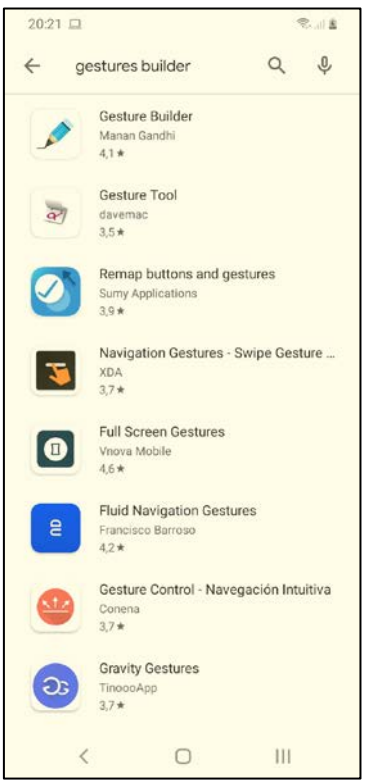
PRACTICA 9: GESTURES

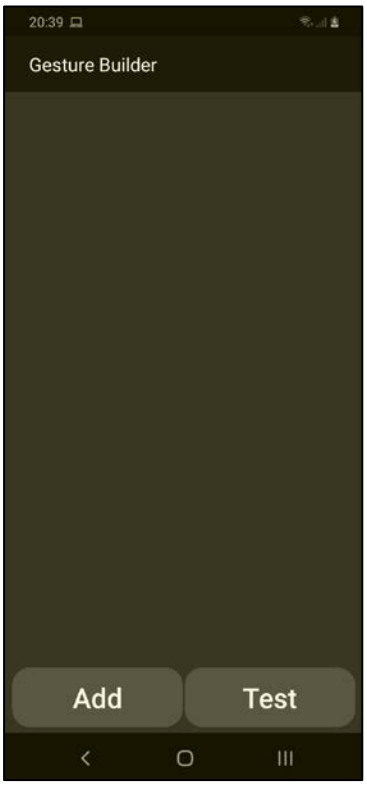
Parte I: Creación de una librería de gestures

Paso 1. Abre un emulador con nivel de API24 y con memoria externa.

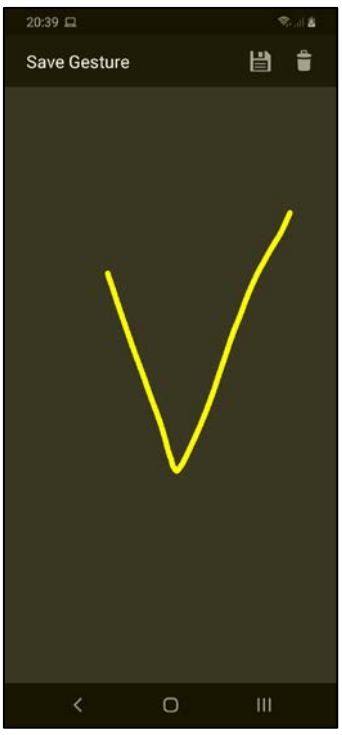
Nos bajaremos una aplicación al móvil y lo haremos con ella: Gesture Builder

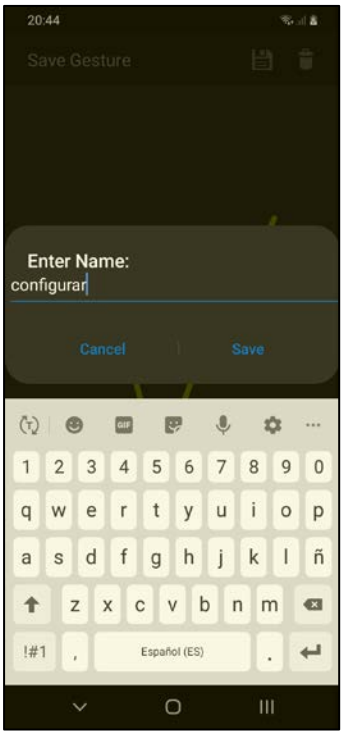
Paso 2. En el menú de programas busca el siguiente icono y abre la aplicación Gestures Builder.





