

Trakabum



Curso: 2ºDAM O

Alumno: Sergio Laín Camús

Tutor: Jose Ramón García Sevilla

ÍNDICE

1 Introducción

- 1.1 Breve descripción del proyecto
- 1.2 Objetivos a perseguir

2 Tipo de Empresa y Estudios de Viabilidad y Mercado

2.1 Empresa y entorno

- 2.1.1 Entorno PESTEL, PORTER, DAFO, CAME
- 2.1.2 Misión, visión y valores
- 2.1.3 Nombre comercial, marca y logotipo

2.2 Estudio de mercado

- 2.2.1 Estudio de mercado: Análisis de la realidad local
- 2.2.2 Consumidores potenciales
- 2.2.3 Segmentación, estrategias y mapa de empatía

2.3 Forma jurídica de la empresa

- 2.3.1 Forma jurídica de la empresa
- 2.3.2 Recursos humanos
- 2.3.3 Impuestos devengados

2.4 Plan de operaciones

- 2.4.1 Plan de inversiones y gastos
- 2.4.2 Punto muerto/Umbral de rentabilidad

2.5 Plan de financiación

2.6 Plan financiero

- 2.6.1 Plan de tesorería
- 2.6.2 Cuenta de resultados

2.7 Tramites de constitución y puesta en marcha

3 Análisis de Requerimientos

- 3.1 Requisitos funcionales
- 3.2 Requisitos no funcionales
- 3.3 Legislación aplicable
- 3.4 Metodología. Fases del proyecto. Tareas y plazos de ejecución

4 Diseño

4.1 Diseño de datos

- 4.1.1 Entidad-relación
- 4.1.2 Diseño lógico-relacional
- 4.1.3 Diseño físico

4.2 Diseño funcional

- 4.2.1 Diagrama de clases
- 4.2.2 Diagrama de casos de uso
- 4.2.3 Especificación de casos de uso
- 4.2.4 Diseño de interfaz

5 Implementación

5.1 Tecnologías a emplear

6 Validación

- 6.1 Definición del procedimiento de evaluación, seguimiento y control del proyecto
- 6.2 Procedimientos para la participación de los usuarios en la evaluación del proyecto

7 Conclusiones

8 Bibliografía

9 Anexo

Introducción

Breve descripción del proyecto

La idea de este proyecto consiste en desarrollar una aplicación para dispositivos móviles en la cual puedes almacenar y hacer un seguimiento de los álbumes que estas escuchando, estuviste escuchando y quieres escuchar en un futuro. Ademas de poder hacer un seguimiento de ellos, cuando los termines podrás darles una clasificación para saber si te ha gustado o no. A estos álbumes les podrás dar etiquetas (Tags) personalizadas o dejar las que ya vienen predeterminadas dependiendo del genero del mismo. Por ejemplo, si añadimos un álbum de Rock Progresivo, podremos mantener el tag de Rock Progresivo que es con el que viene el álbum o ponerle uno totalmente diferente.

Yo como melomano que soy, me encanta escuchar muchos y variados álbumes de diferentes artistas y estilos, pero al también ser una persona que le encanta organizar todo, saltar de álbum en álbum sin haber terminado uno o tal vez sin recordar cuales escuche me suele frustrar y es por lo que estuve buscando en un pasado una aplicación que me sirviera para esto.

Es por ello por lo que encontré una posibilidad perfecta para poder hacerla. Y así se creó la empresa **MATCOM**, la cual desarrollaría el producto **Trakabum**.

Objetivos a perseguir

Este proyecto va dirigido a gente que le guste escuchar música y quiera organizar sus sesiones de escucha o almacenar todo lo que ha ido escuchando y lo que le ha parecido. Mi aplicación permitirá a este publico:

- **Mayor organización** al tenerlo todo en un sitio, en este caso, mi app.
- **Mayor rapidez** gracias al poder buscar álbumes rápidamente.

Tipo de Empresa y Estudios de Viabilidad y Mercado

Empresa y entorno

Vamos a empezar a hablar y detallar todo lo relacionado a lo que viene siendo nuestra empresa, como la creación, el análisis del entorno, la ubicación o el plan financiero de la misma.

Antes de ir con los demás puntos, me gustaría comentar rápidamente la ubicación de la empresa. Esta se ubicará en un coworking llamado Ulab, el cual esta ubicado cerca del Mercado Central. Al ser pocos miembros en la empresa como veremos luego mas tarde, esta es la opción mas económica y fácil para ir empezando con nuestra empresa, a tan solo 176€ al mes *por persona*.



Figura 1: Imagen de la oficina de ULab, Fuente: Alicante Turismo

Entorno PESTEL, PORTER, DAFO, CAME

Se ha hecho un análisis PESTEL y un estudio de 5 fuerzas PORTER. De estos, se han extraído los conceptos y factores mas importantes a tener en cuenta en cuanto a nuestro producto.

Primero vamos a ver el análisis PESTEL:

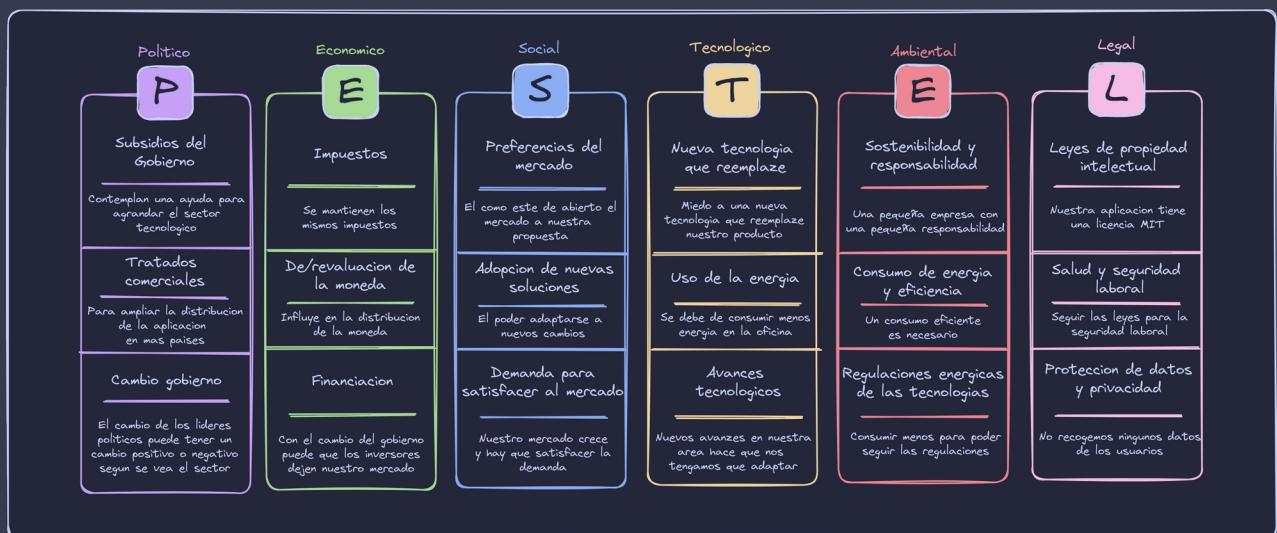


Figura 2: Análisis PESTEL; Fuente: Propia

Ahora vamos a ver el PORTER (Bajo, Medio, Alto):



Figura 3: Análisis PORTER; Fuente: Propia

A continuación se detallan el análisis DAFO y el CAME, donde sacamos nuestras fortalezas, debilidades, amenazas y las oportunidades de nuestro producto de cara al público e internamente.

Análisis DAFO:



Figura 4: Análisis DAFO; Fuente: Propia

Análisis CAME:

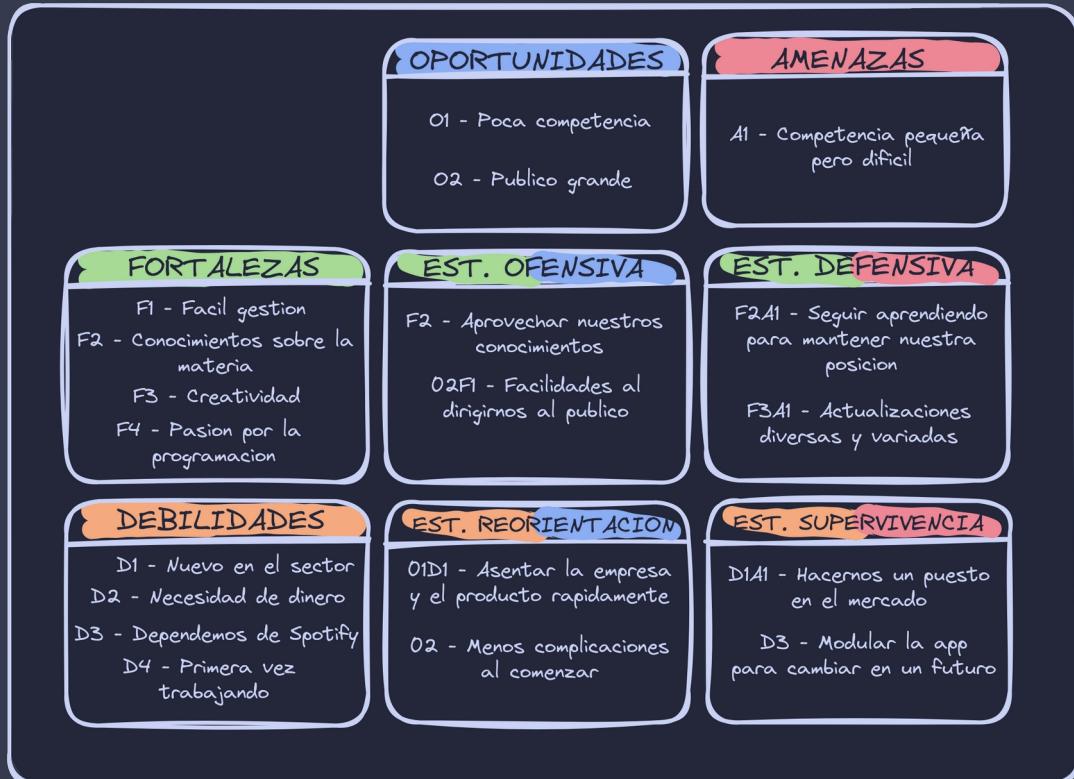


Figura 5: Análisis CAME; Fuente: Propia

Misión, visión y valores

- **Misión:** La misión de la empresa es facilitar a toda persona que le gusta escuchar música y quisiera organizar sus escuchas una aplicación para poder ayudarle a conseguirlo.
- **Visión:** La visión de la empresa es poder obtener los suficientes beneficios para poder seguir con la empresa y desarrollar mas productos en un futuro.
- **Valores:** Los valores de la empresa son sobretodo el compromiso por hacer las cosas lo mejor posible, la calidad e innovación en nuestros productos y el poder crecer junto al cliente.

Nombre comercial, marca y logotipo

El nombre comercial de la empresa que voy a crear para este proyecto se va a llamar MATCOM. El nombre es un resultado de unir dos palabras:

MAT → Micro Arquitecture Technology

COM → Computer

Esta combinación de palabras hacen referencia a que la empresa se dedica a la tecnología de ordenadores.

Para el nombre del proyecto, he elegido el nombre de Trakabum, el cual es otra unión de palabras tal como hemos visto en el nombre de la empresa:

Trak → Track (Monitorear/Seguir)

Abum → Álbum

Esta combinación hace que se entienda cual es el propósito de la aplicación: Monitorear tus álbumes no o ya escuchados.

A continuacion están los logotipos, los cuales tienen un patrón de dibujo a mano, ya que representa a que MATCOM hace todo de manera artesanal y con cariño para todos sus clientes.

El logotipo de la empresa es el siguiente:

El de la aplicación es este:



Figura 6: Logotipo Empresa; Fuente: Propia

Figura 7: Logotipo App; Fuente: Propia

Estudio de mercado

Estudio de mercado: Análisis de la realidad local

Nuestra empresa se dirige a el mercado de oyentes de música, que preferiblemente usen spotify regularmente para escuchar música, con la necesidad de organizar la música que escuchan, fácil y rápido.

