Documento de uso de Inteligencia Artificial

1. Introducción

El uso de herramientas de **Inteligencia Artificial (IA)** se ha convertido en un apoyo fundamental en el desarrollo de software moderno. Estas herramientas permiten aumentar la productividad, reducir tareas repetitivas y facilitar el arranque de nuevas funcionalidades, especialmente en proyectos complejos como aplicaciones basadas en microservicios.

En el contexto de este proyecto, la IA se ha utilizado como **herramienta de asistencia**, nunca como sustituto del criterio técnico ni del conocimiento adquirido en la asignatura. Somos conscientes de que los modelos de IA pueden **alucinar**, generar código incorrecto o proponer soluciones que no encajan con la arquitectura del sistema. Por este motivo, todo el contenido generado ha sido **revisado, adaptado y validado manualmente** antes de integrarse en el proyecto final.

El uso de IA ha sido **responsable y controlado**, centrándose en:

- Acelerar tareas mecánicas o repetitivas.
- Obtener primeras versiones sobre las que iterar.
- Explorar alternativas de diseño o detectar errores.
- Mejorar la calidad del código, los tests y el frontend.

En ningún caso se ha delegado en la IA la toma de decisiones críticas de diseño, seguridad o arquitectura.

2. Uso de la IA por microservicios

A continuación se describe el uso de la IA en cada uno de los microservicios del proyecto. En aquellos microservicios donde el uso ha sido mínimo o inexistente, se indica explícitamente.

2.1. User-auth

En el microservicio **user-auth** hemos utilizado la IA como un recurso clave para fortalecer la seguridad y la robustez de la gestión de identidades. A continuación se detallan los usos principales.

2.1.1. Generación de tests unitarios y de integración

Se ha empleado IA para generar y refinar tests, poniendo foco en la lógica crítica de seguridad:

- Cobertura de **flujos de autenticación** (registro, login, logout).
- Validación de la lógica de **rotación de tokens** (refresh tokens, periodos de gracia y detección de reutilización).
- Escenarios complejos como la **autenticación de dos factores (2FA)**.
- Pruebas de integración para asegurar la correcta comunicación con Redis y la publicación de eventos en Kafka.

Los tests propuestos por la IA nos sirvieron de base para identificar casos borde, como la invalidación de sesiones concurrentes, y los ajustamos manualmente para cumplir con los estándares de seguridad del proyecto.

Ejemplo de prompt utilizado:

```
Genera casos de test con Vitest para el servicio de autenticación (authService.js)
cubriendo casos positivos y negativos de:
- Login exitoso devolviendo tokens
- Login con 2FA habilitado (debe devolver token temporal)
- Renovación de access token con refresh token válido
- Intento de uso de refresh token revocado
```

2.1.2. Generación y mejora de vistas del frontend

Usamos IA como asistente en el desarrollo de las interfaces de usuario relacionadas con la cuenta:

- Creación de esqueletos para los formularios de **login y registro** con validaciones de cliente.
- Implementación del asistente de configuración para el **perfil de usuario** y la activación de 2FA.
- Componentes para la visualización y edición de datos personales.

Todo el código generado fue revisado e integrado en el sistema de diseño de la aplicación para asegurar coherencia visual.

Ejemplo de prompt utilizado:

```
Ayudame a crear una funcionalidad que permita al usuario poder completar su perfil
con pasos, y que se vaya gestionando el progreso visualmente.
```

2.1.3. Resolución de bugs y problemas técnicos

Se utilizó la IA como herramienta de consulta para resolver incidencias técnicas durante el desarrollo:

- Depuración de errores de conexión con **Redis** y la **API Gateway** en entornos dockerizados.
- Análisis de **condiciones** en la lógica de renovación de tokens.
- Correcciones de estilos en la página del **perfil de usuario**.

