



19 DE MAYO DE 2025

# PROYECTO INTEGRADOR

LIVEALBUMARTWORK

EMILIANO ESPINOZA PERALES – 21040337 – INGENIERÍA EN SISTEMAS COMPUTACIONALES

MATERIA: TECNOLOGÍAS INTERACTIVAS GRUPO: 8AY

Docente: Martínez Reyes Octavio Sergio

# Introducción

En la presente propuesta de proyecto integrador de realidad aumentada, se busca combinar Unity y Vuforia Engine para transformar portadas de álbumes musicales en experiencias visuales interactivas. La aplicación funcionará de tal forma que, al apuntar con la cámara de un dispositivo móvil al artwork de un disco, la aplicación identificará automáticamente la imagen y desplegará un modelo 3D animado (con movimiento), personalizado según la portada identificada. Al mismo tiempo, mostrará datos clave del álbum relevantes para el usuario (año de lanzamiento, autores, contexto histórico, tracklist, etc.) y reproducirá un fragmento de audio como preview de alguna o algunas de las pistas del álbum (como un mashup). La interacción táctil con el objeto 3D permitirá al usuario descubrir detalles adicionales y modificar ligeramente la animación en tiempo real.

## Propósito y objetivo general

**Propósito:** Facilitar el acceso a información musical a través de una interfaz inmersiva, capturando la atención del usuario mediante animaciones y audio sincronizado.

**Objetivo general:** Desarrollar un prototipo funcional que realice los siguientes procesos de forma fluida:

1. Reconocimiento de las portadas a través de Vuforia.
2. Consulta de metadatos y assets locales o en un backend (por ejemplo: Firebase).
3. Descarga dinámica de las texturas y clips de audio.
4. Generación de una experiencia de RA que contenga: animación 3D, paneles de información y reproducción de audio.
5. Detección de interacciones táctiles para expandir el contenido.

## Posibles beneficios o impacto social

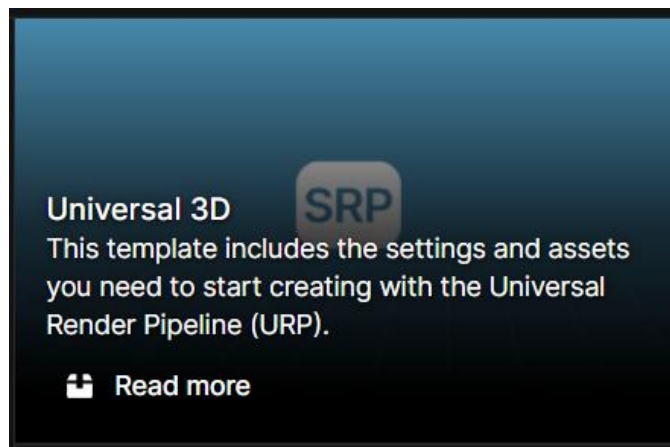
1. **Motivar a entusiastas a explorar la historia de la música**, comprendiendo la evolución de géneros y artistas a través de datos contextuales y curiosidades.
2. **Fomentar la cultura**, ayudando a preservar la memoria de álbumes clásicos y contemporáneos al ofrecer presentaciones dinámicas que trascienden el formato físico (discos y vinilos).

3. **Proporcionar a sellos y artistas independientes una herramienta de marketing innovadora**, aumentando el ‘engagement’ y el alcance de lanzamientos.
4. **Ofrecer accesibilidad tecnológica**, “democratizando” el uso de realidad aumentada al ofrecer una experiencia limpia y directa, sin necesidad de hardware especializado, solamente con un smartphone.
5. **Crear conexión social** a partir de los descubrimientos musicales gracias a la aplicación, permitiendo compartir experiencias y generar conversaciones en torno al contenido cultural.

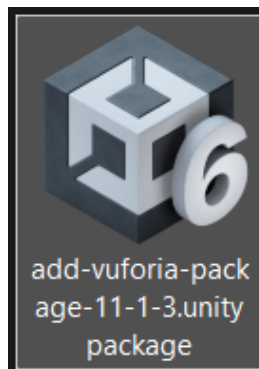
## Procedimiento de desarrollo

### 1. Configuración inicial de Unity y Vuforia

- Crear proyecto Unity (versión 2020.3 LTS o superior).



- Importar paquete Vuforia Engine via Package Manager.



- Registrar licencia básica en Vuforia Developer Portal y pegar App Key en Vuforia Configuration.

## Basic Plan

With the Basic Plan you have access to all Basic features and can try out our Premium features. You can publish apps commercially when using basic features. Click here to [Learn more!](#)

[Generate Basic License](#)

## Upgrade to Premium

Gain access to Premium features such as Model and Area Targets, production support, and more!

[Request Now](#)

Name	Type	Status ▾	Date Modified
LICENCIA_LIVEARTWORK	Basic	Active	May 02, 2025

- Crear base de datos en Vuforia Developer Portal

## AlbumCoverDB [Edit Name](#)

Type: Cloud

License Key: LICENCIA\_LIVEARTWORK



Associating a Cloud Database with a Basic license is for prototyping use only. Please purchase a Cloud Image Add-on to use your app in production. The Basic license will automatically be suspended if the quota for monthly recognitions is used up.

