

Plataforma de valoración y venta de consolas, juegos y accesorios: versión Android



Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:

Juan Romero López

Tutor/es:

Jaume Aragones Ferrero

Septiembre 2016



Universitat d'Alacant
Universidad de Alicante

Contenido

INTRODUCCIÓN.....	6
OBJETIVOS.....	7
METODOLOGÍA	8
TECNOLOGÍAS Y HERRAMIENTAS EMPLEADAS	10
Android Studio	10
Atom	11
phpMyAdmin	11
Trello	11
Balsamiq.....	11
Visio	12
Lucidchart.....	12
Bitbucket	12
SISTEMAS DE INFORMACIÓN.....	13
Esquema E-R	13
Modelo Relacional.....	14
Diccionario de Datos.....	14
Procedimientos.....	15
Mockups.....	16
IMPLEMENTACIÓN	17
Elección de versión mínima.....	17
Implementación del API	17
Pantalla de Inicio de Sesión	20
Pantalla de Registro de Usuario	23
Pantalla principal de la aplicación	25
Pantalla Direcciones de Envío.....	30
Pantalla Editar Perfil Usuario.....	35
Pantalla para escoger avatar usuario	36
Pantalla Detalles Anuncio.....	40
Pantalla de Tasación	43
Pantalla Publicar Anuncio.....	47
Pantalla Buscador.....	49
Otros	51

Creación Icono.....	51
Internacionalización de la Aplicación.....	51
SEGURIDAD.....	52
Contraseña del usuario.....	53
Ataques de inyección SQL.....	53
Denegación de servicio.....	53
PRUEBAS.....	55
Casos de prueba de la pantalla de Inicio de Sesión.....	55
Casos de prueba de la pantalla de Registro de Usuario.....	55
Casos de prueba del Buscador.....	56
Casos de prueba modificar perfil usuario.....	57
Casos de prueba direcciones de envió.....	57
Casos de prueba de reportar precio.....	58
Casos de prueba publicar anuncio.....	58
Casos de prueba reportar anuncio.....	58
Casos de prueba de la pantalla de comentarios.....	59
PROBLEMAS, DIFICULTADES Y SOLUCIONES.....	60
Realizar una petición al api desde la aplicación Android.....	60
Mantener la sesión del usuario (compartir información entre actividades).....	61
Mejorar el rendimiento de la aplicación a la hora de listar las imágenes de los anuncios.....	61
Mostrar caracteres UTF-8 correctamente en las respuestas JSON.....	61
CONCLUSIONES.....	62
Mejoras y Ampliaciones.....	62
BIBLIOGRAFÍAS Y REFERENCIAS.....	64
ANEXOS.....	66
Diccionario de datos.....	66
Accesorio:.....	66
Anuncio:.....	66
Anuncio_img:.....	66
Articulo:.....	66
Articulos_anuncios:.....	67
Atributos_accesorio:.....	67
Atributos_consola:.....	67
Atributos_juego:.....	68

Características_accesorio:	68
Características_consola:	68
características_juego:	68
Comentario:	69
Consola:	69
Direccion_envio:	69
Juego:	69
Localidad:	70
País:	70
Pedido:	70
Plataforma:	70
Provincia:	71
Reportar_anuncio:	71
Reportar_precio:	71
seguimiento_producto:	71
Usuario:	72
Modelo Relacional	73
Procedimientos	77
Mockups	93
Pantalla de Inicio de sesión	93
Pantalla de Registro	93
Pantalla principal	94
Pantalla dirección de envío	94
Perfil de usuario	95
Cambiar avatar	95
Detalles anuncio	96
Tasación	96
Publicar anuncio	97
Buscador	97

INTRODUCCIÓN

El proyecto se basa en la creación de una aplicación Android. Pero, ¿Qué es Android? y ¿Por qué voy a desarrollar una aplicación Android?

Android es la plataforma móvil más importante del mundo, la cuota de mercado referente a los teléfonos inteligentes a nivel global es de un 82,8% en el segundo trimestre del 2015.¹ En cuanto a España, la cuota de mercado es aún mayor, siendo esta de un 93,9% en abril de 2016.²

Estos datos son los que diferencian a Android del resto de las plataformas móviles existentes como iOS o Windows Phone y es que 9/10 smartphones en España usan el sistema operativo Android y 8/10 a nivel global. Y es por eso, que los desarrolladores software de dispositivos Android se ha convertido en uno de los perfiles más demandados en el mercado de trabajo actual.³

Otro de los motivos de desarrollar la aplicación en Android es que los teléfonos de hoy día tienen capacidades de computación similares a los ordenadores, con la diferencia de que el teléfono lo solemos llevar todo el día con nosotros, esto nos permite realizar aplicaciones mucho más cercanas a los usuarios. En cualquier momento del día tiene acceso a nuestras aplicaciones y la facilidad de uso y aprendizaje es mucho mayor que la de utilizar un ordenador, sobre todo en la gente mayor.

¹ IDC - <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>

² Cinco Días -

http://cincodias.com/cincodias/2016/06/15/tecnologia/1465987235_750846.html

³ PC Actual - http://www.pcactual.com/noticias/actualidad/desarrolladores-android-demandados-2_11605

OBJETIVOS

Este proyecto nace con la idea de crear una plataforma que sirva para tasar productos dedicados al mundo de los videojuegos. En mi caso, se trata de analizar, diseñar e implementar una aplicación que permita valorar, comprar y vender consolas, accesorios y videojuegos.

El objetivo es desarrollar una aplicación móvil compatible con dispositivos que utilicen el sistema operativo Android (Android 4.1 o superior). La idea es que “Game2Sell” se convierta en el referente a la hora de realizar la tasación de artículos relacionados con el mundo de los videojuegos. Para ello, la aplicación contará con un sistema de tasación que permitirá a los usuarios conocer cuál es el valor de los artículos que poseen o desean comprar.

Debido a esto, es muy importante que el tasador sea capaz de ir adaptando el valor de los artículos en referencia al mercado actual. Por tanto, el tasador implementado deberá ser capaz de adecuar el precio de los productos en base a la oferta, demanda y número de unidades, porque un producto muy demandado y con pocas unidades en circulación tendrá un mayor precio que un producto con poca demanda y muchas unidades en circulación.

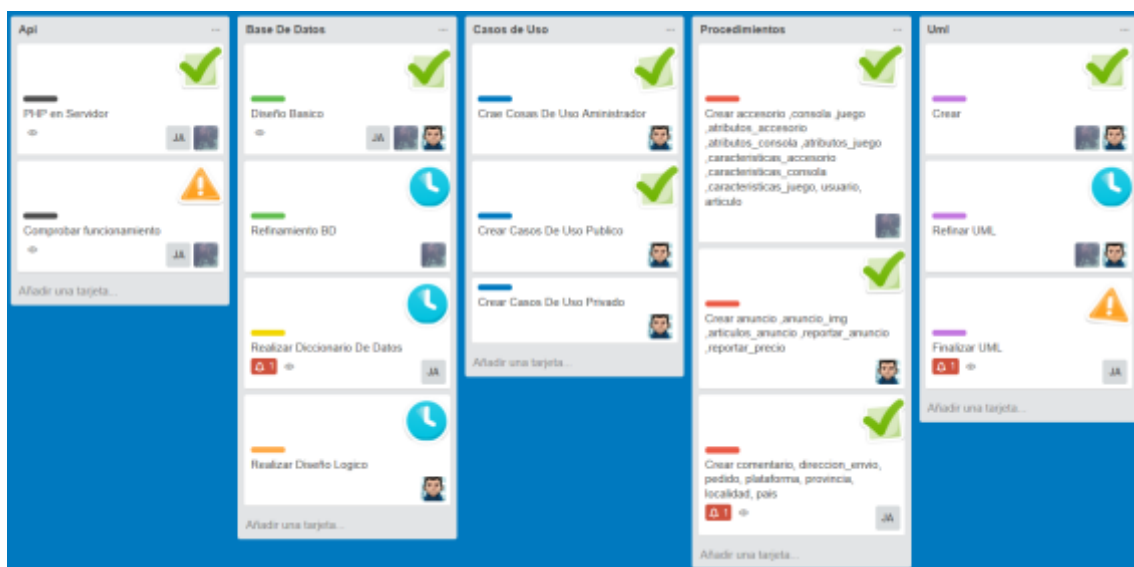
Una forma de mejorar el tasador es dando la posibilidad a los usuarios de indicar de alguna forma su discrepancia con el precio de la tasación y que sean capaces de poder establecer el precio que ellos considere oportuno. La información que obtengamos de ese feedback junto con el número de ventas que se produzcan en la aplicación, nos pueden servir de utilidad para la creación de un algoritmo que permita la modificación automática del precio de tasación en base a estos criterios.

METODOLOGÍA

Este trabajo de fin de grado tiene una parte común con otros TFG en el que compartimos la misma base de datos y el api que ataca a esa base de datos.

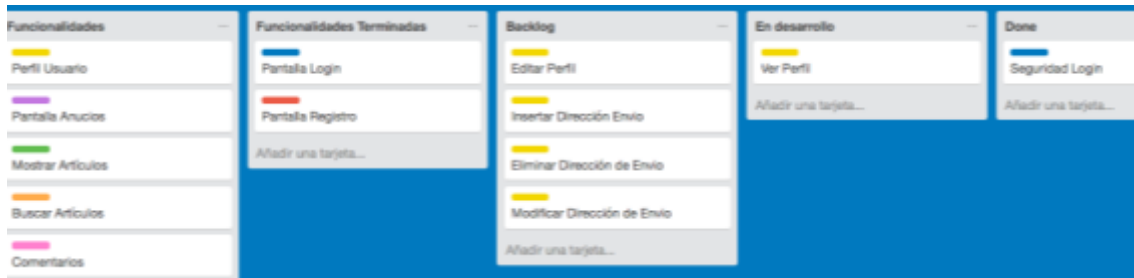
La parte común la hemos decidido implementar siguiendo la metodología de desarrollo SCRUM, entre nosotros hemos establecido el product backlog (las tareas a realizar). Una vez establecido el product backlog hemos realizado varios sprints, durante estos sprints hemos ido realizando “dailys sprints” (reuniones) de unos 15 minutos donde decíamos que hemos hecho en el día anterior, que íbamos a hacer y que problemas nos han surgido. La idea de estas reuniones diarias es comprobar que se estaba realizando el trabajo y no nos atascábamos en ningún punto ya que los problemas surgidos los solucionábamos entre todos.

Para obtener una mayor visibilidad sobre el estado del proyecto de la parte común, hemos creado un panel kanban, donde vemos de una forma sencilla que tareas hemos realizado, las tareas que se encuentran en curso y las que no han comenzado aún.



Como se puede observar en la imagen, las tareas marcadas con el check verde son las finalizadas, las tareas con el reloj azul son las que se encuentran en proceso y las que tienen el símbolo de advertencia son las tareas no comenzadas.

Por otro lado, para realizar la parte individual también he utilizado un tablero kanban pero he planificado las tareas de una forma diferente:



He utilizado un tablero kanban con cinco columnas, la columna de “funcionalidades” donde están todas las funcionales a implementar, la columna de “funcionalidades terminadas” donde están las funcionalidades que ya han finalizado. La columna de “backlog” donde están las funcionalidad que estoy desarrollando divididas en tareas (si procede). En la columna “en desarrollo” están la tarea/funcionalidad que estoy implementado. Y por último en la columna “Done” están las tareas terminada pero que no he probado/subido al repositorio.

A medida que voy implementado código, voy subiendo los cambios al repositorio. Pero tendré en cuenta que los cambios realizados no comprometan los cambios anteriores, ni hagan que el proyecto no se pueda compilar.

El proceso de trabajo es el siguiente:

- Selecciona una tarea de la lista de funcionalidades.
- Descomponer la funcionalidad en tareas y añadir las tareas al backlog.
- Crear una rama en el repositorio.
- Realizar las tareas, a medida que termino una tarea subo dicha tarea a la rama de la funcionalidad.
- Una vez terminadas todas las tareas de la funcionalidad, fusiono (merge) la rama de la nueva funcionalidad con la rama principal.

TECNOLOGÍAS Y HERRAMIENTAS EMPLEADAS

Para llevar a cabo el proyecto he utilizado diversas tecnologías y herramientas seleccionadas para cumplir con ciertos propósitos concretos.

Para la realización de la aplicación para dispositivos Android, he utilizado el lenguaje de programación java. La versión del JDK⁴ utilizado es la 1.7. Para llevar a cabo la aplicación he utilizado el IDE Android Studio.

Por otro lado, para la realización del API⁵. He utilizado el lenguaje de programación PHP, ya que es la opción más sencilla para acceder a bases de datos MySQL. La herramienta utilizada para programar el API en PHP es Atom.

Por último, para realizar la base de datos y los procedimientos almacenados en base de datos he utilizado SQL. La herramienta utilizada para manejar la base de datos y los procedimientos es phpMyAdmin.

Android Studio

A la hora de escoger el IDE⁶ para desarrollar la app, tenía dos opciones Android Studio y Eclipse con el plugin ADT⁷.



Al final me he decantado por Android Studio en su versión 2.X por diversas razones que paso a especificar:

Primero de todo, Android Studio es el IDE oficial para desarrollar aplicaciones para Android. Es un IDE basado en IntelliJ IDEA, creado y mantenido por el equipo de desarrollo de las toolkits de Google.

En segundo lugar, al finalizar el año 2015 desde Google van a dejar de dar soporte al plugin ADT de Eclipse, por tanto, desde la documentación oficial nos instan a migrar los proyectos desde Eclipse con ADT a Android Studio.⁸

⁴ JDK - Java Development Kit

⁵ API – La interfaz de programación de aplicaciones

⁶ IDE - Entorno de Desarrollo Integrado

⁷ ADT - Android Developers Tools

⁸ Soporte Eclipse ADT - <https://developer.android.com/studio/tools/sdk/eclipse-adt.html>

Atom

Atom es un editor de textos open source sencillo y gratuito, que cuenta con una gran cantidad de plugins para programar en cualquier lenguaje de programación. No hay ninguna razón especial por el cual he escogido Atom más allá de que es gratuito y ofrece una gran versatilidad. La versión de Atom utilizada es la 1.9.8.



phpMyAdmin

phpMyAdmin es la herramienta que he utilizado para el manejo de la base de datos MySQL está alojado en el servidor junto con el Api y la página web. La versión de MySQL es la 5.5.50 y la versión de phpMyAdmin utilizada es la 4.2.12 y está alojada sobre un servidor Apache versión 2.2.31.



Trello

Trello es una herramienta que he utilizado a modo de panel Kanban⁹, creando tarjetas asociadas a tareas para darle visibilidad al desarrollo del software, es decir, saber cuánto he hecho, cuanto me falta y que estoy haciendo. Consiguiendo así un mayor control sobre el proceso de desarrollo software y cumplir con el objetivo de entregar el proyecto a tiempo en el plazo acordado.



Balsamiq

He utilizado Balsamiq Mockups en su versión 3.3.12 (versión escritorio).

Balsamiq es la herramienta que he utilizado para realizar los mockups¹⁰ de las pantallas que va a tener mi aplicación. Con el objetivo claro de conseguir un enfoque sobre el diseño básico en etapas más tempranas, es decir, antes de comenzar a diseñar las pantallas en Android Studio ya sé cuál es el diseño que van a tener.



⁹ Kanban – Metodología de trabajo que se centra en la visibilidad del flujo de trabajo

¹⁰ Mockup – boceto básico sobre el diseño que tendrá un aplicación.

Visio

Visio es una herramienta para la realización de diagramas que nos permite simplificar y comunicar información compleja. En mi caso la he utilizado para realizar el diagrama de clases, donde especificamos y describimos los métodos y procesos utilizados en la parte de la base de datos.



Lucidchart

Para la realización de los casos de usos he utilizado la herramienta Lucidchart, dicha herramienta me permite realizar los diferentes casos de uso de manera colaborativa, en la cual muchos participantes pueden ver, editar y borrar al mismo tiempo.



Bitbucket

Es una plataforma para alojar proyectos que utilizan sistemas de control de versiones Git y Mercurial. En mi caso, he utilizado Git. He decidido utilizar este repositorio en detrimento de GitHub porque BitBucket me permite crear repositorios público y privados indistintamente de manera gratuita.

