PROYECTO FIN DE CURSO

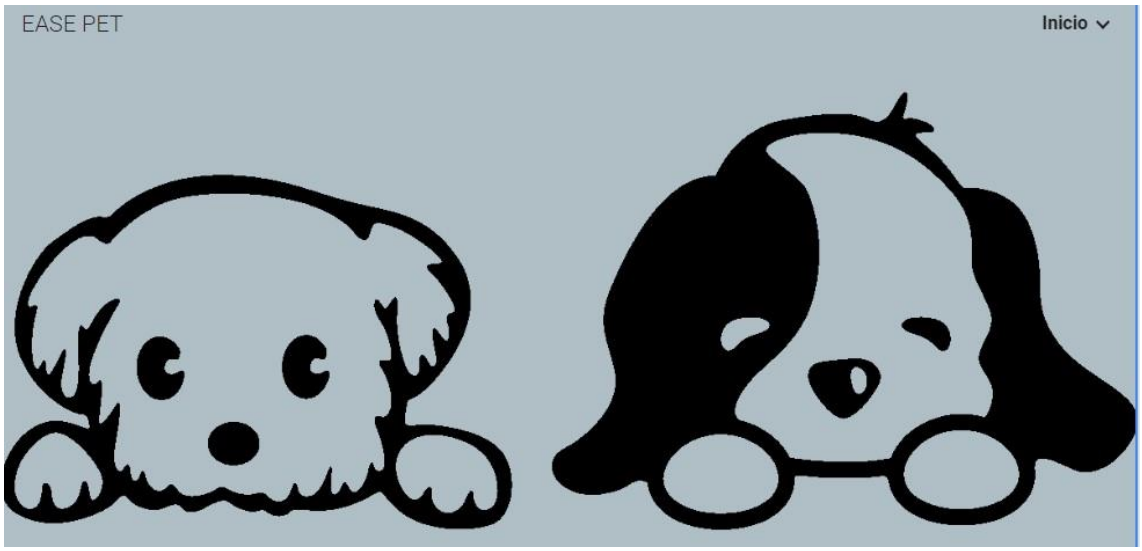
EASE PET

Desarrollo de aplicaciones multiplataforma

Aplicación Android

2022/2023

Tutor: Alberto Imaz



Sonia Borreguero del Pozo

IES CLARA DEL REY

ÍNDICE

Contenido

1. Introducción

Descripción del proyecto

Este proyecto consiste en una aplicación desarrollada en Android Studio llamada Ease Pet que permite a los dueños tener la tranquilidad de pasear a sus mascotas sin miedos a perderlas. La aplicación tiene una serie de pasos, siendo el primero el registro de los dueños para almacenar su información en la base de datos , y posteriormente el registro de la mascota. Una vez registrado ambos datos , se podrá acceder a la aplicación y generar automáticamente el código QR.

Tecnologías usadas

La aplicación se ha desarrollado en Java usando el IDE Android Studio. Para el almacenamiento de datos de usuarios de la aplicación se ha optado por usar SQLite. También, se ha procedido a la realización de pruebas, instalación y simulación dispositivos Android tanto en simuladores como el dispositivos físicos Android.

JAVA

Java es un lenguaje de programación ampliamente utilizado para codificar aplicaciones web. Ha sido una opción popular entre los desarrolladores durante más de dos décadas, con millones de aplicaciones Java en uso en la actualidad. Java es un lenguaje multiplataforma, orientado a objetos y centrado en la red que se puede utilizar como una plataforma en sí mismo. Es un lenguaje de programación rápido, seguro y confiable para codificarlo todo, desde aplicaciones móviles y software empresarial hasta aplicaciones de macrodatos y tecnologías del servidor.

En mi caso, con Ease Pet he decidido optar por este lenguaje, porque es muy versátil y tiene la suficiente potencia para crear una aplicación útil, buena y sencilla. Es el lenguaje que más he usado y con el que me siento más cómoda para desarrollar mi aplicación.

.12.2 SQLITE

La plataforma Android Studio , cuenta con toda una gama de opciones para acceder a los datos propios de una aplicación, incorporando a cada serie las herramientas necesarias para crear y gestionar bases de datos. Esta herramienta es SQLITE.

Es muy útil y necesaria para poder almacenar la información de mi aplicación de forma persistente y sencilla. Como dato a destacar de esta herramienta es que no necesita ni configuración ni el soporte de un servidor.

He decidido hacer uso de esta herramienta en mi IDE porque era necesario usar algún promotor de gestión de base de datos, para almacenar toda la información de los clientes y sus mascotas porque es la manera más rápida y sencilla de tener mis datos almacenados.

Software utilizado

Android studio

¿Qué es Android studio?

Android Studio es un entorno de desarrollo integrado (IDE) oficial que se usa en el desarrollo de apps para Android. Basado en el potente editor de código y las herramientas para desarrolladores de Intelli J IDEA[1]

Principales características que define el IDE Android studio:

- El sistema de compilación es flexible, además de ser compatible con Gradle, la cual permite la automatización de compilaciones de forma flexible y con gran rendimiento.
- Permite al usuario trabajar de forma fluida y con gran cantidad de funciones prácticas.

[1] https://developer.android.com/studio/intro?hl=es-419

- Esta plataforma te permite desarrollar aplicaciones para cualquier dispositivo Android.
- Contiene plantillas de compilación que te ayudan a otorgar funciones comunes de otras apps e importar códigos de muestra
- Modificar fragmentos de código y recursos de una app sin necesidad de que esta se reinicie.
- Integración con GitHub y plantillas de código para ayudarte a compilar funciones de apps comunes y también importar código de muestra.
- Compatibilidad integrada con Google Cloud Platform.

Lenguaje usado para programar en Android studio

Desde siempre, el sistema operativo de Android se ha desarrollado a través del lenguaje de programación Java. En el caso de este proyecto final de grado se ha optado por usar Java.

Es interesante a futuro migrar de Java a Kotlin en la aplicación Ease Pet.

Hardware utilizado

Para el desarrollo de proyecto se ha utilizado los siguientes dispositivos:

- Pórtatil HP laptop 15-da0
- El procesador del portátil es de Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
- Ram 8 GB
- Sistema operativo de 64 bits, procesador basado en x64.
- Dispositivo físico Xiaomi redmi note 6 pro
- Dispositivo físico Samsung Galaxy 6 edge.

El uso de los dispositivos móviles externos es para asemejar más a la realidad como queda finalmente la aplicación. En algunas ocasiones los simuladores de móviles proporcionados en Android studio no representan con la misma claridad los colores, la nitidez y hasta el taño y textura de las letras.

Se han usado dos dispositivos Android de distintas marcas, las más vendidas en el mercado para hacer pruebas con clientes potenciales.