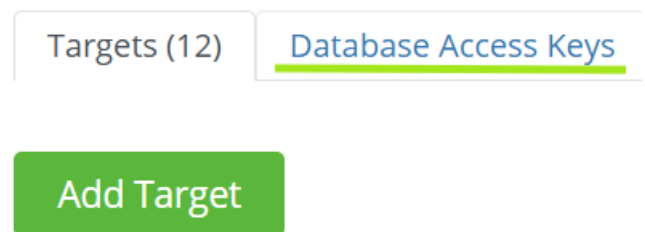
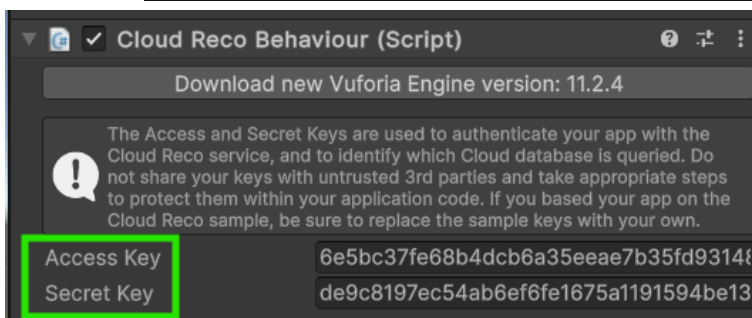
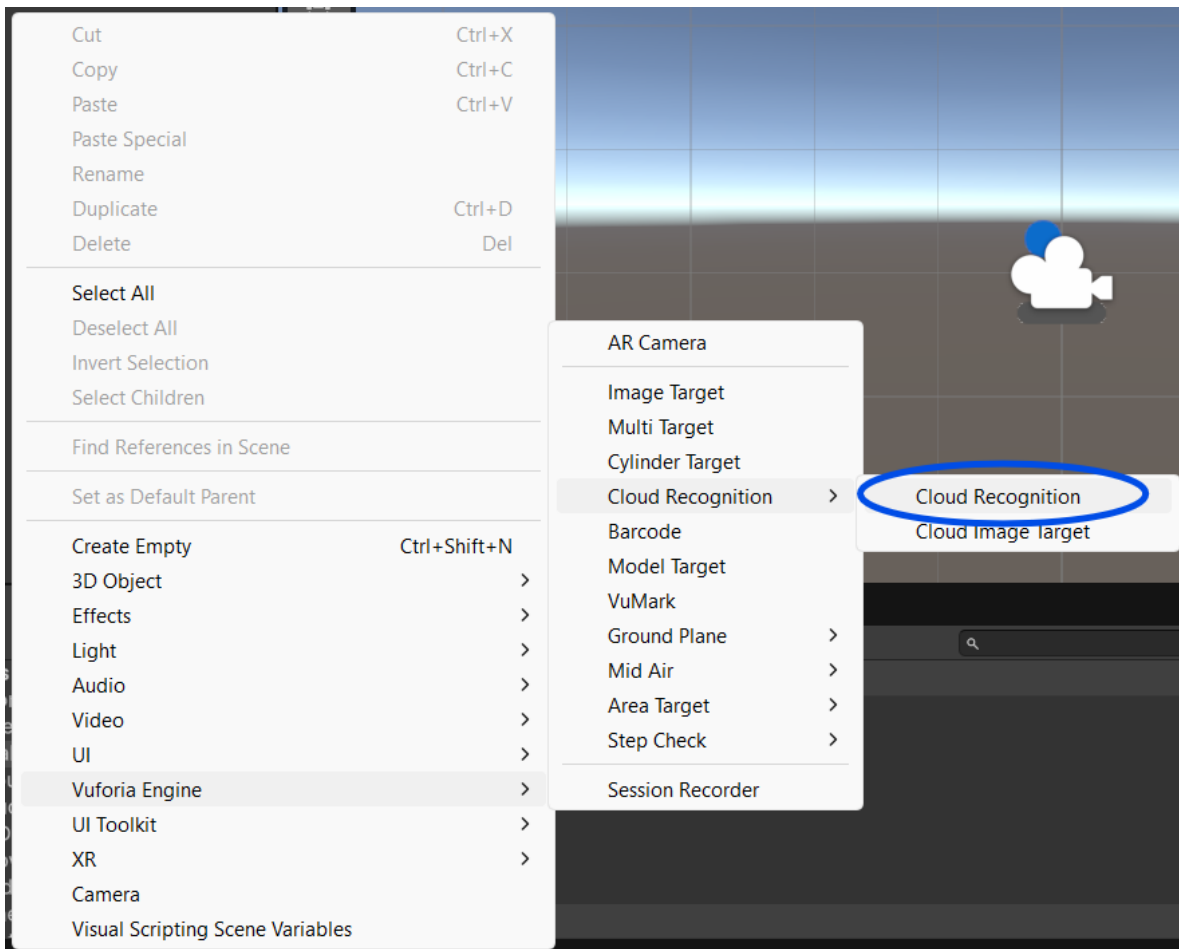
Targets (12)

[Database Access Keys](#)

