

A-YEN



Di Qi Chen Dong

CFGs en Desarrollo de Aplicaciones Multiplataforma

<https://github.com/diqichendong/a-yen-app>

Índice

- [1. Requisitos hardware y software](#)
- [2. Guía de instalación del entorno de desarrollo](#)
- [3. Puesta en marcha del entorno de desarrollo y la aplicación](#)
- [4. Explicación del funcionamiento de la aplicación](#)
 - [4.1. Productos](#)
 - [4.2. TPV \(Ventas\)](#)
 - [4.3. Lista de la compra \(Compra\)](#)
 - [4.4. Registros](#)
- [5. Explicación del código más complejo o interesante](#)
 - [5.1. Hacer fotos](#)
 - [5.2. Obtener registros de compras / ventas](#)
 - [5.3. Guardar un producto](#)
- [6. Mayores dificultades encontradas](#)

1. Requisitos hardware y software

Los requisitos para el desarrollo del proyecto son:

- El IDE Android Studio para el desarrollo de aplicaciones Android.
- Cuenta de Google para los servicios de Firebase.
- *(Opcional)** Teléfono móvil Android 8 o superior para la ejecución de la aplicación.

**Opcional: Se puede crear un dispositivo virtual en Android Studio para ejecutar las aplicaciones.*

Los requisitos para únicamente la ejecución del proyecto son:

- Teléfono móvil Android 8 o superior.
- O el IDE Android Studio para poder utilizar el dispositivo virtual.

2. Guía de instalación del entorno de desarrollo

Para la instalación de Android Studio hay que ir a su página web:

<https://developer.android.com/studio>

descargar el instalador y seguir los pasos de instalación. Es una instalación clásica como la de cualquier otro programa.

Para ver los requisitos del sistema o tener una guía de instalación más extensa, recomiendo visitar el siguiente enlace:

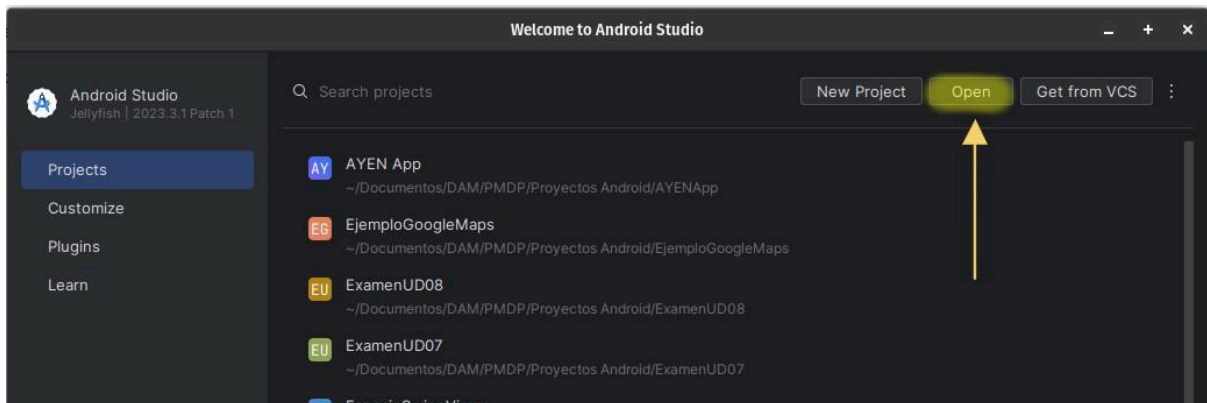
<https://developer.android.com/codelabs/basic-android-kotlin-compose-install-android-studio?hl=es-419#0>

3. Puesta en marcha del entorno de desarrollo y la aplicación

Para poner en marcha el entorno de desarrollo, lo primero que hay que hacer es descargar el proyecto o clonarlo del repositorio de GitHub:

<https://github.com/diqichendong/a-yen-app>

Una vez ya se tiene el proyecto en local, iniciamos Android Studio y lo abrimos.