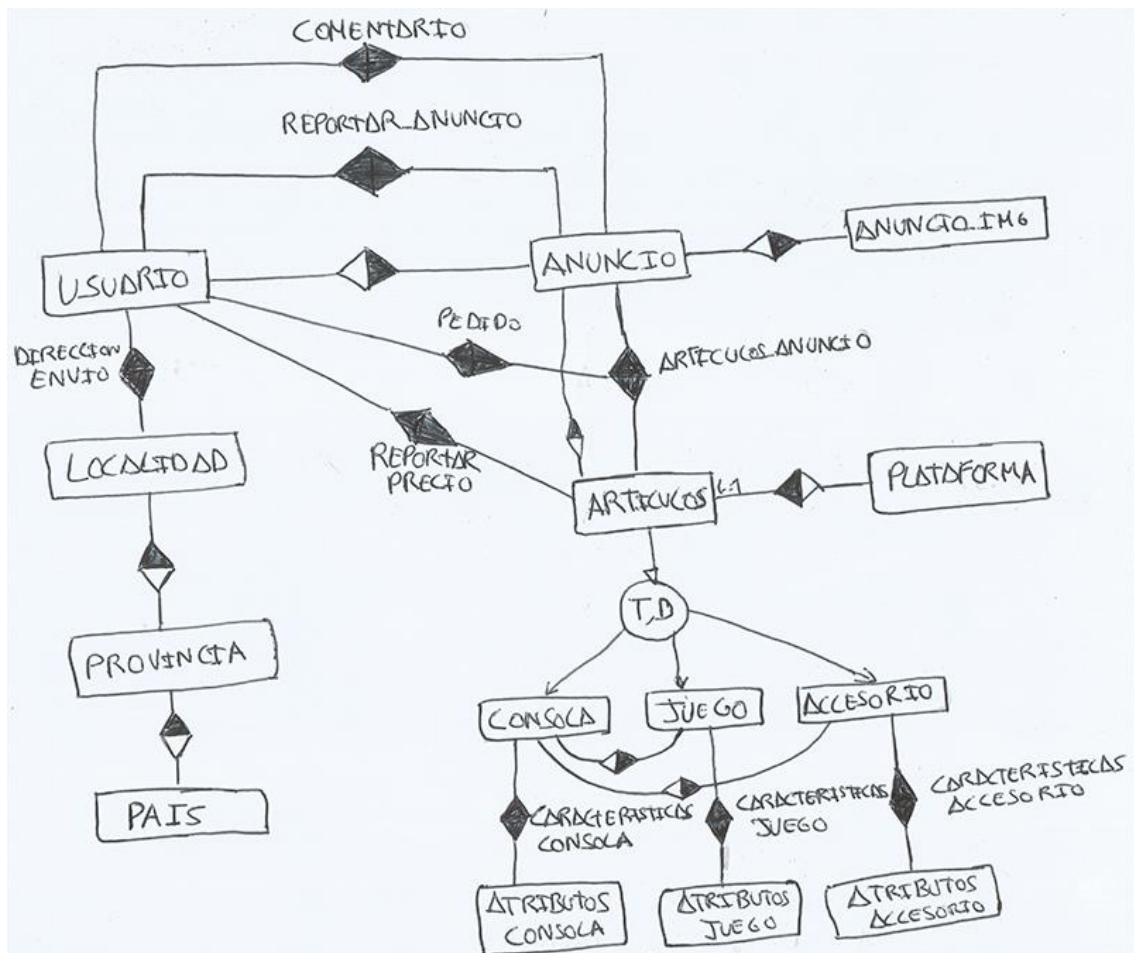


SISTEMAS DE INFORMACIÓN

Para la realización del proyecto he creado una base de datos mysql utilizando phpmyadmin. La principal función de la base de datos es que sirva como mecanismo de persistencia de datos. Para poder almacenar y consultar toda la información referente a la aplicación independientemente de que estemos utilizando la aplicación Android, iOS o la página web.

Esquema E-R

Para visualizar la base de datos de una forma más clara he realizado un esquema Entidad-Relación que representa las entidades de la base de datos así como las relaciones entre las distintas entidades.



Modelo Relacional

El modelo relacional nos aporta de una forma sencilla e intuitiva la representación de la base de datos. El modelo relacional de la base de datos se puede consultar en el [anexo](#).

Diccionario de Datos

Otra forma de representar la información que contienen las tablas es a través del diccionario de datos, por ejemplo el diccionario de datos de la tabla artículo es la siguiente:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		identifica el usuario
nombre	varchar(100)	Sí	<i>NULL</i>	nombre del usuario
apellido	varchar(100)	Sí	<i>NULL</i>	apellidos del usuario
email	varchar(100)	No		email del usuario
user	varchar(100)	No		identificador login del usuario
password	varchar(200)	No		psw del usuario para login
fecha_nacimiento	date	Sí	<i>NULL</i>	fecha de nacimiento del usuario
telefono	int(11)	Sí	<i>NULL</i>	teléfono del usuario
imagen	varchar(100)	Sí	<i>NULL</i>	identifica la imagen del usuario
baja	varchar(1)	No	0	0 indica a un usuario activo y 1 indica el usuario se ha dado de baja

Donde podemos observar el nombre de la columnas, el tipo de datos, si es posible que el valor de los campos sea nulo, el valor por defecto que tendrán los campos si no se establecen y el comentario que sirve como nota aclaratoria.

Para ver el diccionario de datos completo se puede visualizar en el [anexo](#).

Procedimientos

Además de la base de datos, he creado una serie de procedimientos almacenados para mejorar y facilitar la realización de peticiones y consultas a base de datos desde cualquier plataforma (Android, iOS o la página web). Estos procedimientos almacenados serán llamados desde el api.

Por ejemplo, el procedimiento almacenado que se encarga de insertar un usuario en la base de datos es el siguiente:

Sp_InsertaUsuario

Recibe:

- nombre
- apellido
- email
- nombre de usuario
- contraseña
- fecha de nacimiento
- teléfono
- imagen de perfil

Realiza:

- Inserción en la tabla “usuario”.

El procedimiento recibe una serie de parámetros, en caso de que no reciba todos los parámetros o alguno de ellos sean nulos, el valor será el establecido por defecto (se puede observar el diccionario de datos para comprobar esto). El código del procedimiento es el siguiente:

```
BEGIN
  IF EXISTS (SELECT * FROM usuario WHERE user = p_user) then
    SELECT 'El nombre de usuario ya está en uso';
  ELSE
    INSERT INTO usuario (nombre, apellido, email, user, password, fecha_nacimiento, telefono, imagen)
    VALUES (p_nombre, p_apellido, p_email, p_user, p_password, p_fecha_nacimiento, p_telefono, p_imagen);
    SELECT 'Registro realizado con éxito';
  END IF;
END
```

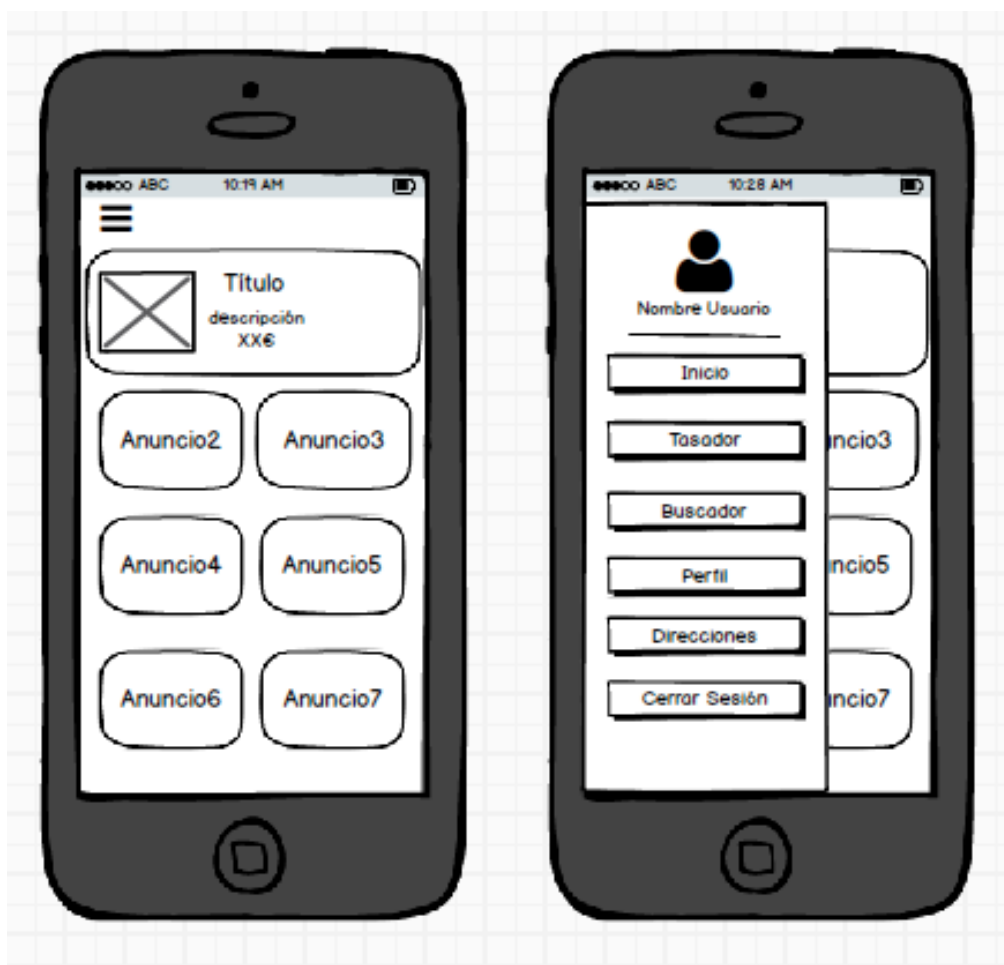
Se tendrá en cuenta que el nombre de usuario no puede estar repetido, si intentáramos registrar un usuario con un nombre ya existente en base de datos, el sistema nos avisaría de que no es posible. Solo se registrará un usuario si su nombre de usuario es único.

Todos los procedimientos se pueden consultar en el [anexo](#).

Mockups

Antes de comenzar a implementar el proyecto, he creado los mockups de la aplicación. Consiguiendo así definir el diseño básico que tendrán las pantallas en fases tempranas del desarrollo (incluso antes de implementar nada de código).

Por ejemplo vamos a ver el mockup de la pantalla principal de la aplicación que es la siguiente:



Todos los mockups creados se pueden visualizar en el [anexo](#).

IMPLEMENTACIÓN

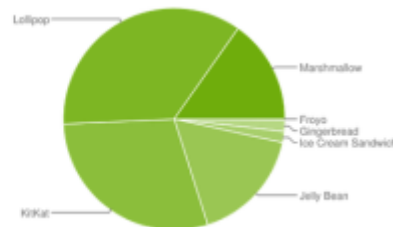
Antes de comenzar a implementar el proyecto, un paso previo es elegir cual será el API mínimo, el cual determinará que versión de Android necesitan los dispositivos para poder ejecutar esta aplicación.

Elección de versión mínima

El API mínimo que he decidido escoger es el API 16, que es la versión Jelly Bean. El motivo de la elección se debe a que según los datos oficiales de Google a fecha 1 de Agosto de 2016, con el API 16 (Android 4.1) y superiores mi aplicación podrá ser ejecutada aproximadamente en el 96,6% de los dispositivos Android.¹¹

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.6%
4.1.x	Jelly Bean	16	6.0%
4.2.x		17	8.3%
4.3		18	2.4%
4.4	KitKat	19	29.2%
5.0	Lollipop	21	14.1%
5.1		22	21.4%
6.0	Marshmallow	23	15.2%

Data collected during a 7-day period ending on August 1, 2016.
Any versions with less than 0.1% distribution are not shown.



Implementación del API

Hemos desarrollado un API en php con el objetivo de simplificar el acceso a los procedimientos almacenados en base de datos desde las diferentes aplicaciones (Android, IOS, Web). Cada servicio del api ataca a un procedimiento almacenado distinto, el resultado de la ejecución se devolverá en JSON.

Para poder tratar con el api en mi aplicación Android he creado una clase JSONParser con el objetivo de que al realizar una petición al api pueda parsear la respuesta, para poder tratar con el resultado. Todas las respuestas tienen dos partes, el resultado y el mensaje. El resultado sirve para saber si la petición ha tenido éxito (indicado con un 1)

¹¹ Datos API Google - <https://developer.android.com/about/dashboards/index.html>

o para indicar que no devuelve ningún resultado o faltan parámetros en la petición (el valor del resultado será un 0).

Por tanto, cuando el resultado es igual a uno, lo que hago es recorrer el mensaje de respuesta y convertir el resultado en un objeto de una clase java. Ejemplo: cuando intento iniciar sesión en la aplicación si las credenciales son válidas, el resultado será 1 y el mensaje contendrá los datos del usuario que inicia sesión, que los utilizaré para crear un objeto del tipo usuario.

El fichero principal del api es “conexión_bd.php” que implementa una clase de nombre “Base de Datos” que contiene varias funciones:

- La función conectar: se encarga de realizar la conexión a una base de datos con los credenciales que estén establecidas en el fichero “login_bd.php”:

```
public function conectar(){
    $this->conexion = mysql_connect(BD_HOST, BD_USER, BD_PASS);

    if ($this->conexion == 0) DIE("Lo sentimos, no se ha podido conectar con MySQL: " . mysql_error());
    $this->db = mysql_select_db(BD_NAME, $this->conexion);

    if ($this->db == 0) DIE("Lo sentimos, no se ha podido conectar con la base datos: " . BD_NAME);

    mysql_set_charset('utf8',$this->conexion);

    return true;
}
```

- La función desconectar: su único objetivo es cerrar la conexión de la base de datos:

```
public function desconectar(){
    if ($this->conectar->conexion) {
        mysql_close($this->$conexion);
    }
}
```

- La función “faltanParametros”, se llamará cuando intentamos realizar una petición y el número de parámetros son incorrectos:

```
public function faltanParametros(){
    $response["success"]=0;
    $response["message"]= "Faltan parámetros";
    echo preg_replace("/\\\\\\\\u{a-f0-9}{4}/e", "iconv('UCS-4LE','UTF-8',pack('V', hexdec('U$1')))", json_encode($response));
}
```

- Por último, la función que se encarga de ejecutar el procedimientos almacenado en base de datos:

```
public function ejecutarProcedimiento($procedimiento){
    //echo($procedimiento);
    $resultset = mysql_query('CALL '.$procedimiento);
    $response = array();

    if (mysql_num_rows($resultset) > 0){
        $response["success"]=1;
        while($r = mysql_fetch_assoc($resultset)){
            $response["message"][] = $r;
        }
    }else{
        $response["success"]=0;
        $response["message"] = "Ningún resultado";
    }

    echo preg_replace("/\\\\\\\\u{a-f0-9}{4}/e", "iconv('UCS-4LE','UTF-8',pack('V', hexdec('U$1')))", json_encode($response));
}
```

El resto de ficheros tienen un comportamiento similar, solo varía el procedimiento al que llaman y los parámetros de los mismo, por ejemplo vamos a ver cómo sería el procedimiento de publicar un comentario:

```
$db = new BaseDatos();

if($db->conectar()){
    if(isset($_GET['p_titulo']) && isset($_GET['p_descripcion']) && isset($_GET['p_id_usuario']) && isset($_GET['p_id_anuncio'])){
        $titulo = $_GET['p_titulo'];
        $descripcion = $_GET['p_descripcion'];
        $idUser = $_GET['p_id_usuario'];
        $idAnuncio = $_GET['p_id_anuncio'];

        $db->ejecutarProcedimiento("Sp_InsertaComentario('".$titulo."', '".$descripcion."', '".$idUser."', '".$idAnuncio."')");
    }else{
        $db->faltanParametros();
    }
    $db->desconectar();
}
```

Como se puede observar, la lógica es la misma para todos:

Primero intenta realizar la conexión a la base de datos, si es posible realizar la conexión a la base de datos, se comprobarán que se está llamando con los parámetros adecuados en caso contrario, se enviará la respuesta de que faltan parámetros y se cerrará la conexión en base de datos. Si por el contrario son correctos los parámetros, se llamará al procedimiento almacenado en base de datos para que intente realizar la acción acordada.