Add Target

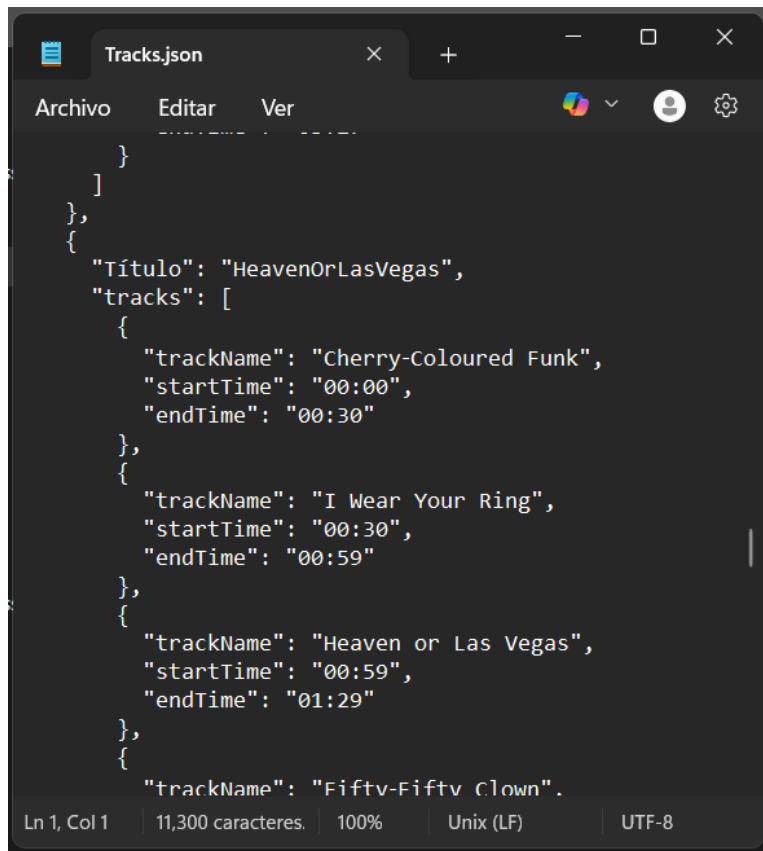
<input type="checkbox"/>	Image	Target Name	Rating ⓘ	Recos ▾	Status ▾	Date Modified
<input type="checkbox"/>		RayOfLight	★★★★★	78	Active	May 15, 2025
<input type="checkbox"/>		AlligatorBitesNeverHeal	★★★★☆	3	Active	May 14, 2025

- Añadir CloudRecoBehaviour en escena y configurar Access Key/Secret Key.



## 2. Preparación de assets y estructura de carpetas

- Carpeta **'Assets/Models'**: importar modelo 3D ajustado correctamente desde Blender u otro programa de modelado.
- Carpeta **'Assets/Resources'**: organizar subcarpetas: Portadas/, TexturasDisco/, Videos/, Audios/, Albums.json, Tracks.json.

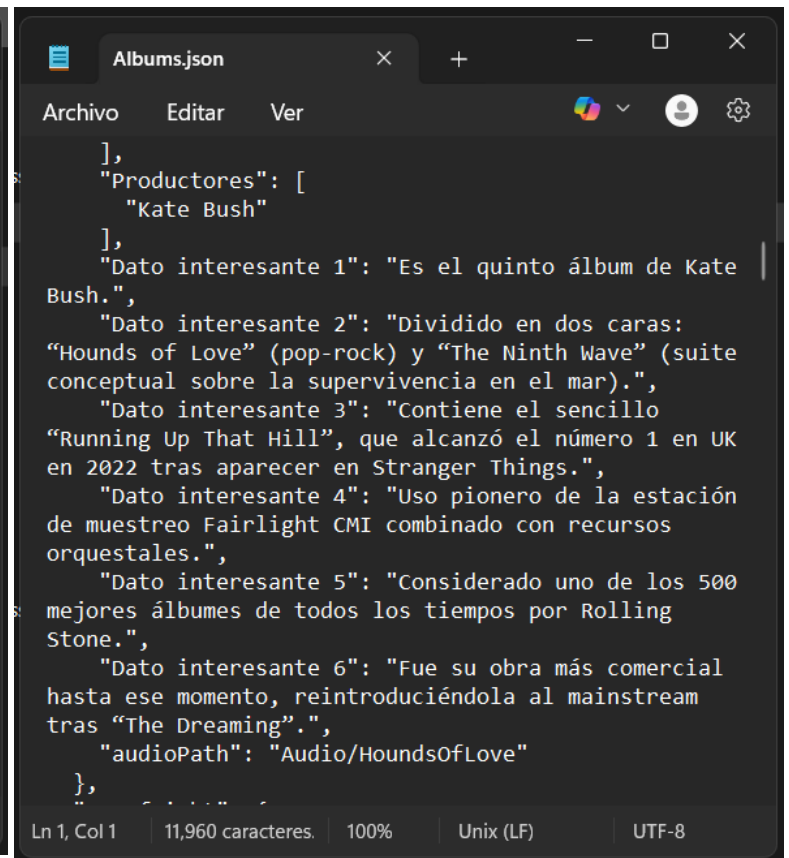


```

{
  "Título": "HeavenOrLasVegas",
  "tracks": [
    {
      "trackName": "Cherry-Coloured Funk",
      "startTime": "00:00",
      "endTime": "00:30"
    },
    {
      "trackName": "I Wear Your Ring",
      "startTime": "00:30",
      "endTime": "00:59"
    },
    {
      "trackName": "Heaven or Las Vegas",
      "startTime": "00:59",
      "endTime": "01:29"
    },
    {
      "trackName": "Fifty-Fifty Clown"
    }
  ]
}

