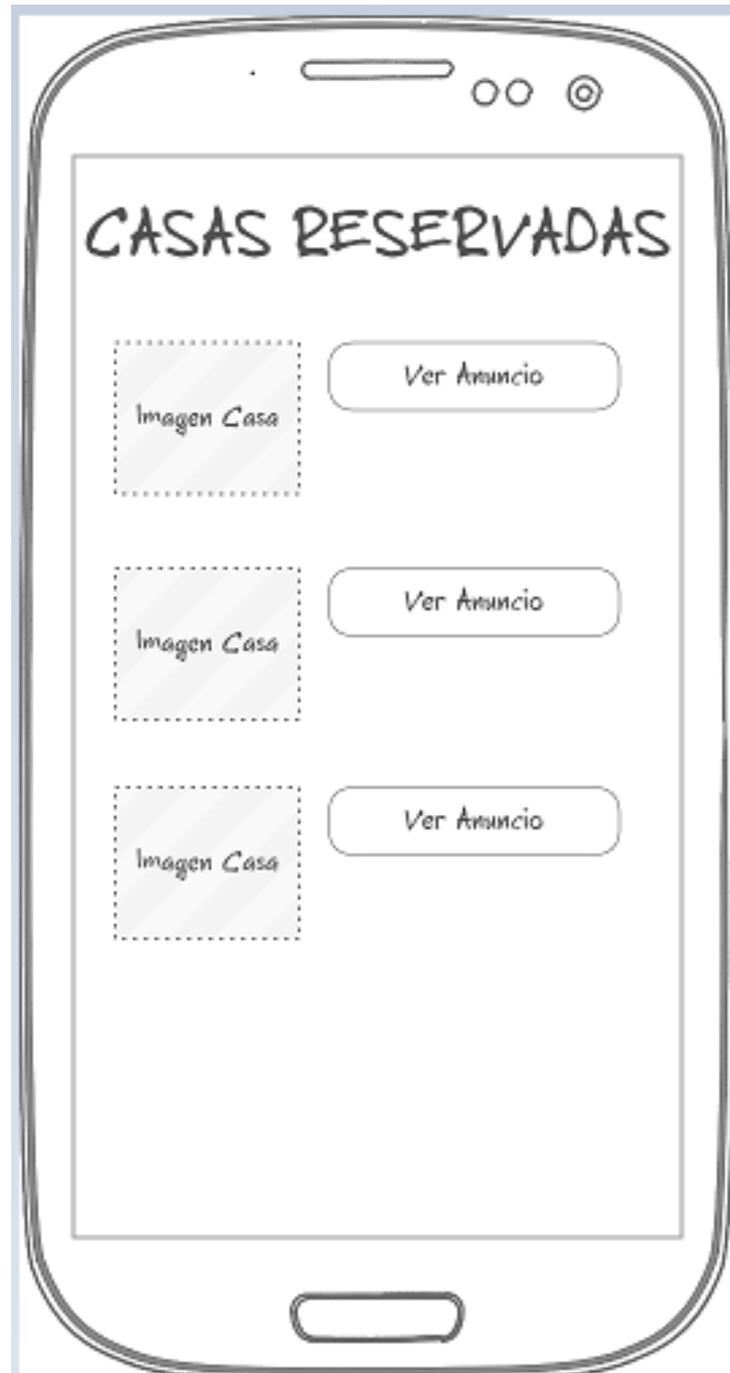


**Figura 16. Pantalla con las Casas Visitadas**

Como se observa en esta pantalla se tiene que poder Ver el Anuncio de las viviendas visitadas y dejar opinión, si todavía no se ha evaluado esa visita.

### Pantalla con las Casas Reservadas

En esta pantalla se encontrarán las Casas Reservadas para una futura visita.

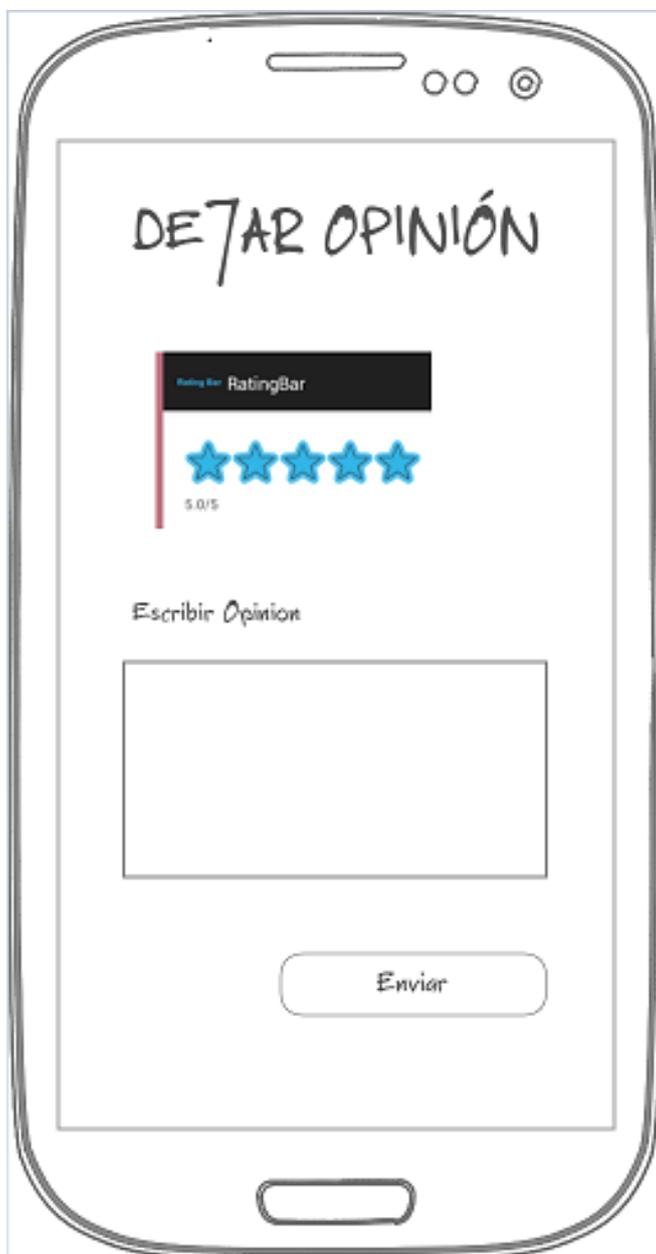


**Figura 17. Pantalla con las Casas Reservadas**

Con esta pantalla no tenemos bastantes opciones, solo está la opción de ver el anuncio de la Vivienda Reservada.

### Pantalla para Dejar Opinión

Con la siguiente imagen se puede comprobar la interfaz de la ventana que tendremos para dejar una opinión.