Pantalla de Inicio de Sesión

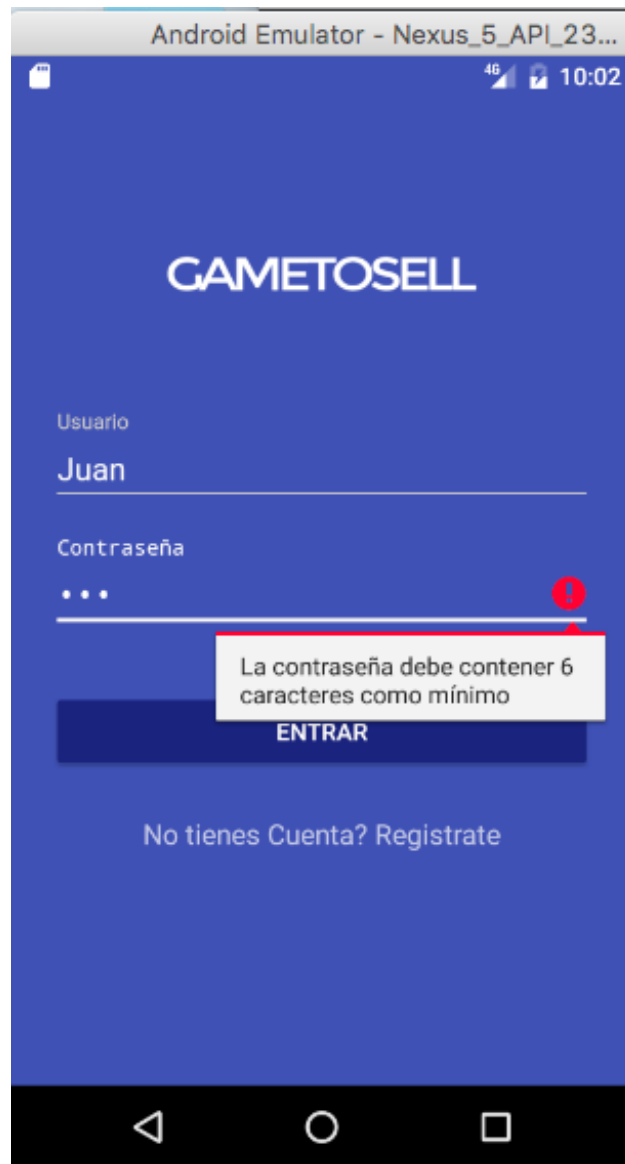
La pantalla de inicio de sesión sirve para que los usuarios puedan acceder con sus credenciales a la aplicación.

La pantalla tendrá varias etiquetas de texto, un botón para realizar la petición y dos campos para introducir texto en el cuál se deberán introducir las credenciales de acceso del usuario, esto es, nombre de usuario y contraseña.



Una vez diseñada la pantalla, debo implementar el comportamiento de dicha pantalla.

Por una parte existe una función de validación de campos para el nombre de usuario y la contraseña. El nombre de usuario debe contener al menos cuatro caracteres y la contraseña un mínimo de seis. Si los campos no cumplen el patrón establecido se mostrará un aviso.



Si pulsamos sobre el texto de “¿No tienes cuenta? Regístrate” se abrirá la pantalla de registro, para que el usuario pueda obtener credenciales de acceso para utilizar la app.

Por último, cuando pulsamos en el botón entrar y las credenciales son correctas, se creará la sesión del usuario y se abrirá la pantalla principal con el usuario conectado.

Cuando el login es correcto creamos la sesión del usuario, al hacer login recibimos todos los datos del usuario en formato json. Con esos datos creamos un objeto de la clase usuario y enviamos ese objeto a la pantalla principal de la siguiente forma:

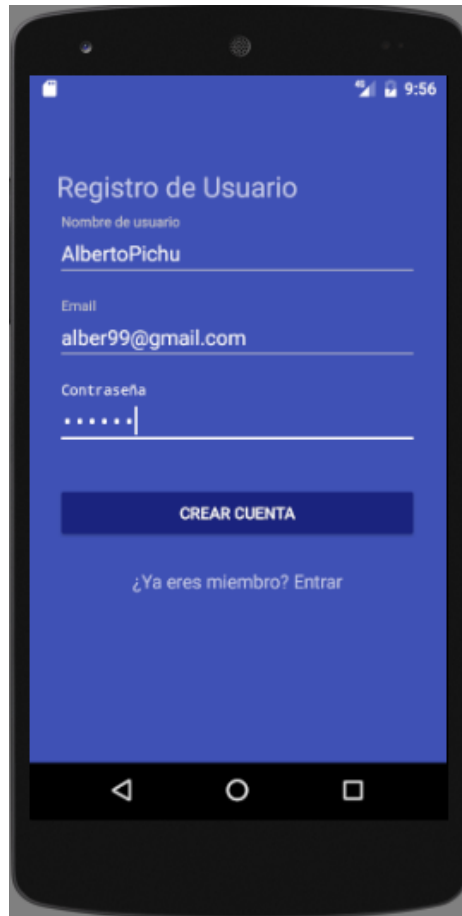
```
Intent i = new Intent(getApplicationContext(), MainActivity.class);  
i.putExtra("sesionUsuario", usuario);  
startActivity(i);
```

Para poder pasar un objeto de una actividad a otra actividad la clase usuario debe implementar la interface *Serializable*. Hacemos esto para poder convertir el objeto en bytes y poder pasarlo a la otra actividad. Para recuperar el objeto usuario desde la otra actividad (actividad principal) lo hacemos de la siguiente forma:

```
Intent i = getIntent();  
userSession = (Usuario)i.getSerializableExtra("sesionUsuario");
```

Pantalla de Registro de Usuario

Esta pantalla permite a los usuarios crear una nueva cuenta. Como se puede observar, para simplificar al máximo el registro en la aplicación, solo se pedirán los tres campos obligatorios: el nombre de usuario, la contraseña y el correo electrónico. Más adelante, en la pantalla de perfil de usuario, se podrá completar, si se desea, el resto de campos. El resultado de la pantalla es el siguiente:



Al igual que la anterior pantalla, al pulsar en el botón de crear cuenta si los campos no son válidos se mostrará un mensaje al usuario para que pueda solucionarlo y vea el motivo por el cual no son correctos los datos introducidos.

Estos son:

- El nombre de usuario al menos debe contar con cuatro caracteres.
- La contraseña deberá tener 6 caracteres como mínimo.

- El correo electrónico deberá ser un correo válido, esto es, que cumpla el patrón establecido en *android.util.Patterns.EMAIL_ADDRESS*.¹²

Si al intentar registrarnos se produce algún fallo se notificará, al igual que cuando intentamos registrar un usuario con un nombre de usuario que previamente ya ha sido registrado por otra persona.

En cambio, cuando el registro se completa de forma satisfactoria se notificará y se redireccionará a la página de inicio de sesión. Para que el usuario puede conectarse a la aplicación y empezar a utilizarla.

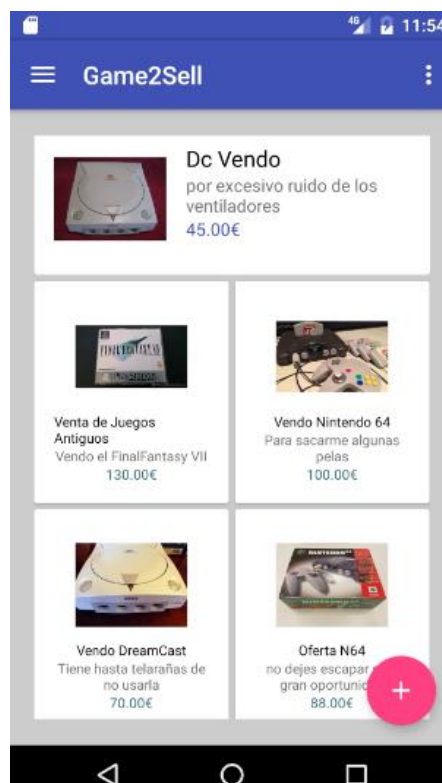
¹² Patrones - <https://developer.android.com/reference/android/util/Patterns.html>

Pantalla principal de la aplicación

En la pantalla principal se cargarán los anuncios existentes, la distribución será la siguiente: un anuncio destacado que ocupará la primera fila de forma completa y el resto, se distribuirán dos anuncios por cada fila.

En un primer momento los anuncios se mostrarán por orden de creación de manera descendente, es decir, primero los anuncios más recientes. Pero la idea en el futuro es modificar el orden de cómo se muestran los anuncios.

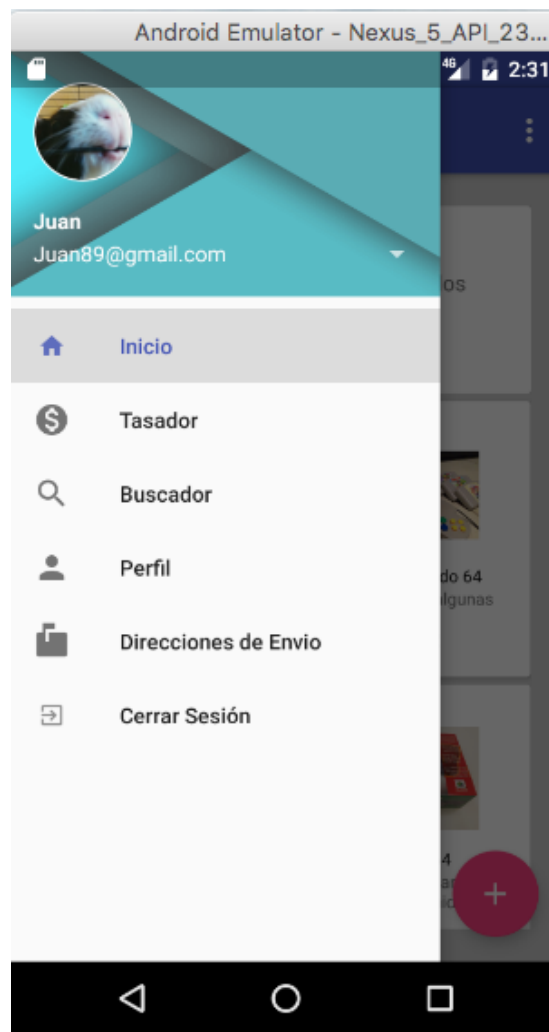
La idea es, mostrar los anuncios más relevantes para los usuarios esto se conseguirá realizando un “seguimiento” del usuario. Es decir, crear un log¹³ de las búsquedas que realiza el usuario (para saber cuáles son sus preferencias), al igual que realizar un seguimiento de los artículos que ha comprado el usuario. Realizando este seguimiento lo que vamos a conseguir es ofrecer al usuario artículos basados en las preferencias y necesidades del usuario en concreto. Lograr que la aplicación se ajuste a las necesidades y gustos del usuario sería un paso importante para mejorar la acogida de la aplicación.



¹³ Almacenará únicamente el texto insertado en la búsqueda y el id del usuario que lo realizó. (Por ejemplo: en una tabla de la base de datos).

Además de mostrar los anuncios otra parte muy importante de la pantalla es el menú principal de la aplicación, al que podemos acceder a él desde la pantalla principal pinchando en el icono de las tres líneas (Burger icon). Desde dicho menú se podrá acceder a todas las funcionalidades que ofrece la aplicación.

El resultado final de la pantalla es el siguiente:



Por tanto, como se puede apreciar al dibujar los anuncios en la pantalla principal se muestra:

- La fotografía del producto, si el anuncio ha sido creado sin la imagen se mostrará una imagen por defecto que indique que el anuncio no tiene imagen disponible para mostrar.
- El título del anuncio.
- La descripción del anuncio.

- El precio del anuncio.

Como he indicado anteriormente el orden en el que aparecen los anuncios actualmente, es en base al orden de creación (los más nuevos primeros).

Para implementar la lista de forma que aparezca un elemento de cabecera destacado (el primero) he utilizado una librería que permite añadir un GridView¹⁴ con cabecera (GridViewWithHeaderAndFooter¹⁵)

Una de las cuestiones fundamentales en lo referente al rendimiento de la aplicación es como cargar las imágenes de los anuncios. Para que la aplicación no se ralentice he utilizado la librería Picasso¹⁶: Esto supone que a la hora de dibujar la imagen en pantalla en el adaptador hago uso de Picasso de tal forma:

```
ImageView imagen = (ImageView) view.findViewById(R.id.imagen);
Picasso.with(mContext)
    .load("http://[REDACTED]image/anuncios/"+item.getId_anuncio()+".png")
    .placeholder(R.drawable.imgdefault)
    .error(R.drawable.imgdefault)
    .into(imagen);
```

Como se puede observar en la imagen, Picasso nos permite establecer una imagen de placeholder (lo que nos permitirá mejorar el rendimiento, porque al hacer scroll no intentará cargar todas las imagen si el scroll está en movimiento) y una imagen de error en caso de que no encuentra la imagen del anuncio que está alojada en el servidor. Para cargar la imagen haremos uso del código identificador del anuncio, pues la imagen del anuncio tendrá como nombre el código identificador del anuncio.

Además de mostrar los anuncios como se puede observar, hay un botón flotante¹⁷ que siempre permanecerá en esa posición independiente de si realizamos scroll o no. Dicho botón nos permite crear un nuevo anuncio (pero eso lo veremos más adelante).

¹⁴ GridView – <https://developer.android.com/guide/topics/ui/layout/gridview.html>

¹⁵ <https://github.com/liaohuqiu/android-GridViewWithHeaderAndFooter>

¹⁶ Picasso - <http://square.github.io/picasso/>

¹⁷ Floatin Action Button -

<https://developer.android.com/reference/android/support/design/widget/FloatingActionButton.html?hl=es>

Por otro parte, tenemos el menú que podemos decir que se divide en dos: la cabecera del menú y los elementos del menú.

En la cabecera el menú se cargan los datos del usuario:

- La imagen de perfil de usuario, que en caso de que no tenga una establecida se mostrará una por defecto. Al pinchar en la imagen de perfil tendremos la posibilidad de cambiar dicha imagen (esta pantalla se explicará más adelante).
- El nombre de usuario.
- El correo electrónico del usuario.

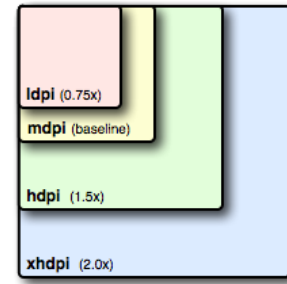
En los elementos del menú, tenemos los siguiente:

- Inicio. Pantalla principal
- Tasador. Pantalla para tasar los artículos sobre los que queremos vender, o simplemente saber cuál es el precio de la tasación.
- Buscador. Desde esta pantalla accedemos a un buscador que nos permitirá filtrar las condiciones de búsqueda.
- Perfil. Esta pantalla posibilita al usuario modificar sus datos.
- Direcciones de Envío. Pantalla que permite añadir, modificar y eliminar las direcciones de envío del usuario.
- Cerrar Sesión. Para cerrar la sesión del usuario.

Para los iconos de los elementos del menú he utilizado los recursos que proporciona google (Material Icons¹⁸), donde nos permite seleccionar el color de los iconos (blanco o negro) y el tamaño. A raíz de esa selección se nos proporciona un fichero comprimido en zip con las imágenes preparadas para ser insertada en el proyecto para adaptarse a todos las densidades de pantalla para Android:

¹⁸ Material Icons – Google Desing - <https://design.google.com/icons>

- mdpi: 160dpi. (res/drawable-mdpi/graphic.png)
- hdpi: 240dpi. (res/drawable-hdpi/graphic.png)
- xhdpi: 320dpi. (res/drawable-xhdpi/graphic.png)
- xxhdpi: 480dpi. (res/drawable-xxhdpi/graphic.png)
- xxxhdpi: 640dpi. (res/drawable-xxxhdpi/graphic.png)



Cuando hablamos de densidad de pantalla (dpi) nos referimos a la cantidad de pixeles que caben un determinado espacio físico (y se miden en puntos por pulgadas).

Es importante tener las imágenes en varios tamaños para conseguir que el diseño se adecue al tamaño de la pantalla¹⁹, si solo utilizamos una única imagen obtendríamos un diseño pobre al visualizarlo en pantallas más grandes o pequeñas.

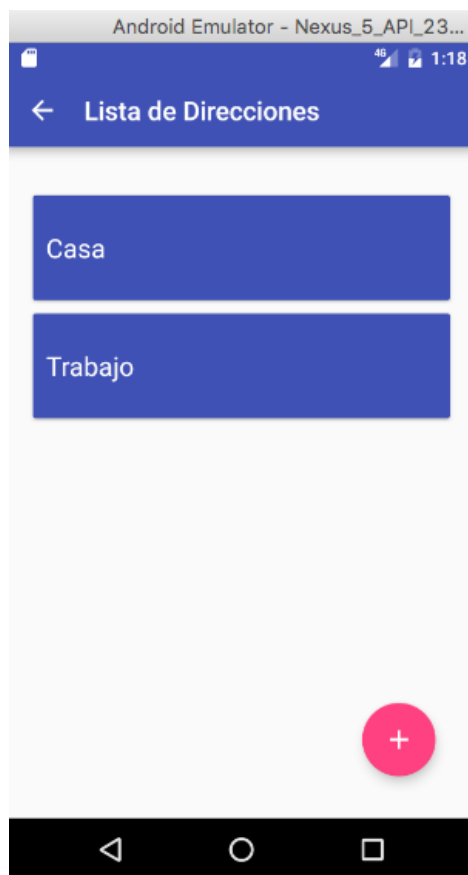
¹⁹ Supporting Multiple Screens - https://developer.android.com/guide/practices/screens_support.html

Pantalla Direcciones de Envío

Cuando accedemos a la pantalla de direcciones de envío desde el menú principal de la actividad lo que nos mostrará será una lista de todas las direcciones de envío que tiene dicho usuario.