```

Ln 1, Col 1 | 11,300 caracteres. | 100% | Unix (LF) | UTF-8



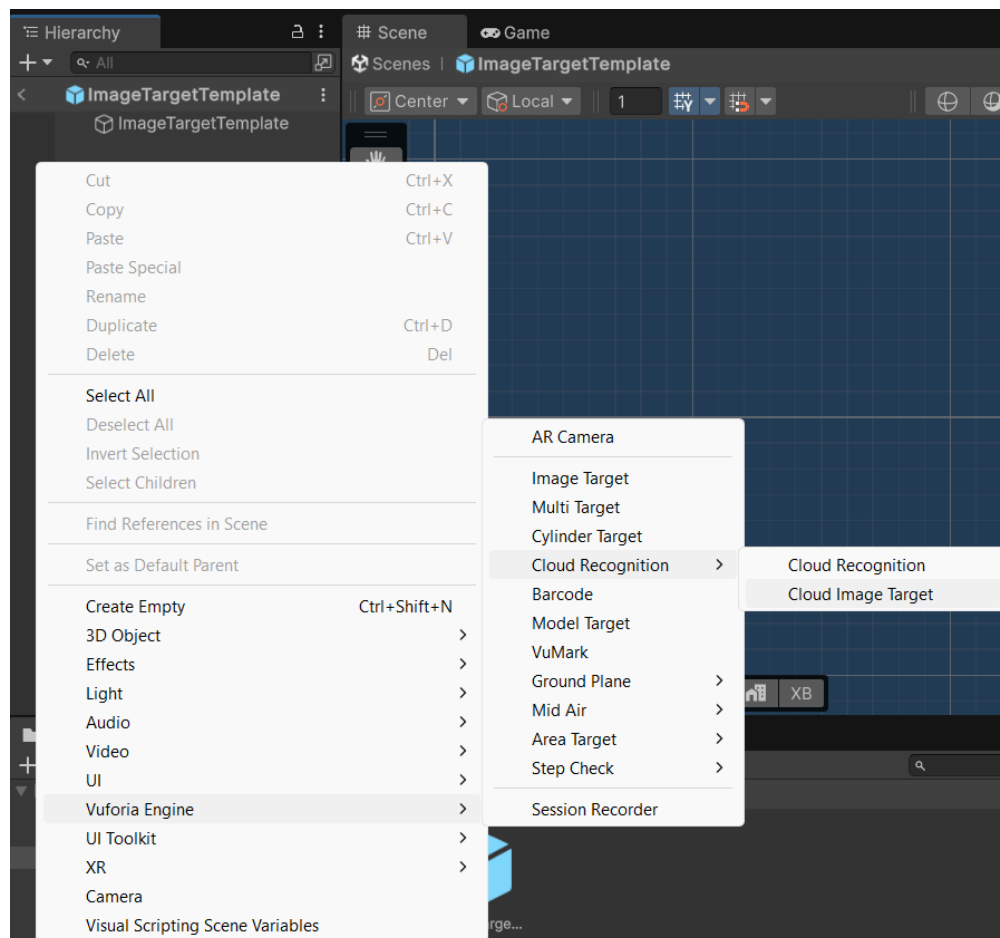
```

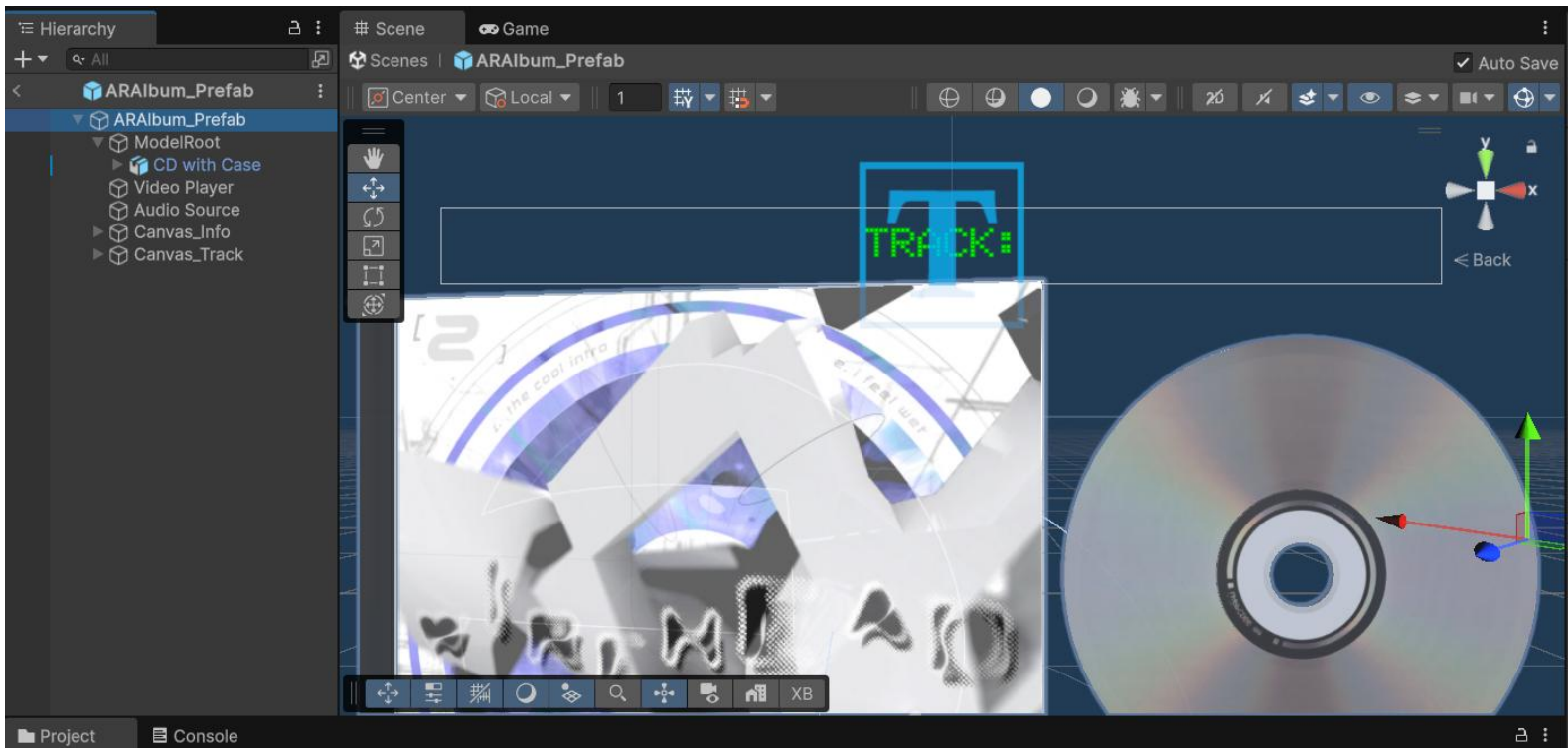
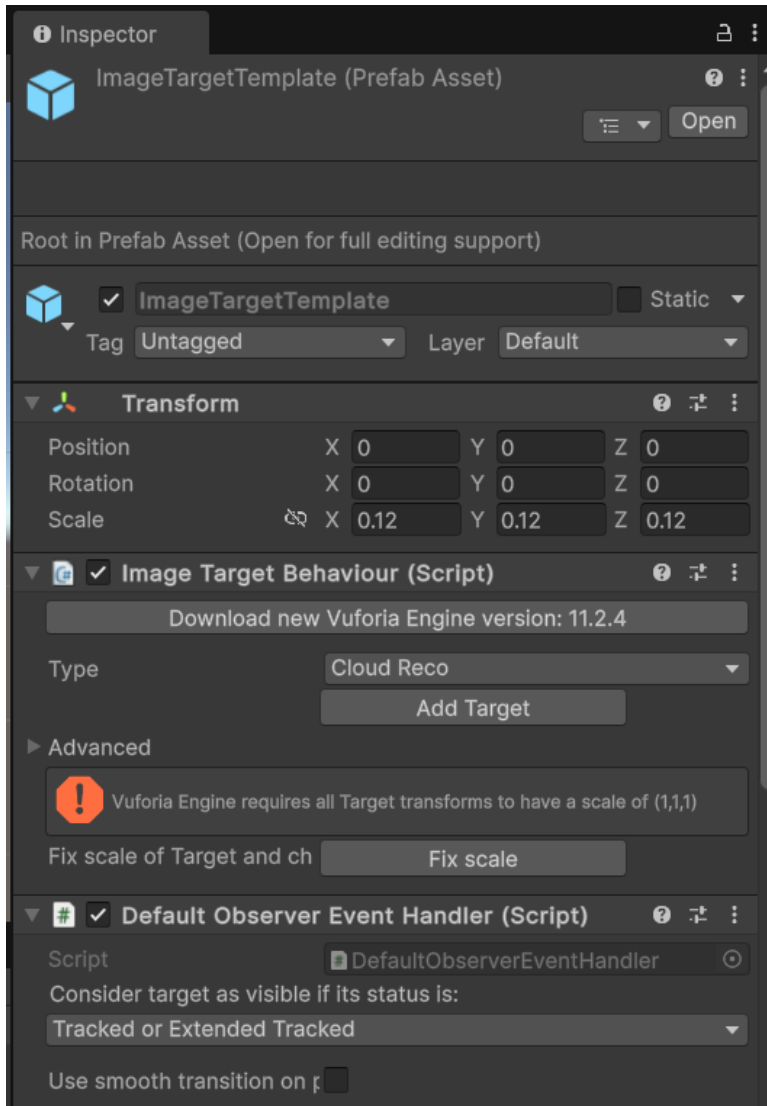
{
  "Productores": [
    "Kate Bush"
  ],
  "Dato interesante 1": "Es el quinto álbum de Kate Bush.",
  "Dato interesante 2": "Dividido en dos caras: 'Hounds of Love' (pop-rock) y 'The Ninth Wave' (suite conceptual sobre la supervivencia en el mar).",
  "Dato interesante 3": "Contiene el sencillo 'Running Up That Hill', que alcanzó el número 1 en UK en 2022 tras aparecer en Stranger Things.",
  "Dato interesante 4": "Uso pionero de la estación de muestreo Fairlight CMI combinado con recursos orquestales.",
  "Dato interesante 5": "Considerado uno de los 500 mejores álbumes de todos los tiempos por Rolling Stone.",
  "Dato interesante 6": "Fue su obra más comercial hasta ese momento, reintroduciéndola al mainstream tras 'The Dreaming'.",
  "audioPath": "Audio/HoundsOfLove"
}