2. DISEÑO

2.1 Descripción del funcionamiento de la app

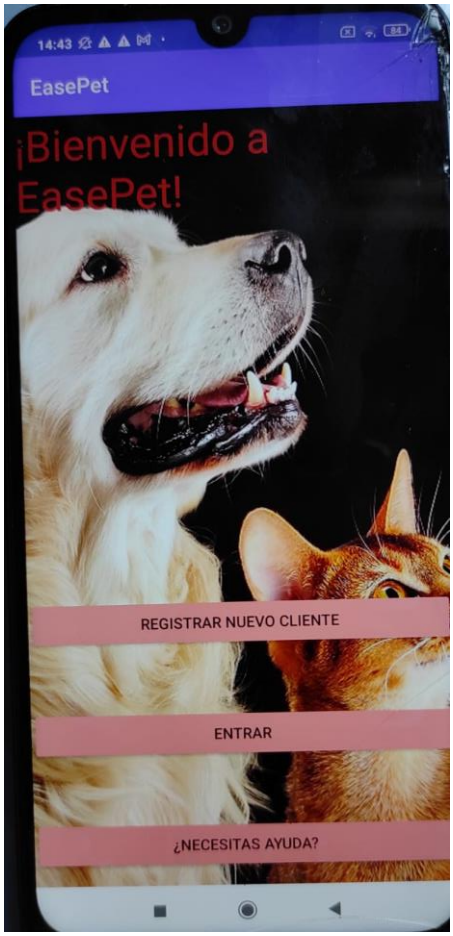
El principal objetivo de Ease pet es que la mayoría de las mascotas estén localizadas y se pueda obtener la información de rastreo de una manera continuada. Con esto evitamos que las mascotas se pierdan y si se pierden encontrarles de una manera rápida, eficaz y segura. Con ello contribuiremos a la tranquilidad de los dueños y el mayor disfrute del momento de pasear con ellos. Con esta idea queremos llegar a conseguir confianza y tranquilidad para los dueños y seguridad para las mascotas.

Ease Pet tiene como propósito ofrecer la tranquilidad y seguridad a los dueños a través de un simple collar sin necesidad de insertar ningún tipo de “chip” a tu mascota y que proporcionará el código con la geolocalización de la mascota a cualquier hora del día.. Generar tranquilidad y seguridad en todo momento que implique sacar a la mascota del domicilio habitual, así con ello cumplimos con el objetivo de salud y bienestar de los objetivos de desarrollo sostenible.

Módulos del sistema

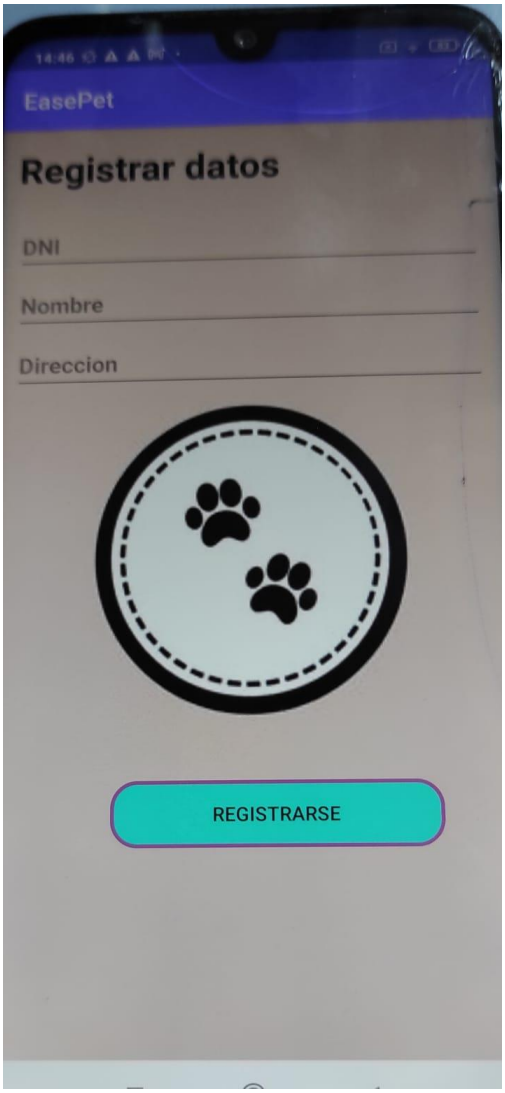
Ventana principal de la aplicación.

Esta es la primera pantalla que se presenta al iniciar la aplicación. Se muestra un texto de bienvenida al usuario. Hay 3 botones. El primero es el paso inicial para entrar en la aplicación. Es necesario registrar los datos del usuario. y con el DNI y el nombre podrá acceder al botón Entrar. Una vez se haya registrado, podrá comenzar a usar la aplicación. Como tercer botón hay una opción de ayuda en el caso de que el usuario tuviera algún tipo de problema o duda respecto al uso de la aplicación.

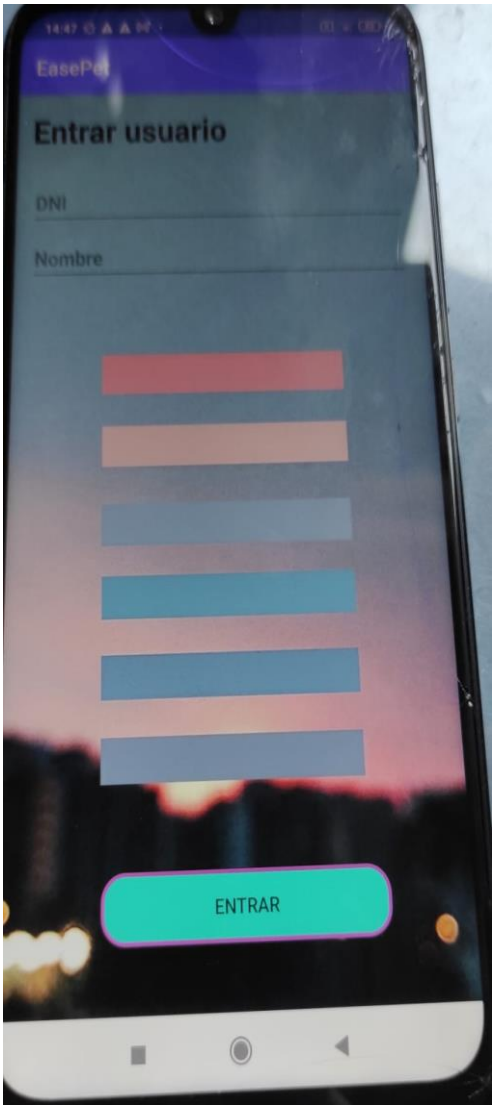


Ventana de registro **Ventana de Login**

Esta ventana es el primer paso para que
El usuario registre sus datos necesarios y
Pueda acceder posteriormente a la aplicación.
Se controlará que no haya 2 usuarios con
El mismo DNI.



Esta ventana recoge
los datos necesarios
Para acceder a la App



Ventana registro de la mascota

En esta ventana se registrará el tipo de mascota que tenga el usuario. Se añadirá el nombre de cada mascota. Si el usuario tuviera más de una mascota, puede seleccionar más de una. Esta ventana contiene un control, el cual obliga a seleccionar Al menos una opción. Hay que recordar que esta aplicación está creada para dueños de mascotas, si no se posee al menos una no será útil esta aplicación para ese usuario.