Desde la pantalla de la lista de direcciones del usuario cabe la posibilidad de añadir una nueva dirección si pulsamos en el botón flotante con el icono más. Además cuando pinchamos sobre un elemento de la lista (una dirección) nos da la posibilidad de modificar y eliminar los datos de la misma.

El resultado de la pantalla es el siguiente:



La lista está implementada con un RecyclerView²⁰, que es un componente que ha sido añadido con la llegada de Android 5.0. con la intención de mejorar los ListView y los GridViews.

²⁰<https://developer.android.com/reference/android/support/v7/widget/RecyclerView.html>

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/cardListAdress"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:clipToPadding="false"
    android:paddingBottom="16dp"
    android:paddingTop="16dp"
    android:scrollbars="vertical" />
```

Cada elemento de la lista es un CardView²¹ (nos permite mostrar los elementos de la lista en forma de “tarjetas”) que únicamente tiene un texto que representa el título que describe la dirección creada por el usuario.

Para dibujar e interactuar con los elementos de la listas (independientemente del tipo de lista que sea) necesitamos un adaptador.

Por tanto al abrir la pantalla lo primero que debemos hacer es recoger los datos, es decir, todas las direcciones del usuario que ha iniciado sesión. Por tanto hacemos una petición al api pasándole como parámetro el código identificador del usuario:

```
HashMap<String, String> params = new HashMap<>();
params.put("p_id_user", args[0]);

Log.d("request", "starting");

JSONObject json = jsonParser.makeHttpRequest(LOGIN_URL, "GET", params);

if (json != null) {
    Log.d("JSON result", json.toString());
    return json;
}
```

Si el json devuelto es correcto y tiene direcciones en base de datos, recorreremos el usuario creando objetos de la clase direcciones y añadiéndolos a un ArrayList de direcciones. Dicho ArrayList será lo que se le pasará al adaptador:

```
private void cargarDireccion(List items) {
    adapter = new AddressAdapter(items, this, userSession);
    recycler.setAdapter(adapter);
}
```

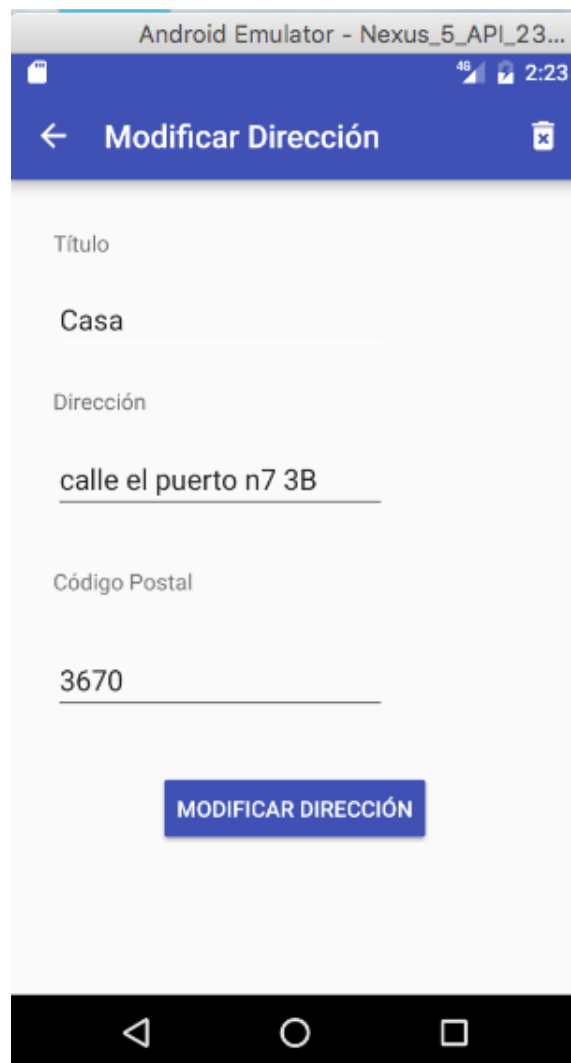
²¹<https://developer.android.com/training/material/lists-cards.html?hl=es>

Lo más importante del adaptador es que al pulsar sobre un elemento se debe abrir la pantalla para modificar la dirección pulsada:

```
@Override
public void onBindViewHolder(AddressViewHolder viewHolder, final int i) {
    viewHolder.titulo.setText(items.get(i).getTitulo());

    viewHolder.titulo.setOnClickListener((v) -> {
        Intent intent = new Intent(context, ModifyAddressActivity.class);
        intent.putExtra("sesionUsuario", sessionUser);
        intent.putExtra("selDireccion", items.get(i));
        context.startActivity(intent);
    });
}
```

Por tanto, le pasaremos a la pantalla de modificación de la dirección de envío el objeto seleccionado (la variable “i” nos indica la posición del elemento de la lista pulsado). La pantalla quedará de la siguiente forma al pulsar sobre un elemento:

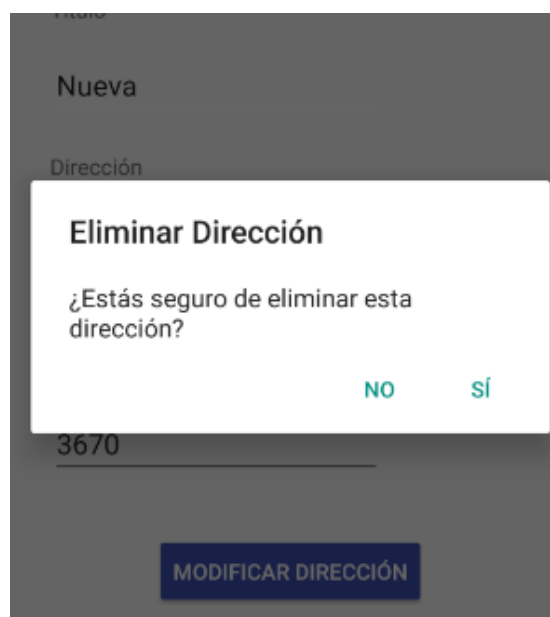


La pantalla carga los datos de la dirección:

- El título que servirá para identificar la dirección de manera sencilla.
- La dirección de envío.
- El código postal de la dirección (que implica la localidad y la provincia).

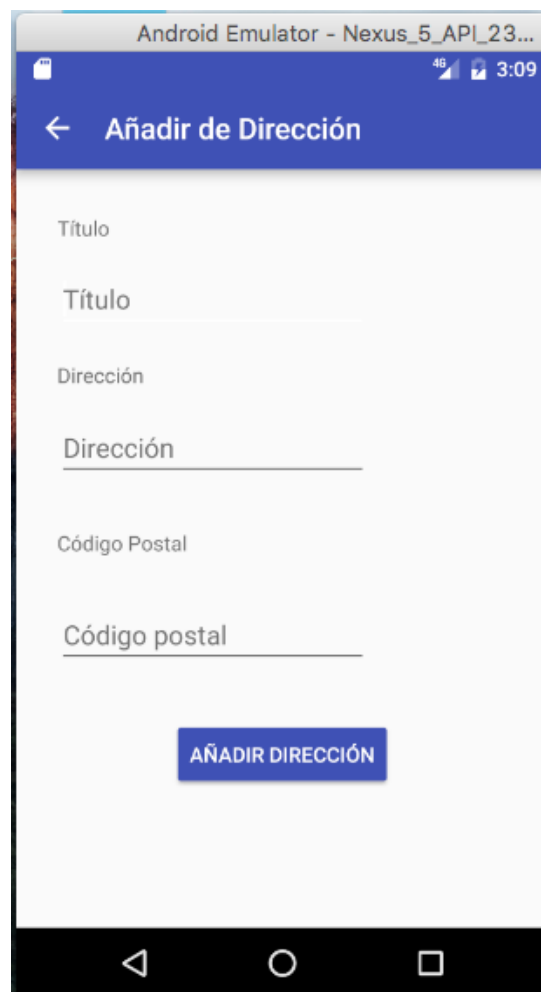
Tendremos la posibilidad de modificar los datos y una vez pulsamos el botón de modificar dirección, se realizarán los cambios introducidos en los campos, eso sí, igual que la inserción se comprobará que ese usuario no tiene otra dirección con el mismo título y que el código postal es válido. Para ello se realiza una petición al api (DameLocalidadPorCP) en el que se pasa como parámetro el código postal introducido y se comprueba en base de datos si existe ese código postal haciendo uso del procedimiento almacenado pertinente (Sp_RecuperarLocalidad) que realiza una búsqueda en la tabla localidad.

Aunque en dicha pantalla no solo podemos modificar los datos, sino que también podemos eliminar la dirección de envío. Para ello, lo único que debemos hacer es pulsar en el icono de la papelera. Al intentar eliminar una dirección, nos preguntará si estamos seguros de querer eliminar la dirección, en caso de que confirmemos se eliminará definitivamente la dirección de envío.



Tanto al modificar como el borrar, si se ha podido realizar correctamente la operación se redireccionará a la pantalla que lista las direcciones de envío donde podremos ver los cambios producidos por dichas acciones.

Además de esto en la pantalla que lista las direcciones si pulsamos en el botón flotante accederemos a la pantalla de inserción de nueva dirección, que es la siguiente:

The image shows a screenshot of an Android emulator window titled "Android Emulator - Nexus_5_API_23...". The emulator displays a mobile app interface with a blue header bar containing a back arrow and the text "Añadir de Dirección". Below the header, there are three text input fields, each with a label above it: "Título", "Dirección", and "Código Postal". At the bottom of the form is a blue button with the text "AÑADIR DIRECCIÓN". The emulator's status bar at the top shows a 4G signal, a battery icon, and the time 3:09. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps buttons.

Esta pantalla tiene el mismo funcionamiento que la pantalla de modificación, salvo que esta pantalla obviamente no precarga ningún tipo de información y la función que realiza es la de insertar una nueva dirección (y no modificarla). Al pulsar en el botón de añadir, realizará las mismas comprobaciones que la modificación: que no esté repetido el título y que el código postal exista. Si todo va bien, se redireccionará a la pantalla que lista las direcciones, con la nueva dirección insertada.

Pantalla Editar Perfil Usuario

Dicha pantalla precargará los datos que tenga el usuario almacenados en base de datos y permitirá modificar dichos campos. Al igual que la pantalla de registro de usuarios, cuando intentamos ejecutar la acción pulsando el botón se comprobará que los campos son correctos. El resultado final de la pantalla es el siguiente:



Android Emulator - Nexus_5_API_23...

← Perfil de Usuario

Nombre
Juan

Apellidos
Romero

Email
Juan89@gmail.com

Teléfono
662062062

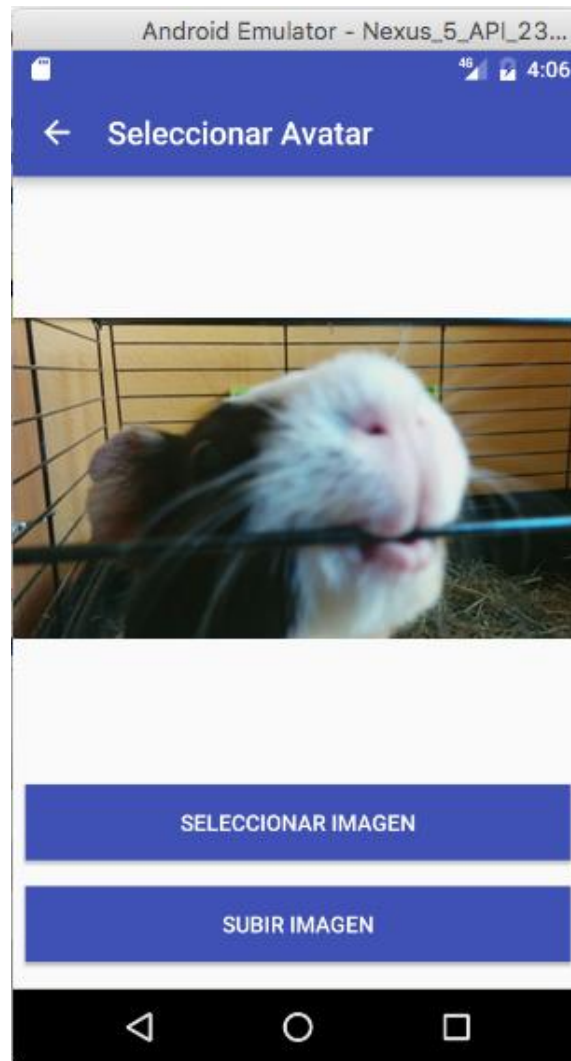
Fecha de nacimiento
1989-05-13

GUARDAR DATOS

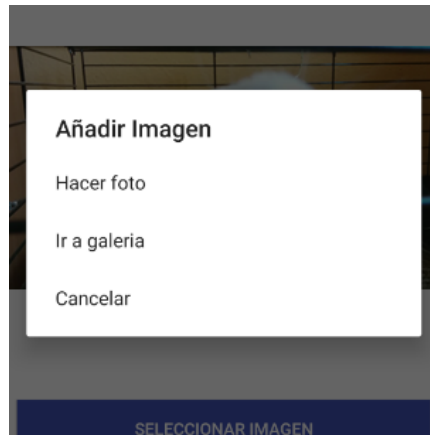
Pantalla para escoger avatar usuario

Esta pantalla nos permite seleccionar/cambiar la imagen de perfil de usuario, para acceder a esta pantalla debemos pulsar en el avatar del usuario que aparece en la cabecera del menú.

Por tanto, la pantalla tendrá dos botones uno para seleccionar la imagen y otra botón para subir la imagen al servidor.



Como se puede observar precarga la imagen actual, además si pulsamos en el botón seleccionar imagen tendremos la posibilidad de seleccionar la imagen desde la galería o podemos hacer una fotografía desde la cámara del móvil.



Para seleccionar la imagen desde la galería realizamos lo siguiente:

```
case "Ir a galeria":
    Intent intent = new Intent();
    intent.setType("image/*");
    intent.setAction(Intent.ACTION_GET_CONTENT);
    startActivityForResult(Intent.createChooser(intent, "Select Picture"), PICK_IMAGE_REQUEST);
    break;
```

El *action get content*²² me permite recuperar datos de un determinado tipo, en este caso imágenes (MIME TYPE = image/*). Por tanto lo que hago es una invocación (con intent) de las imágenes que tengo en el dispositivo.

Si por el contrario, quiero realizar una fotografía para que sea mi avatar:

```
case "Hacer foto":
    Intent cameraIntent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
    startActivityForResult(cameraIntent, CAMERA_REQUEST);
    break;
```

Para poder hacer uso de la cámara debo pedir los permisos necesarios en el manifest²³:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-feature android:name="android.hardware.camera" android:required="true" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

²²https://developer.android.com/reference/android/content/Intent.html#ACTION_GET_CONTENT

²³ <https://developer.android.com/reference/android/Manifest.permission.html>

Una vez seleccionada la imagen, para recuperar la imagen que hemos escogido debo sobrescribir la función `onActivityResult`²⁴ para obtener el resultado de llamar a la cámara de fotos o la galería, en caso de que haya seleccionado una imagen desde la galería:

```
//galeria
if (requestCode == PICK_IMAGE_REQUEST && resultCode == RESULT_OK && data != null && data.getData() != null) {
    Uri filePath = data.getData();
    try {
        bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), filePath);
        foto.setImageBitmap(bitmap);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Si por el contrario he realizado una foto desde la cámara:

```
//foto
if (requestCode == CAMERA_REQUEST && resultCode == RESULT_OK) {
    Bitmap photo = (Bitmap) data.getExtras().get("data");
    foto.setImageBitmap(photo);
}
```

Una vez seleccionada la imagen, cuando pulsamos en el botón “subir imagen”. Haciendo uso de la librería volley²⁵ realizo una petición al fichero `upload.php` pasándole como parámetros el nombre de la imagen, y la imagen.

```
//converting bitmap to string
String image = getStringImage(bitmap);

//Getting Image Name
String name = userSession.getId_usuario();

//Creating parameters
Map<String,String> params = new Hashtable<String, String>();

//Adding parameters
params.put(KEY_IMAGE, image);
params.put(KEY_NAME, name);

//returning parameters
return params;
```

Y ya se encarga el fichero `upload.php` de subir la imagen al servidor:

²⁴[https://developer.android.com/reference/android/app/Activity.html#onActivityResult\(int,%20int,%20android.content.Intent\)](https://developer.android.com/reference/android/app/Activity.html#onActivityResult(int,%20int,%20android.content.Intent))

²⁵<https://github.com/mcxiaoke/android-volley>

```
<?php
if($_SERVER['REQUEST_METHOD']=='POST'){
    $image = $_POST['image'];
    $name = $_POST['name'];

    $path = "image/usuarios/$name.png";

    $actualpath = "http://[REDACTED]/$path";

    file_put_contents($path,base64_decode($image));
    echo "Imagen subida correctamente";
}else{
    echo "Error";
}
?>
```

Si funciona correctamente se mostrará el mensaje de que se ha realizado correctamente, si no se mostrará un mensaje de error avisando de que no ha sido posible subir la imagen al servidor.