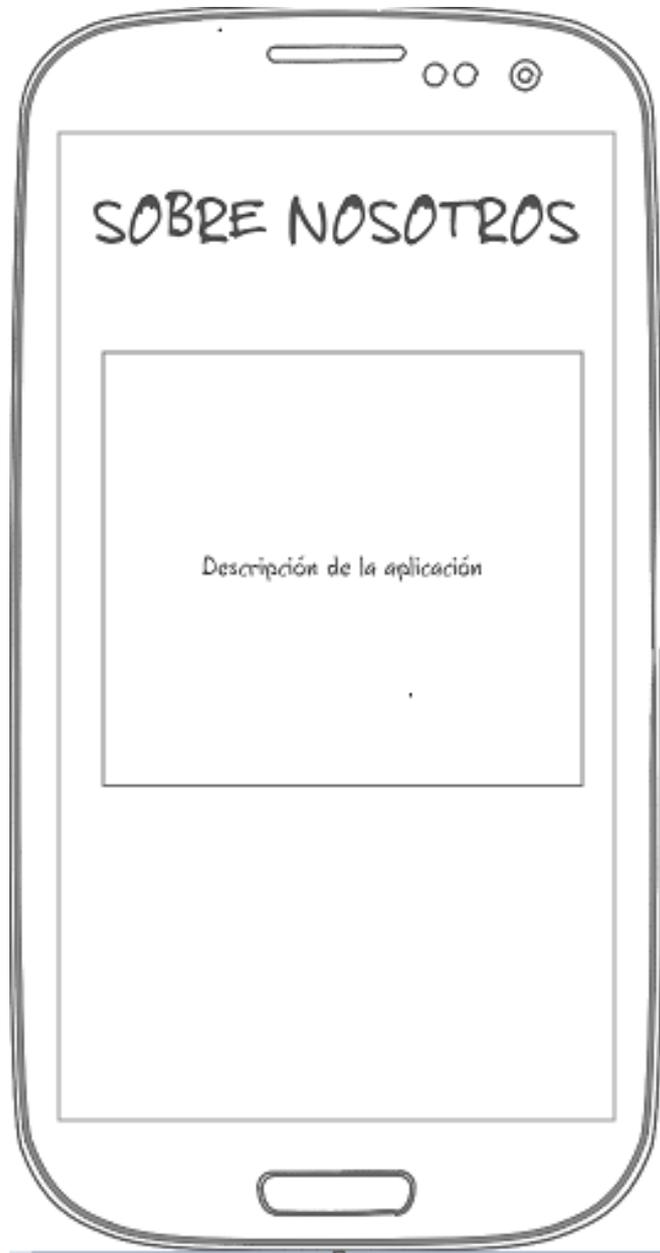


**Figura 18. Pantalla para Dejar Opinión**

En esta pantalla tenemos tres campos destacables. El primero es la barra de estrellas con la que se va a puntuar la experiencia vivida en ese intercambio. Con el siguiente campo se puede detallar más y explicar las razones de esa puntuación. Para terminar, tenemos el botón de Enviar con el que se sube dicha valoración.

**Pantalla Sobre Nosotros**

En esta pantalla aparecerá información relevante sobre la aplicación y datos de contacto por si se quiere contactar con el dueño de la aplicación.



**Figura 19. Pantalla Sobre Nosotros**

### 4.3 CAPA DE PERSISTENCIA

Tras un análisis de cómo sería la base de datos y las tablas que se necesitarían para cumplir con toda la funcionalidad requerida para la aplicación, el esquema de la base de datos quedaría de la siguiente manera:



Figura 20. Esquema de la Base de Datos

Las tablas más importantes son las de **Vivienda** y **Usuario**. En ambas tablas se insertarán los datos cuando el usuario se registre rellenando sus datos y los de la Vivienda.

En cuanto a las siguientes tablas, tenemos, por una parte, las relacionadas con las viviendas, como son: **Visitadas – Reservas e Imágenes\_Vivienda**.

Por otra parte, tenemos aquellas tablas relacionadas con la tabla de Usuario: **Notificaciones – Valoraciones – ImagenPerfil y Reservas**.

Ya para terminar con esta fase de relaciones, tenemos la tabla de **Visitadas**, que podría estar perfectamente relacionada directamente con las tablas de Vivienda y Usuario, pero ya que es una tabla que depende de las Reservas, puesto que no se puede visitar una vivienda sin tener una reserva previamente, he optado con relacionarla con esa tabla.

Como he mencionado anteriormente, la tabla **Reservas** está relacionada tanto con la tabla de Vivienda como con la de Usuario. Esto se debe a que la tabla de Reservas tiene que estar sujeta a los usuarios receptor, que es el que recibe la solicitud de reserva, y solicitante, que como el propio nombre indica, es el usuario que realiza la solicitud de reserva. A su vez, tiene que estar relacionada con una vivienda. Por cada reserva tendremos dos filas. En la primera estará la del solicitante junto a la vivienda del receptor, y en la segunda, la del receptor junto a la vivienda del solicitante.

Las tablas de **Imágenes\_Vivienda** e **ImagenPerfil**, corresponderían a las imágenes de la vivienda que el usuario haya insertado a la hora de registrar la vivienda, y la imagen de perfil que éste haya subido junto a sus datos.

En la tabla de **Valoraciones** tendremos las diferentes opiniones y puntuaciones que los usuarios han realizado. Para aclarar más esta tabla cabe destacar el atributo **ID\_Usu\_Rec\_Not**. Este atributo puede tener tres valores, el 0, el 1 o el 2. Si su valor es 0 significa que el usuario receptor todavía no ha respondido la solicitud de reserva que le han hecho. Si el valor es un 1 significa que el receptor ha visto la solicitud y ha decidido aceptarla. Por último, el valor 2 significa que el usuario ha rechazado la solicitud de reserva.

Como podemos observar, a la hora de realizar el primer esquema de cómo sería la Base de Datos, nos sale un total de ocho tablas. Al finalizar la aplicación, me he quedado únicamente con seis tablas simplificando el esquema original, de tal manera, que me he deshecho de las tablas de “Imágenes\_Vivienda” e “ImagenPerfil” metiendo sus datos en las tablas de “Vivienda” y “Usuario”, respectivamente. También he acabado relacionando la tabla de Valoraciones con la de Visitadas. De esta manera solo se podría evaluar las viviendas que se han visitado previamente.

## 4.4 CAPA DE LÓGICA

En este apartado se representará las acciones que un usuario puede realizar en la aplicación junto con su recorrido, que se hará mediante un diagrama de flujos.



## Diseño de una aplicación Android para el intercambio de viviendas

En el siguiente cuadro podemos observar que significan cada una de las formas e iconos que observaremos en el diagrama.

	Con esta forma se representarán las pantallas principales de la aplicación: Registrarse, Iniciar Sesión, Perfil, Buscador, Notificaciones y Sobre Nosotros.
	Con este icono se representarán las acciones que el usuario puede realizar.
	Con esta forma se representarán los momentos de toma de decisión, que según la elección del usuario se redirigirá a una pantalla u otra.
	Este icono representará el inicio y la finalización de la aplicación que supone el cierre de sesión. Vamos a representar el cierre de sesión solo tras aceptar una solicitud aunque en realidad se puede realizar desde cualquier pantalla de la aplicación.



## 5. DETALLES DE IMPLEMENTACIÓN

### 5.1 INTRODUCCIÓN

Tras presentar el diseño de la aplicación que se va a realizar para este proyecto, en este capítulo veremos las cuestiones relacionadas con la implementación realizada para conseguir nuestro objetivo.

En primer lugar, veremos cómo se ha implementado la app Android. Para ello, se tratará el código Java que da cuerpo a la aplicación.

Acto seguido, veremos cómo se realizó la parte de diseño de las pantallas, en XML en su gran mayoría, y a los que la aplicación tiene acceso.

El siguiente paso será ver cómo se ha implementado la parte de comunicación entre la aplicación y el servidor Web, donde están guardados todos los datos y ficheros relevantes de la aplicación. Esta parte está implementada en PHP.