Ventana generación del código QR

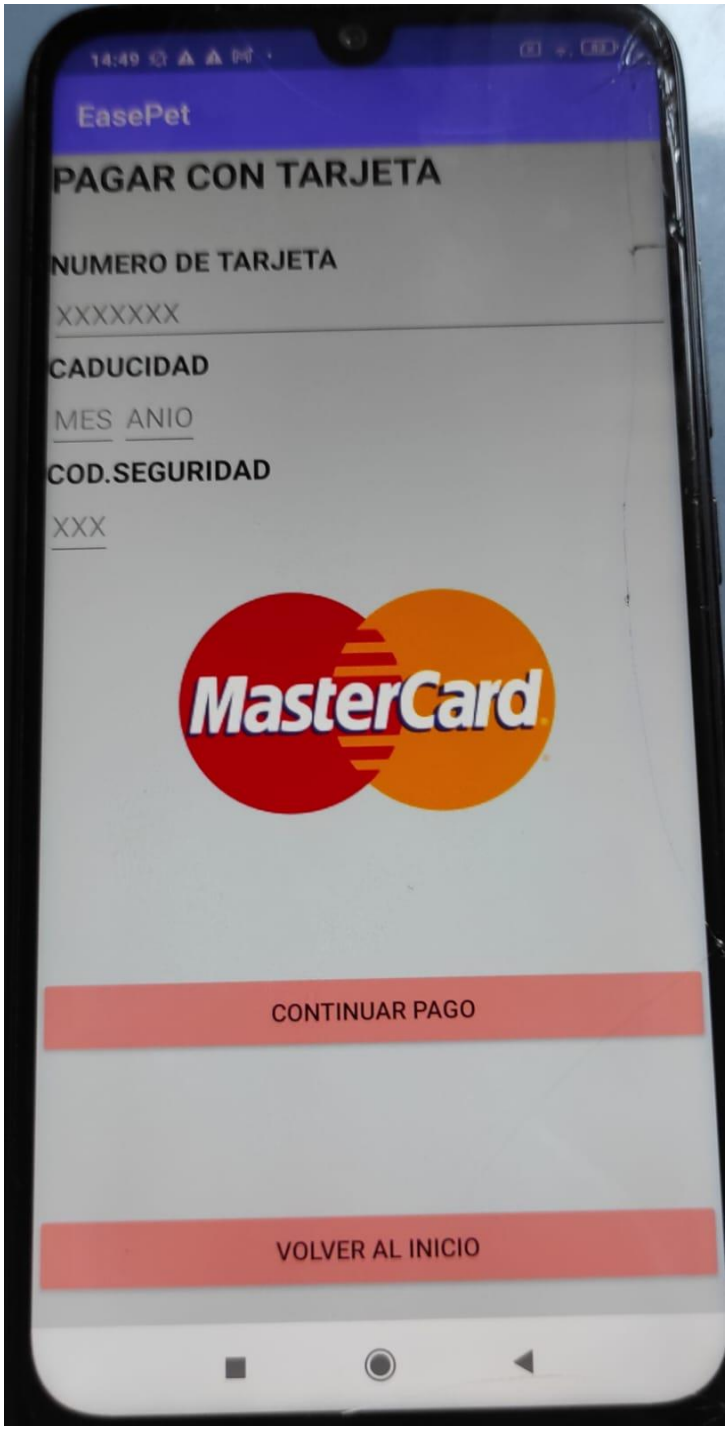
En esta ventana se creará la principal función de la aplicación que es la generación del código QR. En este código se almacenan 3 datos. El nombre del dueño, el teléfono de su clínica veterinaria de referencia y el teléfono del veterinario con el que tiene convenio Ease Pet. De esta manera protegemos los datos del dueño, como su DNI, dirección o teléfono personal.

Ventana elección del collar

En esta ventana el usuario tendrá hasta 6 tipos de collares distintos con los que se le implantará el correspondiente código QR generado. Una vez completado la elección tendrá que continuar a la pasarela de pago.

Ventana pasarela de pago

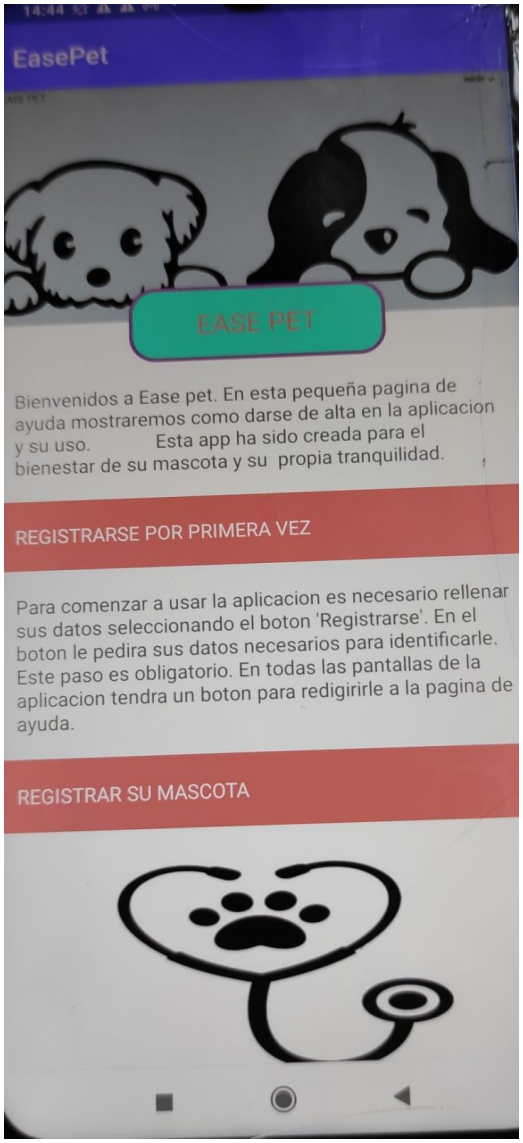
En esta ventana el usuario introducirá los datos personales para completar la compra del collar. Si algún campo no está completo, le saltará un aviso mediante una ventana emergente que le obligará a rellenar los datos correctamente antes de volver a la pantalla de inicio.

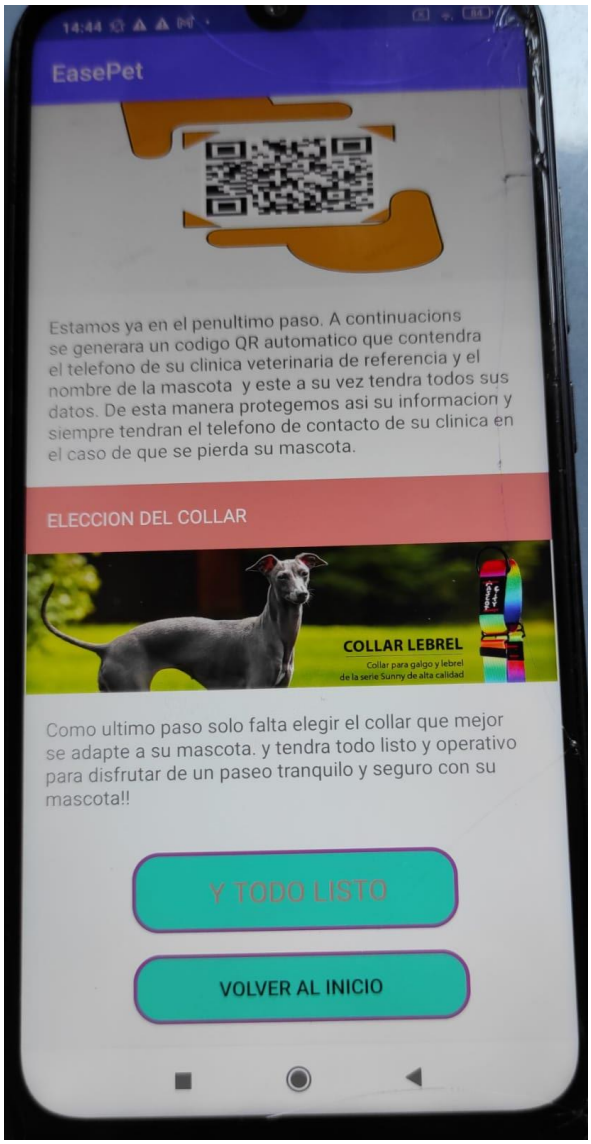


Ventana de Ayuda

Esta ventana está creada y diseñada para aquellos usuarios que se den de alta por primera vez o simplemente que les surja alguna duda respecto a los pasos a seguir en la aplicación.