Este mercado es bastante amplio, ya que la gran mayoría de gente escucha música a menudo en su día a día, ya sea para trabajar, estudiar, descansar o dormir, por lo tanto la ubicación de mis clientes es global (Por todo el mundo).

Un estudio realizado en 2018 por la empresa de investigación de mercado NPD Group encontró que el 73% de los consumidores de música en EE.UU. aún escuchan música que tienen almacenada en su computadora. Estos consumidores dijeron que el proceso de organizar su música y crear listas de reproducción era importante para ellos.

Además, un estudio realizado en 2017 por la compañía de tecnología de audio Sonos encontró que el 62% de los encuestados dijeron que la organización de su colección de música digital era importante para ellos. El mismo estudio también encontró que el 74% de los encuestados había creado al menos una lista de reproducción.

En resumen, hay evidencia de que muchas personas necesitan y valoran la organización de su música, por lo que tenemos un buen mercado al que apuntar.

Consumidores potenciales

La aplicación va dirigida a personas de entre 14 y 40 años, aunque perfectamente la podrían usar mayores y menores a la franja de edad que se indica. También cabe indicar de que para los menores que se registren, se les pondrá sobre un álbum que tenga contenidos para mayores de 18. Como ya dije al principio, la aplicación esta pensada para gente olvidadiza o que le gustaría almacenar y tener a su disposición la cantidad de música que escucha y que le gustaría escuchar.

Al ser una aplicación de móvil también es bastante fácil de instalar y usarla, por lo que con eso esperemos expandir un poco el mercado en un futuro.

Segmentación, estrategias y mapa de empatía

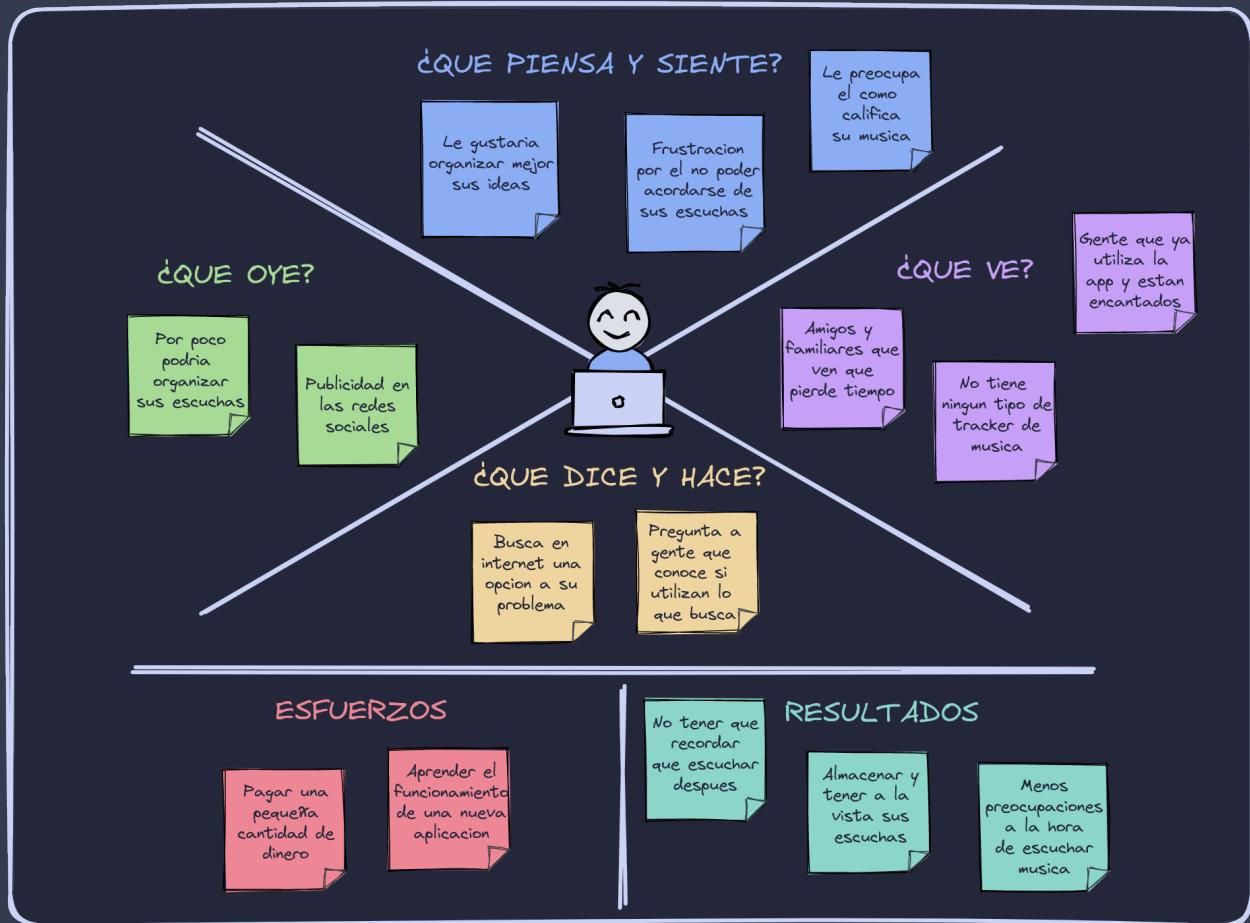


Figura 8: Mapa de empatía; Fuente: Propia

Como veis arriba tenemos el mapa de empatía, el cual nos ayuda a entender al cliente y encontrar el producto efectivo para él.

Tenemos varios tipos de segmentación para nuestro producto. Estas son:

- **Demográfica:** Personas entre 14 y 40 años
- **Psicográfica:** Personas que tengan mala memoria o les guste organizar las cosas que hacen.
- **Geográfica:** Nos dirigimos a gente en todo el mundo.
- **Comportamental:** Usuarios que escuchan música varias veces al día y se centran en la escucha de álbumes de principio a fin.

Ahora vamos a discutir sobre las estrategias . Tenemos diferentes estrategias, las cuales varían dependiendo del tema a requerir:

- **Precio:** La primera aplicación, Trakabum, se lanzará por un precio de 1€. Al ver la competencia, esta se dedica a ofrecer una aplicación de track de música gratis pero con anuncios intrusivos que destrozan la experiencia, donde en algunas se puede pagar para eliminarlos. Lo que hemos hecho desde MATCOM es eliminar esa barrera donde, por solo 1,18€ ya tienes acceso a todo sin publicidad. También esta pensado hacer una demo gratuita en la cual puedes probar la aplicación por el periodo de 2 horas sin interrupciones. Este primer producto busca ser rentable, ademas de dar a conocer a el publico quien es MATCOM y como se nos va a ver desde el principio.
- **Distribución:** La distribución se hará en la tienda de aplicaciones de Google Play, donde la gran mayoría de aplicaciones se encuentran y de donde mas gente suele descargar o comprar apps.
- **Promoción:** La promoción se hará en redes sociales con el nombre de nuestra empresa, ademas de con Google Ads, donde publicaremos la pagina de la aplicación al buscar en internet palabras como: "Track", "Álbum", "Music Tracker"... Cabe destacar de que si el producto es bueno, las reseñas también lo serán, lo que dará un gran boost a nuestra aplicación y compañía.

Forma jurídica de la empresa

Forma jurídica de la empresa

La forma jurídica de la que he optado por ha sido la de ser una Sociedad Limitada Unipersonal (S.L.U.). Las razones son:

- Respondemos con el capital de la empresa (responsabilidad limitada) y nos distanciamos entre el concepto de la persona y la empresa, cosa que no pasa por ejemplo con los autónomos, donde ellos son la empresa y responden con sus ingresos.
- Tenemos que pagar unos menos impuestos ya que tributa un porcentaje fijo de 25% o 30%, mientras que por ejemplo el autónomo es de unos 24% a 48% de sus beneficios.
- Una SLU también transmite una mejor imagen de fiabilidad y rendimiento económico que la de un autónomo

Ademas de tener varias ventajas, también tiene varias desventajas como:

- Tienes que pagar un capital inicial de unos 3000€ y el proceso es un poco largo.
- Dependemos en parte de mas empresas que nos ayuden a poder hacernos mas grandes en los primeros momentos.

Recursos humanos

Ahora vamos a discutir el tema recursos humanos de nuestra empresa.

Primero vamos a empezar por el organigrama de la misma:

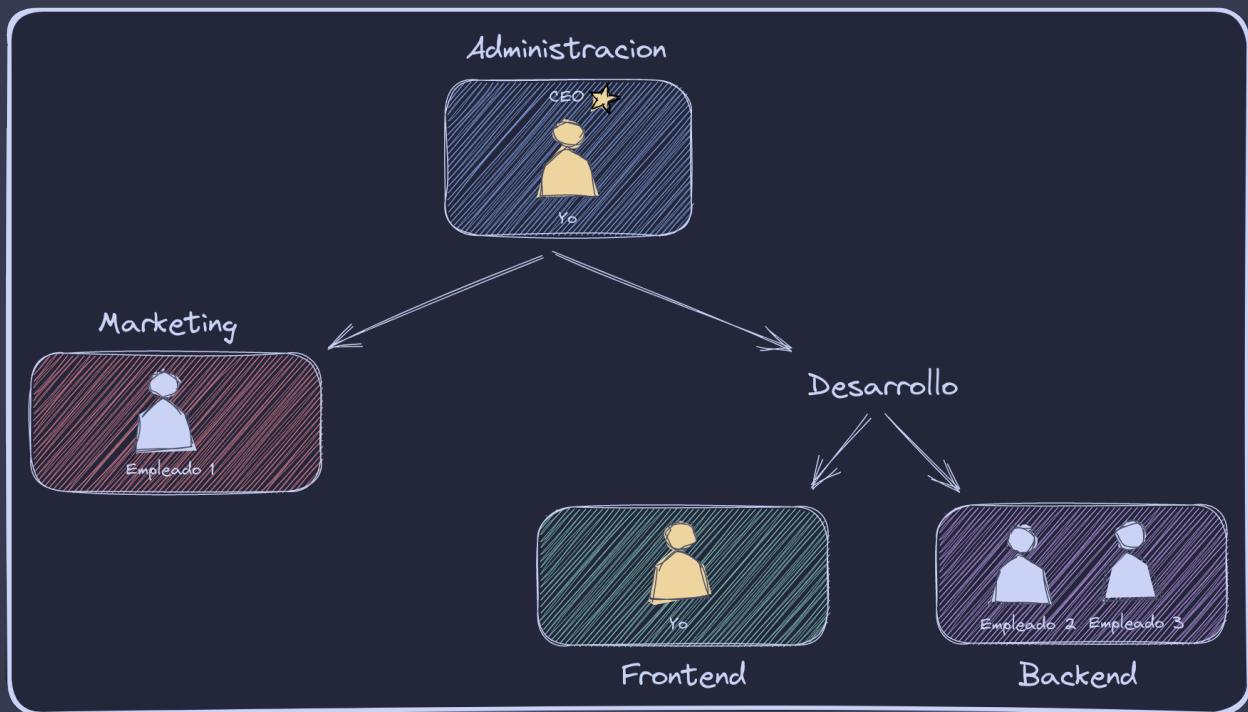


Figura 9: Organigrama empresarial; Fuente: Propia

En el organigrama vemos que yo soy el jefe de la empresa, ademas de dedicarme también al desarrollo frontend de la aplicación. Luego tendré a 3 empleados contratados, uno de ellos en el departamento de Marketing mientras que los otros dos estarán trabajando en el desarrollo del backend de nuestra aplicación.

Departamentos detallados con sus tareas:

- **Administración**: Este departamento se encargará de la gestión financiera, contabilidad, presupuesto y administración general de la empresa.
- **Marketing**: La tarea de este departamento será la de promocionar la aplicación y crear la conciencia que se va a tener sobre nuestra marca.
- **Desarrollo (Frontend)**: Este departamento se encarga de diseñar la interfaz y la experiencia que va a tener el usuario con nuestra aplicación.
- **Desarrollo (Backend)**: Se encarga de el diseño, desarrollo y prueba de la aplicación.