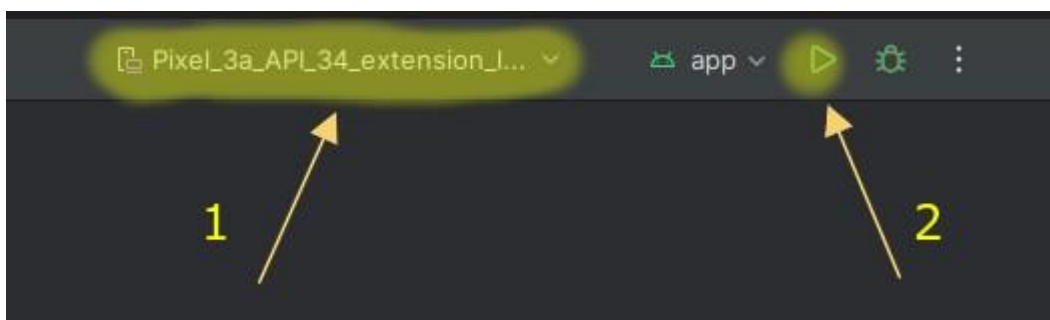
Para poner en marcha la aplicación, se puede hacer de dos formas distintas: ejecutándola desde Android Studio o instalando el .apk de la aplicación.

Para ejecutar la aplicación desde el IDE con el teléfono móvil físico:

- Activar las opciones de desarrollador del móvil (cada fabricante es distinto).
- En las opciones de desarrollador, permitir instalar aplicaciones mediante adb y permitir la depuración por USB.
- A continuación, conecta el móvil al ordenador y permite la transferencia de datos.
- Por último, en la parte superior de Android Studio con el proyecto abierto, selecciona el dispositivo y hacer click en ejecutar.

Para ejecutar la aplicación desde el IDE con un dispositivo virtual:

- En la parte superior de Android Studio con el proyecto abierto, selecciona el dispositivo virtual.
- Hacer click en ejecutar



Para instalar la aplicación mediante el .apk:

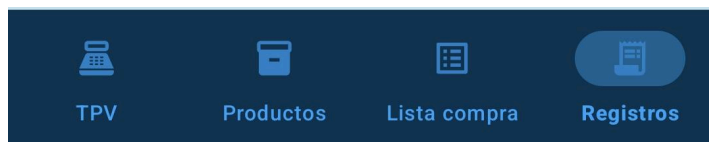
- Descargar el .apk de la raíz del proyecto del repositorio de GitHub.
- Transferir el archivo al móvil, si se ha descargado en el ordenador.
- Al abrir el .apk en el móvil ya empieza la instalación.

<https://github.com/diqichendong/a-yen-app>

4. Explicación del funcionamiento de la aplicación

A-YEN es una aplicación de gestión de un comercio inspirada en la tienda de alimentación de mi familia, Alimentación Yen.

La aplicación consta de 4 vistas principales, las cuales se puede navegar a ellas fácilmente a través del menú inferior.




4.1. Productos



El núcleo de la aplicación, la vista de los productos es una lista con todos los datos de los productos y su respectiva imagen.

Esta lista se puede filtrar por nombre y/o código de barras, permitiendo también escanear un código de barras que es aplicado al filtro directamente.


← Guardar producto



Hacer foto

Subir foto

Código: 5449000000996



Nombre: Coca-Cola 33cl

Precio: 1.20 €

Coste: 0.80 €

Stock: 98


Umbral de compra: 20

GUARDAR

CANCELAR

Cada producto en la lista contiene un par de botones, una para eliminar el producto previa confirmación y el otro para modificar el producto en cuestión llevándote a otra pantalla para su modificación.


← Guardar producto



Hacer foto

Subir foto

Código: Introduce el código



Nombre: Introduce el nombre


Precio: 9.99 €

Coste: 9.99 €

Stock: 123

Umbral de compra: 20

Añadir a compras



Por último, la vista también tiene un botón que te envía a una pantalla similar a la de modificación para que puedas crear un nuevo producto. A diferencia de la pantalla de modificación, esta te permite activar un switch para que se cree una compra en el stock y al coste del nuevo producto.

4.2. TPV (Ventas)

La pantalla del TPV es una clásica caja registradora.