Contiene una pequeña descripción de qué se hará en cada ventana.





2.2 Requisitos del sistema

Para poder instalar esta aplicación necesitaremos un dispositivo móvil con sistema operativo Android superior a Android 5.0 y 20 MB de espacio en memoria. También podremos instalarlo en los emuladores de smartphones modelo Pixel 2 XL API 21 de Android Studio.

2.3 Base de datos

En la aplicación Ease Pet se ha hecho uso de SQLite.

Para trabajar con SQLite es necesario crear una subclase SQLiteOpenHelper, la cual tiene la lógica para crear y actualizar una base de datos. Esta subclase necesita los siguientes tres métodos:

o Constructor de la base de datos

Este constructor llamado “MYDB”. Toma el contexto, el nombre de base de datos que elijamos poner, un cursor opcional el cual en las mayorías de las aplicaciones se puntúa con valor “NULL” y por último la versión de la base de datos. En el caso de esta aplicación hemos decidido ponerlo a 1 declarando previamente la variable más arriba.

```
public MyDB(Context context) { super(context, BDname,  factory: null, BDversion); }
```

```
public static final int BDversion=1;
```

- **Método onCreate()**

En este método se pasa el objeto SQLiteDataBase que necesita rellenar con las tablas y datos iniciales apropiados. En este caso se crean dos tablas distintas. Una para el cliente con sus campos DNI, nombre y dirección. Otra tabla para registrar los datos de la mascota con sus respectivos campos como nombre del perro, gato, hámster y loro.

```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL("Create table " + TABLA_CLIENTES+ "(" + "Dni text PRIMARY KEY," + "Nombre text ," + "Direccion text )
    db.execSQL("Create table " + TABLA_MASCOTAS+ "(" + "Nombreperro text PRIMARY KEY," +
            "Nombregato text ," + "Nombrehamster text , " + "Nombreloro text , " +"Tipo text)");
```

- **Método onUpgrade ()**

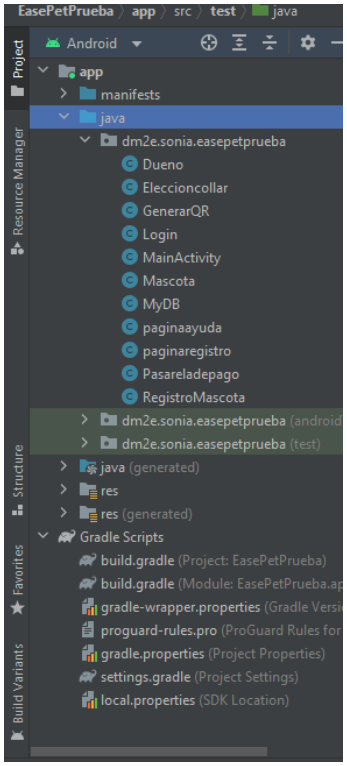
Dentro de este método se realiza la edición de los datos de las tablas hasta encajar con la estructura de la nueva versión. En el caso de esta aplicación ejecutamos 2 Query distintas. Una para eliminar la tabla clientes en el caso de existir otra con el mismo nombre y otra para eliminar la tabla mascota si hubiera otra con el mismo nombre.

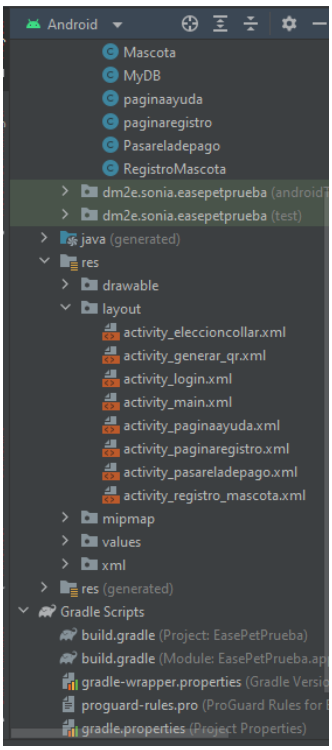
```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
  db.execSQL("DROP TABLE IF EXISTS "+ TABLA_CLIENTES);
    db.execSQL("DROP TABLE IF EXISTS "+ TABLA_MASCOTAS);
    onCreate(db);
```

2.4 Estructura de la App

La estructura de la aplicación de manera externa es la siguiente. Dentro de cada carpeta existen múltiples divisiones, pero de manera más genérica dejaremos esta como primer vistazo.

En la foto de la izquierda se visualiza la parte de la lógica de la aplicación. En la foto de la derechas se visualiza la parte más visual, que será cada una de las pantallas que veremos.