A continuación se define el coste de cada uno de los trabajadores:

Trabajador	Departamento	Salario (€)	S.S. (37%) (€)	Total Gasto (€)
Trabajador 1	Marketing	1125,83	416	1541.83
Trabajador 2	Desarrollo (Back)	1125,83	416	1541.83
Trabajador 3	Desarrollo (Back)	1125,83	416	1541.83
TOTAL	—	3377,49	1248	4625.49

Figura 10: Coste trabajadores; Fuente: Propia

Impuestos devengados

En la siguiente imagen podemos ver el calendario fiscal de la empresa, donde se ven los meses que tendremos que afrontar los pagos del IVA y del impuesto de sociedades. Cabe destacar que este ultimo se empezará a pagar a partir del segundo año, si se han obtenido ingresos el primero.

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
IVA	✗			✗			✗			✗		
IMPUESTO SOCIEDADES (a partir del 2º año)					✗							

Figura 11: Impuestos devengados anuales; Fuente: Propia

Plan de operaciones

Plan de inversiones y gastos

GASTO	CONCEPTO	PRECIO (€)
ASPECTOS ADMINISTRATIVOS	Constitucion de la empresa	3000€
	Gestorias, consultorias	120€ al mes
INFRAESTRUCTURAS (INSTALACIONES)	Ubicacion Fisica	704€ al mes
	Portatiles	1000€ * 4 = 4000€
MARKETING	Plan de marketing y promocion	700€
RECURSOS HUMANOS	Salarios	1125,83€ * 3 = 3377,49€
	Seguridad social	416€ * 3 = 1248€

Figura 12: Plan de inversiones y gastos; Fuente: Propia

Resumen:

GASTOS MENSUALES	INVERSIONES
Gestorias, consultorias: 120€ /mes	Constitucion de la empresa: 3000€
Ubicacion fisica: 704€/mes	Portatiles: 4000€
Salarios: 3377,49€	Plan de marketing y promocion: 700€
S.S.: 1248€	
TOTAL: 5449,49€ /mes	TOTAL: 7700€

Figura 13: Resumen plan de inversiones y gastos; Fuente: Propia

Punto muerto/Umbra de rentabilidad

COSTES FIJOS	COSTES VARIABLES
Gestorias, consultorias: 120€/mes	
Ubicacion fisica: 704€/mes	
Salarios: 3377,49€	
S.S.: 1248€	
Constitucion de la empresa: 3000€	
Portatiles: 4000€	
Plan de marketing y promocion: 700€	
TOTAL: 13149,49€	TOTAL: 0€

Figura 14: Umbral de rentabilidad; Fuente: Propia

El precio de nuestro producto al público es de 1,18€. Al usar una tienda como Google Play, hay un porcentaje de la venta que Google se lleva. En mi caso, al ser un negocio pequeño que no factura mas de 1 millón de euros al año, Google se lleva un 15% de la compra, por lo que la aplicación nos daría un beneficio de 1€. Los costes fijos son 13149,49€/mes, y los variables son 0€. Teniendo esto en cuenta, para calcular el umbral de rentabilidad, debemos de dividir los costos fijos totales entre el coste de nuestro producto. Al ser 1€, la operación es muy fácil.

$$\text{Umbral de rentabilidad} = 13149,49\text{€} / 1\text{€} = 13149 \text{ unidades}$$

Este significa que debemos de vender esa cantidad de unidades al mes para poder alcanzar el punto de equilibrio, ya que si no estaré generando perdidas.

Plan de financiación

OBJETIVO	13149,49€
AHORROS	3000€
CROWDFUNDING	12000€
TOTAL	15000€

Figura 15: Plan de financiación; Fuente: Propia

Teniendo en cuenta que se parte con unos ahorros de 3000€, necesitaremos 10149,49€ para poder comenzar la empresa. Para este dinero que nos falta haremos un crowdfunding, donde se le darán beneficios a las personas que nos ayuden como acceso a nuevas versiones de la aplicación, actualizaciones de como va el producto, atención al cliente inmediata.... También, cabe destacar que se pide un poco mas de dinero en el crowdfunding que lo que se necesita para así tener un colchón para los meses venideros.

Plan financiero

Plan de tesorería

Para mostrar la viabilidad de la empresa, se muestra a continuación un cuadro con todos los gastos e inversiones ademas de los beneficios a lo largo del año de apertura:

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	<th>SEPTIEMBRE</th> <th>OCTUBRE</th> <th>NOVIEMBRE</th> <th>DICIEMBRE</th> <th>TOTAL</th>	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE	TOTAL
ENTRADAS DE DINERO													
Aportaciones Empresario	3000												3000
Creditos solicitados													0
Ventas				8000	9000	8000	7000	6000	5500	10000	9000	9000	72000
Subvenciones publicas													0
Intereses de la cuenta bancaria													0
Crowdfunding	12000												12000
TOTAL ENTRADAS	15000	0	0	8000	9000	8000	7000	6000	5500	10000	9000	9000	86500
SALIDAS DE DINERO													
Compras de activo fijo	4000												4000
Proveedores (mat, prim. o mercan.)													0
Alquileres, renting	704	704	704	704	704	704	704	704	704	704	704	704	8448
Gastos de constitucion	3000												3000
Seguros													0
Devolucion capital													0
Devolucion intereses													0
Seguridad Social (S.S)	1248	1248	1248	1248	1248	1248	1248	1248	1248	1248	1248	1248	14976
Plan de marketing				700						700			1400
Gestorias, consultorias	120			120		120	120			120			600
Subida aplicacion Google Play				25									25
Salarios	3377,49	3377,49	3377,49	3377,49	3377,49	3377,49	3377,49	3377,49	3377,49	3377,49	3377,49	3377,49	40529,88
TOTAL SALIDAS	12449,49	5329,49	5329,49	6174,49	5329,49	5449,49	5449,49	5329,49	5329,49	6149,49	5329,49	5329,49	72978,88
ENTRADAS - SALIDAS	2550,51	-5329,49	-5329,49	1825,51	3670,51	2550,51	1550,51	670,51	170,51	3880,51	3670,51	3670,51	13521,12

Figura 16: Plan de tesorería; Fuente: Propia

Cuenta de resultados

CUENTA RESULTADOS PREVISIONAL		1º AÑO	2º AÑO	3º AÑO
INGRESOS				
Ingresos de explotación		72000		
Ventas previstas		72000		
Ingresos financieros				
Intereses de la cuenta bancaria				
Ingresos extraordinarios		12000		
TOTAL INGRESOS		84000		
GASTOS				
Gastos de explotación		72978,88		
Compras				
Mercancías				
Acuerdos con distribuidores		25		
Servicios				
Alquileres		8448		
Reparaciones				
Asesoría		600		
Seguros				
Constitución Empresa		3000		
Publicidad		1400		
Suministros				
Gastos personal				
Sueldos		40529,88		
Seguridad Social (S.S)		14976		
Amortizaciones				
De inmovilizado		1000		
Gastos de establecimiento				
Gastos extraordinarios				
TOTAL GASTOS		69978,88		
RESULTADOS				
Resultado de explotación Ingresos de explotación - Gastos Financieros		-978,88		
Resultado financiero Ingresos financieros - Gastos financieros		0		
Resultados actividad extraordinaria Ingresos extraordinarios - Gastos extraordinarios		12000		
Resultados actividad ordinaria Res. explotación + Res. financiero + Res. extraordinario		11021,12		

Figura 17: Cuenta de resultados; Fuente: Propia

Tramites de constitución y puesta en marcha

Antes de empezar a trabajar, tenemos que indicar varios aspectos y entregar varios documentos en distintos lugares para poder constituir legalmente la empresa y poder empezar a trabajar. En la siguiente tabla, se indican los trámites a realizar.

TRAMITE	CONCEPTO	LUGAR/ORGANISMO
Certificación negativa del nombre	Para elegir el nombre de la sociedad, y que no debe de ser repetido	Registro Mercantil Central
Estatutos de la sociedad	Indica las normas de funcionamiento de la sociedad	Registro Mercantil Central
Otorgamiento de la escritura pública	Constituye la escritura de la constitución y estatutos de la S.L.U. Requiere la firma delante del notario	Acudir a cualquier Notario Colegiado
Solicitud código identificación	Obtiene el número que identifica legalmente a la empresa	AEAT del domicilio de la empresa
Liquidación del ITP y AJD	Impuesto a pagar por formar la sociedad	AEAT (De la comunidad)
Inscripción en el registro mercantil	Obligación a todas las sociedades para otorgar una personalidad jurídica	Registro provincial mercantil
Declaración censal	Documento en el que se indican las características de la actividad, así como el centro de trabajo y la modalidad tributaria	AEAT del domicilio de la empresa
Inscripción de empresa en S.S.	Obligatorio cuando se contratan nuevos trabajadores	Tesorería de la S.S del domicilio de la empresa
Inscripción de trabajadores en S.S.	Obligatorio cuando se contratan nuevos trabajadores	Tesorería de la S.S del domicilio de la empresa
Comunicar apertura centro trabajo	Es necesario informarlo en todos los casos	Consejería de trabajo de la comunidad autónoma

Figura 18: Trámites de constitución y puesta en marcha; Fuente: Propia

Análisis de Requerimientos

Requisitos funcionales

Una vez terminada la parte de la empresa, ya podemos empezar con la aplicación. Primero vamos a ver la funcionalidad de nuestra aplicación definiendo un par de requisitos funcionales, los cuales definen el como se comporta la aplicación y sus funcionalidades:

FUNCIONALIDAD	DESCRIPCION
Iniciar Sesion	Esta pantalla da la opcion al usuario de iniciar sesion con una cuenta local o con Spotify, para como guardar los datos
Busqueda	En la aplicacion se podran buscar albumes que se encuentren en Spotify, por el nombre del artista
Vista	Se podra cambiar la vista de la aplicacion para poder visualizar los albumes de diferentes (modo cuadricula, pantalla completa, lista compacta...)
Filtrar/Organizar	Se podran filtrar y organizar los albumes a partir de tags que podras asignar, asi como categorias ya creadas mas "globales"
Marcar albums	Se podran marcar albumes como favoritos, terminados o no terminados, asi como futuras escuchas
Escuchar	Se podra reproducir una muestra de cada cancion de un album
Notificaciones	Se enviaran notificaciones de albumes que han salido recientemente de artistas donde ya nos gustaban sus anteriores albumes, asi como recomendaciones y recordatorios
Importar/Exportar Datos	Sera posible exportar los datos locales como las listas o albumes que tengamos en la aplicacion para poder importar los datos en otro dispositivo

Figura 19: Requisitos funcionales; Fuente: Propia

Requisitos no funcionales

Los requisitos no funcionales en lugar de definir funcionalidades del programa como hacen los anteriores, estos son características que no estan relacionados directamente con las características específicas que hemos visto anteriormente. Aquí tenemos los requisitos no funcionales de nuestra aplicación:

CARACTERISTICA	DESCRIPCION
Velocidad/Tiempo respuesta	La velocidad de la aplicación será bastante rápida, con una carga inicial de al menos 1,5 segundos
Usabilidad	La interfaz será clara y sencilla de utilizar para que los clientes funcionen de forma eficiente
Seguridad	La aplicación protegerá la información de los usuarios y garantizará la privacidad de los datos almacenados, mediante medidas de encriptación a través de la contraseña del usuario
Compatibilidad/Requisitos	La aplicación será compatible con la gran mayoría de los dispositivos Android a día de hoy. Como mínimo, Android 4.0, y una conexión a Internet

Figura 20: Requisitos no funcionales; Fuente: Propia

Legislación aplicable

Nuestra aplicación no comparte datos con terceros y nosotros no obtenemos ningún dato por parte del usuario. Al usar la API de Spotify, es esta la que maneja las peticiones y el manejo de datos de la cuenta, pero a nivel local lo único que se almacena son los datos de los álbumes o listas que el usuario tenga. Como ya he dicho, estas se encuentran en local, cifradas y sin compartirse con nadie ya que es el propio usuario el único que tiene acceso a esos datos.

Por lo tanto, no se infinge ningún permiso de privacidad por lo que nuestra aplicación y empresa no se acoge a ninguna normativa.