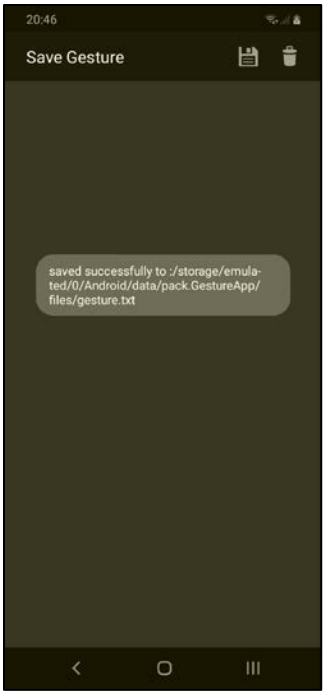
Paso 3. Para añadir una nueva *gesture* a tu librería pulsa el botón “Add gesture” y realiza un trazado sobre la pantalla (por ejemplo un visto bueno) a continuación introduce un nombre asociado a esta *gesture* (por ejemplo “configurar”). El resultado se muestra a continuación:





Paso 4. Si no te gusta como ha quedado el trazado, no tienes más que realizar uno nuevo. El anterior será descartado.

Paso 5. Pulsa el botón "*Done*" para aceptar la *gesture*. Tras pulsar este botón aparecerá un cuadro de texto indicándote donde se acaba de guardar el fichero con la librería de *gestures*.



***NOTA**: Si intentas crear con el emulador una gesture formada por varios trazos (por ejemplo el símbolo "X"), es posible que solo quede almacenado el último trazo. Para que ambos trazos sean reconocidos en la misma gesture, has de introducir el segundo justo a continuación del primero. Puede resultar algo difícil, pero tras un par de intentos lo conseguirás. No te preocupes, introducir una gesture de varios trazos en un dispositivo real no resulta tan complicado como en el emulador. Concretamente, el problema está en el valor FadeOffset que indica el tiempo máximo en milisegundos entre dos trazos de la misma gesture. Si al introducir dos trazos el tiempo entre ellos es mayor que FadeOffset, se considerará que se han introducido dos gestures diferentes. Por defecto, este valor es asignado a 420 milisegundos. El valor resulta adecuado con un dispositivo real, pero muy pequeño para el emulador. En el ejemplo descrito más adelante daremos un valor más alto a FadeOffset si queremos trabajar de forma más cómoda con el emulador.*

Paso 6. Trata de introducir las *gestures* mostradas en la siguiente captura. Para mejorar el porcentaje de reconocimientos correctos, puede ser interesante introducir varias veces la misma *gesture* con trazados alternativos. Esto se consigue dándole el mismo nombre a dos *gestures*.

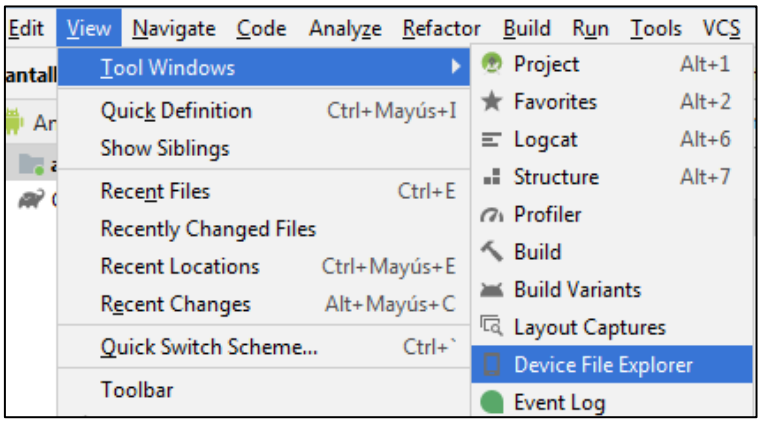
Paso 7. Cada vez que una nueva *gesture* es introducida aparece una ventana de texto que indica el fichero donde está almacenada nuestra librería. En este caso será:

"/storage/emulated/0/Android/data/pack.GestureApp/files/gestures.txt".