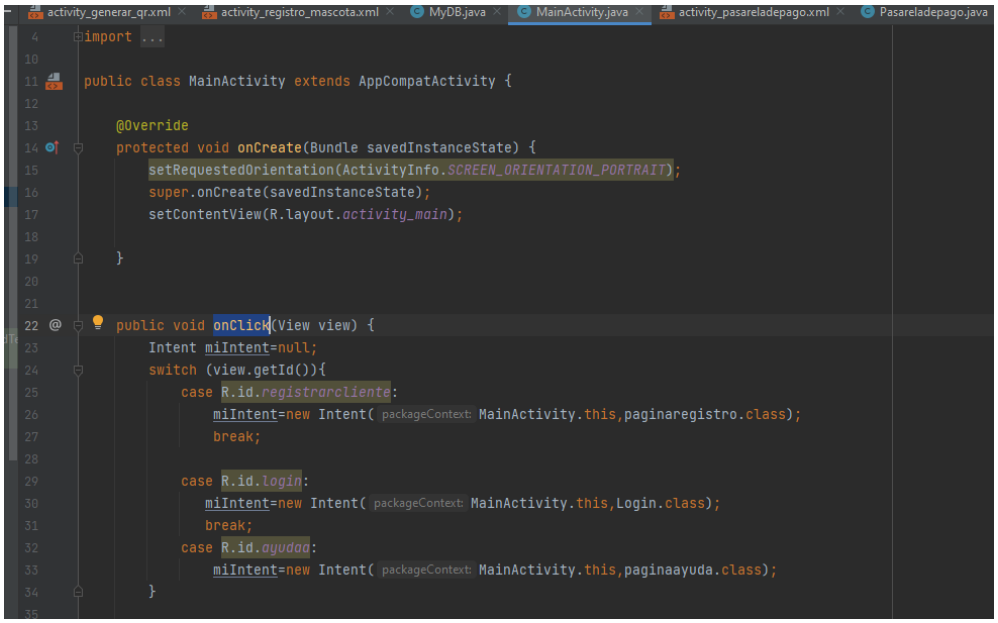




Activities

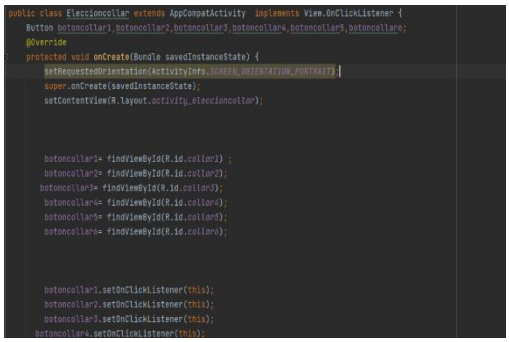
- **MainActivity**

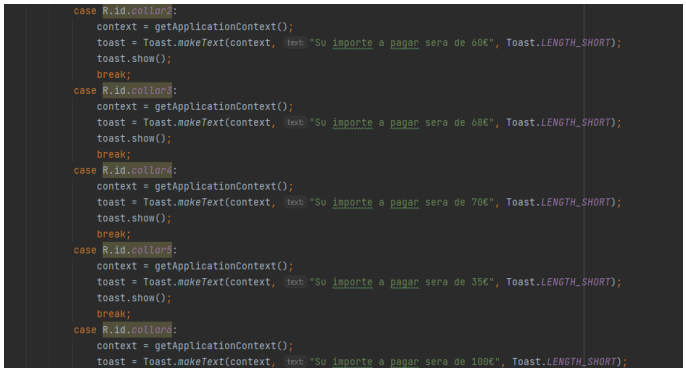
Contiene el código de la actividad principal. En ella tenemos 3 botones que con el método onClick elegimos que ventana queremos que se nos abra en el menú principal.



- **EleccionCollar**

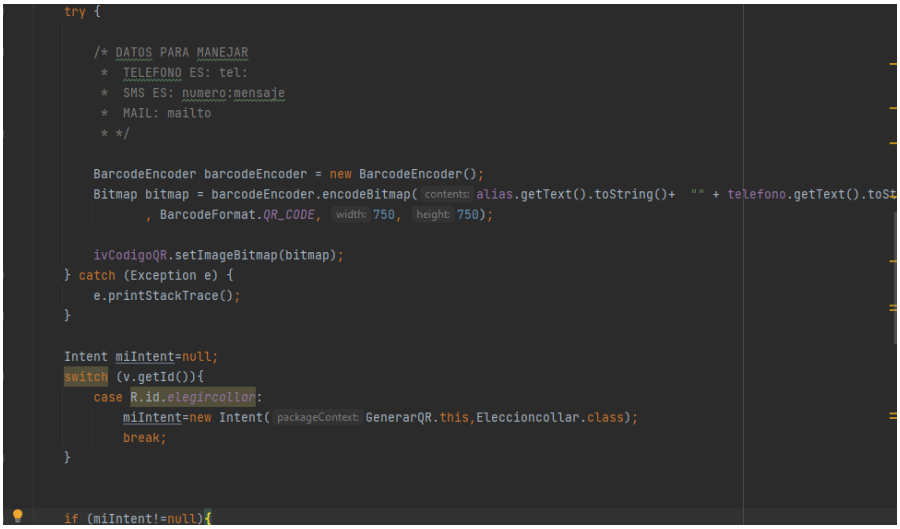
En esta actividad tendemos la lógica de los collares elegidos. Cada collar lleva asignado un botón específico con el cual podemos identificar qué collar ha elegido el usuario para comprar. En función de ello y mediante el switch tendremos diferentes mensajes que saldrán por pantalla. Como última información presente en esta actividad estará el Intent, que nos llevará a una nueva ventana, que será la pasarela de pago.





GenerarQR

En esta actividad tendremos la lógica para generar el código QR. Es la parte más importante de la aplicación ya que gracias a esta clase se hará posible este código. Se ha hecho uso de librerías externas llamadas “ZXING”. Con estas usaremos BarcodeEncoder y Bitmap para poder guardar los datos del nombre y teléfono de la clínica veterinaria.



```
try {

    /* DATOS PARA MANEJAR
     *  TELEFONO ES: tel:
     *  SMS ES: numero:mensaje
     *  MAIL: mailto
     * */

    BarcodeEncoder barcodeEncoder = new BarcodeEncoder();
    Bitmap bitmap = barcodeEncoder.encodeBitmap( contents: alias.getText().toString()+  "" + telefono.getText().toSt
           , BarcodeFormat.QR_CODE,  width: 750,  height: 750);

    ivCodigoQR.setImageBitmap(bitmap);
} catch (Exception e) {
    e.printStackTrace();
}

Intent miIntent=null;
switch (v.getId()){
    case R.id.elegircollar:
        miIntent=new Intent( packageContext: GenerarQR.this,Elecciоncollar.class);
        break;
}


if (miIntent!=null){
```

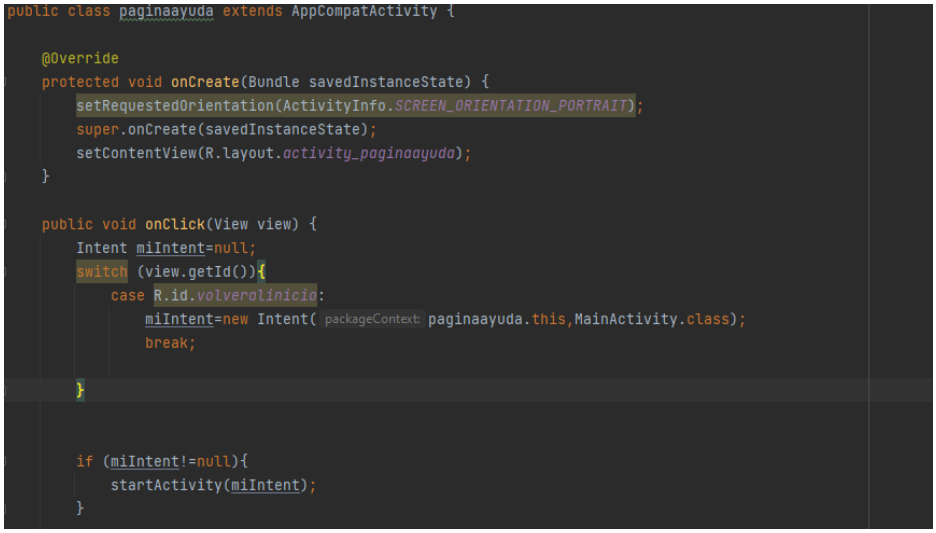
- **Login**

En esta actividad se controlará que los datos introducidos por el usuario existan. Para ello se hace una comprobación a la base de datos para ver si realmente el usuario está dado de alta o no.

```
public void login(View view)
{
    MyDB myDb=new MyDB( context: this);
    SQLiteDatabase db=myDb.getReadableDatabase();

    String[] columns={"Dni","Nombre"};
    EditText dni=findViewById(R.id.dnilogin);
    EditText nombre=findViewById(R.id.nombrelogin);
    String[] cValues={dni.getText().toString(),nombre.getText().toString()};
    Cursor cursor=db.query( table: "clientes",columns, selection: "Dni = ? AND Nombre = ?",cValues, groupBy: null, having: null, orderBy: null);
    if(cursor!=null)
    {
        if(cursor.moveToFirst()){
            Intent intent=new Intent( packageContext: this, RegistroMascota.class);
            intent.putExtra( name: "Dni",dni.getText().toString());

            startActivity(intent);
```