Para todo esto, tendremos un apartado en nuestra aplicación donde se explicarán los términos de uso y la política de privacidad de la misma, dando la información que se ha explicado anteriormente.

Metodología. Fases del proyecto. Tareas y plazos de ejecución

Debemos de saber como vamos a organizar el desarrollo de la aplicación y cual va a ser, vamos a decir que la hoja de ruta para el programa. Para ello, tenemos primero que definir cual va a ser la metodología escogida, ademas de los plazos en los que ejecutaremos cada tarea.

La metodología escogida ha sido la metodología incremental. Esta se centra en, basándose en una idea clara de producto, ir construyéndolo poco a poco, partiendo de una base donde se le irán añadiendo nuevas funcionalidades. Se ha escogido esta metodología sobre otras debido a el poder lanzar una aplicación funcional cuanto antes para poder empezar a ganar beneficios y de ahí, ir subiendo e ir añadiendo todo lo demás poco a poco. Si los clientes nos piden alguna funcionalidad, podremos añadirla al ciclo del producto fácilmente, por lo que al final salimos beneficiándonos las dos partes.

A continuación se muestra un mapa de la metodología aplicada a nuestro producto:

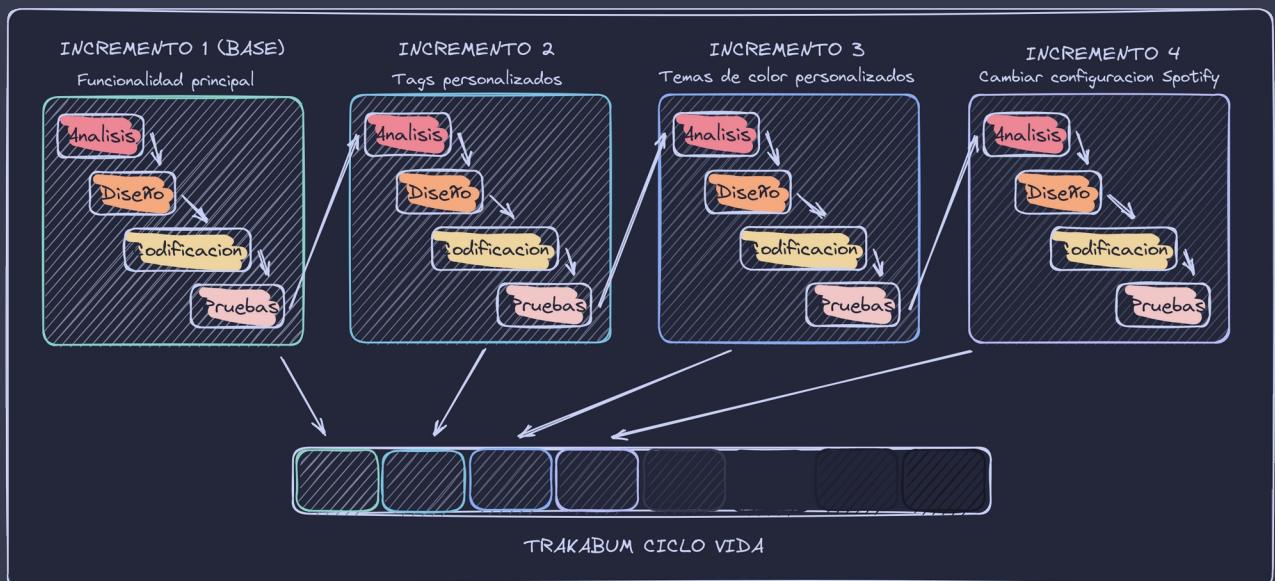


Figura 21: Ciclo de vida; Fuente: Propia

Después de exponer la metodología, pasamos a las fases del proyecto.

El proyecto cuenta con las siguientes fases:

- **Análisis:** En esta fase recopilamos y analizamos los requisitos de nuestra aplicación, se definen el alcance del proyecto, los objetivos y los requisitos funcionales y no funcionales de la aplicación.
- **Diseño:** En esta fase se *dibuja* la arquitectura de la aplicación y se crea un plan detallado para su implementación. También se determinan las herramientas y tecnologías que se usarán en la construcción de la aplicación.

- **Codificación:** En esta fase se escribe el código de la aplicación y se realiza la integración de los diferentes módulos. También se realizan pruebas unitarias para verificar que cada modulo funciona correctamente.
- **Prueba:** En esta fase se llevan a cabo pruebas mucho mas amplias de la aplicación para asegurarse de que funciona de acuerdo a los requisitos funcionales que indicamos anteriormente en la etapa del análisis. Se prueban diferentes escenarios y se corrigen errores.
- **Entrega del incremento:** En esta fase se entrega la aplicación con la funcionalidad implementada para que el cliente la pruebe y nos de su feedback.

Los plazos de ejecución de la aplicación, basándonos en las fases ya antes definidas, son los siguientes:

Fase	Complejidad	Duracion (dias)
Analisis	Media-Alta	7
Diseño	Alta	16
Codificacion	Media-Alta	14
Pruebas	Media	12
FIN		40

Figura 22: Resumen de las fases; Fuente: Propia

Diseño

Diseño de datos

Entidad-relación

A continuación se muestra el modelo entidad-relación de nuestra base de datos:

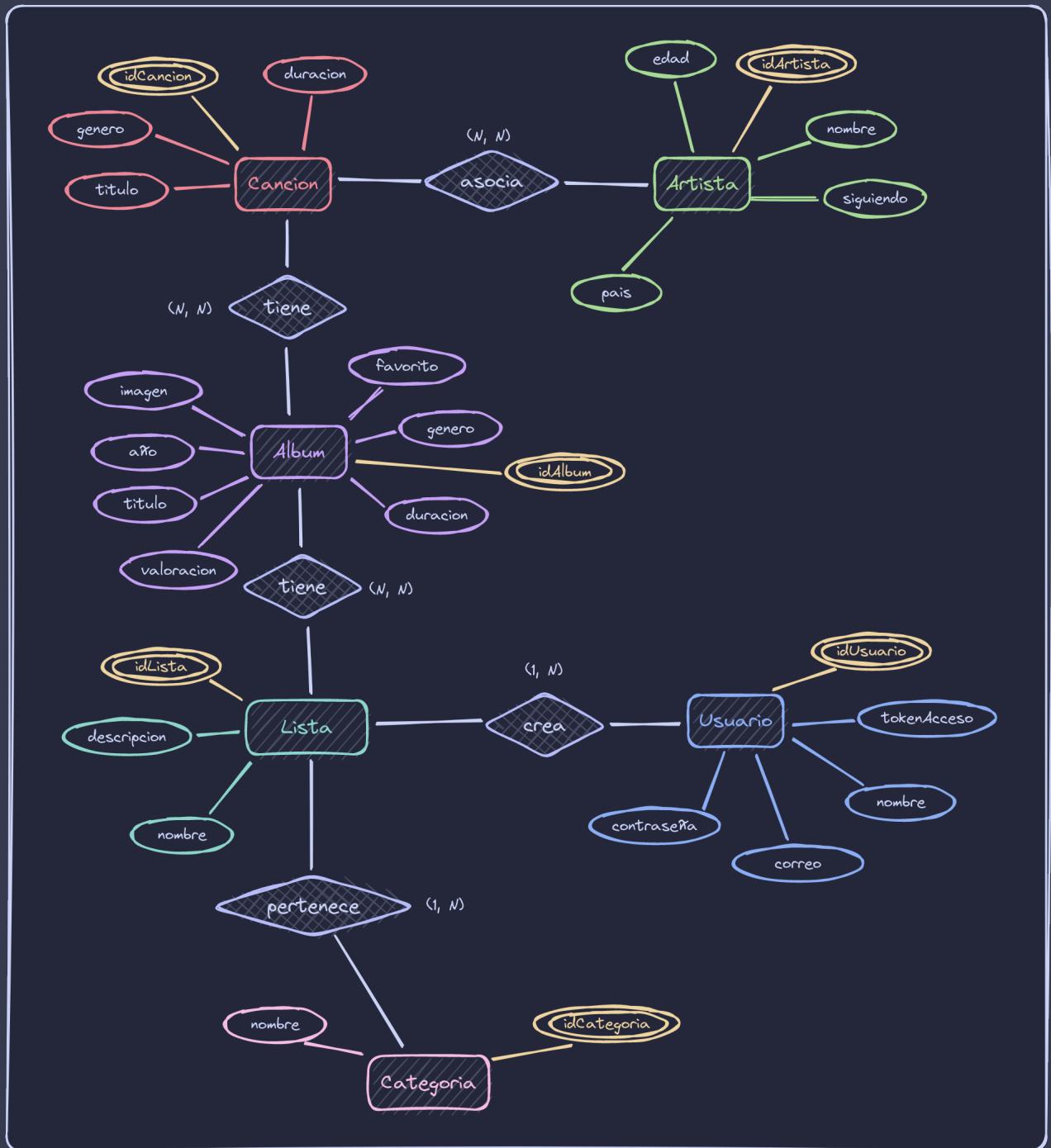


Figura 23: Entidad-relación; Fuente: Propia

Diseño lógico-relacional

Una vez visto el diseño entidad-relación, vamos a mostrar con mas detalle cada entidad de la base de datos, mostrando las claves y atributos que forman la base:

- **USUARIO** (idUsuario, nombre, correo, contraseña, tokenAcceso):
 - **PK:** idUsuario
- **CATEGORIA** (idCategoria, nombre):
 - **PK:** idCategoria
- **LISTA** (idLista, idCategoria, nombre, descripcion):
 - **PK:** idLista
 - **FK:** idCategoria → **CATEGORIA**
- **ALBUM** (idAlbum, titulo, año, genero, duracion, valoracion, imagen, favorito):
 - **PK:** idAlbum
- **ALBUM_LISTA** (idAlbum, idLista):
 - **PK:** idAlbum, idLista
 - **FK:** idAlbum → **ALBUM**
 - **FK:** idLista → **LISTA**
- **ARTISTA** (idArtista, nombre, pais, edad, siguiendo):
 - **PK:** idArtista
- **ARTISTA_CANCIÓN** (idArtista, idCancion):
 - **PK:** idArtista, idCancion
 - **FK:** idArtista -> **ARTISTA**
 - **FK:** idCancion → **CANCION**
- **CANCION** (idCancion, idAlbum, titulo, genero, duracion):
 - **PK:** idCancion
- **CANCION_ALBUM** (idCancion, idAlbum):
 - **PK:** idCancion, idAlbum
 - **FK:** idCancion -> **CANCION**
 - **FK:** idAlbum → **ALBUM**

Diseño fisico

Por ultimo, para terminar con la base de datos se muestra a continuacion el diseño fisico de la misma donde se ven los tipos de datos de los atributos junto a lo demas que hemos visto anteriormente en los otros diseños:

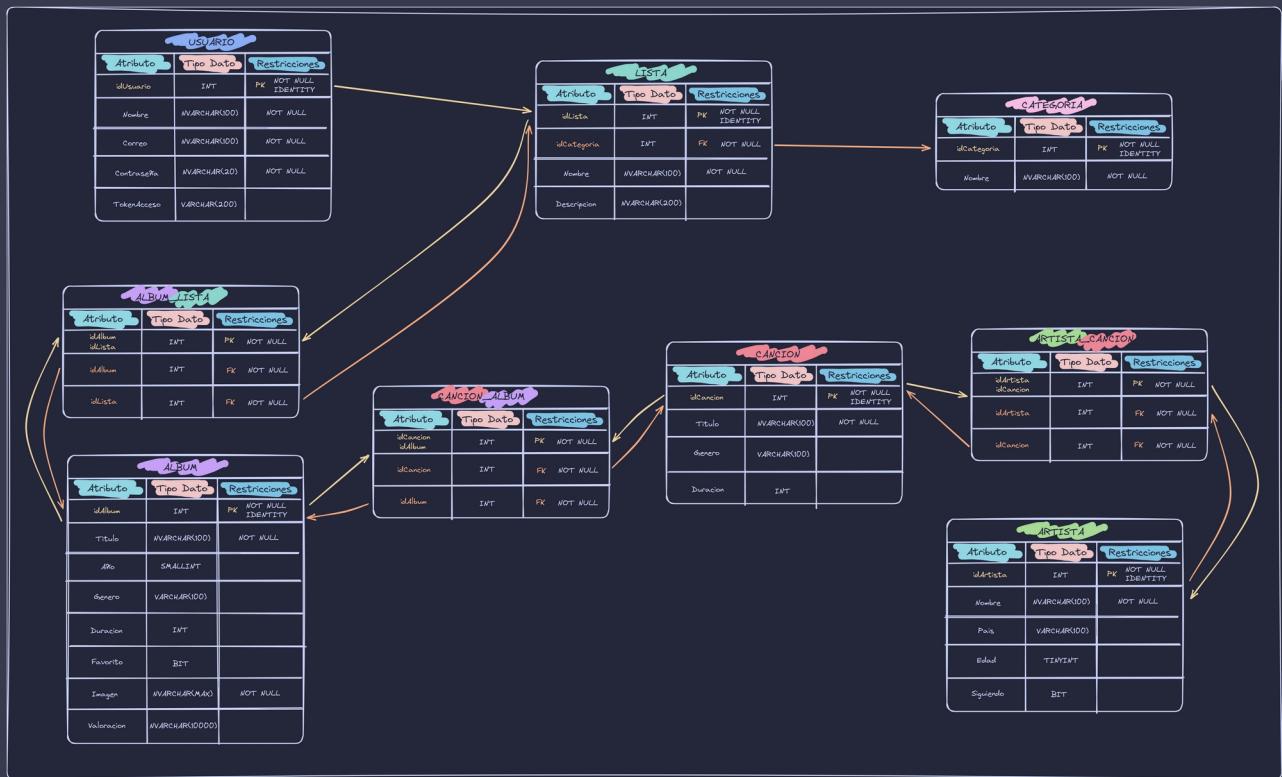


Figura 24: Diseño Fisico; Fuente: Propia

Diseño funcional

Diagrama de clases

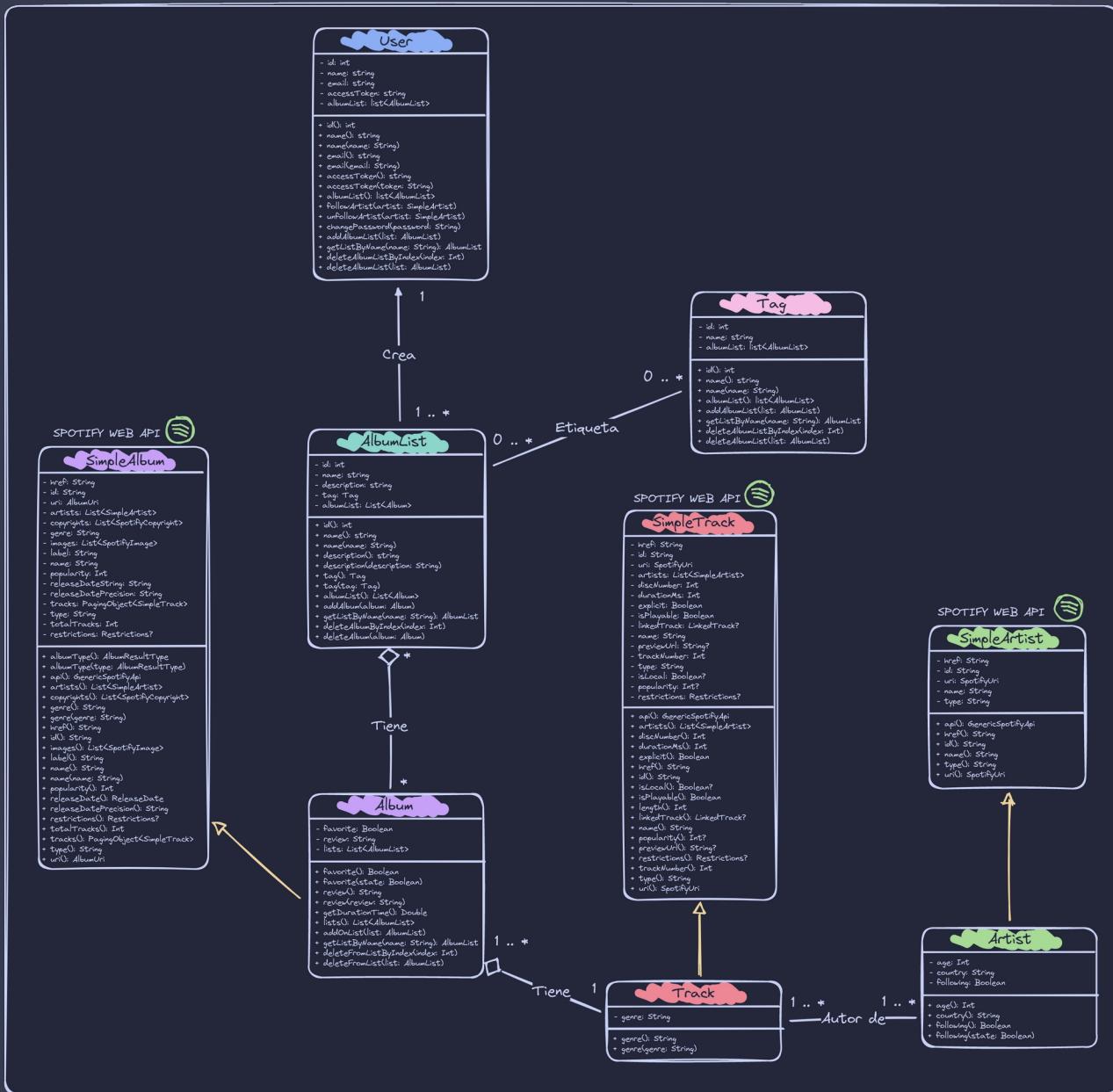


Figura 25: Diagrama de clases; Fuente: Propia

Diagrama de casos de uso



Figura 26: Diagrama de casos de uso; Fuente: Propia

Especificación de casos de uso

Busqueda de álbumes	
Descripción	El usuario puede buscar álbumes específicos de manera rápida y eficiente, usando palabras clave como el nombre del artista o álbum
Actores	Usuario
Precondiciones	<ul style="list-style-type: none"> - El usuario ha iniciado sesión en la aplicación - La aplicación está en la pantalla de búsqueda
Curso normal	<ol style="list-style-type: none"> 1. El usuario ingresa una palabra clave y le da al botón de buscar 2. La aplicación realiza una búsqueda y muestra los resultados que coinciden con la palabra clave 3. El usuario selecciona un álbum de los resultados para ver más detalles sobre él
Alternativas	<ol style="list-style-type: none"> 1.1. Si el usuario no ingresa una palabra válida, se le informa sobre ello 2.1. Si no se encuentran resultados, se muestra un mensaje con que no se encontró ningún álbum
Postcondiciones	<ul style="list-style-type: none"> - Se muestran los resultados de la búsqueda - El usuario puede ver más detalles de un álbum que sale en los resultados

Figura 27: Búsqueda de álbumes; Fuente: Propia

Visualización detalle album

Descripción	El usuario puede ver en detalle toda la información de un álbum como el año de salida, género, número de canciones, etc.
Actores	Usuario
Precondiciones	<ul style="list-style-type: none"> - El usuario ha iniciado sesión en la aplicación - El usuario tiene álbumes a la vista para poder ver
Curso normal	<ol style="list-style-type: none"> 1. El usuario selecciona un álbum que tenga a la vista 2. La aplicación muestra una nueva pantalla con toda la información sobre el álbum 3. El usuario puede dejar su opinión sobre el álbum, añadirlo a nuevas listas, o marcarlo como favorito
Alternativas	<p>3.1. Si el usuario escoge el escribir una valoración, le saldrá un "pop-up" en donde podrá seleccionar la nota, además de más contenido sobre el álbum</p> <p>3.2. Si el usuario escoge el añadir el álbum a una nueva lista, le aparecerá un "pop-up" con las listas ya existentes, para que seleccione una. También se le puede dar a un botón para añadir una nueva lista</p>
Postcondiciones	El usuario puede salir de la pestaña dándole al botón de volver atrás arriba a la izquierda

Figura 28: Visualización en detalle de un álbum; Fuente: Propia

Gestionar álbumes	
Descripción	El usuario puede agregar a listas, marcar como favorito, ocultar o eliminar de una lista.
Actores	Usuario
Precondiciones	- El usuario ha iniciado sesión en la aplicación
Curso normal	<ol style="list-style-type: none"> 1. El usuario accede una lista o busca el álbum a través de la búsqueda 2. El usuario puede agregar el álbum a otra lista 3. El usuario puede modificar la visibilidad del álbum o si es favorito o no 4. El usuario puede eliminar el álbum de una lista
Alternativas	<p>2.1. Si el usuario desea agregar un álbum a una nueva lista, la aplicación le permitirá crear una nueva lista directamente</p> <p>4.1. Si el usuario desea eliminar un álbum de una lista la aplicación muestra las listas a las que el álbum pertenece para elegir de cuál se quiere quitar</p>
Postcondiciones	- La aplicación no permite la modificación de datos como la imagen del álbum, el nombre o la cantidad de canciones

Figura 29: Gestionar Álbumes; Fuente: Propia

Gestion listas personalizadas

Descripción	El usuario puede crear, modificar y eliminar listas de álbumes personalizadas
Actores	Usuario
Precondiciones	- El usuario ha iniciado sesión en la aplicación
Curso normal	<p>1. Selecciona el botón para crear una nueva lista de álbum personalizada</p> <p>2. La aplicación muestra un diálogo para que el usuario introduzca los datos sobre la lista</p> <p>3. Una vez se introduzcan se le da al botón de crear y ya tendriamos una nueva lista</p> <p>4. Para editarla o eliminarla nos iremos a seleccionar una ya existente y le daremos al botón de editar para que nos de la opción de modificarla o eliminarla</p>
Alternativas	<p>3.1. Si el usuario crea una lista que ya existe, se le informará de que no puede hacerlo</p> <p>4.1. Si el usuario desea borrar una lista que ya tiene álbumes, se le dará un mensaje de aviso</p>
Postcondiciones	

Figura 30: Gestionar listas personalizadas; Fuente: Propia

Recomendación personalizada

Descripción	La aplicación puede proporcionar al usuario con recomendaciones personalizadas de álbumes basadas en sus preferencias e historial
Actores	Usuario
Precondiciones	<ul style="list-style-type: none"> - El usuario ha iniciado sesión en la aplicación - El usuario ha escuchado suficiente música en la aplicación para que se genere un historial de preferencias.
Curso normal	<ol style="list-style-type: none"> 1. El usuario ingresa en la pestaña de recomendaciones 2. La aplicación muestra una lista de recomendaciones 3. El usuario puede seleccionar varios álbumes y añadirlos a sus listas
Alternativas	<ol style="list-style-type: none"> 1.1. Si el usuario no tiene suficiente historial para que la aplicación le pueda sugerir álbumes, se le pondrá un mensaje de que siga usando la app hasta que se haya recolectado suficiente información
Postcondiciones	<ul style="list-style-type: none"> - El usuario tiene acceso a recomendaciones personalizadas indefinidamente

Figura 31: Recomendación personalizada; Fuente: Propia

Escuchar segmento	
Descripción	Permitir al usuario escuchar brevemente un segmento de una canción o álbum antes de decidir si agregarlo a sus listas de reproducción o no.
Actores	Usuario
Precondiciones	<ul style="list-style-type: none"> - El usuario ha iniciado sesión en la aplicación - El usuario ha buscado un álbum o canción
Curso normal	<ol style="list-style-type: none"> 1. El usuario selecciona un álbum o canción 2. La aplicación proporciona una opción de escuchar brevemente el segmento de un álbum o canción 3. El usuario selecciona la opción de escuchar 4. La aplicación reproduce un segmento
Alternativas	4.1. Si el usuario quiere escuchar otro segmento, le puede dar de nuevo
Postcondiciones	<ul style="list-style-type: none"> - El usuario puede decidir después de escuchar si quiere añadir el álbum a alguna lista