```

Ln 1, Col 1 | 11,960 caracteres. | 100% | Unix (LF) | UTF-8

- Crear los siguientes prefabs:





### 3. Implementación de Cloud Recognition (Script **CloudRecoHandler.cs**):

- Inicializa y arranca el reconocimiento en la nube (Cloud Reco).
- Responder al evento de detección de una portada: detiene temporalmente el escaneo y clona un ImageTarget preparado como plantilla.
- Escalar y alinear ese ImageTarget al tamaño de la portada.
- Instanciar el prefab que contiene el modelo 3d como hijo directo del ImageTarget.
- Pasar al prefab el identificador del target hallado para que los demás controladores carguen los datos correctos.
- Reactivar el escaneo tras un breve retraso para que el usuario pueda detectar otro álbum.

```
C# using UnityEngine; Untitled-1
1  using UnityEngine;
2  using Vuforia;
3
4  public class CloudRecoHandler : MonoBehaviour
5  {
6      [Header("Vuforia")]
7      public CloudRecoBehaviour cloudRecoBehaviour;
8      public ImageTargetBehaviour ImageTargetTemplate;
9
10     [Header("Prefabs")]
11     public GameObject albumPrefab;
12
13     void Awake()
14     {
15         if (cloudRecoBehaviour == null)
16             cloudRecoBehaviour = GetComponent<CloudRecoBehaviour>();
17         //
18         cloudRecoBehaviour.RegisterOnInitializedEventHandler(OnInitialized);
19         cloudRecoBehaviour.RegisterOnNewSearchResultEventHandler(OnNewSearchResult);
20     }
21
22     void OnInitialized(CloudRecoBehaviour crb)
23     {
24         Debug.Log("Cloud Reco inicializado y listo para escanear.");
25     }
26
27     public void OnNewSearchResult(CloudRecoBehaviour.CloudRecoSearchResult result)
28     {
29         Debug.Log("Target detectado: " + result.TargetName);
30         //
31         cloudRecoBehaviour.enabled = false;
32         //
33         var obs = cloudRecoBehaviour
34             .EnableObservers(result, ImageTargetTemplate.gameObject)
35             as ImageTargetBehaviour;
36         //
37         if (obs == null)
38         {
39             Debug.LogError("No se pudo crear el ImageTarget para: " + result.TargetName);
40             return;
41         }
42         //
43         var albumGO = Instantiate(albumPrefab, obs.transform);
44         albumGO.transform.localPosition = Vector3.zero;
45         albumGO.transform.localRotation = Quaternion.identity;
46         albumGO.transform.localScale = Vector3.one;
47         //
48         var ctrlData = albumGO.GetComponent<AlbumController>();
49         ctrlData.targetName = result.TargetName;
50         ctrlData.audioSource = albumGO.GetComponentInChildren<AudioSource>();
51         ctrlData.infoText = albumGO.transform.Find("Canvas_Info/InfoText")
52             .GetComponent<TMPPro.TextMeshProUGUI>();
53         ctrlData.trackText = albumGO.transform.Find("Canvas_Track/InfoTrack")
54             .GetComponent<TMPPro.TextMeshProUGUI>();
55         //
56         var visual = albumGO.GetComponent<AlbumVisualController>();
57         visual.SetupVisuals(result.TargetName);
58         //
59         Invoke(nameof(RestartScanning), 3f);
60     }
61
62     public void RestartScanning()
63     {
64         if (!cloudRecoBehaviour.enabled)
65         {
66             cloudRecoBehaviour.enabled = true;
67             Debug.Log("Reanudando Cloud Reco.");
68         }
69     }
70 }
```