Pantalla Detalles Anuncio

Para acceder a esta pantalla debemos pulsar sobre un anuncio de la pantalla principal, dicha pantalla estará dividida en dos pestañas, por un lado los detalles del anuncio que contiene:

- La fotografía del anuncio.
- El título del anuncio.
- El usuario que ha publicado el anuncio.
- El precio del anuncio.
- La descripción del anuncio.

El resultado final de la pestaña de detalles del anuncio es el siguiente:



Además de comprar el producto (pantalla que veremos más adelante) en esta pantalla podemos denunciar el anuncio en caso de que consideremos que el anuncio no es adecuado por el motivo que sea. Para ello, si pulsamos en el texto donde pone "Reportar Anuncio" se nos abrirá una ventana modal en la que vamos a poder denunciar el

anuncio. Para ello, únicamente deberemos justificar el motivo por el cual hemos reportado el anuncio escribiendo un comentario.

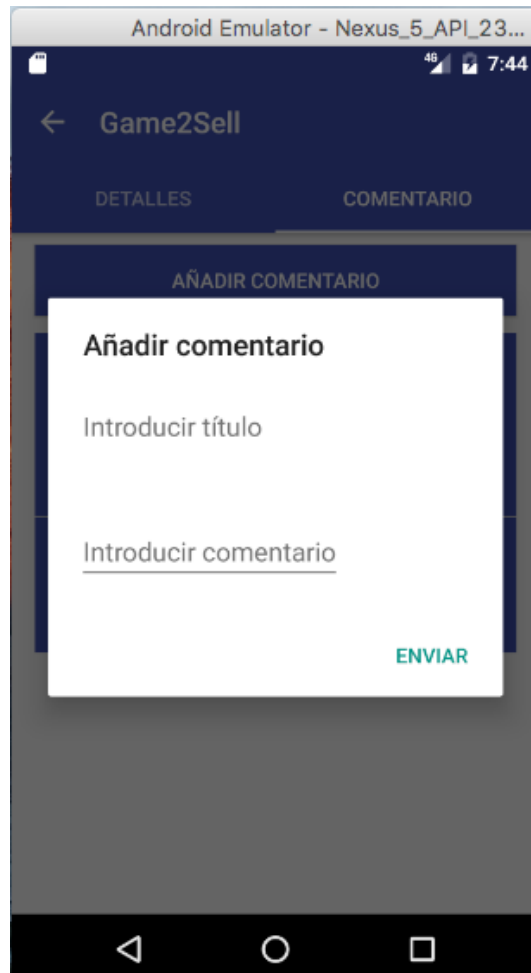
Por otro lado, tenemos la pestaña de comentarios que contiene una lista y cada elemento de la lista representa un comentario, cada comentario contiene:

- El avatar de perfil del usuario.
- El nombre del usuario.
- La fecha en la que se publicó el comentario.
- El título del comentario.
- La descripción del comentario.

El resultado final de la pestaña de comentarios es el siguiente:



Además de listar los comentarios, podemos pulsar en el botón de “añadir comentario”, al pulsar sobre el botón se nos abrirá un ventana (AlertDialog) que nos pedirá el título del comentario y el comentario en sí.



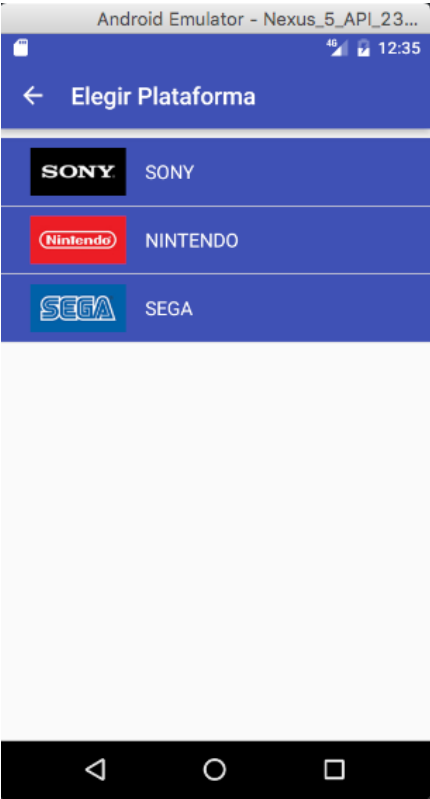
Al introducir el comentario, se mostrará el nuevo mensaje listado en los comentarios.

Pantalla de Tasación

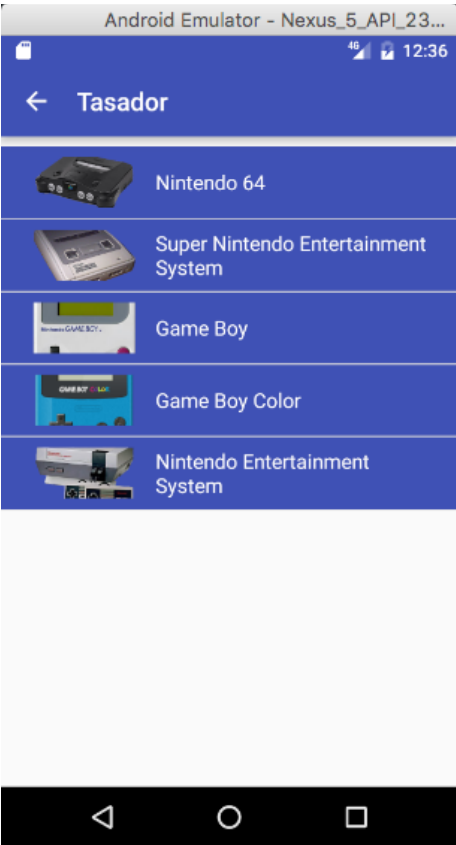
Para realizar la tasación hay que navegar por una serie de pantallas, la primera pantalla se basa en la elección del tipo de artículo, es decir, un artículo puede ser una videoconsola, un videojuego o un accesorio. Por tanto deberemos seleccionar una de estas categorías.



La segunda pantalla, mostrará aquellas plataformas de las que exista al menos algún tipo de artículo de la categoría seleccionada.



En tercer lugar, se listarán aquellos productos que pertenezcan a la plataforma escogida y a la categoría seleccionada.



Y por último, se mostrará toda la información del artículo escogido para tasar. Esto es:

- La imagen del artículo seleccionado.
- El nombre del producto.
- El precio de tasación del artículo.
- La descripción del artículo.

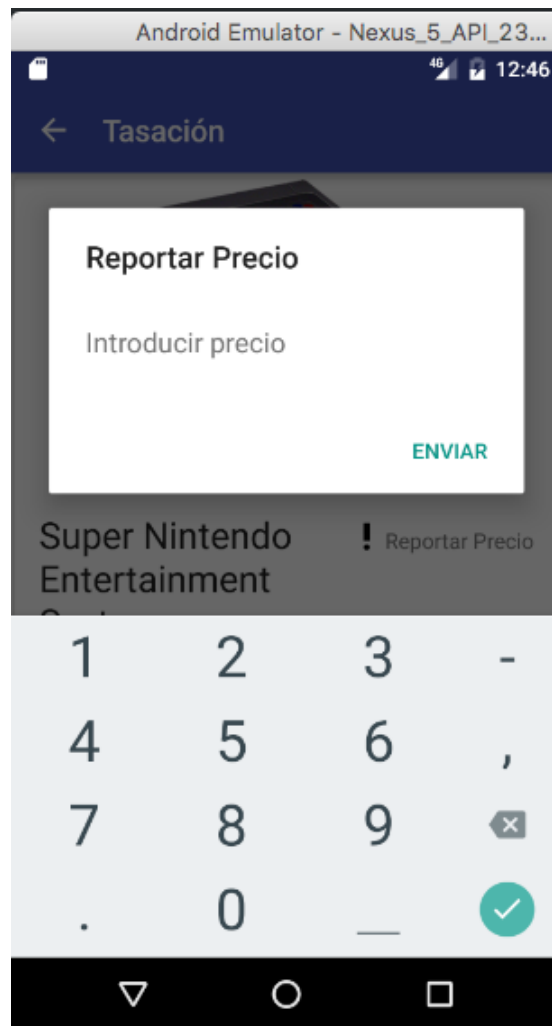
El resultado final de la pantalla de tasación es la siguientes:



Como se puede observar en la pantalla de tasación podemos crear un anuncio, para ello deberemos pulsar sobre el botón “vender”, el funcionamiento de esta pantalla la veremos más adelante.

Además de crear un anuncio y ver el precio de la tasación del artículo, tenemos la posibilidad de “reportar” el precio de tasación. Es decir, si no estamos de acuerdo con el precio de la tasación vamos a poder enviar el precio de tasación que nosotros

consideremos oportuno. Esto servirá para adecuar el precio de tasación al valor de mercado actual.



No todos los reportes se tendrán en cuenta de igual forma. Los precios introducidos por usuarios con mejor “reputación” y más “fiables” tendrán una consideración mayor. En cambio, por ejemplo un usuario que reporte un precio y no haya realizado ninguna compra o venta no se tendrá en consideración.

La idea es crear un disparador (trigger) en base de datos que después realizar un reporte de precio, sea capaz de realizar un cálculo para modificar el precio de tasación final. Consiguiendo así modificar el precio de tasación en base al feedback depositado por los usuarios en base de datos.

Pantalla Publicar Anuncio

Para acceder a esta funcionalidad, únicamente accederemos desde la tasación de artículos.

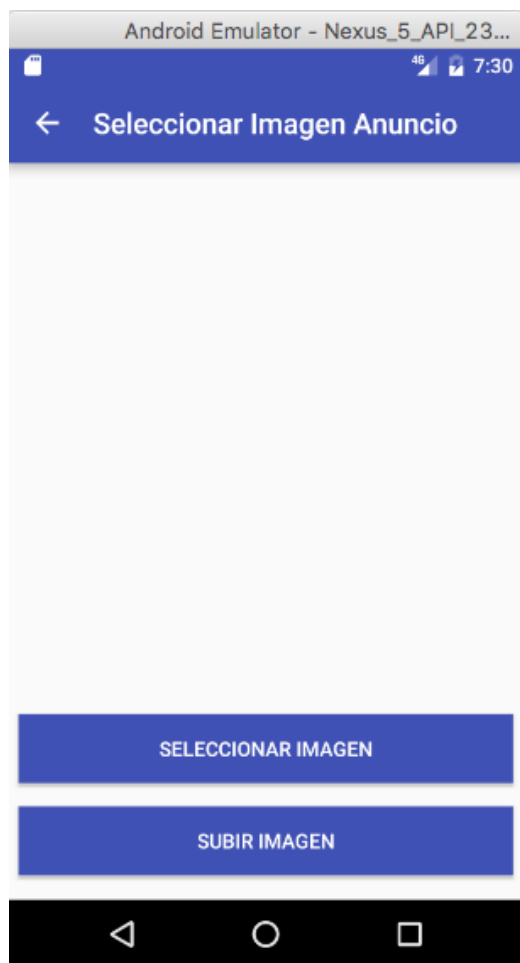
El precio de la tasación del artículo es solo una aproximación al valor de mercado del producto, pero eso no indica que debe ser el precio que deben establecer los usuarios en la aplicación, el precio será elegido por el vendedor en base a lo que el crea oportuno.

Para publicar un anuncio lo único que deberemos hacer es introducir el título que tendrá el anuncio, la descripción del producto a vender y el precio de venta. Una vez hemos rellenado todos los campos.



Obviamente como en el resto de pantallas anteriores cuando intentamos publicar el anuncio se validarán los campos, de forma que el título y la descripción no sean vacíos y el precio del anuncio sea mayor que cero.

Después solo nos faltará seleccionar la imagen que tendrá el anuncio:



La pantalla de selección de imagen de anuncio funciona del mismo modo que la de selección de imagen de perfil (su funcionamiento está explicado en el apartado de la imagen de perfil).

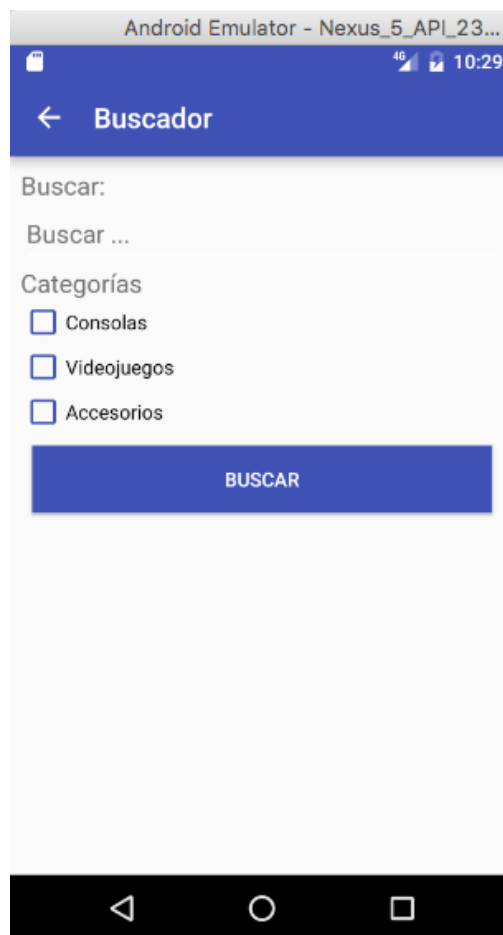
Si la publicación del anuncio se ha realizado de forma correcta se mostrará un mensaje en pantalla informando de que se ha realizado correctamente y se redireccionará a la pantalla principal, donde aparecerá el nuevo anuncio publicado.

Pantalla Buscador

Accedemos a esta funcionalidad desde el menú, cuando seleccionamos el buscador.

El buscador ofrece la posibilidad de buscar por categorías de anuncios, es decir, podemos buscar anuncios que contengan videojuegos, consolas y/o accesorios. La idea es que el buscador en el futuro, se le añadan más opciones de filtrado (pues está preparado para ello) como pudiera ser el precio.

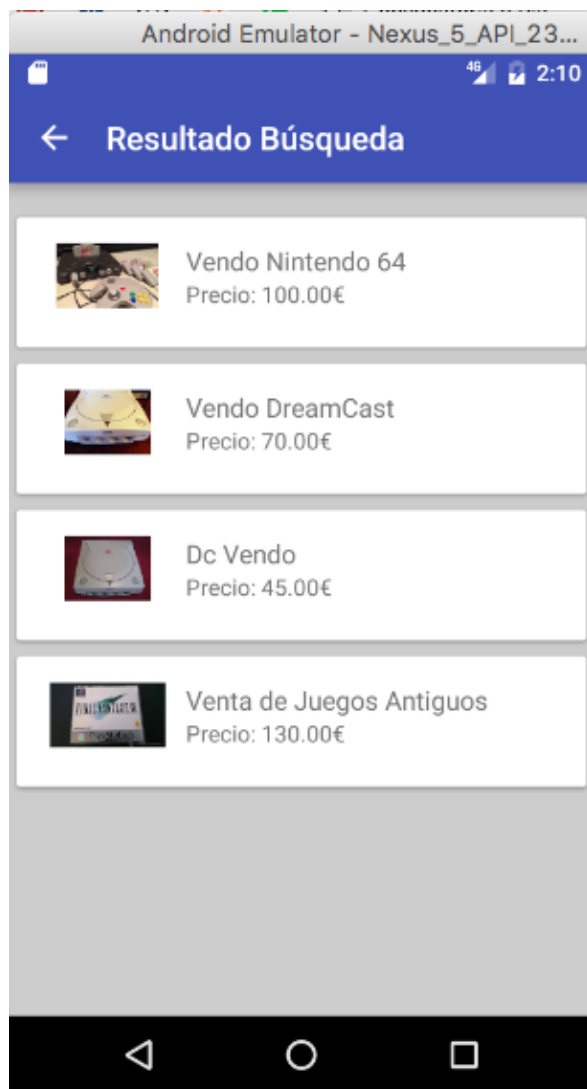
El buscador funciona de la siguiente manera, una vez introducido el texto a buscar, se deberá marcar al menos una de las categorías. La búsqueda se realiza tanto en el título del anuncio como en la descripción del anuncio.