- **Paginaayuda**

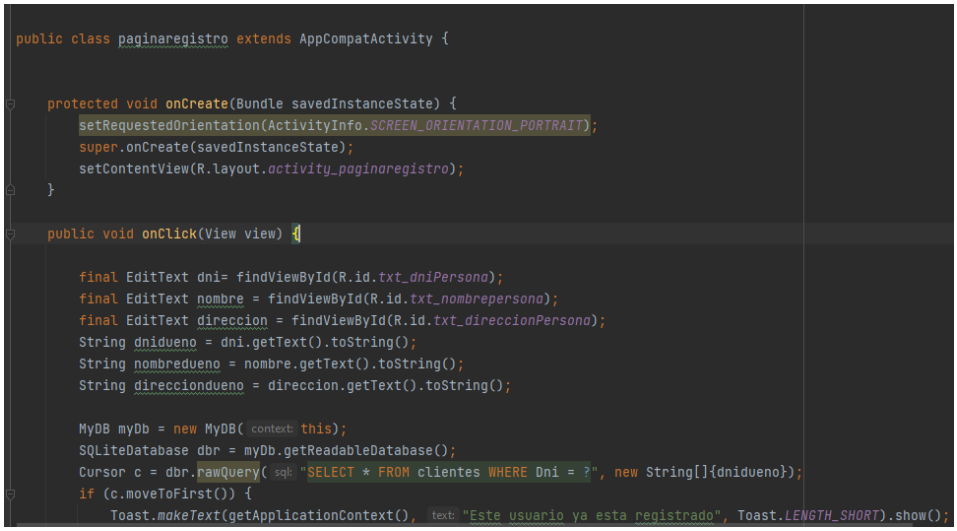
En esta actividad se implementa simplemente el botón “ volver al inicio” después de que el usuario lea los pasos para aprender a usar la aplicación.



```
public class paginaayuda extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_paginaayuda);
    }

    public void onClick(View view) {
        Intent miIntent=null;
        switch (view.getId()){
            case R.id.volveralinicio:
                miIntent=new Intent( packageContext: paginaayuda.this,MainActivity.class);
                break;

        }


        if (miIntent!=null){
            startActivity(miIntent);
        }
```

- **Paginaregistro**

En esta actividad el usuario se dará de alta introduciendo por primera vez sus datos , como el DNI, nombre y dirección. Posteriormente se hará un guardado en la tabla de “clientes”. Si en algún caso el usuario se registra y existe otro usuario con el mismo DNI saltará una ventana emergente informado que el cliente con ese DNI ya existe y te obligará a cambiar de DNI.

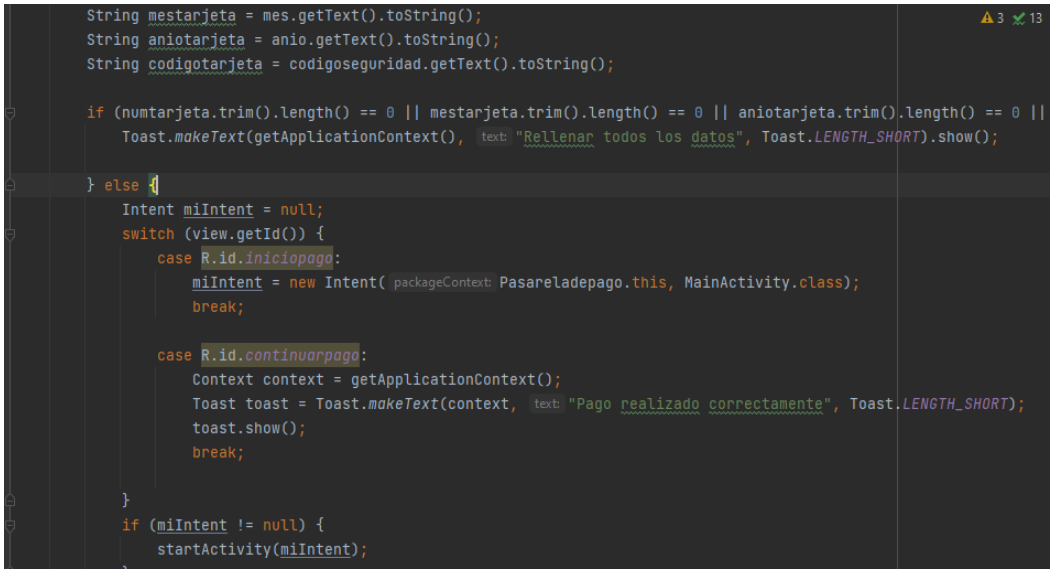


```
public class paginaregistro extends AppCompatActivity {


    protected void onCreate(Bundle savedInstanceState) {
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_paginaregistro);
    }

    public void onClick(View view) {

        final EditText dni= findViewById(R.id.txt_dniPersona);
        final EditText nombre = findViewById(R.id.txt_nombrepersona);
        final EditText direccion = findViewById(R.id.txt_direccionPersona);
        String dnidueno = dni.getText().toString();
        String nombredueno = nombre.getText().toString();
        String direcciondueno = direccion.getText().toString();

        MyDB myDb = new MyDB( context: this);
        SQLiteDatabase dbr = myDb.getReadableDatabase();
        Cursor c = dbr.rawQuery( sql: "SELECT * FROM clientes WHERE Dni = ?", new String[]{dnidueno});
        if (c.moveToFirst()) {
            Toast.makeText(getApplicationContext(),  text: "Este usuario ya esta registrado", Toast.LENGTH_SHORT).show();
```

- **Pasareladepago**

En esta actividad se hará un registro y control en la pasarela de pago. Se ha implementado en el código un control de tal manera que, si el usuario no rellena todos los campos, le saltará una ventana emergente informándole que es necesario completar todos los campos.

Finalmente tenemos un botón que, en el caso de ser seleccionado, redirigirá al usuario a la ventana de inicio.



```
String mestarjeta = mes.getText().toString();
String aniotarjeta = anio.getText().toString();
String codigotarjeta = codigoseguridad.getText().toString();

if (numtarjeta.trim().length() == 0 || mestarjeta.trim().length() == 0 || aniotarjeta.trim().length() == 0 ||
    Toast.makeText(getApplicationContext(), text: "Rellenar todos los datos", Toast.LENGTH_SHORT).show();

} else {
    Intent miIntent = null;
    switch (view.getId()) {
        case R.id.iniciopago:
            miIntent = new Intent( packageContext: Pasareladepago.this, MainActivity.class);
            break;

        case R.id.continuarpago:
            Context context = getApplicationContext();
            Toast toast = Toast.makeText(context, text: "Pago realizado correctamente", Toast.LENGTH_SHORT);
            toast.show();
            break;

    }
    if (miIntent != null) {
        startActivity(miIntent);
```