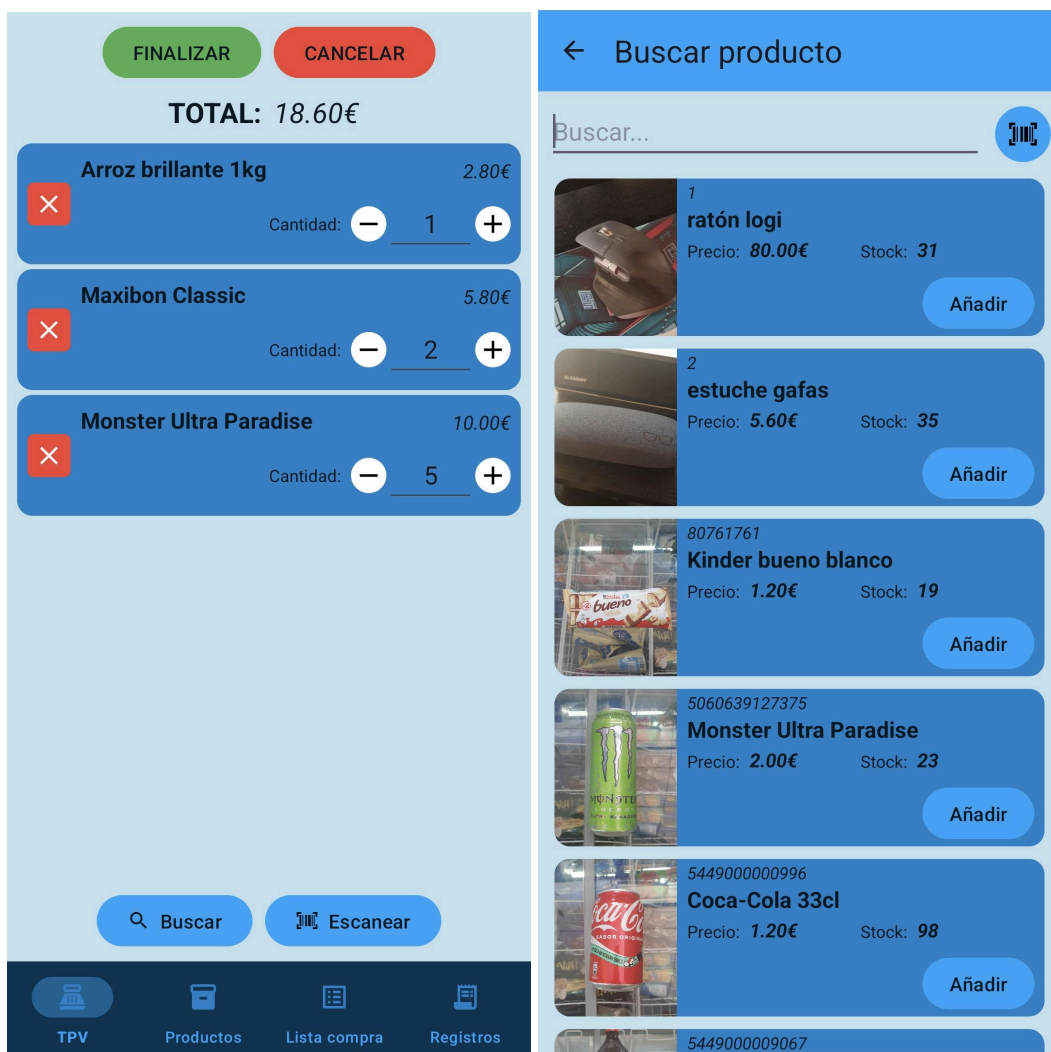
En esta vista tienes dos opciones de añadir productos a una venta mediante dos botones que están situados en la zona inferior.

El primer método es buscando el producto ya sea por nombre o por código en un buscador y el otro es directamente escaneando el código de barras del producto.

Una vez añadido los productos a la venta puedes cambiar la cantidad manualmente y si se añade de nuevo el producto, se aumenta la cantidad en una unidad.

Para eliminar los productos de la lista de la venta, puedes hacerlo mediante un botón situado a la izquierda o estableciendo la cantidad a cero.

Por último, en el TPV existen dos acciones finales para la venta, cancelar la venta que lo único que haría es reiniciar todo o finalizar la venta la cual efectuaría la venta.