Figura 32: Escuchar segmento; Fuente: Propia

Notificaciones artistas	
Descripción	Permitir al usuario recibir notificaciones de actualizaciones de artistas a los que sigue, como nuevos lanzamientos de álbumes
Actores	Usuario
Precondiciones	<ul style="list-style-type: none"> - El usuario ha iniciado sesión en la aplicación - El usuario ha seguido a uno o más artistas
Curso normal	<ol style="list-style-type: none"> 1. La aplicación comprueba periódicamente si hay actualizaciones de los artistas que sigue el usuario 2. Si se encuentra una actualización, la app envía una notificación al usuario 3. El usuario recibe la notificación y puede seleccionar para ver más detalles sobre la misma
Alternativas	
Postcondiciones	<ul style="list-style-type: none"> - El usuario recibe notificaciones cada vez que haya una nueva actualización

Figura 33: Notificaciones de artistas; Fuente: Propia

Diseño de interfaz

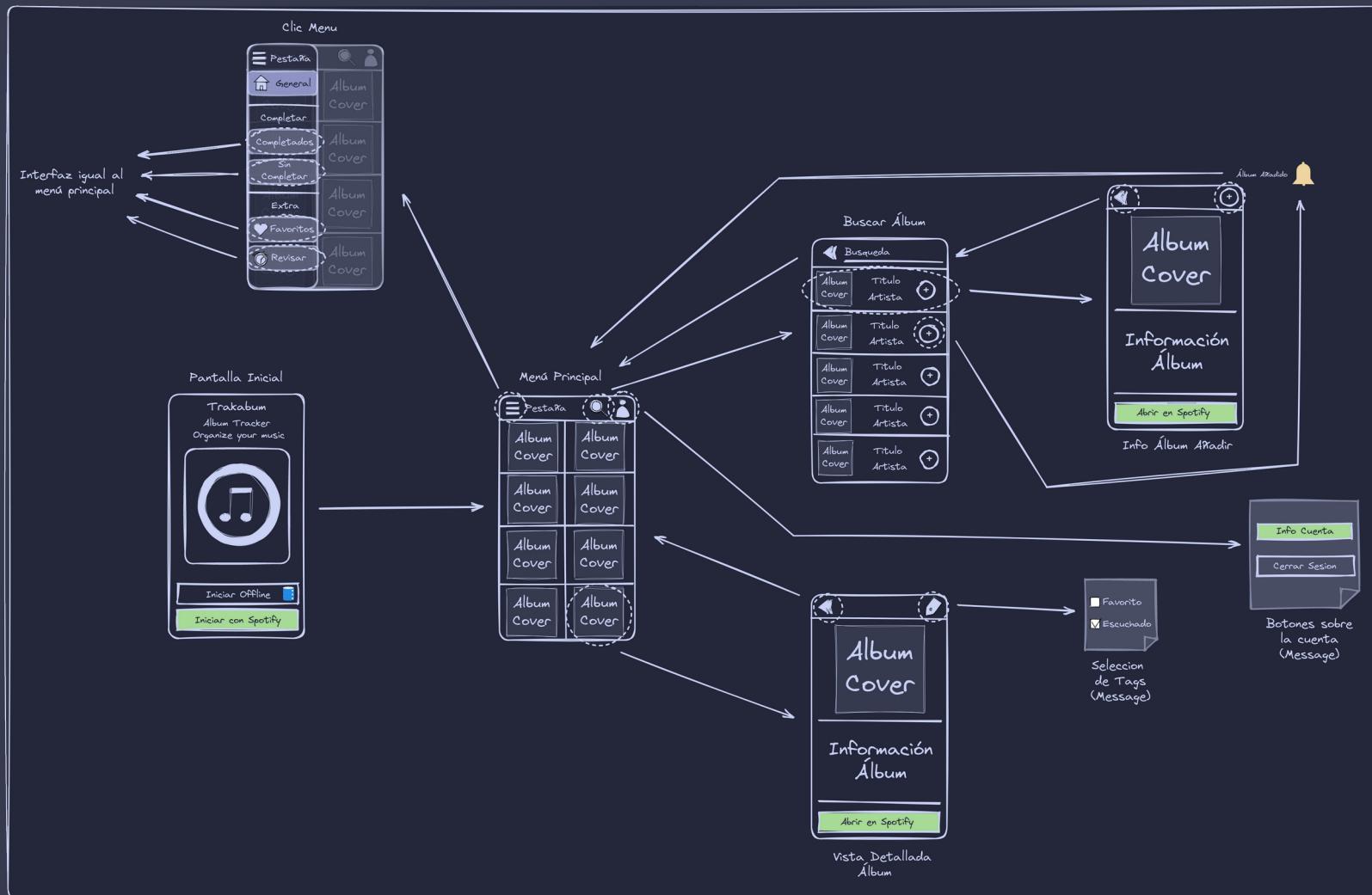


Figura 34: Diseño interfaz; Fuente: Propia

Implementación

Tecnologías a emplear

Para el desarrollo de Trakabum, se han utilizado varias herramientas diversas dependiendo de los campos de implementación de la aplicación. Las herramientas son las siguientes:

- **IDE:** Android Studio

El IDE oficial de Android. Este cuenta con bastantes herramientas para hacer el desarrollo de la aplicación mas sencillo y rápido, como la autocompletación o el formateo de código, descargar dependencias automáticamente, indicar errores antes de compilar el código o el poder debugear la aplicación lo cual es súper útil a la hora de testear tu aplicación.

- **Control de versiones:** Git

Git nos ayuda con el control de las versiones de nuestra aplicación, para así tener controlado cada cambio que se ejecuta en nuestra aplicación, para poder volver a un estado anterior si vemos que no quedamos satisfechos con la nueva versión.

- **Base de datos:** SQLite

Esta base de datos relacional como el nombre indica es muy ligera, ademas de ser gratuita, y es por ello que es una de las mas utilizadas en el desarrollo de una aplicación de Android.

- **Cliente Git:** Gitkraken

Este cliente nos ayuda a trabajar con git con una potente interfaz grafica embebida por comandos, para facilitar el uso de git.

- **Lenguaje programación:** Kotlin

Kotlin es un lenguaje moderno, con una sintaxis simple y gran potencial el cual es el que mas se esta usando actualmente para el desarrollo de aplicaciones de Android. Gracias a estos factores, se ha decidido programar Trakabum en este lenguaje y no en Java, el otro lenguaje mas usado, ya en desuso.

- **Gestor de repositorios:** Github

Github es la plataforma mas grande para alojar el código de tu aplicación, la cual utiliza git (del cual ya he hablado anteriormente).

- **Plataforma de prototipado:** Penpot

Penpot es un software open source el cual nos ayuda a crear los diseños de las interfaces de nuestra aplicación.

- **Software de seguimiento de proyectos:** Jira

Un software de gestión de proyectos en grupos el cual nos ayuda a organizarnos y saber cuales son las tareas de quienes.

Validación

Definición del procedimiento de evaluación, seguimiento y control del proyecto

Para asegurar el mejor funcionamiento de la aplicación, esta pasará por una fase de evaluación y testeo, donde se hará un seguimiento de las tareas y si se cumplen como está previsto o no.

Primero se comenzaran con las **pruebas unitarias**.

A continuacion se indican los casos de uso y las pruebas unitarias de cada uno de ellos con lo que se espera obtener:

Estas pruebas validan si cada modulo de la aplicación (los casos de uso definidos anteriormente) están bien y preparados para poder ser implementados en la aplicación final. Lo dicho, se harán pruebas viendo los caminos de control de cada modulo, por separado, y luego todos juntos, en las pruebas de integración.

Al desarrollar la aplicación en Kotlin, nos vamos a servir de la biblioteca de Junit, la cual es muy popular en los entornos Java, y nos sirve poder automatizar las pruebas que se van a realizar de cada modulo.

Si alguna prueba falla, deberemos de volver al proceso de codificación de la aplicación para hallar el error y solucionarlo.

Una vez que todas las pruebas salgan correctas, tendremos que juntar todos los módulos de la aplicación y ver que funcionan juntos. A esto se le llaman **pruebas de integración**, que como ya he dicho anteriormente, serán las siguientes a las pruebas unitarias.

Una vez que se haya cumplido con las pruebas de integración, por ultimo, se harán unas **pruebas de usabilidad** en donde se evaluará la experiencia del usuario al usar la aplicación, y ver si es intuitiva, agradable a la vista y fácil de utilizar. Para esto, se puede lanzar una beta y esperar un feedback de los usuarios para tomar decisiones acorde a lo que se nos comente.

CARACTERÍSTICA	RESULTADO ESPERADO
Busqueda Álbumes	
No se ingresa una palabra valida	Mensaje de error, informando al usuario
No se encuentra ningun album	Mensaje de error, informando al usuario
Los datos introducidos son los correctos y tienen resultado	Álbumes disponibles a elegir
Visualización detallada	
El usuario le da a "escribir valoracion" pero luego no escribe nada	Mensaje de error, informando al usuario
Añade a una nueva lista el album seleccionado, o lo agrega a una ya existente	Añadido exitoso
Gestionar álbumes	
Se quiere eliminar un album añadido a los favoritos	Mensaje de advertencia, informando al usuario
El usuario intenta añadir el mismo album a la misma lista	Mensaje de error, informando al usuario
Gestionar listas	
Se quiere crear una lista con un nombre que ya existe	Mensaje de error, informando al usuario
El usuario va a eliminar un album que tiene álbumes	Mensaje de advertencia, informando al usuario
Recomendación personalizada	
El usuario se mete en la pestaña cuando no se han obtenido los datos suficientes	Mensaje de error, informando al usuario
Escuchar brevemente segmento	
El usuario le da a la opción de escuchar	Se escucha un fragmento, siempre aleatorio
Notificaciones artistas	
Notificación enviada al haber una actualización de un artista	Usuario recibe la notificación
Usuario no sigue a un artista	Usuario no recibe la notificación

Definición de procedimientos para la participación de los usuarios en la evaluación del proyecto

Gracias a el sistema de reseñas que ofrece la Play Store, nos podremos fijar en las criticas que la aplicación reciba para poder arreglarlas o hacer frente a ellas.

Ademas del sistema de reseñas, también pasaremos una encuesta a todos los usuarios donde les haremos un par de preguntas sobre la aplicación para que nos dejen su feedback y así podamos planear cambios, mejoras o nuevas funcionalidades para un futuro.

A continuacion se muestra un ejemplo de lo que seria la encuesta ya antes mencionada:



Encuesta Valoración
Trakabum

1. ¿Con que frecuencia utilizas nuestra aplicación?

A diario
 Varias veces a la semana
 Una vez a la semana
 Ocasionalmente
 Nunca

2. ¿Encuentras útil el seguimiento de tus álbumes?

Si
 No
 No estoy seguro

3. ¿Crees que nuestra aplicación la ha ayudado?

Si
 No
 No estoy seguro

4. ¿Recomendarías nuestra aplicación a un amigo?

Si
 No
 No estoy seguro

5. ¿Qué mejorarías de nuestra aplicación?

Diseño de la interfaz de usuario
 Funcionalidad adicional (Especificar): _____
 Rendimiento de la aplicación
 Mejora en la organización (Especificar): _____
 Otras (Especificar): _____

6. Puntuación final

1-3
 4-6
 7-9
 10

Conclusiones

En resumen, este proyecto ha dado luz a **Trakabum**, una aplicación innovadora y funcional que aborda la necesidad de poder hacer un seguimiento de la música que escuchamos fácilmente.

A lo largo del proyecto he aplicado todos los conocimientos aprendidos en el Grado Superior, así como otros conocimientos que había aprendido externamente al grado.

Durante la fase de análisis y diseño, se llevo a cabo el investigar las necesidades de los usuarios para saber que producto podía triunfar y cual no, así como toda la parte de el diagrama de clases y los diagramas de base de datos.

Luego vino la parte del desarrollo del prototipo. Este consistió en una gran investigación para, en mi caso, poder implementar una API externa a un proyecto, ya que lo demás no estuvo al mismo nivel de dificultad.

Además del producto final, este proyecto me permitido conocer en gran parte lo que es desarrollar una aplicación propia, así como el trabajo en equipo, la gestión del tiempo, la resolución de problemas y la toma de decisiones.