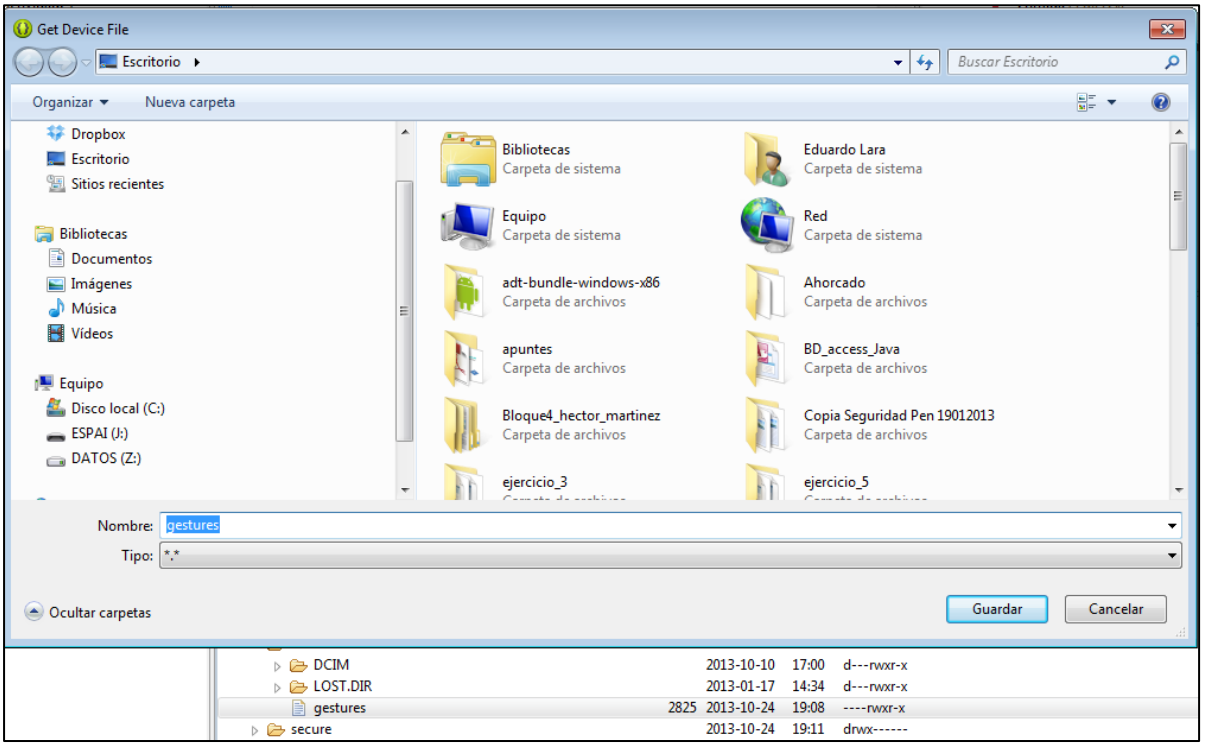
Importante: Dependiendo de la aplicación que hayas escogido para hacer los gestos a falta de Gesture Builder de Manam Gandhi estará en un lugar o en otro. Con la app Gesture Builder Tool de Emedp la dirección será:

"/storage/emulated/0/Android/data/migueldp.runeforge/files/gestures.txt".

Utiliza la vista *File Explorer* de Eclipse para localizar este fichero.