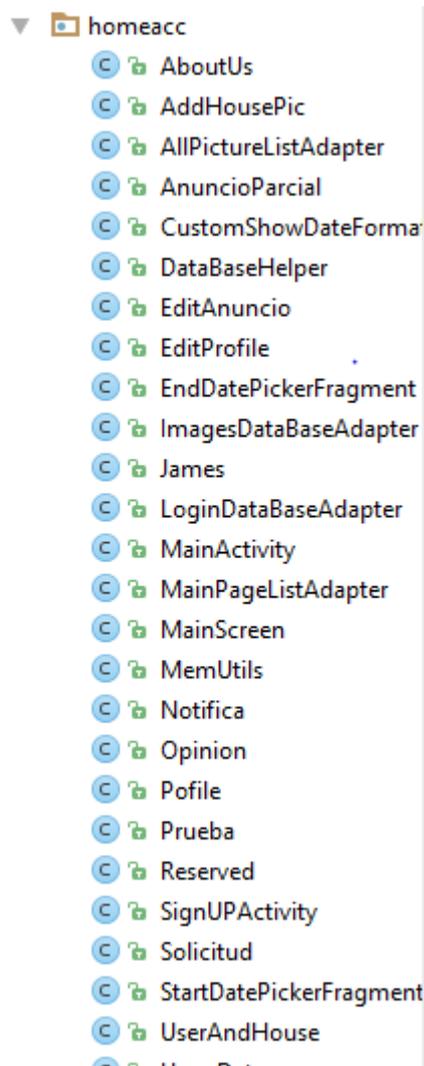
### 5.2 APLICACIÓN ANDROID

La aplicación se ha realizado implementando la app en diferentes paquetes Java. De esta manera el código está mejor organizado y se puede acceder a las clases y archivos deseados cómodamente. El proyecto está separado en dos directorios principales. Por un lado, tenemos *src*, que contiene el código fuente en Java, y por el otro tenemos *res*, con ficheros XML donde vienen definidas las interfaces de la aplicación, y algunas imágenes en formato *png* que se utilizan como iconos en diferentes partes de la aplicación.

Vamos a empezar entrando en detalle del directorio *src* o código Java, donde podemos encontrar la estructura en paquetes del proyecto. En la figura 23[ver en la siguiente página] podemos ver cómo están estructuradas algunas de las clases Java del paquete “HomeAcc”, el paquete principal del proyecto.

En este paquete encontramos todas las clases Java que realizan la gran mayoría de las funcionalidades de la aplicación. Para poner un ejemplo vamos a hablar de la clase ***SignUpActivity***:

En esta clase viene definido el comportamiento de la aplicación a la hora de registrarse un usuario, la parte de inserción de datos personales para ser más exactos. Esta clase se asocia con el código XML, del que hablaremos más tarde, mediante la llamada “***setContentView(R.layout.activity\_sign\_up)***”, siendo “***activity\_sign\_up***” la actividad donde encontraremos la interfaz de la pantalla de registro. Con esta clase comprobaremos que un usuario ha rellenado todos los campos requeridos y que se cumplan con los requisitos de la Base de Datos como, por ejemplo, controlar que dos personas no tengan el mismo nombre de usuario. A su vez esta clase se dirige mediante la actividad *onClick()* a otra clase, donde el usuario añade su vivienda.



**Figura 23.** Carpeta HomeAcc

Por otra parte, tenemos el paquete de persistencia, que es uno de los paquetes más importantes de la aplicación. En este paquete podemos encontrar dos componentes que se utilizan y a los que se llama desde prácticamente todas las clases. Estos dos componentes son **APIManager** y **DataManager**.

Vamos a empezar describiendo **DataManager**; este componente contiene todas las variables que luego se irán utilizando entre clases y entre búsquedas e implementaciones, es decir, en este componente encontramos variables estáticas, que no cambian su valor, que se van utilizando para conectar diferentes situaciones. Por ejemplo, tenemos la variable **"userName"**. Se guarda el valor de esta variable a la hora de conectarse el usuario. De esta manera, cuando necesitamos conocer el nombre del usuario para alguna función, como es el caso de solicitar la reserva, solo tenemos que dirigirnos a **DataManager.userName**.

En cuanto a **APIManager**, es un componente más complejo y, según se mire, más importante. En este componente tenemos las diferentes funciones con las que nos conectamos al servidor. En la figura 24, que aparece a continuación de este párrafo, podemos ver un ejemplo de un método, de los más sencillos en **APIManager**, con el que se obtiene los datos de nuestro anuncio, que como se puede observar se utiliza el *userName* de **DataManager** como se ha explicado anteriormente para conocer el nombre de nuestro usuario.

```
public static JSONArray obtenerDatosAnuncio() {
    JSONArray json = new JSONArray();
    try {
        String url = DataManager.strUrl + "apis/datosAnuncio.php";
        HttpClient httpClient = new DefaultHttpClient();
        HttpContext localContext = new BasicHttpContext();
        HttpPost httpPost = new HttpPost(url);
        MultipartEntity entity = new MultipartEntity();

        entity.addPart("userDestination", new StringBody(DataManager.userName));

        httpPost.setEntity(entity);
        HttpResponse response = httpClient.execute(httpPost,
            localContext);
        InputStream inputStream = response.getEntity().getContent();
        String result;
        if (inputStream != null) {
            result = convertInputStreamToString(inputStream);
            json = new JSONArray(result);
        }
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return json;
}
```

**Figura 24. Ejemplo de método de APIManager.**

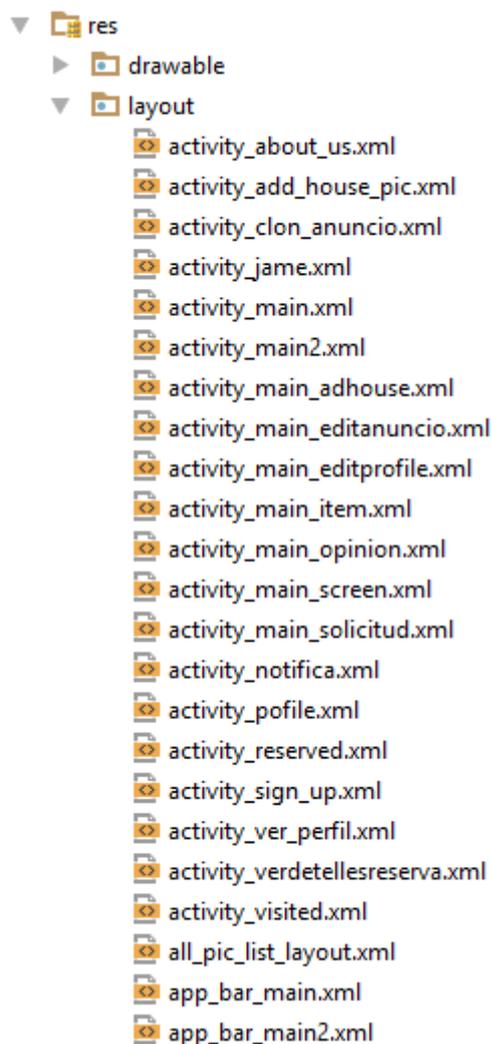
No vamos a entrar ahora en detalle de cómo se envía y se recibe la información, puesto que lo miraremos en el apartado 5.4, aunque sí quería aclarar que para llamar a estos métodos se realiza a través de **Async Tasks**, una clase que permite el uso apropiado y fácil del hilo de interfaz de usuario. Esta clase permite realizar operaciones en segundo plano y publicar los resultados sobre el hilo de interfaz de usuario sin tener que manipular los hilos y / o *handlers*. Cuando empieza su ejecución muestra un diálogo de espera, y lo oculta al terminar. En el anexo podemos comprobar ejemplos de clases de **AsyncTask**.

### 5.3 INTERFACES CON XML

En este apartado vamos a hablar de cómo se realizaron las interfaces de las pantallas, para las cuales se ha utilizado principalmente el lenguaje XML.

Estas interfaces las podemos encontrar en el directorio `res`, junto a otros archivos estáticos, que como se ha explicado anteriormente, son elementos que no cambian. Entre estos archivos podemos encontrar “*styles*” dentro de la carpeta “*values*”. En este archivo podemos encontrar los diferentes estilos que más tarde podemos utilizar en una gran variedad de elementos de nuestra aplicación, como, por ejemplo, dar forma a los botones. Otro archivo destacable dentro de *values* es `colors.xml`, donde podemos definir los colores que se van a utilizar en nuestra aplicación.