Los planes de futuro para **MATCOM** son los de crecer como empresa, así como el de actualizar nuestra aplicación a partir de la sugerencias de los usuarios y de lo que tengamos en la lista de características planeadas.

Bibliografía

Parte Empresa

- Análisis de entornos:
 - Hubspot (2022): Análisis PESTEL: qué es, cómo se hace y ejemplos útiles.
Recuperado de <https://blog.hubspot.es/marketing/crear-analisis-pester>
 - IBEschool (2022): Las 5 fuerzas de Porter: Que son, ejemplos y como aplicarlo
Recuperado de <https://www.iebschool.com/blog/las-5-fuerzas-porter-marketing-digital/>
 - Aulacm (2022): Análisis DAFO: Guía con Ejemplos + 3 Herramientas Gratuitas.
Recuperado de <https://aulacm.com/analisis-dafo-ejemplo-plantilla/>
 - Ingenioempresa (2020): Análisis CAME: ¿Que es y como se hace?
Recuperado de: <https://www.ingenioempresa.com/analisis-came/>
- Mapa de empatía:
 - Anaivars (2020): Como hacer un mapa de empatía paso a paso
Recuperado de: <https://anaivars.com/mapa-de-empatia/>
- Plan de financiación:
 - JRA Economistas (2019): ¿De donde obtienes el dinero para montar tu negocio? El plan de financiación
Recuperado de: <https://jraeconomistas.com/plan-de-financiacion/>
- Tramites de constitución y puesta en marcha:
 - IPYME (2023): Sociedad Limitada: Creación y puesta en marcha
Recuperado de: <http://www.ipyme.org/Publicaciones/SRLCreacionPuestaEnMarcha.pdf>

Parte Aplicación

- Requisitos funcionales/no funcionales:
 - Requeridos Blog (2018): **Requerimientos Funcionales y no Funcionales, ejemplos y tips**
Recuperado de: <https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a>
- Metodología:
 - J. R. García y otros (2020): **Entornos de desarrollo - Unidad 1: Desarrollo de software**
Recuperado de: Aules (Temario de 1º DAM, link no disponible)
- Entidad-relación/Diseño lógico-relacional/Diseño fisico:
 - Carlos Mora (2021): **DB - Unidad 2: E-R Diagramas | Unidad 3: Modelo Relacional**
Recuperado de: Aules (Temario de 1º DAM, link no disponible)
- Diagrama de clases:
 - J. R. García / R. Sánchez (2020): **Entornos de desarrollo - Unidad 7: Introducción al modelado UML**
Recuperado de: Aules (Temario de 1º DAM, link no disponible)
- Diagrama y especificación de casos de uso:
 - J. R. García / R. Sánchez (2020): **Entornos de desarrollo - Unidad 8: Diagramas de casos de uso**
Recuperado de: Aules (Temario de 1º DAM, link no disponible)
- Evaluación, seguimiento y control del proyecto:
 - J. R. García (2020): **Entornos de desarrollo - Unidad 5: Diseño y realización de pruebas**
Recuperado de: Aules (Temario de 1º DAM, link no disponible)

Anexo

Para finalizar, quería explicar en mas detalle como funciona el *workflow* de la aplicación con la API y como funcionan las requests con la misma. A continuacion se encuentra el gráfico explicativo:

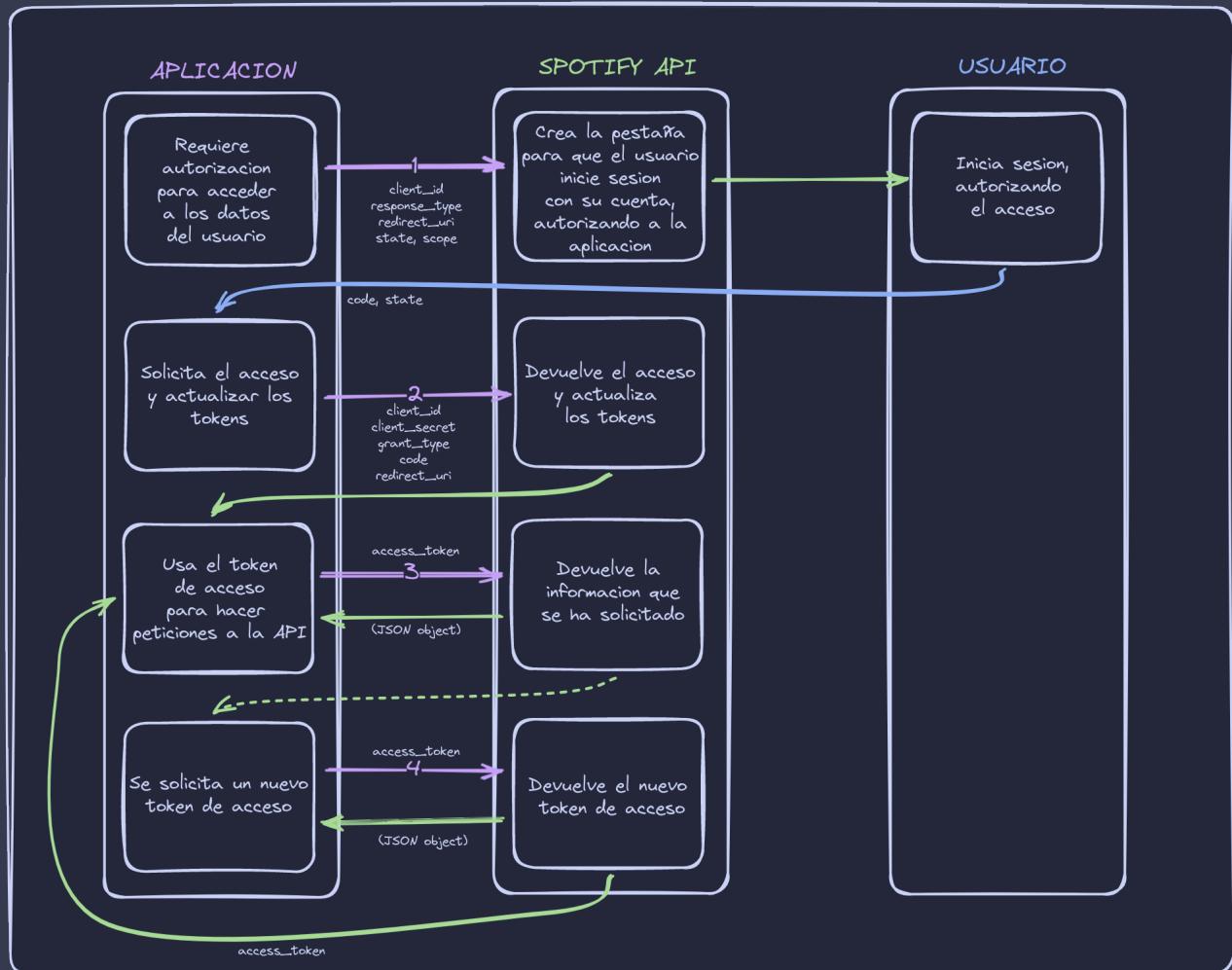


Figura 37: Spotify API Workflow; Fuente: Propia

Vamos a ir mas en detalle con ello, poniendo ejemplos desde el codigo de nuestra aplicación tambien.

1. Lo primero que hace la aplicación es lanzar una petición para autorizar el acceso con la cuenta del usuario. En Trakabum esta petición se realiza en la aplicación de login, al darle al botón de iniciar la aplicación con Spotify.

```

Trakabum
├── git
├── assets
└── docs
- src
  └── androidTest
  └── main
    └── java
      └── com
        └── slc2223
          └── trakabumkt
            ├── DefaultActivity.kt
            ├── KotlinUtils.kt
            ├── LoginActivity.kt
            ├── Model.kt
            ├── SpotifyImplicitLoginActivityImpl.kt
            ├── SpotifyPlaygroundApplication.kt
            └── UIUtils.kt
            └── VerifyLoggedInUtils.kt
      └── res
      └── AndroidManifest.xml
  └── test
  └── .gitignore
  └── build.gradle
  └── proguard-rules.pro
- gradle
  └── build.gradle
  └── gradle.properties
  └── gradlew
  └── gradlew.bat
  └── local.properties
  └── settings.gradle
  └── .gitignore
  └── LICENSE
  └── README.md

 LoginActivity.kt
 1 package com.slc2223.trakabumkt
 2
 3 import androidx.appcompat.app.AppCompatActivity
 4 import android.os.Bundle
 5 import com.adamatzman.spotify.auth.implicit.startSpotifyImplicitLoginActivity
 6 import com.slc2223.trakabumkt.databinding.ActivityLoginBinding
 7
 8 /**
 9  * This activity is the home screen of the application, responsible for handling Spotify authentication.
10  * It initiates the Spotify authentication activity and handles the response from it.
11  * @see <a href="https://developer.spotify.com/documentation/android/quick-start/authenticating-with-spotify">Authenticating with Spotify</a>
12  */
13 class LoginActivity : AppCompatActivity() {
14     private lateinit var binding: ActivityLoginBinding // ViewBinding for the Activity
15
16     /**
17      * Initiates the Spotify authentication activity.
18      * Creates the listener for the Spotify login button.
19      * @param savedInstanceState The saved instance state of the activity.
20      */
21     override fun onCreate(savedInstanceState: Bundle?) {
22         super.onCreate(savedInstanceState)
23         setContentView(R.layout.activity_login)
24         binding = ActivityLoginBinding.inflate(layoutInflater)
25         setOnCreateViewBinding(binding)
26
27         binding.btnSpotifyLoginAct.setOnClickListener {
28             startSpotifyImplicitLoginActivityImpl() // Starts the authentication activity of Spotify
29         }
30     }
31 }

```

LSP Inactive -> 2 NvimTree 11:15 Todo/42

Figura 38: Actividad de Login; Fuente: Propia

- Al darle al boton se crea una nueva actividad la cual llama a la api donde se crea la pestaña de login a traves de auth2. Aqui podemos ver los datos que se pasan, como el estado, el id del cliente, la uri de redireccion (la cual debemos de indicar en el android manifest)...

```

Trakabum
├── git
├── assets
└── docs
- src
  └── androidTest
  └── main
    └── java
      └── com
        └── slc2223
          └── trakabumkt
            ├── DefaultActivity.kt
            ├── KotlinUtils.kt
            ├── LoginActivity.kt
            ├── Model.kt
            ├── SpotifyImplicitLoginActivityImpl.kt
            ├── SpotifyPlaygroundApplication.kt
            └── UIUtils.kt
            └── VerifyLoggedInUtils.kt
      └── res
      └── AndroidManifest.xml
  └── test
  └── .gitignore
  └── build.gradle
  └── proguard-rules.pro
- gradle
  └── build.gradle
  └── gradle.properties
  └── gradlew
  └── gradlew.bat
  └── local.properties
  └── settings.gradle
  └── .gitignore
  └── LICENSE
  └── README.md

 LoginActivity.kt
 1 package com.slc2223.trakabumkt
 2
 3 import android.content.Intent
 4 import android.widget.Toast
 5 import com.adamatzman.spotify.SpotifyImplicitGrantApi
 6 import com.adamatzman.spotify.SpotifyScope
 7 import com.adamatzman.spotify.auth.implicit.AbstractSpotifyAppImplicitLoginActivity
 8
 9 /**
10  * This class is used to handle the implicit grant flow as well as to authenticate the user and get the access token.
11  */
12 class SpotifyImplicitLoginActivityImpl : AbstractSpotifyAppImplicitLoginActivity() {
13     override val state: Int = 1337 // This is used to prevent CSRF attacks, read more in the docs.
14     override val clientId: String = "2cfe9773ab0c4f48cc82e021c49a8be02" // This is the client id of the app in the Spotify Dashboard.
15     override val redirectUri: String = "spotify-sdk://auth" // This is the redirect URL of the app in the Spotify Dashboard.
16     override val useDefaultRedirectHandler: Boolean = false // This is used to prevent the SDK from automatically handling the redirect URI.
17     override fun getRequestedScopes(): List<SpotifyScope> = SpotifyScope.values().toList() // This is used to request the scopes that the app needs.
18
19 /**
20  * This method is called when the authentication is completed successfully.
21  * @param spotifyApi The Spotify API instance that can be used to make requests.
22  */
23 override fun onSuccess(spotifyApi: SpotifyImplicitGrantApi) {
24     val model = Application as SpotifyPlaygroundApplication
25     model.credentialsStore.setSpotifyApi(spotifyApi)
26     toast(this, "Authentication completed correctly!", Toast.LENGTH_LONG)
27     startActivity(Intent(this, DefaultActivity::class.java))
28 }
29
30 /**
31  * This method is called when the authentication fails.
32  * @param errorMessage The error message.
33  */
34 override fun onFailure(errorMessage: String) {
35     toast(this, "Auth failed: $errorMessage", Toast.LENGTH_LONG)
36 }
37 }

```

LSP Inactive -> 2 NvimTree 1:1 Todo/42

Figura 39: Acticidad de Spotify Auth; Fuente: Propia

Si se inicia correctamente sesion entonces se ira a la funcion de `onSuccess()`, si no si irá a la de `onFailure()`.