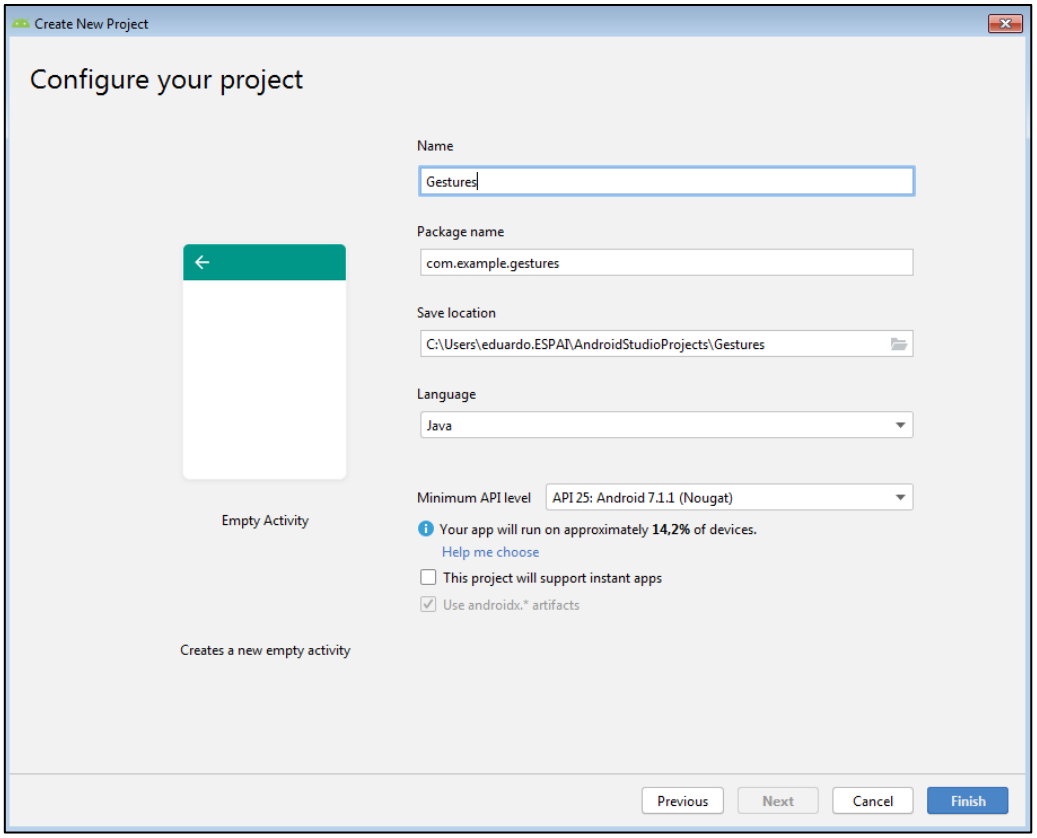


Paso 8. Selecciónalo y pulsa el botón "guardar" para guardarlo en tu ordenador.