4.3. Lista de la compra (Compra)

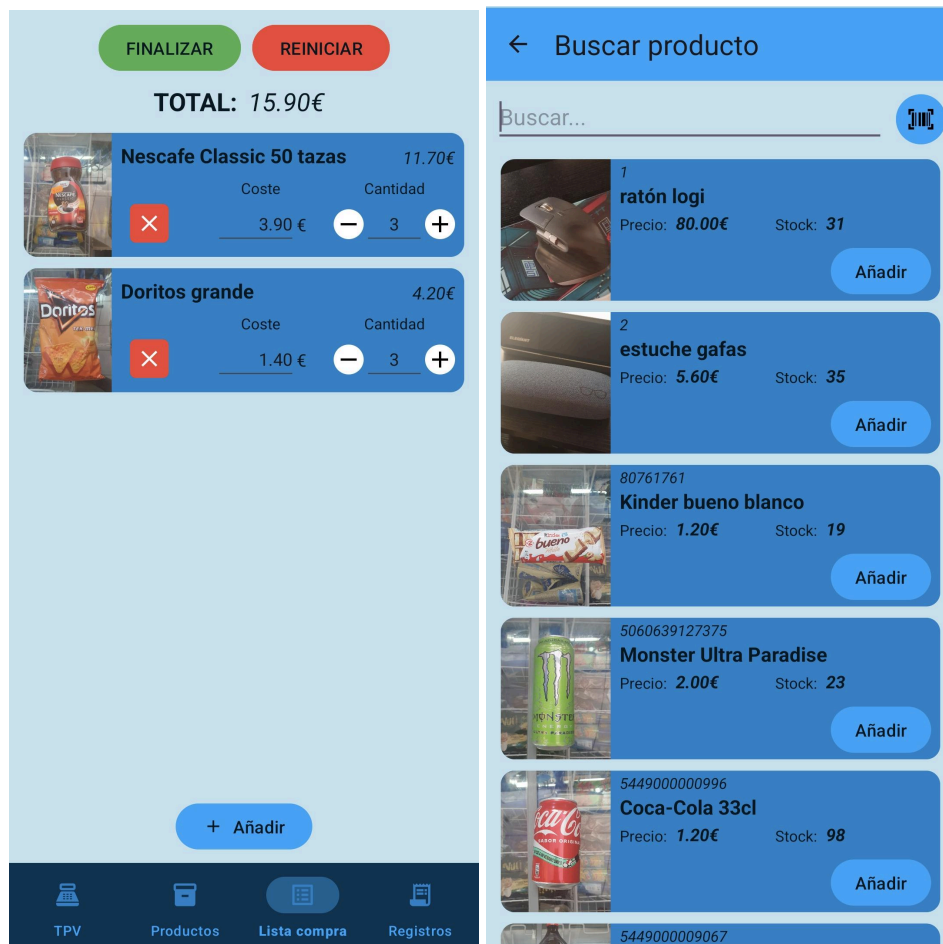
La vista de la lista de la compra genera una lista automáticamente de los productos que tengan un stock por debajo de un umbral de compra que establecemos nosotros a los productos que creamos.

Podemos eliminar productos de la lista accionando el botón rojo situado al lado de las imágenes de los productos.

También podemos editar el nuevo coste que tiene el producto que vas a comprar y la cantidad que te vas a llevar.

Si se desea, también se pueden añadir productos a la lista de la compra aunque el stock no esté por debajo del umbral establecido mediante un botón en la parte inferior.

Por último, al igual que el TPV, tiene dos acciones finales, reiniciar la lista que crea de nuevo la lista con los productos con stock por debajo del umbral descartando la lista anterior, y finalizar compra que efectúa la compra descartando todos los productos con cantidad a cero.



4.4. Registros

Finalmente, tenemos la vista de los registros que se divide en dos pestañas: ventas y compras.

Ambas pestañas nos ofrecen introducir un rango de fechas para filtrar el historial de ventas o compras. Por defecto se mostrarán los registros del día actual.

Los registros de compras y ventas se mostrarán en una lista con un botón que nos enviarán a una pantalla para ver los detalles de esa compra o venta.