En la siguiente figura podemos observar parte de la estructura del directorio `res`, un directorio muy importante, por no decir imprescindible, a la hora de realizar las interfaces de nuestra aplicación.



**Figura 25. Carpeta `res`**

En la figura anterior destaco los archivos de la carpeta *layout*, porque son los archivos donde vienen definidas las diferentes interfaces de las pantallas.

Volviendo al tema de las interfaces de las pantallas, cabe destacar que desde un principio se ha seguido la misma dinámica. Primero se toman los elementos y controles Android existentes y se van variando con colores y formas más adecuadas para la aplicación, de manera que la interfaz de la aplicación se adapte a la aplicación y el parecido entre las diferentes pantallas indica que la aplicación no está muy cargada.

### 5.4 PHP Y SERVIDOR WEB

Antes de entrar en más detalle, voy a hacer una breve descripción de lo que son PHP y JSON.

PHP (*Pre Hypertext -processor*) es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

JSON, acrónimo de *JavaScript Object Notation*, es un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de *JavaScript*, aunque hoy, debido a su amplia adopción como alternativa a XML, se considera un formato de lenguaje independiente.

En cuanto a mi proyecto he utilizado mucho el PHP y JSON como intermediarios entre los datos de mi aplicación y la base de datos donde está todo almacenado.

Como vimos en el capítulo anterior, el *APIManager* se conecta con los archivos PHP, que están guardados en una carpeta situada en el servidor. Con estos archivos PHP obtenemos, insertamos, borramos y modificamos datos de la base de datos. También nos sirven como comprobadores, como a la hora de registrar un usuario se comprueba que el nombre de usuario no está ya registrado en la base de datos, o cuando queremos iniciar sesión se comprueba que los datos introducidos son correctos.

Para el intercambio de datos entre PHP y nuestro código utilizamos JSON, mediante el método Post del protocolo HTTP. A este método se le pasa el mensaje JSON a enviar, y la ruta relativa del servicio web a utilizar. En caso de correcto funcionamiento devuelve la respuesta, en caso contrario, devuelve *fail* y el error será presentado de forma agradable al usuario en capas superiores.

A continuación, veremos un ejemplo de esta situación, desde que se llama a la *API* desde nuestra clase JAVA hasta que se devuelven los resultados.

El ejemplo que vamos a ver es el de la inserción de una opinión a un usuario después de haber visitado su vivienda.

Para empezar, vamos a mostrar la parte de código JAVA de la clase "*Opinion.class*".

```

buttonEnviar.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        DataManager.opinion=opinion.getText().toString();
        DataManager.estrellas=puntuacion.getRating();
        new async_InsertarOpinion().execute();
    }
});

}

private class async_InsertarOpinion extends AsyncTask<String, Void, String>
{
    private ProgressDialog dialog;
    String result;

    @Override
    protected void onPreExecute() {
        dialog = ProgressDialog.show(Opinion.this, null, null, true);
        dialog.setContentView(R.layout.progress_dlg );
        dialog.getWindow().setBackgroundDrawable(new
ColorDrawable(android.graphics.Color.TRANSPARENT));
        dialog.show();
    }
    @Override
    protected String doInBackground(String... params) {
        try {
            result= APIManager.insertarOpinion();
        } catch (JSONException e) {
            e.printStackTrace();
        }
        return null;
    }

    @Override
    protected void onPostExecute(String res) {

        if (dialog.isShowing()) {
            dialog.dismiss();
        }
        if (result.equals("success")){
            Toast.makeText(Opinion.this, "Se ha registrado la opinión.
Gracias.", Toast.LENGTH_SHORT).show();
            Intent i = new Intent(Opinion.this, Visited.class);
            startActivity(i);
        }

        else Toast.makeText(Opinion.this, "No se ha podido registrar la
opinión. Vuelva a intentarlo más tarde.", Toast.LENGTH_SHORT).show();
    }
}

```

Figura 26. Parte del código de la clase Opinion desde donde se accede a la API.

Lo primero que hay que destacar es que se accede a la API una vez se ha pulsado el botón enviar. Tras pulsar dicho botón, se guardan los valores de la opinión, tanto el comentario como la puntuación dada. Acto seguido, se llama a la clase *async\_InsertarOpinion* donde se realizan tres acciones claves. Primero se accede al

método **PreExecute()**, de esta manera, antes de ejecutar la parte de la API aparece un diálogo, que en realidad, es una barra de progreso. Tras ello, nos dirigimos al método **doInBackground()**, que accede a la API y guarda el resultado que devuelve ésta en la variable *result*. Ya para finalizar, tenemos el método **onPostExecute()**, que analiza lo devuelto por la API. En caso de éxito aparece el mensaje "Se ha registrado la opinión. Gracias" y volvemos a la clase **Visited**, donde está el listado de las viviendas visitadas. En caso contrario, aparece el mensaje "No se ha podido registrar la opinión. Vuelva a intentarlo más tarde." y permanecemos en la misma pantalla.

A continuación, vamos a analizar la parte de la API:

```

public static String insertarOpinion() throws JSONException {
    JSONObject json = new JSONObject();
    try {
        String url = DataManager.strUrl + "apis/insertarOpinion.php";
        HttpClient httpClient = new DefaultHttpClient();
        HttpContext localContext = new BasicHttpContext();
        HttpPost httpPost = new HttpPost(url);
        MultipartEntity entity = new MultipartEntity();

        entity.addPart("userName", new StringBody(DataManager.userName));
        entity.addPart("userDestination", new
StringBody(DataManager.userDestination));
        entity.addPart("opinion", new StringBody(DataManager.opinion));
        entity.addPart("estrellas", new
StringBody(Float.toString(DataManager.estrellas));
        entity.addPart("id_visitada", new
StringBody(Integer.toString(DataManager.id_visitada));

        httpPost.setEntity(entity);
        HttpResponse response = httpClient.execute(httpPost,
            localContext);
        InputStream inputStream = response.getEntity().getContent();
        String result;
        if (inputStream != null) {
            result = convertInputStreamToString(inputStream);
            json = new JSONObject(result);
        }
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return json.getString("status").toString();
}
    
```

**Figura 27. Método InsertarOpinion de la clase APIManager.**

En este método empezamos indicando la url del servidor donde está almacenado nuestro archivo php. Tras ello, realizamos una conexión httpPost, pasándole los valores necesarios para efectuar de manera satisfactoria nuestra inserción. Acto seguido, y tras recibir el mensaje que nos devuelve el archivo php, en este caso un *String* con valor

**success** si la operación ha tenido éxito o **fail** en caso contrario, integramos los valores mediante JSON y devolvemos el mensaje entregado por el archivo php.

Ahora pasamos a ver la parte de php:

```
<?php
include_once('includes/connect_database.php');
include_once('variables/variables.php');

$username = $_POST["userName"];
$userDestination = $_POST["userDestination"];
$opinion = $_POST["opinion"];
$estrellas = $_POST["estrellas"];
$estrellas = floatval($estrellas);
$id_visitada = $_POST["id_visitada"];
$id_visitada = intval($id_visitada);

$sql_query = "INSERT INTO tbl_opiniones(escritoPor, escritaPara, Opinion, Estrellas,id_visitada)
              VALUES ('{$username}','{$userDestination}','{$opinion}','{$estrellas}','{$id_visitada}')";
$result = $connect->query($sql_query); // or die ("Error :".mysql_error())

$sql_query="UPDATE tbl_visited_houses set opinada=1 where id = '{$id_visitada}'";
$result = $connect->query($sql_query);
if ( $result) {
    echo '{"status":"success"}';
}
else {echo '{"status":"fail"}';}

include_once('includes/close_database.php');
?>
```

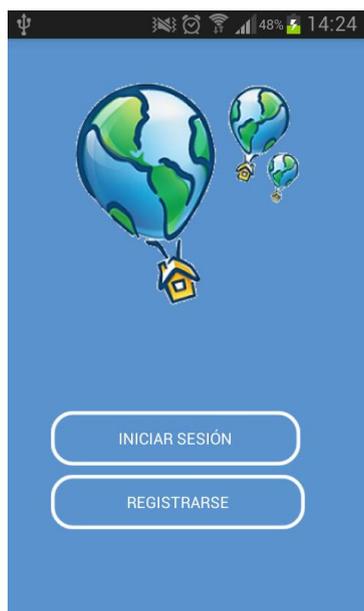
**Figura 28.** Archivo php para la inserción de opiniones.