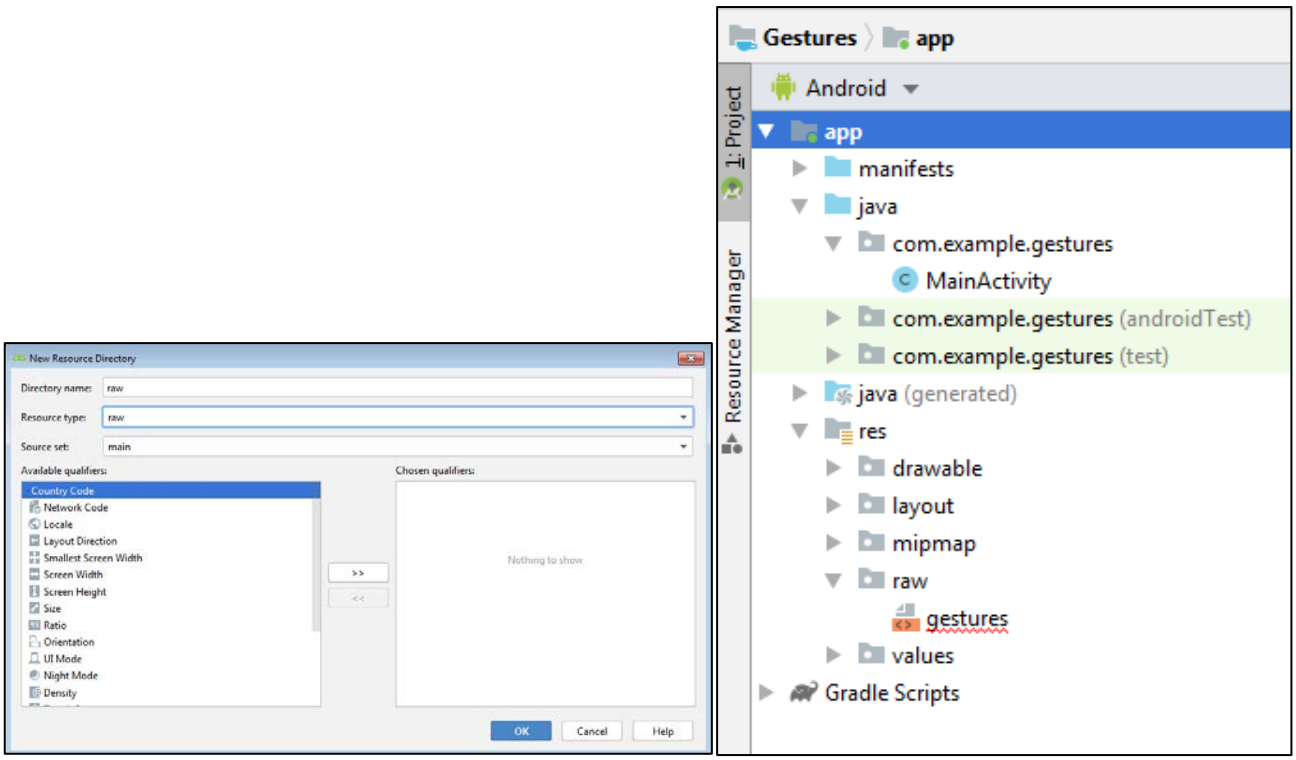


Parte II: Añadiendo la librería de gestures a nuestra aplicación

Paso 1. Para utilizar la librería que acabas de guardar, crea un nuevo proyecto de nombre GESTURES, con los siguientes datos:



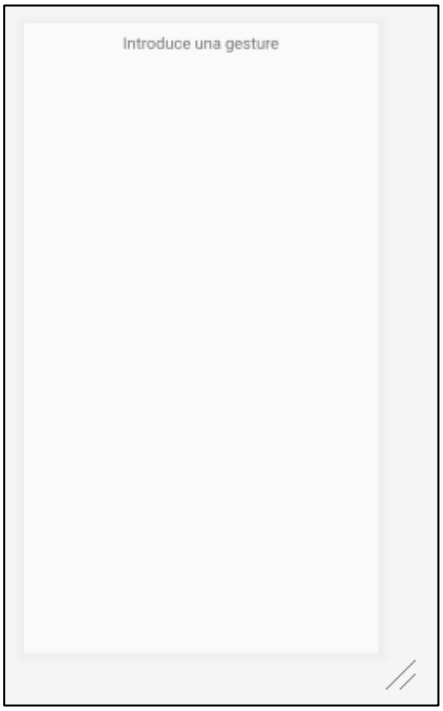
Paso 2. El siguiente paso va a consistir en crear la carpeta res/raw en el proyecto y copiar en ella el fichero que has guardado (gestures) en el ejercicio anterior.



Paso 3. Reemplaza res/layout/activity_main.xml por el siguiente código:

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:text="Introduce una gesture"
        android:textSize="8pt"
        android:layout_margin="10dip"/>

    <TextView
        android:id="@+id/salida"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"/>

    <android.gesture.GestureOverlayView
        android:id="@+id/gestures"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:gestureStrokeType="multiple"
        android:fadeOffset="800"/>
</LinearLayout>
```



El *layout* anterior está formado por un LinearLayot que contiene: un TextView con un título, un TextView para mostrar la salida del programa y un GestureOverlayView que será utilizado para introducir los *gestures*. En esta última etiqueta el parámetro gestureStrokeType indica que permitimos *gestures* formados por varios trazos. El parámetro fadeOffset ha sido explicado en el apartado anterior.

Paso 4. Reemplaza el código de la actividad por:

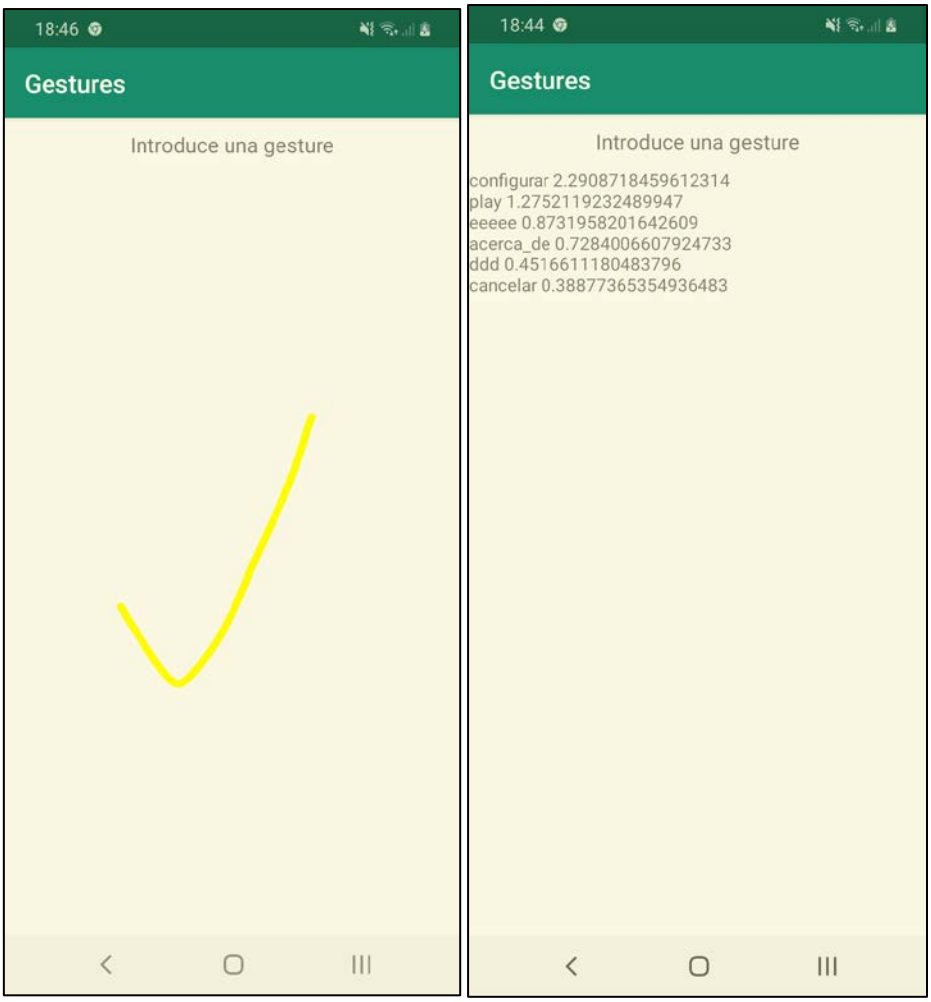
MainActivity.java