VentasCompras

InicioFin

27/05/202404/06/2024

30/05/2024 11:35:04

C-202405301135043660.00€

30/05/2024 14:12:21

C-20240530141221176.30€

01/06/2024 10:48:47

C-2024060110484711.40€

01/06/2024 11:04:50

C-2024060111045092.75€

01/06/2024 16:21:45

C-202406011621451569.00€

02/06/2024 11:19:32

C-20240602111932274.40€

02/06/2024 11:21:41

C-20240602112141498.40€

TPV

Productos


Lista compra

Registros


← Detalles

Código: V-20240602134411


Fecha: 02/06/2024 13:44:11




1x Jumpers pequeño0.50€




1x Arroz brillante 1kg2.80€




1x Nescafe Classic 50 tazas5.00€




1x Bollycao Pack 3ud1.99€



1x Macibon Classic2.90€



1x Coca-Cola 2L2.50€



1x Coca-Cola 33cl1.20€

TOTAL: 20.09€

Cerrar

5. Explicación del código más complejo o interesante

5.1. Hacer fotos

```
cameraActivityLauncher = activity.registerForActivityResult(  
    new ActivityResultContracts.StartActivityForResult(),  
    result -> {  
        if (result.getResultCode() == Activity.RESULT_OK) {  
            guardarProductoActivity.setImgProducto(fotoUri);  
        }  
    }  
);
```

Este es el launcher que lanza la pantalla para hacer fotos y si se devuelve algún resultado (se ha realizado una foto), la foto se muestra en la pantalla de guardar producto.

```
private File createImageFile() throws IOException {  
    String timeStamp = new SimpleDateFormat( pattern: "yyyyMMdd_HH:mm:ss", Locale.getDefault()).format(new Date());  
    String imageFileName = "JPEG_" + timeStamp + "_";  
    File storageDir = guardarProductoActivity.getExternalFilesDir(Environment.DIRECTORY_PICTURES);  
    return File.createTempFile(imageFileName, suffix: ".jpg", storageDir);  
}
```

Este método de crea un archivo temporal para poder guardar la foto que tomemos. Primero creamos un nombre para el archivo, luego seleccionamos el directorio donde se va a almacenar temporalmente y por último devolvemos el archivo temporal.

```
public void abrirCamara() {  
    Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);  
    cameraIntent.putExtra(MediaStore.EXTRA_SIZE_LIMIT, value: 500 * 500);  
    if (cameraIntent.resolveActivity(guardarProductoActivity.getPackageManager()) != null) {  
        File fotoFile = null;  
        try {  
            fotoFile = createImageFile();  
        } catch (IOException ex) {  
            Log.e( tag: "Camara", ex.getMessage());  
        }  
        if (fotoFile != null) {  
            fotoUri = FileProvider.getUriForFile(guardarProductoActivity, authority: "com.example.ayenapp.fileprovider", fotoFile);  
            cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, fotoUri);  
            cameraActivityLauncher.launch(cameraIntent);  
        }  
    }  
}
```

Por último tenemos este método, que básicamente preparamos el intent, que es una operación que nos enviará a la pantalla para hacer fotos. Si el intent es correcto, creamos el archivo temporal para la foto, creamos la URI que es lo que utilizaremos para ver la foto en pantalla y finalmente lanzamos la pantalla de la cámara.

5.2. Obtener registros de compras / ventas

La obtención de los registros tanto de compras como de ventas son iguales, sólo que apuntan a nodos diferentes. Tomaré como ejemplo aquí al método *de getCompras*.

```
public void getCompras(LocalDateTime min, LocalDateTime max) {  
    registroComprasFragment.setBarraCarga(View.VISIBLE);  
  
    DatabaseReference databaseRef = FirebaseDatabase.getInstance().getReference(NODO_BASE);  
  
    Query query = databaseRef.orderByChild(path: "fecha")  
        .startAt(Util.formatearFechaHora(min))  
        .endAt(Util.formatearFechaHora(max));  
  
    diqichendong  
    query.addListenerForSingleValueEvent(new ValueEventListener() {  
        2 usages  diqichendong  
        @Override  
        public void onDataChange(@NonNull DataSnapshot snapshot) {  
            List<Compra> compras = new ArrayList<>();  
  
            for (DataSnapshot ventaSnapshot : snapshot.getChildren()) {  
                Compra compra = ventaSnapshot.getValue(Compra.class);  
                compras.add(compra);  
            }  
  
            registroComprasFragment.setCompras(compras);  
            registroComprasFragment.setBarraCarga(View.GONE);  
        }  
  
        diqichendong  
        @Override  
        public void onCancelled(@NonNull DatabaseError error) {  
            Toast.makeText(registroComprasFragment.getContext(),  
                "No ha sido posible obtener las compras",  
                Toast.LENGTH_SHORT).show();  
        }  
    });  
}
```

En primer lugar, muestra por pantalla la barra de carga para avisar al usuario.