Lo primero que observamos es la conexión al servidor web, mediante el archivo connect\_database. Luego guardamos en variables los valores recibidos desde la API mediante el POST. Acto seguido, insertamos la opinión en nuestra tabla “tbl\_opiniones” de la base de datos y actualizamos el campo opinada de la tabla “tbl\_visited\_house” para asegurarnos de esta manera que no se pueda opinar más de una vez sobre la misma visita. Para terminar, devolvemos “success” si todo ha ido bien o “fail” en caso contrario.

## 6. PRUEBAS

En esta sección veremos el resultado final de nuestra aplicación mediante capturas de pantalla y explicaciones de cada una.

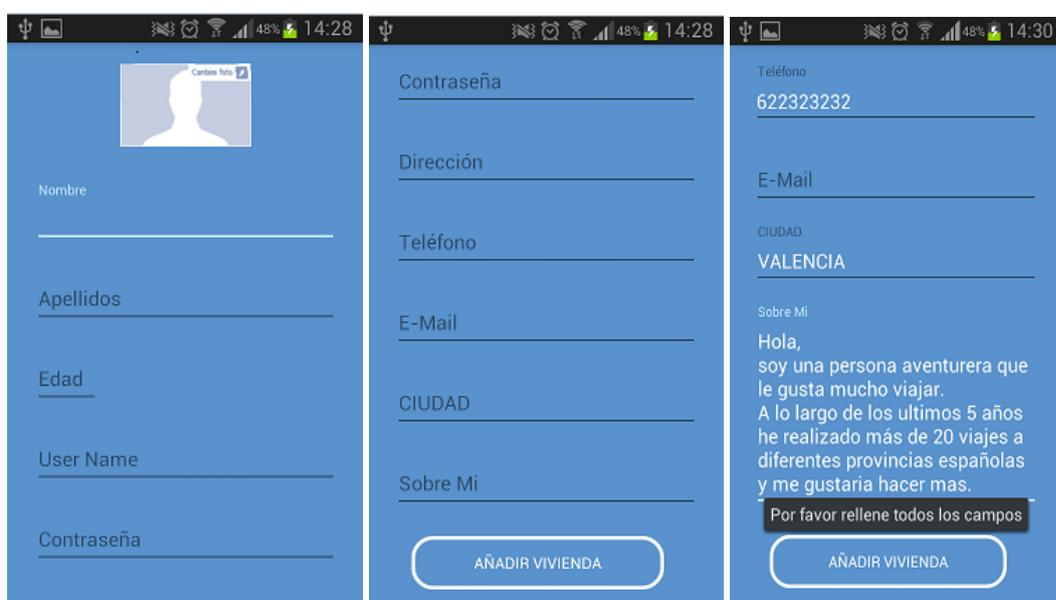
### Apertura de la aplicación



**Figura 29. Pantalla de apertura de la aplicación**

En el momento de abrir la aplicación nos aparece esta pantalla, compuesta por dos botones, el de inicio de sesión y el de registrarse.

### Registrarse



**Figura 30. Pantallas de registro de usuario.**

## Diseño de una aplicación Android para el intercambio de viviendas

Mediante esta pantalla el usuario inserta sus datos personales para poder registrarse en el sistema. Como se puede comprobar, si se deja un campo del formulario sin completarse aparece el mensaje “Por favor rellene todos los campos”. Una vez realizado este paso correctamente, pasamos al registro de una vivienda, que queda de la siguiente manera:

**Figura 31. Pantalla de registro de vivienda.**

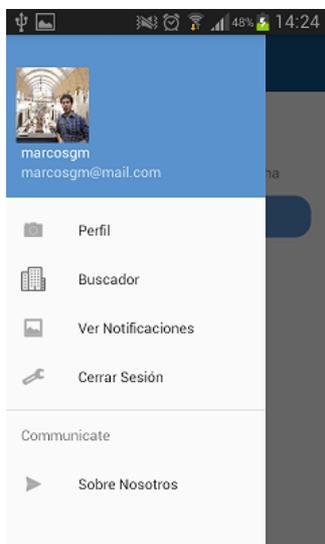
### Inicio de sesión

**Figura 32. Pantalla de Inicio de Sesión.**

Para iniciar sesión, el usuario solo tiene que introducir su nombre de usuario y su contraseña. En caso de que el nombre de usuario o contraseña estén erróneos, aparece el mensaje “Login failed”. Si los datos introducidos son correctos nos dirigimos a la siguiente pantalla.

## Menú de Navegación

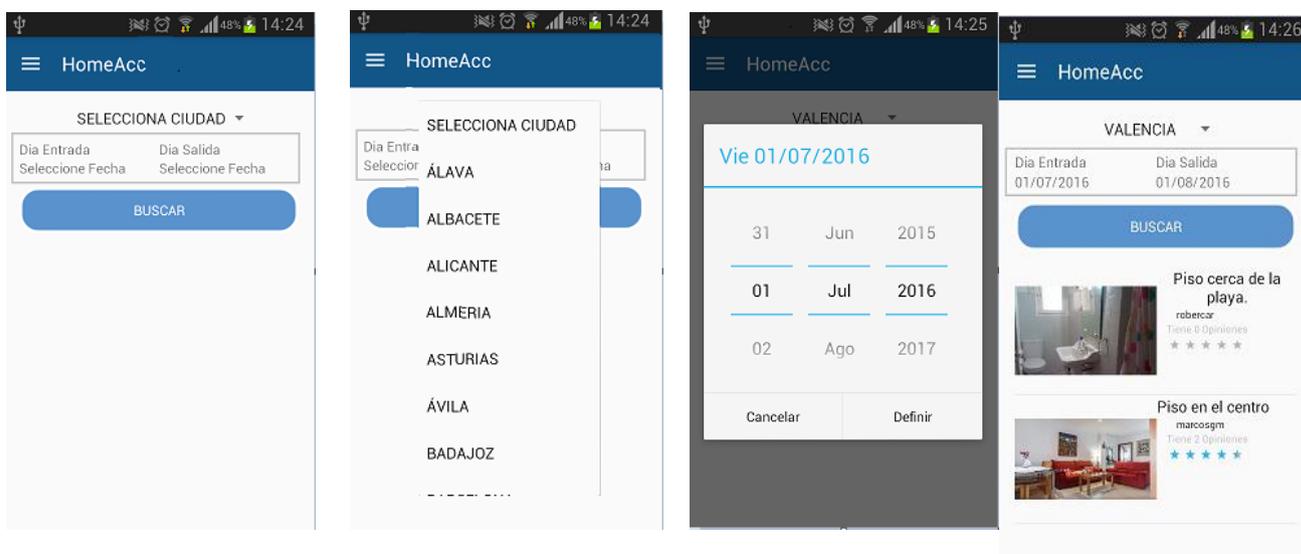
Antes de pasar a describir las demás pantallas, quería aclarar el menú de navegación. Este menú te permite dirigirte a las principales pantallas de la aplicación.