En caso de que el resultado de la búsqueda no arroje ningún tipo de resultado, se mostrará un mensaje indicando que no se ha encontrado ningún anuncio con el texto a buscar introducido.

Si al realizar la búsqueda únicamente se devuelve un único resultado, se abrirá directamente la pantalla de detalles del anuncio.

En caso contrario, si el resultado de la búsqueda es mayor a un anuncio, se listarán todos los anuncios que coincidan con la cadena a buscar. El resultado es el siguiente:

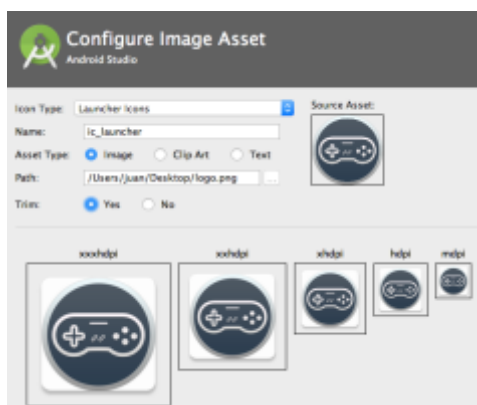


Al pulsar sobre cualquier anuncio de la lista de resultados de búsqueda, se abrirá la vista de detalles del anuncio en cuestión.

Otros

Creación Icono

Para obtener el icono para la aplicación Android, una vez lo hemos creado podemos utilizar la herramienta Android Asset Studio²⁶ que nos generara un zip exportable a Android Studio con el icono para todos los tamaños de densidad (MDPI, HDPI, XXHDPI, XXXHDPI, etc). O por el contrario, podemos hacerlo desde Android Studio desde File -> New -> Image Asset.



Internacionalización de la Aplicación

Android dispone de unos ficheros que nos permiten independizar los recursos como imágenes o textos del código fuente. Gracias a esto, aparte de lograr una menor dependencia, nos permite crear aplicaciones en diversos idiomas de manera sencilla.

Debido a que todos los textos que aparecen en la aplicación están en el fichero `res/values/strings.xml` vamos a poder crear otros ficheros `strings.xml` para otros idiomas por ejemplo para el inglés sería `res/values-en/strings.xml`. Y por tanto todos los usuarios que tengan el sistema configurado en inglés tendrán la app en inglés.

Por tanto, a pesar de que el idioma actual de la aplicación es el español sería muy sencillo traducirlo a cualquier idioma, ya que solo deberíamos crear un fichero `strings.xml` y no deberemos tocar ninguna línea de código para que funcione.

Además de esto, Android permite la localización, por ejemplo podríamos tener el fichero `values-pt-rBR/strings.xml` y `values-pt-rPt/strings.xml` para diferenciar entre el portugués de Portugal y el de Brasil.

²⁶ Android Asset Studio - <https://romannurik.github.io/AndroidAssetStudio>

SEGURIDAD

Uno de los puntos fundamentales a la hora de desarrollar software es la seguridad, pues es clave tener en cuenta la seguridad porque permite reducir los costes asociados a la reparación de defectos y minimizar la posibilidad de que un atacante pueda comprometer el software o los datos que procesa dicho software.

Uno de los objetivos asociados a la seguridad es intentar reducir la superficie de ataque, reduciendo así las oportunidades de un atacante, y los medios disponibles para comprometer el software.

Será clave por tanto desarrollar la aplicación de forma que garanticemos unos conceptos claves asociados a la seguridad:

Confidencialidad: el objetivo es protegernos frente al filtrado de la información, para ello deberemos utilizar técnicas de cifrado y hashing para que en caso de robo de información, dicha información sea inteligible o irrecuperable.

Por tanto, toda la información sensible que maneje nuestra aplicación deberá estar cifrada. Como por ejemplo la contraseña que está cifrada utilizando SHA-512.

Integridad: el objetivo es protegernos frente a las alteraciones de la información no autorizadas. Para ello, deberemos utilizar funciones de resumen para garantizar que la información no ha sido alterada. Deberemos también validar las entradas de datos para evitar ataques de inyección de SQL y garantizar así la integridad de la base de datos.

Para asegurar la integridad en nuestra aplicación, filtraré todos los campos de entrada para evitar que un atacante pueda realizar un ataque de inyección de sql. No permitiendo la entrada de caracteres que permitan realizar tal ataque.

Disponibilidad: debemos tener en cuenta que un software vulnerable puede provocar fallos y parones en el servicios (Denegación de servicio).

Autenticación: en el proceso de validación y verificación de la identidad de un usuario en un principio vamos a utilizar un único factor, dicho factor estará basado en el conocimiento (una contraseña). Un fallo en la autenticación permitiría ataques como el robo de identidad.

En un futuro la autenticación de los usuarios tendrá un segundo factor ya sea algo que posean los usuarios (por ejemplo un dispositivo móvil) o basados en una característica física (huella digital por ejemplo).

Autorización: tener en cuenta que estar autenticado no implica que se esté autorizado. Deberemos tener en cuenta el escalado de privilegios si procede. En mi caso, el escalado de privilegios no procede ya que solo hay un tipo de usuario en la aplicación y por tanto no deberé tener en cuenta la autorización.

Auditorias: es importante realizar auditorías de seguridad porque nos aportará evidencias forense en caso de ataques.

Contraseña del usuario

A la hora de almacenar las contraseñas en base de datos, lo que vamos a hacer es aplicar una función de resumen SHA-512. El motivo por el cual no almacenamos la contraseñas en textos plano es sencillo, porque si almacenamos la contraseña en texto plano cualquier atacante fácilmente podría conseguir las credenciales de acceso.

Para ello he creado una clase que se encarga de aplicar el algoritmo de la función de resumen SHA-512 sobre una cadena de texto. Su uso principal será para tratar con la contraseña.

Ataques de inyección SQL

Para evitar los ataques mediante inyección de SQL deberé filtrar las entradas de datos para evitar los caracteres “especiales” que permiten realizar dichos ataques.

Es muy importante realizar el filtrado en todas las posibles entradas de textos porque un ataque de inyección SQL puede suponer una gran pérdida de datos, por ejemplo podrían hacer un “Drop table” (sentencia que permite eliminar tablas en base de datos).

Denegación de servicio

Un atacante podría hacer un atacante de DDoS a través de una botnet (red de ordenadores zombis bajo el poder de un atacante) afectando así a la disponibilidad de nuestra aplicación.

Defenderse de un ataque de denegación de servicio es difícil, porque no podemos diferenciar entre las peticiones reales (tráfico legítimo) y las peticiones del atacante. Por ello, la mejor forma de defenderse ante un ataque de denegación de servicio es guardar y monitorizar los logs en busca de actividades sospechosas, monitorizar la red en busca de vulnerabilidades. El cloud computing nos puede servir de utilidad siempre y cuando nos permita aprovechar la escalabilidad automática, de forma que automáticamente dependiendo del consumo nos proporcione los recursos necesarios.

PRUEBAS

Para realizar las pruebas sobre la aplicación voy a realizar pruebas funcionales.

El objetivo de realizar pruebas es encontrar el mayor número de errores posibles realizando el menor número de pruebas.

Las pruebas son importantes porque de ellas dependen el éxito del proyecto, las pruebas las realizamos para contribuir al éxito del proyecto en la satisfacción al cliente.

Las pruebas más relevantes realizadas son las siguientes:

Casos de prueba de la pantalla de Inicio de Sesión

Id	Nombre de Usuario	Contraseña	Resultado Esperado
C1	Juan	12	"La contraseña debe contener 6 caracteres como mínimo."
C2	Ana	123456	"El nombre de usuario debe contener al menos 4 caracteres".
C3	Pepe	123456	"El usuario o la contraseña no son correctos".
C4	Juan	123456	Pantalla Principal, con la sesión del usuario Juan creada.

*En la base de datos existe el usuario "Juan" con la contraseña "123456".

Casos de prueba de la pantalla de Registro de Usuario

Id	Nombre de Usuario	Contraseña	Email	Resultado Esperado
C1	Abc	Pass123	Jaime8@mail.es	"El nombre de usuario debe contener al

				menos 4 caracteres.”
C2	Jaime87	123	Jaime8@mail.es	“La contraseña debe contener 6 caracteres como mínimo”
C3	Jaime87	Pass123	Jaime2@mail	“Introduce un email válido”
C4	Juan	Pass123	Jaime8@mail.es	“Nombre de usuario no disponible”
C5	Jaime87	Pass123	Jaime8@mail.es	“Cuenta creada con éxito”

*En la base de datos existe el usuario “Juan”.

Casos de prueba del Buscador

Id	Cadena a buscar	Categoría	Resultado Esperado
C1	-	Consolas	El campo de búsqueda debe contener tres caracteres como mínimo.
C2	N64	-	Debe seleccionar una o más categorías
C3	N64	Artículos	Ningún resultado.
C4	N64	Consolas y Videojuegos.	Lista de resultados con una consola y dos videojuegos.
C5	N64	consolas	La vista de detalles del artículo buscado. (cuando solo devuelve un único resultado).

*En la base de datos hay consola de nombre N64 y dos videojuegos que en el nombre contienen la palabra N64.

Casos de prueba modificar perfil usuario

Id	Nombre	Apellidos	Email	Teléfono	Fecha	Resultado
					Nacimiento	Esperado
C1	Juan	Romero	juan@gmail	662062062	13-05-1989	Introduce un email válido
C2	Juan	Romero	juan@gmail.com	66206206W	13-05-1989	Teléfono incorrecto
C3	Juan	Romero	juan@gmail.com	662062059	13-dd-1sad	Fecha incorrecta
C4	Juan	Romero	juan@gmail.com	662062062	13-05-1989	Cambios realizados con éxito.

*Tanto el nombre como el apellido no tiene ningún tipo de control, porque son campos opcionales y pueden ser nulos (al igual que la fecha de nacimiento).

Casos de prueba direcciones de envío

Id	Título	Dirección	Código Postal	Resultado
				Esperado
C1	Casa	Plaza Asimov n13 3B	03670	Ya existe una dirección de envío, con ese título.
C2	Trabajo	Calle Vostok 1	82382	El código postal no existe.
C3	Trabajo	Avenida Gagarin	03670	Dirección de envío insertada correctamente.

*En base de datos el usuario tiene una dirección de envío con el título "casa".

*La pantalla de modificación de dirección de envío tiene los mismos casos de prueba.

Casos de prueba de reportar precio

Id	Precio	Resultado Esperado
C1	-	Precio incorrecto
C2	0	Precio incorrecto
C3	5	Reporte realizado

Casos de prueba publicar anuncio

Id	Título	Descripción	Precio	Resultado Esperado
C1	-	Vendo por no usar.	80	Título incorrecto
C2	Vendo consola Atari	-	80	Descripción incorrecta
C3	Vendo consola Atari	Vendo por no usar.	-	El precio debe ser mayor a 0€
C4	Vendo consola Atari	Vendo por no usar	80	Anuncio publicado con éxito.

Casos de prueba reportar anuncio

Id	Mensaje	Resultado Esperado
C1	-	Debe introducir el motivo de denuncia del anuncio.
C2	Artículo robado.	Reporte Realizado

Casos de prueba de la pantalla de comentarios

Id	Título	Comentario	Resultado Esperado
C1	-	El artículo parece que tiene algunos desperfectos en la caja.	Falta el título del comentario.
C2	¿Podrías bajar el precio?	-	Falta el contenido del comentario.
C3	¿Podrías bajar el precio?	El artículo parece que tiene algunos desperfectos en la caja.	Comentario realizado con éxito.

PROBLEMAS, DIFICULTADES Y SOLUCIONES

Realizar una petición al api desde la aplicación Android

Cuando intentamos realizar un cálculo complejo o acceder a la red, este es nuestro caso, en Android el hilo principal (el hilo del interfaz de usuario) se bloquea y la aplicación se cierra.

Para solventar este problema, debemos crear una clase dentro del activity que extienda de AsyncTask para poder realizar la petición al api. AsyncTask lo que nos permite es crear un hilo de ejecución nuevo, este hilo secundario comparte las variables con el hilo principal y nos ofrece la posibilidad de ejecutar esos cálculos complejos o los accesos a la red sin que la aplicación se cierre. Es decir, para la ejecución de tareas asíncronas utilizaremos AsyncTask.



Dicha clase tendrá tres métodos que deberemos sobrescribir para realizar las peticiones a nuestro api que son:

- `onPreExecute`: en este método realizamos los trabajos previos a la ejecución. Por obtener el valor de un campo de texto.
- `doInBackground`: obtener el objeto json que nos devuelve la petición. Este es el único método que se ejecuta en el hilo secundario, por tanto es donde deberemos hacer el acceso a la red para que la aplicación no falle.
- `onPostExecute`: en este método lo usaremos para mostrar en la interfaz de usuario el resultado de a ver realizado la petición. Por ejemplo un mensaje indicando que ha ido bien o mal.

Mantener la sesión del usuario (compartir información entre actividades)

Uno de los problemas surgidos es el de mantener la sesión del usuario mientras vamos navegando a través de todas las actividades de la aplicación.

Para solventar dicho problema, he creado una clase usuario, que implementa la interfaz serializable para poder convertir el objeto a bytes y pasarlo entre actividades.

El funcionamiento es el siguiente, una vez que hemos iniciado sesión creamos un objeto de la clase Usuario y desde entonces vamos pasando el objeto de actividad en actividad. Con esto conseguimos mantener la sesión del usuario hasta que cerremos sesión, que en ese caso lo que haremos será borrar el objeto y llamar a la pantalla de inicio de sesión.

Mejorar el rendimiento de la aplicación a la hora de listar las imágenes de los anuncios.

Uno de los principales problemas cuando trabajamos con grandes listas ya sean “Listviews”, “GridViews” o “RecyclerViews” y estas contienen imágenes el rendimiento se ve afectado, pues cuando realizamos scroll intentará cargar todas las imágenes consiguiendo así realentizar la aplicación.

Para que esto no ocurra, he utilizado la librería Picasso que se encarga de solucionar este problema, lo solventa de forma que no intenta cargar la imagen de primeras si no que de primeras carga una imagen de “placeholder” y en el momento que el scroll para se encarga de mostrar la imagen que se encuentra en el servidor.

Mostrar caracteres UTF-8 correctamente en las respuestas JSON

Uno de los problemas surgidos al intercambiar información con JSON es que ciertos caracteres no los escribía bien (acentos, eñes, etc.).

La manera de solventar el problema ha sido utilizando la “preg_replace” que se encarga de reemplazar los caracteres, consiguiendo así poder recibir en la petición de forma correcta las palabras que contienen acentos y eñes.

```
preg_replace("/\\\\\\u([a-f0-9]{4})/e", "iconv('UCS-4LE','UTF-8',pack('V', hexdec('U$1')))", json_encode($response));
```

CONCLUSIONES

Una vez finalizado el proyecto, puedo concluir que he cumplido con los objetivos propuestos inicialmente.

He logrado implementar una base de datos sql que servirá como mecanismo de persistencia, he creado una serie de procedimientos almacenados que facilitan la interacción con la base de datos a la hora de realizar las consultas y las operaciones sobre ella.

Además he creado un api en php para realizar las llamadas a los procedimientos almacenados. Las respuestas que obtengan las distintas aplicaciones (en mi caso la app Android) al realizar las llamadas al api estarán parseadas en json.

Y por último, he desarrollado la aplicación Android que permite: registrar usuarios, iniciar sesión con un usuario, publicar anuncios, buscar anuncios, comprar productos, vender productos, tasar productos, etc.