- **Registromascota**

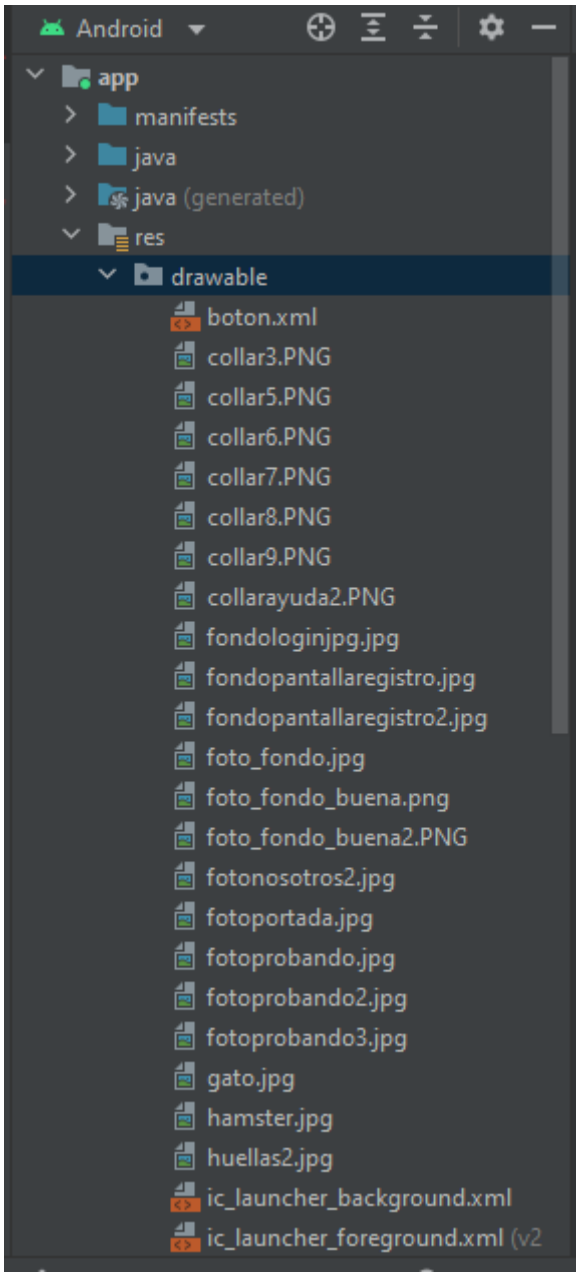
En esta actividad guardamos los nombres de las mascotas. Hay implementado un control necesario para que el usuario no pase a la siguiente ventana sin haber seleccionado al menos una mascota.

```
ch4 = (CheckBox) findViewById(R.id.loro);


if (ch1.isChecked() == true) {
    tipomascota = "perro";

}

if (ch2.isChecked() == true) {
    tipomascota = "gato";

}
if (ch3.isChecked() == true) {
    tipomascota = "hasmter";
}
if (ch4.isChecked() == true) {
    tipomascota = "loro";
}
if (tipomascota.trim().length()==0) {
    Context context = getApplicationContext();
    Toast toast = Toast.makeText(context, text: "No has seleccionado ninguna mascota. Debe seleccionar al menos un
    toast.show();

} else {

    final EditText nombreperroedit = findViewById(R.id.nombreperro);
```

Carpeta RES

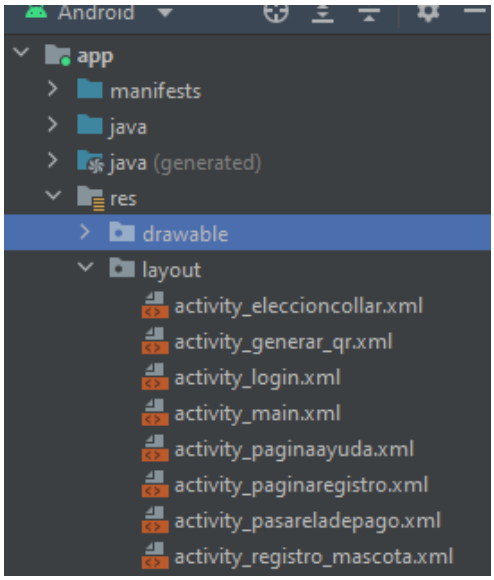
Drawable

Dentro de la carpera res tenemos una subcarpeta llamada “Drawable” en la cual se almacena de forma local cada una de las fotos que han sido usadas en la aplicación. También hay un archivo llamado botón.xml en el cual se registra diferentes colores a lo largo de los botones de la aplicación.



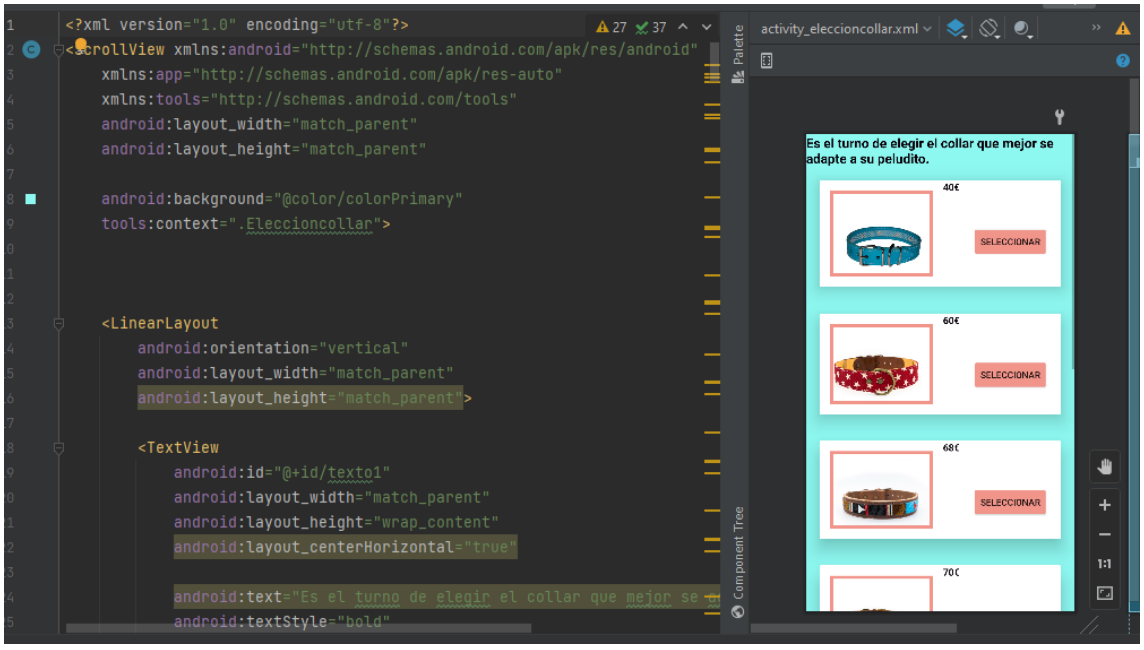
Layout

En esta subcarpeta dentro de res. Se encuentran las correspondientes ventanas de cada actividad. En ellas s eimplementan la organización de las imágenes, textos, botones. A continuación adjunto ejemplo de un Layout.