- Si se inicia sesion correctamente entonces se refrescaran los tokens de acceso y se solicitará el acceso a la aplicación para usar la cuenta del usuario que ha iniciado sesión. En el código, vemos que el token se va directamente a otra clase global de donde se accederá en cada request a la API. Esta clase es la de **Model**:

```

trakabum
├── .git
├── assets
├── docs
└── src
    ├── .gradle
    ├── .idea
    ├── androidTest
    ├── build
    ├── libs
    └── java
        └── com
            └── slc2223
                └── trakabumkt
                    ├── DefaultActivity.kt
                    ├── KotlinUtils.kt
                    ├── LoginActivity.kt
                    ├── Model.kt
                    ├── SpotifyImplicitLoginActivityImpl.kt
                    ├── SpotifyPlaygroundApplication.kt
                    └── Utils.kt
                    └── VerifyLoggedInUtils.kt
            └── res
            └── AndroidManifest.xml
    └── test
    └── .gitignore
    └── build.gradle
    └── proguard-rules.pro
    └── build.gradle
    └── gradle
    └── gradle.properties
    └── gradlew
    └── gradlew.bat
    └── local.properties
    └── settings.gradle
    └── .gitignore
    └── LICENSE
    └── README.md

```

```

2 O K KotlinUtils.kt X LoginActivity.kt X Model.kt X SpotifyImplicitLo... X SpotifyPlayground... X VerifyLoggedInUtil... X
Model.kt
1 package com.slc2223.trakabumkt
2
3 import com.adamstzman.spotify.auth.SpotifyDefaultCredentialStore
4
5 /**
6  * This is a singleton object that holds the [SpotifyDefaultCredentialStore] instance.
7  * Basically, this is used to store the Spotify app credentials across the application for searching functionality.
8 */
9
10 object Model {
11     val credentialStore by lazy {
12         SpotifyDefaultCredentialStore(
13             clientId = "2ef0f773a8c4f68ca82db21c49a4bed02", // Client ID of the app in the Spotify Developer Dashboard
14             redirectUri = "spotify-sdk://auth", // Redirect URI of the app in the Spotify Developer Dashboard
15             applicationContext = SpotifyPlaygroundApplication.context // Context of the Spotify Playground application
16         )
17     }
18 }

```

LSP Inactive -#2 NvimTree 1:1 Todo/42

Figura 40: Modelo de la API; Fuente: Propia

4. Ahora estamos en la DefaultActivity. Tenemos acceso a la API, por lo que solo hay que esperar a que el usuario haga alguna petición; en este caso dandole al botón de buscar un álbum.

```

trakabum
├── .git
├── assets
├── docs
└── src
    ├── .gradle
    ├── .idea
    ├── androidTest
    ├── build
    ├── libs
    └── java
        └── com
            └── slc2223
                └── trakabumkt
                    ├── DefaultActivity.kt
                    ├── KotlinUtils.kt
                    ├── LoginActivity.kt
                    ├── Model.kt
                    ├── SpotifyImplicitLoginActivityImpl.kt
                    ├── SpotifyPlaygroundApplication.kt
                    └── Utils.kt
                    └── VerifyLoggedInUtils.kt
            └── res
            └── AndroidManifest.xml
    └── test
    └── .gitignore
    └── build.gradle
    └── proguard-rules.pro
    └── build.gradle
    └── gradle
    └── gradle.properties
    └── gradlew
    └── gradlew.bat
    └── local.properties
    └── settings.gradle
    └── .gitignore
    └── LICENSE
    └── README.md

```

```

2 O K KotlinUtils.kt X LoginActivity.kt X Model.kt X SpotifyImplicitLo... X SpotifyPlayground... X VerifyLoggedInUtil... X
DefaultActivity.kt
14 /**
15  * This is the default activity that will be launched once the user has logged in correctly.
16 */
17 class DefaultActivity : AppCompatActivity() {
18     private lateinit var binding : ActivityDefaultBinding // ViewBinding for the Activity
19     private var albums: List<SimpleAlbum>? = emptyList() // List of albums found by the search
20     private var currentAlbumIndex = 0 // Index of the current album being displayed
21
22     /**
23      * This method is called when the activity is created.
24      * It creates the listeners for the buttons of searching and navigating through the album results.
25      * @param savedInstanceState The saved instance state of the activity.
26     */
27     override fun onCreate(savedInstanceState: Bundle?) {
28         super.onCreate(savedInstanceState)
29         binding = ActivityDefaultBinding.inflate(layoutInflater)
30         setContentView(binding.root)
31
32         binding.btnSearch.setOnClickListener {
33             searchAlbum(binding.txtAlbumName.text.toString())
34
35         binding.btnNext.setOnClickListener {
36             if (albums?.isEmpty() == true || currentAlbumIndex + 1 > albums?.count()!!) { // If there are no albums or the index is out of bounds
37                 toast(this, "There are no more albums.", Toast.LENGTH_SHORT)
38             }
39             currentAlbumIndex++
40
41             loadAlbum()
42         }
43
44         binding.btnPrevious.setOnClickListener {
45             if (albums?.isEmpty() == true || currentAlbumIndex - 1 < 0) { // If there are no albums or the index is out of bounds
46                 toast(this, "You have the first album so you cannot go lower on the results.", Toast.LENGTH_SHORT)
47             }
48             currentAlbumIndex--
49
50             loadAlbum()
51         }
52     }
53
54 }

```



```

2 O K KotlinUtils.kt X LoginActivity.kt X Model.kt X SpotifyImplicitLo... X SpotifyPlayground... X VerifyLoggedInUtil... X
DefaultActivity.kt
56 /**
57  * This method is called to search for an album using the Spotify API and gets the results to the albums list.
58  * @param albumName The name of the album to search for.
59 */
60 private fun searchAlbum(albumName: String) {
61     if (albumName.isEmpty()) {
62         toast(this, "Please enter an album name...", Toast.LENGTH_SHORT)
63         return
64     }
65
66     lifecycleScope.launch {
67         albums = withContext(Dispatchers.IO) {
68             guardVaildSpotifyApi(class@SpotifyImplicitLoginActivityImpl::class.java) { api ->
69                 api.search.searchAlbum(albumName).items
70             }
71         }
72
73         if (albums?.isEmpty() == true) {
74             toast(this@DefaultActivity, "No albums found.", Toast.LENGTH_SHORT)
75             returnLaunch
76         }
77
78         withContext(Dispatchers.Main) {
79             currentAlbumIndex = 0
80             loadAlbum()
81         }
82     }
83 }
84
85 /**
86  * This method is called to load the album information into the UI, based on the current album index.
87 */
88 private fun loadAlbum() {
89     binding.txtAlbumName.text = albums?[currentAlbumIndex].artists[0].name
90     binding.lblAlbumYear.text = albums?[currentAlbumIndex].releaseDate?.year.toString()
91     Glide.with(applicationContext)
92         .load(albums?[currentAlbumIndex].images[0].url)
93         .transition(DrawableTransitionOptions.withCrossFade())
94         .into(binding.imgAlbumCover)
95 }

```

LSP Inactive -#2 NvimTree 1:1 Todo/42

Figura 41-42: Actividad Busqueda de Álbumes; Fuente: Propia

5. Cuando se le de al boton de buscar se hará una peticion a la API para buscar los albumes con ese nombre a partir de lo que nos ha escrito el usuario. La aplicación lanza un hilo para la busqueda. Si el token de acceso ha caducado, se refrescará antes de hacer la peticion. La API procesa la peticion y nos la devuelve en un JSON. Aquí hay un ejemplo de petición y JSON recibido:

```
matt@matt-arch ~
❯ http GET 'https://api.spotify.com/v1/albums/4aawyAB9vmqN3uQ7FjRGTy?market=ES' \ Authorization:'Bearer undefined...undefined'
```

Figura 43: Petición HTTP a la API; Fuente: Propia

```
{ petition.json
 1 {
 2   "album_type": "compilation",
 3   "total_tracks": 9,
 4   "available_markets": ["CA", "BR", "IT"],
 5   "external_urls": { "spotify": "string" },
 6   "href": "string",
 7   "id": "2up30PMp9Tb4dAKM2erWXQ",
 8   "images": [
 9     {
10       "url": "https://i.scdn.co/image/ab67616d00001e02ff9ca10b55ce82ae553c8228",
11       "height": 300,
12       "width": 300
13     }
14   ],
15   "name": "string",
16   "release_date": "1981-12",
17   "release_date_precision": "year",
18   "restrictions": { "reason": "market" },
19   "type": "album",
20   "uri": "spotify:album:2up30PMp9Tb4dAKM2erWXQ",
21   "copyrights": [{ "text": "string", "type": "string" }],
22   "external_ids": { "isrc": "string", "ean": "string", "upc": "string" },
23   "genres": ["Egg punk", "Noise rock"],
24   "label": "string",
25   "popularity": 0,
26   "artists": [
27     {
28       "external_urls": { "spotify": "string" },
29       "followers": { "href": "string", "total": 0 },
30       "genres": ["Prog rock", "Grunge"],
31       "href": "string",
32       "id": "string",
33       "images": [
34         {
35           "url": "https://i.scdn.co/image/ab67616d00001e02ff9ca10b55ce82ae553c8228",
36           "height": 300,
37           "width": 300
38         }
39       ],
40       "name": "string",
41       "popularity": 0,
42       "type": "artist",
43       "uri": "string"
44     }
45   ],
46   "tracks": {
47     "href": "https://api.spotify.com/v1/me/shows?offset=0&limit=20",
48     "limit": 20,
49     "next": "https://api.spotify.com/v1/me/shows?offset=1&limit=1",
50     "offset": 0,
51     "previous": "https://api.spotify.com/v1/me/shows?offset=1&limit=1",
52     "total": 4,
53     "items": [
54       {
55         "artists": [
56           {
57             "name": "string",
58             "id": "string",
59             "type": "artist",
60             "uri": "string"
61           }
62         ],
63         "name": "string",
64         "popularity": 0,
65         "type": "track",
66         "uri": "string"
67       }
68     ]
69   }
70 }
```

```
57     "external_urls": { "spotify": "string" },
58     "href": "string",
59     "id": "string",
60     "name": "string",
61     "type": "artist",
62     "uri": "string"
63   }
64   ],
65   "available_markets": [ "string" ],
66   "disc_number": 0,
67   "duration_ms": 0,
68   "explicit": false,
69   "external_urls": { "spotify": "string" },
70   "href": "string",
71   "id": "string",
72   "is_playable": false,
73   "linked_from": {
74     "external_urls": { "spotify": "string" },
75     "href": "string",
76     "id": "string",
77     "type": "string",
78     "uri": "string"
79   },
80   "restrictions": { "reason": "string" },
81   "name": "string",
82   "preview_url": "string",
83   "track_number": 0,
84   "type": "string",
85   "uri": "string",
86   "is_local": false
87 }
88 ]
89 }
90 }
```

6. Una vez se reciba el JSON esta pasa a un objeto album, del que podemos sacar todas las propiedades que tiene el JSON.

Figura 44-45: Ejemplo JSON recibido; Fuente: Propia