Mejoras y Ampliaciones

Al fin y al cabo, las aplicaciones están en permanente evolución por tanto una serie de mejoras y ampliaciones que tendrá la aplicación en el futuro son las siguientes:

- Simplificar el proceso de tasación desde la aplicación, para ello, lo único que deberemos hacer para tasar un artículo es realizar una foto con la cámara del móvil para detectar cual es el artículo y cuál es su precio estimado.
- Añadir un salt en la tabla usuario, que sirva para mejorar la seguridad del almacenamiento de las contraseñas. Porque aplicar solo una función de resumen sobre la contraseña no es suficiente, un atacante podría averiguar las credenciales haciendo uso de las contraseñas típicas (123456, password, etc). Porque obviamente al aplicar hash sobre una cadena genera siempre la misma salida. Por tanto, si añadimos un salt estamos eliminando la posibilidad de que el resultado pueda buscarse a partir de una lista de pares precalculados de hash (usamos el hash para dificultar el precálculo).
- Añadir un segundo factor a la autenticación, para obtener mayor seguridad y aprovechar las ventajas de los móviles más modernos. Dicho de otra manera,

permitir conectarse en la aplicación mediante huella dactilar (factor de autenticación basado en una característica física), en aquellos móviles que disponga de lector de huellas. Hay que tener en cuenta que a mayor número de factores de autenticación habrá mayor seguridad pero causarán mayor molestia en los usuarios, por eso en el futuro no sería recomendable disponer de más de dos factores de autenticación.

- Realizar auditorías de seguridad, en la que se registrará y monitorizará las actividades de forma que podamos establecer un historial en caso de que sea necesario. La información registrada deberá estar cifrada y no registraremos información sensible.
- Dar la posibilidad de iniciar sesión/registrarse con la cuenta de Gmail, flexibilizando así el modo de acceder a la app. ²⁷
- Adaptar los anuncios destacados en base a las preferencias, gustos y necesidades de los usuarios. Para ello se monitorizará las búsquedas que realizan los usuarios en el buscador de la aplicación para descubrir sus preferencias, así como también ver cuáles han sido sus compras en la aplicación.
- Añadir más posibilidades de filtrado a la hora de realizar las búsquedas. Dar la posibilidad de filtrar anuncios que cumplan con un precio que este en un determinado intervalo o que sea mayor o menor a un determinado precio.
- Ofrecer la posibilidad de añadir anuncios a la lista de deseados, esto no servirá también para descubrir las preferencias del usuario y poder adecuar el orden de muestra de anuncios en base a sus preferencias.
- Añadir la funcionalidad de seguir usuarios y la posibilidad de comunicarse con ellos de manera privada.
- Posibilidad de compartir los anuncios mediante redes sociales como por ejemplo whatsapp, twitter y Facebook.

²⁷ Sign-in Gmail - <https://developers.google.com/identity/sign-in/android/start-integrating>

BIBLIOGRAFÍAS Y REFERENCIAS

El gran libro de Android. 4 Edición. (2014) Jesús Tomás Gironés.

Android Developers. (2016). Developer.android.com. <https://developer.android.com/>

Trello. (2016). Trello.com. <https://trello.com/>

Atom. (2016). Atom.io. <https://atom.io/>

Lucidchart. (2016). Lucidchart.com. <https://www.lucidchart.com/es>

phpMyAdmin. (2016). Phpmyadmin.net. <https://www.phpmyadmin.net/>

Android Studio. (2016). Developer.android.com. <https://developer.android.com/studio/>

Material Design. (2016). Material.android.com. <https://material.google.com/>

Datos API Google. (2016). Developer.android.com.
<https://developer.android.com/about/dashboards/index.html>

Patterns. (2016). Developer.android.com.
<https://developer.android.com/reference/android/util/Patterns.html>

Floatin Action Button. (2016). Developer.android.com.
<https://developer.android.com/reference/android/support/design/widget/FloatingActionButton.html?hl=es>

Picasso. (2016). Square.github.io. <http://square.github.io/picasso/>

GridView. (2016). Developer.android.com.
<https://developer.android.com/guide/topics/ui/layout/gridview.html>

GridView with Header and Footer. (2015). Liaohuqiu.
<https://github.com/liaohuqiu/android-GridViewWithHeaderAndFooter>

Material Icons. (2016). Design.google. <https://design.google.com/icons>

Supporting Multiple Screens. (2016). Developer.android.com.
https://developer.android.com/guide/practices/screens_support.html

RecyclerView. (2016). Developer.android.com.

<https://developer.android.com/reference/android/support/v7/widget/RecyclerView.html>

CardView. (2016). Developer.android.com.

<https://developer.android.com/training/material/lists-cards.html?hl=es>

Action get content. (2016). Developer.android.com.

<https://developer.android.com/reference/android/content/Intent.html> -

[ACTION_GET_CONTENT](#)

Android Manifest. (2016). Developer.android.com.

<https://developer.android.com/reference/android/Manifest.permission.html>

onActivityResult. (2016). Developer.android.com.

[https://developer.android.com/reference/android/app/Activity.html - onActivityResult\(int, int, android.content.Intent\)](https://developer.android.com/reference/android/app/Activity.html#onActivityResult(int,int,android.content.Intent))

Android Volley. (2015). Mcxiaoke. <https://github.com/mcxiaoke/android-volley>

Android Asset Studio. (2015). Romannurik. <https://romannurik.github.io/AndroidAssetStudio>

Sign-in Gmail. (2016). Developer.android.com. <https://developers.google.com/identity/sign-in/android/start-integrating>

IDC. (2015). Idc.com. <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>

Cinco Días. (2016). Cincodiar.com.

http://cincodias.com/cincodias/2016/06/15/tecnologia/1465987235_750846.html

PC Actual. (2013). Pcactual.com.

http://www.pcactual.com/noticias/actualidad/desarrolladores-android-demandados-2_11605

ANEXOS

Diccionario de datos

Accesorio:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		
id_consola	int(11)	No		

Anuncio:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		identifica el anuncio
título	varchar(100)	No		identifica el título del anuncio
descripcion	varchar(1000)	No		descripción del anuncio
precio	float(10,2)	No		precio puesto por el usuario del anuncio
id_usuario	int(11)	No		
id_articulo	int(11)	No		
activo	int(1)	No	0	

Anuncio_img:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		almacena el identificador de la imagen
id_anuncio	int(11)	No		

Artículo:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		identifica el artículo
nombre	varchar(100)	No		nombre del artículo
descripcion	varchar(1000)	No		descripción del artículo

precio	float(10,2)	No		precio del artículo
img	varchar(100)	No		imagen del artículo
id_plataforma	int(11)	No		

Articulos_anuncios:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		identifica los productos del anuncio
id_anuncio	int(11)	No		
id_articulo	int(11)	No		

Atributos_accesorio:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		
capacidad	varchar(200)	No		
color	varchar(200)	No		

Atributos_consola:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		
tipo	varchar(200)	No		
modelo	varchar(200)	No		
capacidad	varchar(200)	No		
formato	varchar(200)	No		
color	varchar(200)	No		

Atributos_juego:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		
formato	varchar(200)	No		
idioma	varchar(200)	No		
genero	varchar(200)	No		
desarrollador	varchar(200)	No		
edad	varchar(200)	No		

Caracteristicas_accesorio:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		Características de una consola en concreto
id_accesorio	int(11)	No		
id_atributos_accesorio	int(11)	No		

Caracteristicas_consola:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		Características de una consola en concreto
id_consola	int(11)	No		
id_atributos_consola	int(11)	No		

caracteristicas_juego:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		Características de una consola en concreto
id_juego	int(11)	No		
id_atributos_juego	int(11)	No		

Comentario:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		identifica el comentario
titulo	varchar(100)	Sí	<i>NULL</i>	título del comentario
descripcion	varchar(1000)	No		descripción del comentario
fecha	timestamp	No	CURRENT_TIMESTAMP	fecha de creación del comentario
id_usuario	int(11)	No		
id_anuncio	int(11)	No		

Consola:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		

Direccion_envio:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		identifica la dirección de envío
direccion	varchar(1000)	No		dirección de envío
id_usuario	int(11)	No		
id_localidad	int(11)	No		
titulo	varchar(200)	No		

Juego:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		
id_consola	int(11)	No		

Localidad:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		
id_provincia	int(11)	No		
nombre	varchar(200)	No		
cp	int(5)	No		

País:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		identifica al país
nombre	varchar(1000)	No		

Pedido:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		identificadores de pedidos
fecha	timestamp	No	CURRENT_TIMESTAMP	fecha en la que se realiza el pedido
id_usuario	int(11)	No		
id_producto	int(11)	No		
estado	varchar(100)	No	Pendiente	

Plataforma:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		identifica la plataforma
nombre	varchar(100)	No		nombre de la plataforma

Provincia:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		identifica la provincia
nombre	varchar(100)	No		nombre de la provincia
id_pais	int(11)	No		

Reportar_anuncio:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		identifica el reporte del anuncio
descripcion	varchar(1000)	No		descripción del reporte del anuncio
revisado	varchar(1)	No	0	
id_usuario	int(11)	No		
id_anuncio	int(11)	No		

Reportar_precio:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		identifica el reporte
precio	float(10,2)	No		precio que se reporta
revisado	varchar(1)	No	0	
id_usuario	int(11)	No		
id_articulo	int(11)	No		

seguimiento_producto:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		identifica el seguimiento del producto
id_usuario	int(11)	No		
id_producto	int(11)	No		

Usuario:

Columna	Tipo	Nulo	Predeterminado	Comentarios
id	int(11)	No		identifica el usuario
nombre	varchar(100)	Sí	<i>NULL</i>	nombre del usuario
apellido	varchar(100)	Sí	<i>NULL</i>	apellidos del usuario
email	varchar(100)	No		email del usuario
user	varchar(100)	No		identificador login del usuario
password	varchar(200)	No		psw del usuario para login
fecha_nacimiento	date	Sí	<i>NULL</i>	fecha de nacimiento del usuario
telefono	int(11)	Sí	<i>NULL</i>	teléfono del usuario
imagen	varchar(100)	Sí	<i>NULL</i>	identifica la imagen del usuario
baja	varchar(1)	No	0	0 indica a un usuario activo y 1 indica el usuario se ha dado de baja

Modelo Relacional

ACCESORIO (id, id_consola)

CP: id

C.Aje: id_consola → CONSOLA

ANUNCIO (id, titulo, descripción, precio, id_usuario, id_articulo, activo)

CP: id

C.Aje: id_usuario → USUARIO

C.Aje: id_articulo → ARTICULO

ANUNCIO_IMG (id, id_anuncio)

CP: id

C.Aje: id_anuncio → ANUNCIO

ARTICULO (id, nombre, descripción, precio, img, id_plataforma)

CP: id

C.Aje: id_plataforma → PLATAFORMA

ARTICULOS_ANUNCIOS (id, id_anuncio, id_artiuclo)

CP: id

C.Aje: id_anuncio → Anuncio

C.Aje: id_articulo → Articulo

ATRIBUTOS_ACCESORIO (id, capacidad, color)

CP: id

ATRIBUTOS_CONSOLA (id, tipo, modelo, capacidad, formato, color)

CP: id

ATRIBUTOS_JUEGO (id, formato, idioma, genero, desarrollador, edad)

CP: id

CARACTERISTICAS_ACCESORIO (id, id_accesorio, id_atributos_accesorio)

CP: id

C.Aje: id_accesorio → ACCESORIO

C.Aje: id_atributos_accesorio → ATRIBUTOS_ACCESORIO

CARACTERISTICAS_CONSOLA (id, id_consola, id_atributos_consola)

CP: id

C.Aje: id_consola → CONSOLA

C.Aje: id_atributos_consola → ATRIBUTOS_CONSOLA

CARACTERISTICAS_JUEGO (id, id_juego, id_atributos_juego)

CP: id

C.Aje: id_juego → JUEGO

C.Aje: id_atributos_juego → ATRIBUTOS_JUEGO

COMENTARIO (id, titulo, descripción, fecha, id_usuario, id_anuncio)

CP: id

C.Aje: id_usuario → USUARIO

C.Aje: id_anuncio → ANUNCIO

CONSOLA (id)

CP: id

DIRECCION_ENVIO (id, dirección, id_usuario, id_localidad, titulo)

CP: id

C.Aje: id_usuario → USUARIO

C.Aje: id_localidad → LOCALIDAD

JUEGO (id, id_consola)

CP: id

C.Aje: id_consola → CONSOLA

LOCALIDAD (id, id_provincia, nombre, cp)

CP: id

C.Aje: id_provincia → PROVINCIA

PAIS (id, nombre)

CP: id

PEDIDO (id, fecha, id_usuario, id_producto, estado)

CP: id

C.Aje: id_usuario → USUARIO

C.Aje: id_producto → ARTICULOS_ANUNCIOS

PLATAFORMA (id, nombre)

CP: id

PROVINCIA (id, nombre, id_pais)

CP: id

C.Aje: id_pais → PAIS

REPORTAR_ANUNCIO (id, descripción, revisado, id_usuario, id_anuncio)

CP: id

C.Aje: id_usuario → USUARIO

C.Aje: id_anuncio → ANUNCIO

REPORTAR_PRECIO (id, precio, revisado, id_usuario, id_articulo)

CP: id

C.Aje: id_usuario → USUARIO

C.Aje: id_articulo → ARTICULO

USUARIO (id, nombre, apellido, email, user, password, fecha_nacimiento, teléfono, imagen, baja)

CP: id

Procedimientos

Sp_DameUsuario

Recibe:

- nombre
- password

Realiza:

- Comprobación en la tabla “usuario”.

Devuelve:

- Si existe un usuario con dicho nombre y contraseña.

Notas Adicionales:

- El password estará cifrado.

Sp_InsertaUsuario

Recibe:

- nombre
- apellido
- email
- nombre de usuario
- contraseña
- fecha de nacimiento
- teléfono
- imagen de perfil

Realiza:

- Inserción en la tabla “usuario”.

Sp_BorraUsuario

Recibe:

- código identificativo de un usuario (id_usuario)

Realiza:

- Actualización en la tabla “usuario”

Procedimiento:

- Actualiza el usuario, asignándole al campo “baja” un 1. Los usuarios con un 1 en el campo baja, son usuarios no activos de la aplicación.

Sp_InsertaConsola

Recibe:

- nombre
- descripción
- precio
- imagen
- plataforma a la que pertenece la consola

Realiza:

- Inserción en la tabla “artículo” y en la tabla “consola”.

Procedimiento:

- Inserción de un artículo y en la tabla consola almacena el id de la consola insertada.

Sp_InsertaVideojuego

Recibe:

- nombre
- descripción
- precio
- imagen
- plataforma a la que pertenece el videojuego

Realiza:

- Inserción en la tabla “artículo” y en la tabla “juego”.

Procedimiento:

- Inserción de un artículo y en la tabla juego almacena el id del videojuego insertado.

Sp_InsertaAtributosConsola

Recibe:

- id_consola
- tipo (puede tener los valores “sobremesa” o “portátil”)
- modelo (“FAT”, “SLIM”, etc)
- capacidad (500Gb, etc)
- formato (NTSC, PALo NTSC-J)
- color

Realiza:

- Inserta en la tabla “atributos_consola” y en la tabla “caracteristicas_consola”.

Procedimiento:

- Inserta los atributos de consola, el id consola y el id atributo en "características_consola".

SP_DameTodasConsolas

Realiza:

- Consulta en la tabla "artículo"

Devuelve:

- Todas las consolas de la base de datos.

Sp_DameConsolasPorPlataforma

Recibe:

- id de plataforma

Realiza:

- Consulta en la tabla "artículo"

Devuelve:

- Todas las consolas que pertenezcan a dicha plataforma.

Sp_BuscarArticulo

Realiza:

- Consulta en la tabla "artículo"

Devuelve:

- Todos los artículos (consolas, videojuegos, accesorios) que tengan en el nombre la cadena a buscar recibida por parámetro.

Sp_ActualizarArticulo

Recibe:

- descripción
- precio
- imagen
- plataforma a la que pertenece el artículo.

Realiza:

- Actualización en la tabla "artículo"

Procedimiento:

- Actualiza los datos de un artículo ya sea una consola, videojuego o accesorio.

Sp_ActualizarUsuario

Recibe:

- Campos a modificar
- Identificador del usuario a modificar

Realiza:

- Actualización en la tabla “usuario”

Procedimiento:

- Permite actualizar los campos del usuario, todos salvo el nombre de usuario.

Sp_DameAtributosConsola

Recibe:

- código identificador de la consola.

Realiza:

- Consulta en la tabla “atributos_consola”

Devuelve:

- Todos los atributos pertenecientes a la consola recibida por parámetros.

Sp_ActualizarAtributosJuego

Recibe:

- código identificador del atributo a modificar.

Realiza:

- Actualización en la tabla “atributos_juego”

Procedimiento:

- Actualizar los atributos de un juego

Sp_DameTodosJuegos

Devuelve:

- Todos los videojuegos de la base de datos.

Sp_DameJuegosPorPlataforma

Recibe:

- código de plataforma.

Realiza:

- Consulta en la tabla “artículo”

Devuelve:

- Todos los videojuegos que pertenezcan a la plataforma recibida por parámetros.

Sp_DameJuegosPorConsola

Recibe:

- Código identificador de la consola.