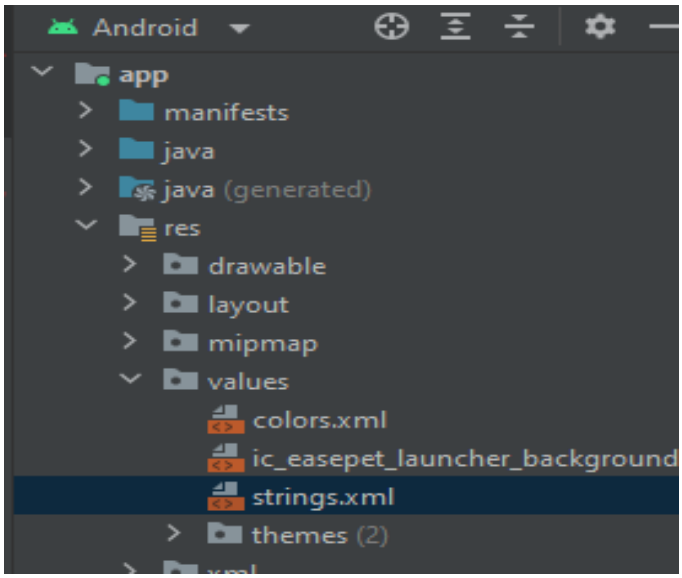
Activity_eleccioncollar

En este ejemplo se muestra en la parte de la izquierda el código necesario para meter cada dato en su correspondiente lugar. En la parte de la derecha nos muestra como ejemplo como lo veríamos en nuestro simulador o dispositivo móvi.



Values

En esta carpeta se almacenan más recursos utilizados en la aplicación como colores, las variables de tipo String usadas.




```
<resources>
    <string name="app_name">EasePet</string>
    <string name="saludo">¡Bienvenido a EasePet!</string>
    <string name="saludosegundo">Para acceder es necesario darse de alta como cliente.</string>
    <string name="registrarcliente">Registrar nuevo cliente</string>
    <string name="saludoaltacliente">Registrar datos</string>
    <string name="entrarcliente"> Entrar </string>
    <string name="registrarmascota">Registrar Mascota</string>
    <string name="preguntamascotatipo">¿Que tipo de mascotas tienes?</string>
    <string name="dudacliente">¿Necesitas ayuda?</string>
    <string name="bienvenidos">Bienvenidos a Ease pet. </string>
    <string name="login">Entrar </string>
    <string name="portadagr">Generacion QR</string>
    <string name="explicaciongr">¡Como ultimo paso nos toca generar el codigo QR! Rellena los campos necesarios para regis
    <string name="saludologin">Entrar Usuario </string>
    <string name="eleccioncollar">¡Muchas gracias por confiar en Ease Pet! Lo antes posible se lo enviaremos a su domicili
    <string name="preguntacollar">Es el turno de elegir el collar que mejor se adapte a su peludito.</string>
```

3. Planificación

3.1 Secuencia de tareas

- Presentación de las ideas, creación de los diagramas de clases necesarios para ubicarnos y saber cuantas tablas vamos a necesitar en nuestra base de datos, qué atributos son los mas acertados en cada tabla.

- Creación de la interfaz de la app , primeros bocetos , valorando posibles cambios y adaptaciones de la interfaz. Primeros pasos en la creación de la base de datos.

- Comprobamos la aplicación con datos de prueba. Medimos posibles errores, metemos datos ficticios en las tablas creadas previamente. Implementamos usuarios y perfiles.

- Implementación final de la interfaz .Optimización de la aplicación y pruebas unitarias.

Temporización de las taras del proyecto

Descripción	Semana	Fecha de tutoría
Inicio FCT	Semana 0	21 Marzo
Presentación de la idea principal de la aplicación. Creación del diagrama de clases con sus respectivas tablas. Creación del diseño de la base de datos	Semana 1	30 Marzo
Creación de la interfaz de la app , primeros bocetos , valorando posibles cambios y adaptaciones de la interfaz. Primeros pasos en la creación de la base de datos	Semana 2-3	20 de Abril
Comprobamos la aplicación con datos de prueba. Medimos posibles errores, metemos datos ficticios en las tablas creadas previamente. Implementamos usuarios y	Semana 4-5	11 de Mayo

4. Pruebas

Pruebas en el dispositivo físico.

Se hacen pruebas en el dispositivo físico Xiaomi redmi note 6 pro. Con esto comprobamos que la instalación se hace correctamente. Vemos posibles fallos o bugs desde la consola de Android studio. Se han observado casos de problemas del SDK con la versión. Esta prueba en dispositivo físico hace que se parezca más a la realidad a al hora de comercializar la aplicación.

Pruebas de accesibilidad

Las pruebas de accesibilidad se realizan al final del desarrollo de la aplicación móvil.

En estas pruebas observamos posibles alteraciones en la gama de colores en las imágenes, en el color tamaño o funcionalidad de los botones. Analizamos que podría suceder si giramos la pantalla del dispositivo o se hay algún tipo de deformidad al girarlo.

Pruebas de funcionamiento de software

Se realizarán pruebas de funcionamiento de la aplicación basadas en los casos de uso donde se comprueba que la aplicación funciona tal como se espera

5. Propuestas de mejora en un futuro

Realizando el proyecto se ha llegado a los objetivos que estaban marcados desde el principio, pero también se ha visto varias propuestas en las cuales se puede mejorar las aplicaciones desarrolladas.

- Implementar mayor control en el registro de mascotas. Puede suceder que un usuario no se de cuenta y registre 2 mascotas del mismo tipo y mismo nombre.
- En relación a la base de datos. Estaría muy interesante tener un servidor en la nube en donde se almacene los datos del cliente y el registro de sus mascotas.
- En la pasarela de pago hay dos botones. El de pago y el de volver al inicio. Se podría implementar un control mayor, por si acaso el usuario se equivoca del

collar seleccionado que le deje ir para atrás sin obligarle a meter los datos bancarios.

- Publicar la aplicación en Play store.
- Se podría implementar un acceso directo a un número de atención al cliente desde la misma aplicación.

6. Conclusión final

Ease pet se ha creado con la intención de ayudar a los dueños de mascotas. Con ello pretendemos que cuando llegue el momento de sacar a sus mascotas pierdan el miedo a perderlas y nunca más encontrarlas. Es cierto que existe a día de hoy el concepto del "chip" que se les pone sobre todo a las mascotas como perros o gatos. En las mascotas como un hámster, loro u otro tipo de mascotas no es tan común hacer el uso de este chip.

Con ello en Ease pet queremos incluir poco a poco diferentes mascotas, para así abarcar lo máximo posible y que cada tipo de mascota tenga la oportunidad de ser rastreado.

Con Ease pet se busca que de una manera más rápida y ágil se puedan poner en contacto con los dueños en el caso de que la mascota se pierda.

De esta manera se evita el tener que ir con la mascota a la clínica veterinaria más cercana o incluso como hemos recogido en un formulario, algunas personas optarían ppr llevarlo directamente a la policía. Con ello ahorramos tiempo a la hora de localizar al dueño y en conclusión ahorramos tiempo de sufrimiento al dueño.

Haber elegido un IDE como Android studio ha facilitado mucho el trabajo, porque en el 2º año escolar hemos dado bastante temario y bastantes ejemplos de aplicaciones que podemos desarrollar con Android studio.

Para finalizar, este trabajo en el ámbito personal me ha ayudado a repasar muchos conceptos que tenía aún pendientes de matizar y ha reafirmar que la programación móvil no es un camino sencillo, pero sí es un camino gratificante y bonito cuando puedes tener en directo en tu dispositivo móvil físico la obra maestra que estás creando.

7. Bibliografía

https://talently.tech/blog/que-es-android-studio/

https://developer.android.com/studio/intro?hl=es-419

https://www.w3schools.blog/android-tutorial

PDF y documentos

https://developer.android.com/guide?hl=es-419