**Figura 33. Menú navegador.**

La primera parte del menú aparece cierta información del usuario como el nombre de usuario, cuenta de correo electrónico y la foto del perfil. En la segunda parte tenemos por un lado, el “Perfil” donde accedemos a las pantallas de Casas Visitadas, Casas Reservadas, Editar Perfil, Editar Anuncio y por otro lado, tenemos “Buscador de Viviendas”, “Ver Notificaciones”y “Cerrar Sesión”. En la tercera parte, tenemos el acceso a “Sobre Nosotros”, donde hay información del administrador por si se desea contactar con él por alguna incidencia.

## Buscador de Viviendas

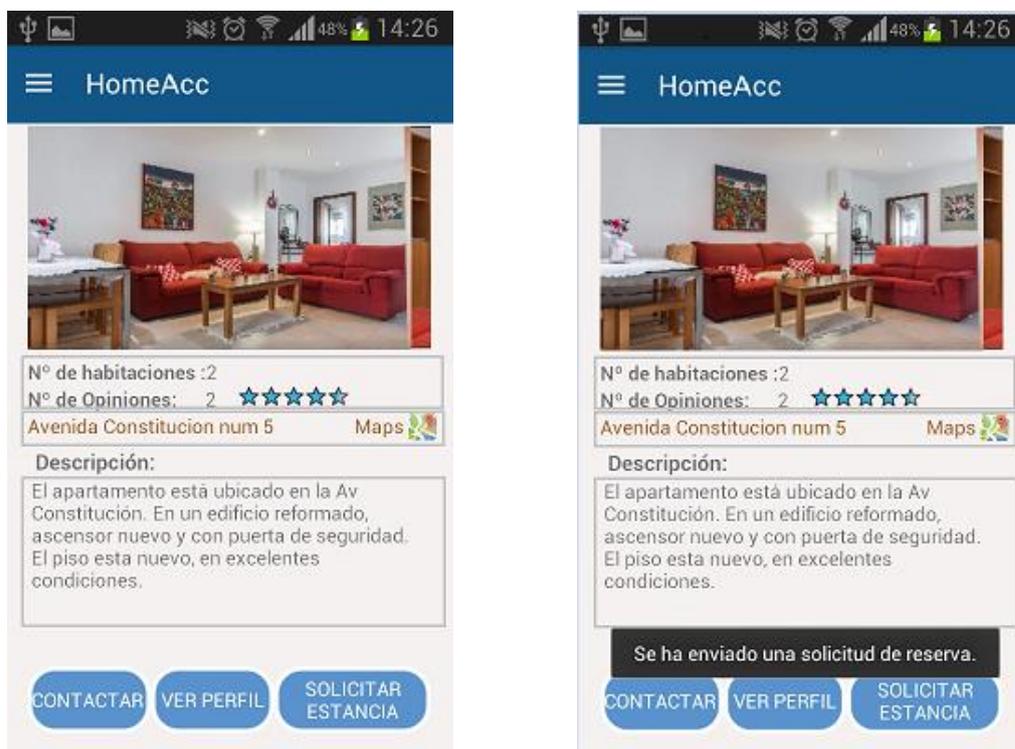


**Figura 34. Pantalla Inicial y de Buscador de Viviendas**

De izquierda a derecha, la primera pantalla es la primera que aparece a la hora de iniciar sesión o cuando se accede a esta pantalla desde otra antes de realizar una búsqueda. Las

dos siguientes pantallas muestran como le aparecen al usuario los listados de las provincias disponibles y un seleccionador de fechas para elegir el día de entrada y día de salida que desea. En la cuarta pantalla, se muestra como aparecen los anuncios tras seleccionar los datos deseados. En nuestro ejemplo, aparecen los anuncios de Valencia, disponible entre el día 01/07/2016 al 01/08/2016.

### Ver Anuncio



**Figura 35 Pantalla de Anuncio.**

Cuando se quiere visualizar un anuncio accedemos a esta pantalla. La primera parte está dedicada a las imágenes que el usuario ha subido de la vivienda. Para visualizar las demás imágenes hay que deslizar el dedo de izquierda a derecha. Acto seguido, aparece la información que más destacan los usuarios como son, el número de habitaciones y el número de viviendas con una descripción gráfica en formato de estrellas, con la media de puntuaciones que se le han realizado al usuario. La siguiente parte muestra la dirección de la vivienda con una opción, "Maps", con la que se puede acceder a Google Maps para saber el posicionamiento de la vivienda. A continuación, está la descripción de la vivienda y para terminar, tenemos tres botones: el de contactar, ver perfil del usuario y solicitar estancia. De hecho, la pantalla de la derecha muestra como aparece el mensaje después de solicitar la estancia.

## Ver Perfil

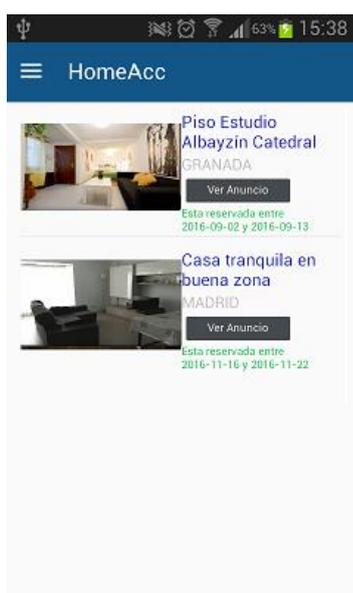


**Figura 36. Pantalla de Perfil de usuario.**

La pantalla de ver perfil tiene el aspecto que se muestra en esta imagen. Está dividida en tres partes. Información de datos personales del usuario, una breve descripción hecha por el usuario sobre sí mismo y, para terminar, las opiniones recibidas por el usuario. Pulsando en una opinión accedemos al perfil del usuario que ha realizado dicha opinión.

## Casas Reservadas

En este apartado aparecen las casas reservadas por el usuario, pero que todavía no se han visitado.



**Figura 37. Pantalla de Reservas de Viviendas**

En esta pantalla encontramos información sobre las diferentes reservas en curso. Por una parte, tenemos el título del anuncio acompañado de la ciudad donde se encuentra. Tenemos la opción de ver el anuncio de la vivienda y, para finalizar, el periodo de fechas en el que la vivienda está reservada.

### Casas Visitadas



**Figura 38. Pantalla de Viviendas Visitadas**



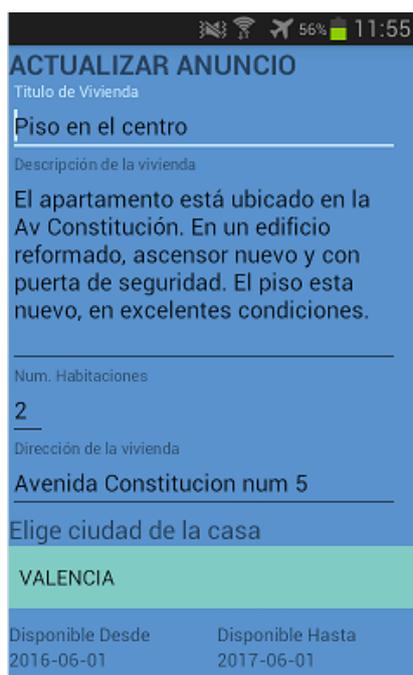
**Figura 39. Pantalla de Dejar Opinión.**

Otra de las opciones del perfil, casas visitadas, tiene una parte igual a la de las reservadas. La parte del título, ciudad y ver anuncio. Por otra parte, tenemos la opción de Dejar Opinión accediendo, de esta manera, a la pantalla de la derecha. En caso en el que ya se ha dejado una opinión previamente se muestra un mensaje indicándolo y desactivando la opción de dejar opinión para esa visita.