A continuación, se crea la referencia a la base de datos de donde queremos traernos los datos.

Luego, la parte interesante, se crea una consulta (query) a la cuál le indicamos que los datos que obtengamos lo ordenemos por el nodo de “fecha” de la base de datos y le aplicamos los filtros de fecha. La fecha en la base de datos está guardada con el formato “yyyy-MM-dd HH:mm:ss”, por lo tanto, el filtro se aplicará correctamente

tan solo indicando que la fecha mínima y máxima en ese formato en el *startAt* y *endAt*.

Por último, a la consulta que creamos, le añadimos un listener de eventos ya que los servicios de Firebase son asíncronos. Cuando el servicio de Firebase obtenga los resultados ya se creará la lista de registros, se mostrará en la vista de los registros y terminará la barra de carga.

5.3. Guardar un producto

```
public void guardarProducto(Producto producto, Uri uri) {
    guardarProductoActivity.setBarraCarga(View.VISIBLE);

    DatabaseReference databaseRef = FirebaseDatabase.getInstance().getReference(NODO_BASE);
    StorageReference storageRef = FirebaseStorage.getInstance().getReference();

    List<Task<Void>> tareas = new ArrayList<>();

    if (uri != null) { // Si hay cambio de foto
        tareas.add(storageRef.child(producto.getFoto()).putFile(uri).continueWithTask(task -> Tasks.forResult(null)));
        tareas.add(databaseRef.child(producto.getCodigo()).setValue(producto));
    } else { // No hay cambio de foto
        tareas.add(databaseRef.child(producto.getCodigo()).setValue(producto));
    }

    Tasks.whenAll(tareas).addOnSuccessListener(unused -> {
        Toast.makeText(guardarProductoActivity, "Producto guardado correctamente", Toast.LENGTH_SHORT).show();
        guardarProductoActivity.finish();
    }).addOnFailureListener(e -> {
        Toast.makeText(guardarProductoActivity, "No se ha podido guardar el producto", Toast.LENGTH_SHORT).show();
    }).addOnCompleteListener(task -> {
        guardarProductoActivity.setBarraCarga(View.GONE);
    });
}
```

En primer lugar, mostraremos la barra de carga al usuario.

A continuación, creamos las referencias tanto de la base de datos como la del almacén de ficheros para las fotos.

Luego, creamos una lista de tareas para guardar las tareas de guardar en la base de datos y la de guardar la foto en el almacén de ficheros.

Si estamos editando un producto y cambiamos la foto o el producto es nuevo, será necesario guardar el producto en base de datos y su nueva imagen en el almacén de ficheros. Y si solo estamos editando un producto y no cambiamos su foto, sólo será necesario guardar el producto en base de datos.

Por último, le añadimos los listeners de eventos a la lista de tareas ya que los servicios de Firebase son asíncronos, y cuando terminen todas las tareas, se actuará en consecuencia.

6. Mayores dificultades encontradas

La mayor dificultad que me he encontrado desarrollando esta aplicación es la utilización de los servicios de Firebase y entender su asincronismo.

Como se vió en el apartado anterior, al final utilicé las lista de *Task* e implementar los listeners directamente a la lista con *Tasks.whenAll()*, pero al principio lo hice anidando los listeners y era todo un lío.

Por ejemplo, el *setValue()* del *DatabaseReference* le añadía su listener y dentro del callback de ese listener, realizaba el *putFile()* del *StorageReference*.

Otro pequeño bache que encontré fue el mostrar la foto que había tomado en pantalla. En un principio, tomaba la foto y la URI que devolvía la asignaba directamente al bitmap del componente *ImageView* de la pantalla y hacía que perdiese mucha calidad la foto y se veía muy borrosa. Finalmente, investigando un rato, ví que se podía guardar la imagen en un archivo temporal para que no perdiese información, y luego convertirla en la URI para poder mostrarla en la pantalla.

Y por último está el problema de las librerías para implementar el escáner de códigos de barras. Antes de llegar a la librería que estoy utilizando, había probado unas 4 o 5 anteriores, y cada una con su propia forma de implementar, y al final no funcionaban, o estaban obsoletas ,o simplemente no entendía cómo funcionaban. Pero finalmente encontré la que estoy utilizando ahora que está basada en una librería que había probado antes y estaba desactualizada.