```
package com.example.gestures;

import ...

public class MainActivity  extends AppCompatActivity implements GestureOverlayView.OnGesturePerformedListener {

    private GestureLibrary libreria;
    private TextView salida;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        libreria = GestureLibraries.fromRawResource( context: this, R.raw.gestures);
        if (!libreria.load()) {
            finish();
        }
        GestureOverlayView gesturesView = (GestureOverlayView) findViewById(R.id.gestures);
        gesturesView.addOnGesturePerformedListener(this);
        salida = (TextView) findViewById(R.id.salida);
    }

    public void onGesturePerformed(GestureOverlayView ov, Gesture gesture) {
        ArrayList<Prediction> predictions = libreria.recognize(gesture);
        salida.setText("");
        for (Prediction prediction : predictions){
            salida.append(prediction.name+" " + prediction.score+"\n");
        }
    }
}
```

En este código se comienza declarando dos campos de la clase librería contendrá la librería de *gestures* creada en el ejercicio anterior y salida que corresponde al TextView donde escribiremos los resultados. En el constructor, tras realizar las operaciones habituales, se carga la librería de *gestures* desde los recursos y en caso de no ser cargada finaliza la aplicación. A continuación, asocia el GestureOverlayView de main.xml a el objeto gestureView y se indica que nuestra clase será el escuchador de este elemento. Finalmente se asocia el TextView donde queremos sacar la salida al objeto salida.

El método onGesturePerformed se introduce para implementar la interfaz OnGesturePerformedListener. Este método tiene dos parámetros, el GestureOverlayView donde se ha introducido el *gesture* y el objeto Gesture que ha sido introducido. El primer paso consiste en reconocer el *gesture* comparándolo con la lista de nuestra librería. El resultado es una lista ordenada de Predictions con las *gestures* que considera más parecidas a la introducida. Tras borrar salida, se recorre todos los elementos de esta lista mostrando el nombre del *gesture* (prediction.name) y la puntuación de reconocimiento (prediction.score). Resulta complicado fijar un umbral, pero una puntuación inferior a 1 se suele considerar demasiado baja para tenerla en cuenta como predicción.

Paso 5. Ejecuta la aplicación y estudia las puntuaciones obtenidas.


18:46
Gestures
Introduce una gesture
18:44
Gestures
Introduce una gesture
configurar 2.2908718459612314
play 1.2752119232489947
eeeee 0.8731958201642609
acerca_de 0.7284006607924733
ddd 0.4516611180483796
cancelar 0.38877365354936483