## Editar Perfil y Editar Anuncio



**Figura 40. Pantalla de Editar Perfil.**



**Figura 41. Pantalla de Editar Anuncio.**

Con estas dos pantallas podemos actualizar nuestros datos. En la de la izquierda, tenemos la de editar perfil donde podemos actualizar cualquier dato, imagen incluida, de nuestro perfil. En la imagen de la derecha tenemos la pantalla de editar anuncio, con la que actualizamos los datos de nuestro anuncio, el más importante, las fechas de disponibilidad.

## Ver Notificaciones



**Figura 42. Pantalla de Ver Notificaciones.**

## Diseño de una aplicación Android para el intercambio de viviendas

En esta pantalla aparecen las diferentes notificaciones que recibe un usuario. Aquellas de color verde son las que indican que es una reserva aceptada, las de rojo reserva rechazada y las de un gris claro solicitud de reserva. Para eliminar las notificaciones aceptadas y rechazadas se desliza con el dedo de izquierda a derecha. En cuanto a las solicitudes de reserva se accede a ver su detalle pulsando en el botón “Detalles Reserva” desde donde se puede aceptar o rechazar la solicitud.



## 7. CONCLUSIONES

Para terminar el proyecto, voy a exponer las conclusiones del trabajo. Para empezar, voy a explicar las conclusiones técnicas, seguidas de las personales y terminar con ampliaciones del proyecto.

### 7.1 TÉCNICAS

En este proyecto me he desarrollado en varios aspectos técnicos. Para empezar, cabe destacar la parte del Android, utilizando Android Studio, un entorno que nunca había utilizado antes y con el que he ido aprendiendo las bases de la programación para Android, me ha ayudado a desarrollarme en este ambiente y a saber manejar este tipo de tecnología.

Por otra parte, está el tema de las interfaces. Para este aspecto he utilizado XML, con el que no tenía mucha relación hasta empezar este trabajo. Gracias a este lenguaje se me ha facilitado mucho la tarea de las interfaces. También aprendí a manejar este lenguaje cada vez más, entrenando también mi parte de diseño.

En cuanto al PHP, tenía una pequeña noción de su utilización, pero no tan desarrollada como después de la realización del trabajo. En este aspecto aprendí a *parsear* los datos con JSON y manipular datos entre un archivo PHP, un archivo en nuestro servidor web y nuestro código JAVA.

Para terminar este apartado, quiero hablar de MySQL del que ya tenía una idea, pero realizando este trabajo he avanzado mucho y aumentado mis conocimientos en este sistema de gestión de bases de datos.

### 7.2 PERSONALES

Creo que este proyecto me ha ayudado mucho para fomentar la base de la programación de aplicaciones móviles. Dado que es un campo donde se pide cada vez más personal y es muy demandado no viene nada mal.

También he visto como un proyecto de una aplicación móvil se puede hacer eterno, sobre todo para gente sin experiencia como yo, cuando crees que te vas acercando al final siempre salen cosas nuevas que implementar y detalles por corregir, lo que te hace comprender que para hacer un proyecto profesional que aparentemente parece sencillo por detrás hay bastante trabajo y detalles a tener en cuenta, más si no tienes unos requisitos previos establecidos.

### 7.3 POSIBLES AMPLIACIONES DEL PROYECTO

En principio no tengo pensado avanzar en el proyecto a parte de su finalidad principal, la de terminar el Trabajo de Fin de Grado, por lo que el trabajo no tendrá ampliaciones.

Sin embargo, nunca se sabe y puede que algún día decida ampliarlo y subirlo al “Play Store”. Para ello me gustaría tener en consideración los siguientes puntos:

- Un servicio de mensajería diferente al de Google. Esta decisión está basada sobre todo por asuntos de privacidad. Sería tener un servicio de mensajería donde los usuarios pueden contactar entre ellos.
- Revisar la contraseña, número de teléfono y correo electrónico de los usuarios. Para la primera es suficiente con tener que poner la contraseña dos veces a la hora de registrarse. En cuanto a las otras dos, estaría bien comprobarlos enviando un mensaje con un código, que habrá que validar, y un correo electrónico con lo mismo.
- Receptor de Notificaciones del móvil. Cuando un usuario recibe una notificación, ya sea de aceptación, rechazo o simplemente una solicitud de reserva, el usuario le llega una notificación a su teléfono móvil.

## 8. BIBLIOGRAFÍA

### Documentos Web:

Curso de Android

<http://www.sgoliver.net/blog/curso-de-programacion-android/>

Tutorial Android

<http://droideando.blogspot.com.es/2011/02/tutorial-del-primer-proyecto-en-android.html>

Android Developer:

<http://developer.android.com/index.html>

Tutoriales en los siguientes canales de Youtube:

<https://www.youtube.com/channel/UCrjb6hMiNztfhH5mW-RBe4A>

<https://www.youtube.com/channel/UCubLHJktculMCJgUBsct9uQ>

Curso de PHP

<http://php.net/>

Información sobre MySQL:

<https://www.mysql.com/>

Crear Un *Web Service* Para Android Con Mysql, Php y Json:

<http://www.hermosaprogramacion.com/2015/05/crear-un-webservice-para-android-con-mysql-php-y-json/>

Tutorial de Android con MySQL:

<https://www.simplifiedcoding.net/android-json-tutorial-to-get-data-from-mysql-database/>

Usando *layouts* basadas en XML con Android

<http://androideity.com/2011/07/11/usando-layouts-basadas-en-xml-con-android/>

Buena utilización de Android Studio:

<http://android-developers.blogspot.com/2016/04/android-studio-2-0.html>

Aprender a utilizar PHP:

[www.w3schools.com/php](http://www.w3schools.com/php)

Sitio web oficial XML

[www.w3.org/XML/](http://www.w3.org/XML/)

Foro oficial Android:

<http://developer.android.com/community/index.html>

Universidad Politécnica de Valencia, Android: Programación de aplicaciones para móviles:

<http://www.androidcurso.com/>

## Libros

Gargenta, M., Learning Android, O'Reilly Media, 2011.

García de Jalón, J., Rodríguez, J.I., Mingo, I., Imaz A., Brazález A., Larzabal, A., Calleja, J., García, J., Aprenda Java como si estuviera en primero, Universidad de Navarra, 1999.

Hashimi S., Komatineni S., Pro Android, Apress, 2009

Karch, M., Android for Work Productivity for Professionals, Apress, 2010.

## 9.ANEXO

En este anexo mostraré la clase VerAnuncio donde quedará claro lo explicado en la implementación.