#### 4. Control de datos y audio (Script AlbumController.cs):

- Leer y parsear dos archivos JSON al inicio: uno con los metadatos generales de cada álbum (artista, año, sello, curiosidades...) y otro con la información del tracklist.
- Cuando recibe su el nombre del target (estandarizado para consistencia), busca en esos JSON el nodo correspondiente. Si no lo encuentra, avisa con un error.
- Cargar dinámicamente el clip de audio preview almacenado en Resources y empezar a reproducirlo.
- Mostrar primero la información esencial (título, lanzamiento, artista), luego rotar curiosidades en pantalla cada pocos segundos, y finalmente, a medida que la música avanza, actualizar el texto de "Track: ..." para indicar al usuario en qué fragmento se encuentra.
- Aplicar un pequeño detalle estético al texto en función del álbum detectado.

```
1  using UnityEngine;
2  using TMPPro;
3  using System.Collections;
4  using SimpleJSON;
5
6  public class AlbumController : MonoBehaviour
7  {
8      [Header("Identificador")]
9      public string targetName;
10
11     [Header("Componentes")]
12     public AudioSource audioSource;
13     public TextMeshProUGUI infoText;
14
15     //
16     private JSONNode db;
17
18     void Awake()
19     {
20         //
21         var txtAsset = Resources.Load<TextAsset>("Albums");
22         if (txtAsset == null)
23         {
24             Debug.LogError("No se encontró Resources/Albums.json");
25             return;
26         }
27         db = JSON.Parse(txtAsset.text);
28     }
29
30     void Start()
31     {
32         if (db == null || db[targetName] == null)
33         {
34             Debug.LogError($"No hay datos para '{targetName}' en Albums.json");
35             return;
36         }
37
38         var node = db[targetName];
39
40         //
41         string audioPath = node["audioPath"];
42         var clip = Resources.Load<AudioClip>(audioPath);
43         if (clip != null) audioSource.PlayOneShot(clip);
44         else Debug.LogWarning($"Audio no encontrado en Resources/{audioPath}");
45
46         //
47         StartCoroutine(CycleInfo(node));
48     }
49
50     IEnumerator CycleInfo(JSONNode node)
51     {
52         //
53         var infos = new System.Collections.Generic.List<string>();
54
55         //
56         string title = node["Título"];
57         string release = node["Lanzamiento"];
58         string artist = node["Artista"];
59         infos.Add($"{{title}} {{(release)}}\nBy {{artist}}");
60     }
```

```
60
61     //
62     var sellos = node["Sello"].AsArray;
63     var productores = node["Productores"].AsArray;
64     infos.Add($"Sello: {string.Join(", ", sellos)}");
65     infos.Add($"Productores: {string.Join(", ", productores)}");
66
67     //
68     for (int i = 1; i <= 6; i++)
69     {
70         string key = $"Dato interesante {i}";
71         if (node[key] != null)
72             infos.Add(node[key]);
73     }
74
75     //
76     int idx = 0;
77     while (true)
78     {
79         infoText.text = infos[idx];
80         idx = (idx + 1) % infos.Count;
81         yield return new WaitForSeconds(5f);
82     }
83
84 }
85
```

## 5. Control visual e interactividad (Script AlbumVisualController.cs):

- Recibe el nombre del álbum y asigna las texturas al CD case (la portada) y al disco 3D (la textura personalizada).
- Prepara el VideoPlayer con el clip adecuado y asocia su salida a un RenderTexture que luego se muestra cuando el usuario lo solicite.
- Inicialmente configura el plano de vídeo para que muestre la portada, de modo que siempre se vea la imagen del álbum.
- En cada fotograma, hace girar el disco sobre su propio eje horizontal, simulando el rodar de una rueda de coche, sin que se desplace de su posición.
- Atiende a los clics provenientes de los colliders para pausar/reanudar la rotación y la música, así como para alternar entre la portada estática y la reproducción del vídeo en el plano.

```
1  using UnityEngine;
2  using UnityEngine.EventSystems;
3  using UnityEngine.Video;
4
5  public class AlbumVisualController : MonoBehaviour, IPointerClickHandler
6  {
7      [Header("Renderers")]
8      public MeshRenderer discRenderer;
9
10     [Header("Video Overlay")]
11     public GameObject videoScreenPlane; // tu plane hijo de CDCase
12     public VideoPlayer videoPlayer;
13
14     bool showingVideo = false;
15
16     /// <summary>
17     /// Llamar desde CloudRecoHandler tras instanciar el prefab.
18     /// </summary>
19     public void SetupVisuals(string albumKey)
20     {
21         // 1) Textura del disco
22         var discTex = Resources.Load<Texture2D>($"CD textures/{albumKey}");
23         if (discTex != null)
24             discRenderer.material.mainTexture = discTex;
25         else
26             Debug.LogWarning($"Textura disco no encontrada: {albumKey}");
27
28         // 2) Video
29         var vpClip = Resources.Load<VideoClip>($"Videos/{albumKey}");
30         if (vpClip != null)
31             videoPlayer.clip = vpClip;
32         else
33             Debug.LogWarning($"Video no encontrado: {albumKey}");
34
35         // Aseguramos que el plane esté oculto al inicio
36         videoScreenPlane.SetActive(false);
37     }
38
39     void Update()
40     {
41         // 3) Gira el disco continuamente
42         discRenderer.transform.Rotate(Vector3.up, 45f * Time.deltaTime, Space.Self);
43     }
44
45     public void OnPointerClick(PointerEventData eventData)
46     {
47         // 4) Alterna video on/off al tocar
48         if (!showingVideo)
49         {
50             videoScreenPlane.SetActive(true);
51             videoPlayer.Play();
52             showingVideo = true;
53         }
54         else
55         {
56             videoPlayer.Stop();
57             videoScreenPlane.SetActive(false);
58             showingVideo = false;
59         }
60     }
61 }
62
```