Realiza:

- Consulta en la tabla “artículo”.

Devuelve:

- Todos los juegos que pertenezcan a la consola recibida por parámetro.

Sp_InsertaAccesorio

Recibe:

- nombre
- descripción
- precio
- imagen
- plataforma a la que pertenece el accesorio.

Realiza:

- Inserción en la tabla artículos y en la tabla accesorio

Procedimiento:

- Inserción en la tabla artículos y en la tabla accesorio almacena el id del accesorio que acabamos de insertar y el id de la consola a la que pertenece el accesorio.

Sp_DameAtributosJuego

Recibe:

- id de un juego

Realiza:

- Consulta en la tabla “atributos_juego”.

Devuelve:

- Todos sus atributos.

Sp_InsertaAtributosAccesorio

Recibe:

- id de un accesorio

Realiza:

- Inserta en la tabla "atributos_accesorio".

Procedimiento:

- Inserta una serie de características sobre ese accesorio (capacidad y color)

Sp_InsertarAtributosJuego

Recibe:

- id del juego

Realiza:

- Consulta en la tabla "atributos_juego".

Procedimiento:

- Inserta una serie de características sobre ese juego (formato, idioma, género, desarrollador, edad)

Sp_ActualizarAtributoAccesorio

Recibe:

- El código identificador del atributo de accesorio a modificar.

Realiza:

- Actualiza en la tabla "atributos_accesorio".

Procedimiento:

- Actualizar los atributos de un accesorio pasado por parámetro.

Sp_DameTodosAccesorios

Realiza:

- Consulta en la tabla "artículo".

Devuelve:

- Todos los accesorios.

Sp_DameAccesoriosConsola

Recibe:

- El código identificador de una consola.

Realiza:

- Consulta en la tabla "artículo".

Devuelve:

- Todos los accesorios que pertenezcan a la consola pasada por parámetro.

Sp_DameAccesoriosPorPlataforma

Recibe:

- El código identificador de una plataforma.

Realiza:

- Consulta en la tabla "artículo".

Devuelve:

- Todos los accesorios que pertenezcan a la plataforma recibida por parámetros.

Sp_InsertaComentario

Recibe:

- título
- descripción
- id_usuario
- id_anuncio

Realiza:

- Inserción en la tabla comentarios

Procedimiento:

- Permite insertar a cualquier usuario un comentario

Sp_BorrarComentario

Recibe:

- id del comentario a borrar.

Realiza:

- Borra un comentario

Procedimiento:

- Permite borrar al administrador los comentarios

Sp_ActualizarComentario

Recibe:

- id
- titulo
- descripción.

Realiza:

- Editar un comentario

Procedimiento:

- Permite al administrador editar comentarios pasando

Sp_DameComentarioPorUsuario

Recibe:

- id_usuario

Devuelve:

- Los comentarios de un usuario pasado por parámetro.

Sp_DameComentarioPorAnuncio

Recibe:

- id_anuncio

Devuelve:

- Los comentarios de un anuncio pasado por parámetro.

Sp_InsertarDireccionEnvio

Recibe:

- dirección
- id_usuario
- id_localidad
- titulo

Realiza:

- Inserta una dirección de envío ("Registro realizado con éxito")

Procedimiento:

- Permite insertar al usuario direcciones de envío.

Sp_BorrarDireccionEnvio

Recibe:

- id de la dirección

Realiza:

- Borra una dirección de envío

Procedimiento:

- Permite a los usuarios borrar una dirección de envío pasada por parámetro.

Sp_EditarDireccionEnvio

Recibe:

- id
- dirección
- id_localidad
- id_usuario
- titulo.

Realiza:

- Editamos la dirección de envío.

Procedimiento:

- Permite editar a los usuarios su dirección de envío pasada por parámetro.

Comprobamos:

- Si la dirección y localidad esta repetida ("La dirección esta repetida")

Sp_DameDireccionesEnvioPorUsuario

Recibe:

- id_usuario.

Devuelve:

- Las direcciones de envío de un usuario pasado por parámetro.

Sp_InsertarPlataforma

Recibe:

- Nombre de la plataforma

Realiza:

- Inserta una plataforma

Procedimiento:

- Permite insertar al administrador plataformas pasadas por parámetro.

Sp_BorrarPlataforma

Recibe:

- id de la plataforma

Realiza:

- Borra una plataforma

Procedimiento:

- Permite borrar al administrador plataformas pasadas por parámetro.

Sp_DamePaíses

Devuelve:

- Todos los países

Sp_DameProvinciaPorPais

Recibe:

- id_pais

Devuelve:

- Todas las provincias de un país pasado por parámetro.

Sp_DameLocalidadPorProvincia

Recibe:

- id_provincia

Devuelve:

- Las localidades de una provincia pasada por parámetro.

Sp_InsertarPedido

Recibe:

- id_usuario
- id_producto

Realiza:

- Inserta un pedido

Procedimiento:

- Permite al usuario insertar un pedido pasado por parámetro.

Sp_BorrarPedido

Recibe:

- id_pedido

Realiza:

- Borra un pedido

Procedimiento:

- Permite al usuario borrar un pedido pasado por parámetro.

Sp_DamePedidosPorUsuario

Recibe:

- id_usuario

Devuelve:

- Los pedidos comprados por un usuario pasado por parámetro.

Sp_DamePedidosPorProducto

Recibe:

- id_producto

Devuelve:

- Los pedidos realizados para un producto pasado por parámetro.

Sp_DameAnuncios

Devuelve:

- Todos los anuncios

Sp_InsertaAnuncio

Recibe:

- titulo
- descripción
- precio
- id_usuario

Realiza:

- Inserción en la tabla anuncio y en la tabla artículos_anuncio

Procedimiento:

- Inserción en la tabla anuncio y en la tabla artículos_anuncio almacena el id del anuncio que acabamos de insertar y el id del artículo que pertenece a ese anuncio.

Sp_ActualizarAnuncio

Recibe:

- descripción
- precio
- titulo
- id_anuncio

Realiza:

- Actualización de la tabla anuncio

Procedimiento:

- Permite editar a los usuarios el anuncio pasada por parámetro.

Sp_DeshabilitarAnuncio

Recibe:

- Id_anuncio

Realiza:

- Actualizamos la tabla “anuncio”

Procedimiento:

- Actualiza el anuncio pasado por parámetro, asignándole al campo “activo” un 1. Los anuncios con un 1 en el campo activo, son anuncios no habilitados de la aplicación los cuales se pondrá así si el artículo es comprado por alguien.

Sp_InsertaAnuncioImg

Recibe:

- Id_anuncio

Realiza:

- Inserta en la tabla “anuncio_img”

Procedimiento:

- Permite insertar al usuario imágenes a un anuncio pasado por parámetro.

Sp_BorrarAnuncioImg

Recibe:

- Id_anuncio_img

Realiza:

- Borrar en la tabla “anuncio_img”

Procedimiento:

- Permite borrar al usuario la imagen pasada por parámetro.

Sp_DameAnuncioImgs

Recibe:

- Id_anuncio

Realiza:

- Consulta en la tabla "anuncio_img"

Devuelve:

- Todas las imágenes de un anuncio pasado por parámetro.

Sp_InsertaReporteAnuncio

Recibe:

- descripcion
- Id_anuncio
- Id_usuario

Realiza:

- Inserta en la tabla "reportar_anuncio"

Procedimiento:

- Permite crear al usuario un reporte de un anuncio pasado por parámetro.

Sp_ActualizarReporteAnuncio

Recibe:

- descripcion
- Id_reporte

Realiza:

- Actualiza en la tabla "reportar_anuncio"

Procedimiento:

- Permite actualizar al usuario un reporte de un anuncio pasado por parámetro.

Sp_ReporteAnuncioRevisado

Recibe:

- Id_reporte_anuncio

Realiza:

- Actualiza en la tabla “reportar_anuncio”

Procedimiento:

- Actualiza un reporte de un anuncio pasado por parámetro asignándole al campo “revisado” un 1. Los anuncios reportados con un 1 en el campo revisado, son reportes de anuncio revisados de la aplicación.

Sp_DameReporteAnuncioNoRevisados

Devuelve:

- Los reportes de anuncios que no están revisados, es decir el campo revisado es “0”.

Sp_DameReporteAnuncioPorAnuncio

Recibe:

- id_anuncio

Devuelve:

- Los reportes de anuncios realizados para un anuncio pasado por parámetro.

Sp_DameReporteAnuncioPorUsuario

Recibe:

- id_usuario

Devuelve:

- Los reportes de anuncios realizados para un usuario pasado por parámetro.

Sp_DameReporteAnuncioRevisados

Devuelve:

- Los reportes de anuncios que están revisados, es decir el campo revisado es “1”.

Sp_DameReporteAnuncioTodos

Devuelve:

- Los reportes de anuncios realizados por los usuarios.

Sp_InsertaReportePrecio

Recibe:

- descripcion
- Id_articulo
- Id_usuario

Realiza:

- Inserta en la tabla “reportar_precio”

Procedimiento:

- Permite crear al usuario un reporte de un precio de un artículo pasado por parámetro.

Comprobamos:

- Si el artículo existe ("El artículo no existe")

Sp_ActualizarReportarPrecio

Recibe:

- precio
- Id_reporte

Realiza:

- Actualiza en la tabla "reportar_precio"

Procedimiento:

- Permite actualizar al usuario un reporte de un precio pasado por parámetro.

Sp_ReportePrecioRevisado

Recibe:

- Id_reporte_precio

Realiza:

- Actualiza en la tabla "reportar_precio"

Procedimiento:

- Actualiza un reporte de un precio pasado por parámetro asignándole al campo "revisado" un 1. Los precio reportados con un 1 en el campo revisado, son reportes de precio revisados de la aplicación.

Sp_DameReportePrecioNoRevisado

Devuelve:

- Los reportes de precios que no están revisados, es decir el campo revisado es "0".

Sp_DameReportePrecioPorArticulo

Recibe:

- id_articulo

Devuelve:

- Los reportes de precios realizados para un artículo pasado por parámetro.

Sp_DameReporteAnuncioPorUsuario

Recibe:

- id_usuario

Devuelve:

- Los reportes de precios realizados para un usuario pasado por parámetro.

Sp_DameReportePrecioRevisados

Devuelve:

- Los reportes de precios que están revisados, es decir el campo revisado es "1".

Sp_DameReportePrecioTodos

Devuelve:

- Los reportes de precios realizados por los usuarios.

Mockups

Pantalla de Inicio de sesión



Pantalla de Registro



Pantalla principal



Pantalla dirección de envío



Perfil de usuario

A hand-drawn sketch of a mobile app screen titled "Datos Usuario". The screen displays a form with the following fields: "Nombre" (Name), "Apellidos" (Last names), "Email", "Fecha Nacimiento" (Date of birth) with a calendar icon, and "Teléfono" (Phone number). Each field is represented by a rectangular input box. At the bottom of the form is a "Guardar" (Save) button. The screen is framed by a dark grey border representing the phone's bezel and home button.

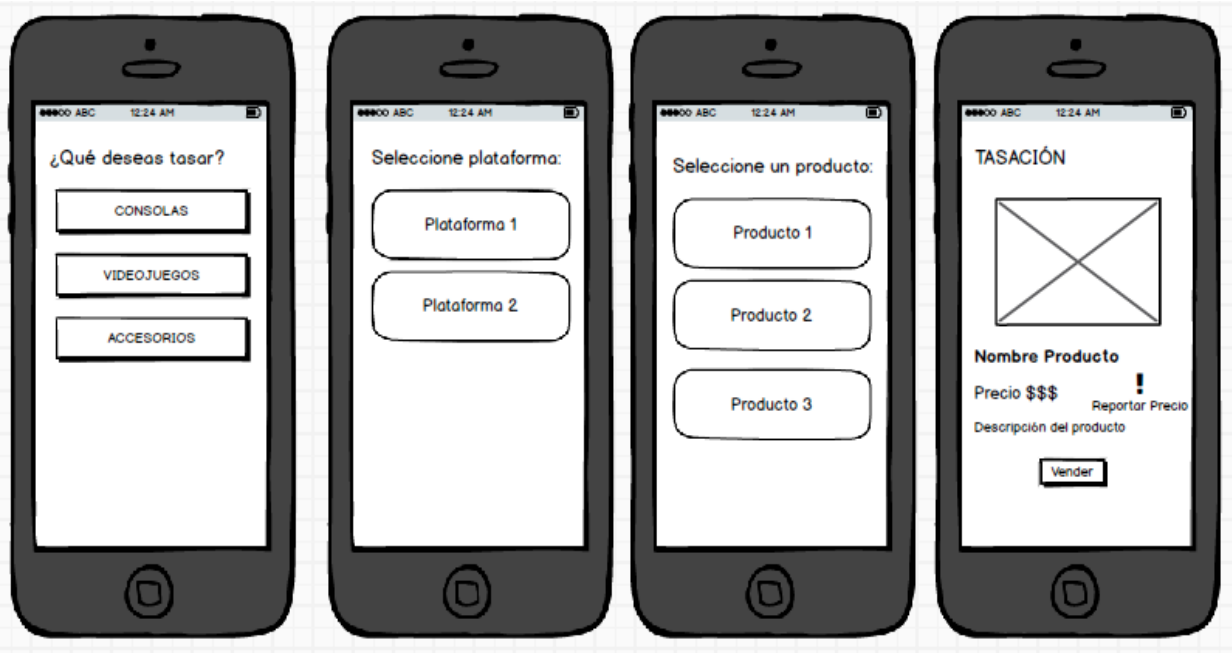
Cambiar avatar

A hand-drawn sketch of a mobile app screen titled "Seleccionar Avatar". The screen displays a square placeholder for a profile picture, containing a simple line drawing of a smiling person. Below the placeholder are two buttons: "Seleccionar Imagen" (Select image) and "Subir Imagen" (Upload image). The screen is framed by a dark grey border representing the phone's bezel and home button.

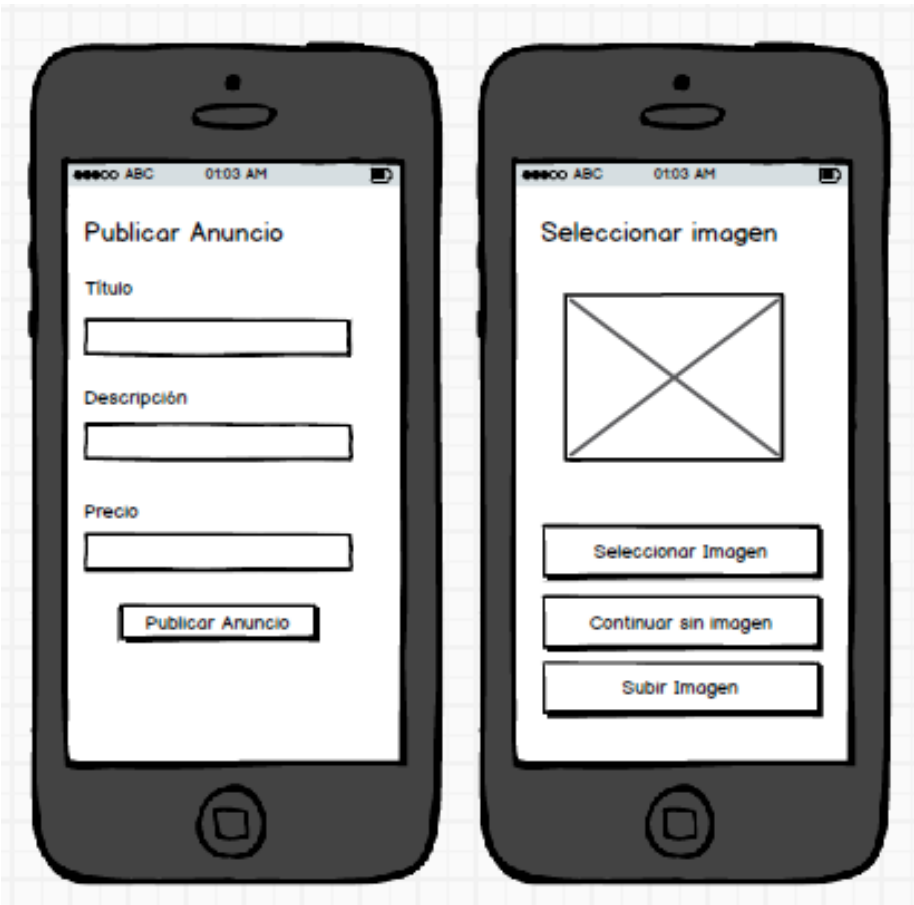
Detalles anuncio



Tasación



Publicar anuncio



Buscador