### Clase de los anuncios

```

package com.example.tfg.homeacc;

import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.graphics.Rect;
import android.graphics.drawable.ColorDrawable;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.design.widget.NavigationView;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.RatingBar;
import android.widget.TextView;
import android.widget.Toast;

import com.example.tfg.manager.APIManager;
import com.example.tfg.manager.DataManager;
import com.github.snowdream.android.widget.SmartImageView;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.lucasr.twowayview.TwoWayView;

import java.util.ArrayList;

public class VerAnuncio extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener {
    String username;
    Button contact, solicitud, perfil;
    String s;
    TwoWayView piclist;
    ArrayList Imagenes = new ArrayList();
    TextView NumRooms, Descripcion, Direccion, NumOpiniones, verMaps;
    RatingBar mediaEstrellas;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main2);
    //Recuperar variables
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    NavigationView navView = (NavigationView) findViewById(R.id.nav_view);
    View headerView = navView.getHeaderView(0);
    TextView Nav_userName = (TextView)
headerView.findViewById(R.id.nav_header_name);
    TextView Nav_Email = (TextView) headerView.findViewById(R.id.nav_header_email);

    //Barra de MEnu de Navegacion
    Nav_userName.setText(DataManager.userName);
    Nav_Email.setText(DataManager.userEmail);
    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
        this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
    drawer.setDrawerListener(toggle);
    toggle.syncState();

    NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
    navigationView.setNavigationItemSelectedListener(this);
    contact = (Button) findViewById(R.id.buttonSolicitar);
    solicitud = (Button) findViewById(R.id.buttonSolicitud);
    perfil = (Button) findViewById(R.id.buttonPerfil);

    contact.setText("Contactar "); // + DataManager.userDestination);
    solicitud.setText("Solicitar Estancia");
    perfil.setText("Ver Perfil");

    NumRooms = (TextView) findViewById(R.id.textView15);
    Descripcion = (TextView) findViewById(R.id.textView10);
    Direccion=(TextView) findViewById(R.id.textViewDireccion);
    verMaps=(TextView) findViewById(R.id.textView11);
    NumOpiniones=(TextView) findViewById(R.id.textView16);
    mediaEstrellas=(RatingBar) findViewById(R.id.ratingBarOpiniones);

    descargarListado();

    // Acción al pulsar botón contactar
    contact.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(Intent.ACTION_SENDTO); // it's not
ACTION_SEND
            intent.setType("text/plain");
            intent.putExtra(Intent.EXTRA_SUBJECT, "Intercambio de viviendas");
            intent.putExtra(Intent.EXTRA_TEXT, "");
            intent.setData(Uri.parse("mailto:" + DataManager.destinationMail)); //
or just "mailto:" for blank
            intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK); // this will make such
that when user returns to your app, your app is displayed, instead of the email
app.
            startActivity(intent);
        }
    });
}

```

```

// Acción al pulsar botón ver perfil
perfil.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        DataManager.userPerfil=DataManager.userDestination;
        Intent verPerfil = new Intent(getApplicationContext(),
VerPerfil.class);
        startActivity(verPerfil);
    }
});

////// Acción al pulsar botón solicitar reserva
solicitud.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        DataManager.solicitudSaliente = DataManager.userName;
        DataManager.solicitudReceptor = DataManager.userDestination;
        new async_insertarNotif().execute();
    }
});

//////Acción al pulsar Maps
verMaps.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String map = "http://maps.google.co.in/maps?q=" +
DataManager.direccion + ", " + DataManager.ciudadCasa;
        Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(map));
        startActivity(i);
    }
});
}

private void descargarListado() {
    Imágenes.clear();
    new async_obtenerDatosAnuncio().execute();
}

@Override
protected void onPause() {
    super.onPause();
}

@Override
public boolean onNavigationItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id == R.id.nav_camera) {
        Intent i=new Intent(verAnuncio.this,Pofile.class);
        startActivity(i);
        // Handle the camera action
    } else if (id == R.id.nav_gallery) {
        Intent i=new Intent(verAnuncio.this,Notifica.class);
        startActivity(i);
    } else if (id == R.id.nav_home) {

    } else if (id == R.id.nav_manage) {

    } /*else if (id == R.id.nav_share) {

    } */ else if (id == R.id.nav_send) {
        Intent i=new Intent(verAnuncio.this,AboutUs.class);
        startActivity(i);
    }
}

```

```

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    drawer.closeDrawer(GravityCompat.START);
    return true;
}

private class async_obtenerDatosAnuncio extends AsyncTask<String, Void, String> {
    private ProgressDialog dialog;
    JSONArray result;

    @Override
    protected void onPreExecute() {
        dialog = ProgressDialog.show(verAnuncio.this, null, null, true);
        dialog setContentView(R.layout.progress_dlg);
        dialog.getWindow().setBackgroundDrawable(new
ColorDrawable(android.graphics.Color.TRANSPARENT));
        dialog.show();
    }
    @Override
    protected String doInBackground(String... params) {
        try {
            result = APIManager.obtenerDatos();
        }
        catch (Exception ex) {
            ex.printStackTrace();
        }
        return "";
    }
    @Override
    protected void onPostExecute(String res) {

        if (dialog.isShowing()) {
            dialog.dismiss();
        }
        try {
            String numhabs=(result.getJSONObject(0).getString("room_cnt"));
            String descripcion=(result.getJSONObject(0).getString("room_des"));
DataManager.destinationMail=(result.getJSONObject(0).getString("destinationMail"));
DataManager.roomTitle=(result.getJSONObject(0).getString("room_title"));
DataManager.ciudadCasa=(result.getJSONObject(0).getString("ciudadCasa"));
DataManager.imagenDestinacion=(result.getJSONObject(0).getString("home_img0"));
DataManager.direccion=(result.getJSONObject(0).getString("direccion"));
if((result.getJSONObject(0).getString("estrellas")==null) DataManager.mediaEstrella
s=0;
            else
DataManager.mediaEstrellas=Float.parseFloat(result.getJSONObject(0).getString("estr
ellas"));
DataManager.numOpiniones=(result.getJSONObject(0).getString("numOpiniones"));
                NumRooms.setText(numhabs);
                Descripcion.setText(descripcion);
                NumOpiniones.setText(DataManager.numOpiniones);
                Direccion.setText(DataManager.direccion);
                mediaEstrellas.setRating(DataManager.mediaEstrellas);
        } catch (JSONException e) {
            e.printStackTrace();
        }
        for(int i=0;i<5;i++) {

```

```

for(int i=0;i<5;i++) {
    try {
        String obt = "home_img" + i;
        String img=result.getJSONObject(0).getString(obt);
        if (!img.equals("false")){
            Imagenes.add(img);
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}

piclist.setAdapter(new ImagenAdapter(getApplicationContext()));
}

private class ImagenAdapter extends BaseAdapter {
    Context ctx;
    LayoutInflater inflater;
    SmartImageView smartImageView;

    public ImagenAdapter(Context applicationContext) {
        this.ctx=applicationContext;

        inflater=(LayoutInflater) ctx.getSystemService(LAYOUT_INFLATER_SERVICE);

        @Override
        public int getCount() {
            return Imagenes.size();
        }

        @Override
        public Object getItem(int position) {
            return position;
        }

        @Override
        public long getItemId(int position) {
            return position;
        }

        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
            ViewGroup
            viewGroup=(ViewGroup) inflater.inflate(R.layout.activity_main_adhouse,null);

            smartImageView=(SmartImageView) viewGroup.findViewById(R.id.imagen1);
            String
            urlfinal="http://compartecasagratis.com/apis/uploads/"+Imagenes.get(position).toString();

            Rect rect = new
            Rect(smartImageView.getLeft(), smartImageView.getTop(), smartImageView.getRight(), smartImageView.getBottom());
            smartImageView.setImageUrl(urlfinal,rect);

            return viewGroup;
        }
}

private class asvnc insertarNotif extends AsvncTask<String, Void, String> {

```

```

private class async_insertarNotif extends AsyncTask<String, Void, String> {
    private ProgressDialog dialog;
    Boolean result;

    @Override
    protected void onPreExecute() {
        dialog = ProgressDialog.show(verAnuncio.this, null, null, true);
        dialog setContentView(R.layout.progress_dlg );
        dialog.getWindow().setBackgroundDrawable(new
ColorDrawable(android.graphics.Color.TRANSPARENT));
        dialog.show();
    }
    @Override
    protected String doInBackground(String... params) {
        try {
            result = APIManager.insertarNotif();
        }
        catch ( Exception ex ) {
            ex.printStackTrace();
        }
        return "";
    }
    @Override
    protected void onPostExecute(String res) {

        if (dialog.isShowing()) {
            dialog.dismiss();
        }

        if (result) {
            Toast.makeText(verAnuncio.this, "Se ha enviado una solicitud de
reserva.", Toast.LENGTH_SHORT).show();
        }
        else Toast.makeText(verAnuncio.this, "No se ha podido enviar la
solicitud de reserva por algun error.", Toast.LENGTH_SHORT).show();
    }
}
}

```