## 6. Pruebas y depuración en Android

- Configurar **Player Settings** → **Android**:

**Player**

Company Name	EmilianoEspinozaPerales
Product Name	LiveAlbumARtwork
Version	0.1

▼ **Resolution and Presentation**

**Resolution**

Run Without Focus	<input checked="" type="checkbox"/>
Fullscreen Mode	Fullscreen Window
Resizable Activity	<input checked="" type="checkbox"/>
Hide Navigation Bar	<input checked="" type="checkbox"/>
Render outside safe area	<input checked="" type="checkbox"/>
Optimized Frame Pacing	<input checked="" type="checkbox"/>

**Resolution Scaling**

Resolution Scaling Mode	Disabled
Reset resolution on window resize	<input type="checkbox"/>
Blit Type	Always

**Supported Aspect Ratio**

Aspect Ratio Mode	Native Aspect Ratio
-------------------	---------------------

**Miscellaneous**

Show Loading Indicator	Don't Show
------------------------	------------

**Orientation**

Default Orientation*	Auto Rotation
Auto Rotation Behavior	User

**Allowed Orientations for Auto Rotation**

Portrait	<input checked="" type="checkbox"/>
Portrait Upside Down	<input checked="" type="checkbox"/>
Landscape Right	<input checked="" type="checkbox"/>
Landscape Left	<input checked="" type="checkbox"/>

Use 32-bit Display Buffer\* ☒

Disable Depth and Stencil\* ☐

Render Over Native UI\* ☐

\* Shared setting between multiple platforms.

▼ **Other Settings**

**Rendering**

Color Space\* Linear

MSAA Fallback Downgrade

Auto Graphics API ☐

**Graphics API**

OpenGLES3

Vulkan

Require ES3.1 ☐

Require ES3.1+AEP ☐

Require ES3.2 ☐

Multithreaded Rendering\* ☒

Static Batching ☒

Dynamic Batching ☐

Sprite Batching Threshold 300

Sprite Batching Max Vertex Count 65535

GPU Skinning\* GPU (Batched)

Graphics Jobs (Experimental) ☐

**Identification**

Override Default Package Name ☒

Package Name com.EmilianoEspinozaPerales.LiveAlbumARTwork

Version\* 0.1

Bundle Version Code 1

Minimum API Level Android 10.0 (API level 29)

Target API Level Android 10.0 (API level 29)

**Configuration**

Scripting Backend IL2CPP

Api Compatibility Level\* .NET Standard 2.1

Editor Assemblies Compatibility Level\* Default (.NET Framework)

IL2CPP Code Generation Faster runtime

C++ Compiler Configuration Release

IL2CPP Stacktrace Information Method Name

Use incremental GC\* ☒

Allow downloads over HTTP\* Not allowed

Mute Other Audio Sources\* ☐

Active Input Handling\* Input System Package (New)

Target Architectures

ARMv7 ☒

ARM64 ☒

x86-64 (Magic Leap 2) ☐

Enable Armv9 Security Features for Arm64 ☐

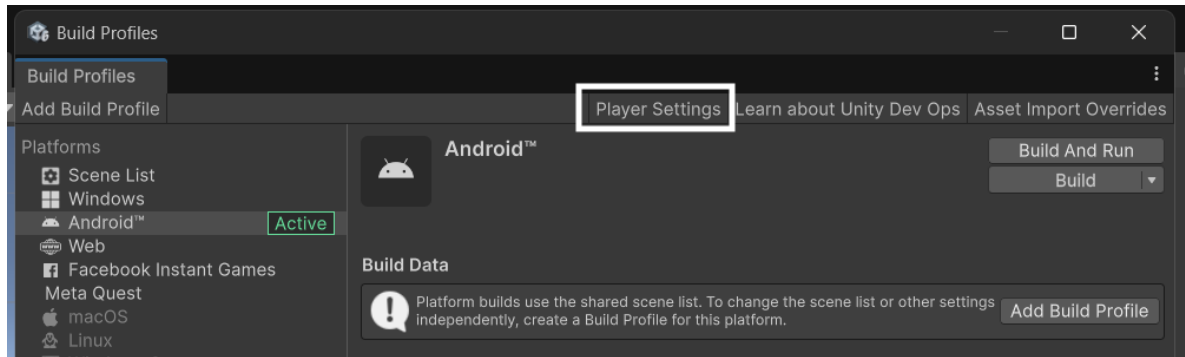
Split APKs by target architecture ☐

Install Location Prefer External

Internet Access Require

Write Permission Internal

- Generar APK (build) e instalar en dispositivo.



## Herramientas utilizadas

- **Unity 2020.3 LTS:** entorno de desarrollo, edición de escena y prefabs.
- **Vuforia Engine:** reconocimiento de imágenes en la nube (Cloud Recognition).
- **Blender:** ajuste del modelo 3d de CD case y disco.
- **Visual Studio / VS Code:** edición de scripts C#.
- **SimpleJSON:** biblioteca para parsear JSON en Unity.
- **TextMeshPro:** renderizado de texto de alta calidad en canvases.
- **Adobe Photoshop:** diseño de imágenes y texturas personalizadas para el modelo.
- **Audacity:** Edición de tracks.