Ejemplo de prompt utilizado:

```
Tengo un error de "Connection refused" al intentar conectar a Redis desde los
tests de integración en el pipeline de CI, pero funciona en local.
Aquí está mi configuración de docker-compose y el archivo de workflow de GitHub
Actions.
¿Qué configuración me falta para que el servicio de test vea al contenedor de
Redis?
```

2.1.4. Herramientas de IA utilizadas

Principalmente hemos utilizado el nuevo IDE desarrollado por Google, **Antigravity**, aprovechándonos de su funcionalidad estrella, el modo agente, que nos ha permitido usar **Claude** para la generación de código y tests, y **Gemini** para consultas más específicas sobre seguridad y arquitectura de microservicios. Al igual que en otros módulos, todas las sugerencias han sido validadas por el equipo de desarrollo, comprobando exhustivamente el codigo generado y modificando manualmente aquellas partes que no cumplían con nuestras necesidades.

2.2. Beats-upload

En el microservicio **beats-upload** la Inteligencia Artificial se ha utilizado como **herramienta de apoyo** durante el desarrollo. A continuación se detallan los casos de uso y herramientas empleadas.

2.2.1. Generación de tests unitarios y de integración

La IA se ha empleado para **generar y refinar tests unitarios y de integración** utilizando **Vitest**, con especial foco en:

- Cobertura de los **servicios principales** (beatService, waveformService, kafkaConsumer).
- Tests para los **middlewares de autenticación y autorización** (authMiddlewares, authorizationMiddleware, validationMiddleware).
- Validación de la lógica de **subida de archivos a S3** con URLs prefirmadas.
- Escenarios de error como archivos con extensiones/MIME types inválidos, límites de tamaño excedidos y usuarios sin permisos.

Ejemplo de prompt utilizado:

```
Genera tests unitarios con Vitest para el servicio beatService.js cubriendo:
- Generación de URLs prefirmadas para subida a S3
- Validación de extensiones de archivo (mp3, wav, flac, aac)
- Validación de tipos MIME
- Límite de tamaño de archivo
- Usuario sin permisos suficientes
- Casos de error de conexión con S3
Mockea las dependencias externas como S3, toobusy-js y el cliente de Space.
```

2.2.2. Generación y mejora de vistas del frontend

La IA se utilizó como asistente en el desarrollo de las interfaces relacionadas con la subida de beats:

- Creación de componentes para el **formulario de subida de beats** con drag & drop.
- Implementación de **barras de progreso** para la subida directa a S3.
- Visualización de **formas de onda** (waveforms) de los archivos de audio.
- Estilos y feedback visual para estados de carga, éxito y error.

Todo el código generado fue revisado y adaptado al sistema de diseño de la aplicación.

Ejemplo de prompt utilizado:

```
Crea un componente React para subir archivos de audio con drag & drop.
Debe mostrar una barra de progreso durante la subida y validar que el archivo sea
de tipo audio (mp3, wav, flac) antes de enviarlo.
Usa estilos acordes a una aplicación tipo SoundCloud.
```

2.2.3. Resolución de bugs y problemas técnicos

La IA se utilizó como herramienta de consulta para resolver incidencias técnicas durante el desarrollo:

- Depuración de errores de **CORS con S3** y URLs prefirmadas.
- Problemas de configuración de **CloudFront** para la distribución de archivos.
- Análisis de condiciones de carrera en la **subida de archivos grandes**.
- Configuración del **cliente de Kafka** para la publicación de eventos.

Ejemplo de prompt utilizado:

```
Tengo un error de CORS al intentar subir un archivo directamente a S3 usando una
URL prefirmada generada desde el backend.
El error dice "No 'Access-Control-Allow-Origin' header is present".
Aquí está mi configuración de bucket S3 y el código que genera la URL prefirmada.
¿Qué configuración de CORS necesito en el bucket?
```

2.2.4. Herramientas de IA utilizadas

Durante el desarrollo del microservicio **beats-upload** se han utilizado las siguientes herramientas:

- **GitHub Copilot**, integrado en el Visual Studio Code para autocompletado y sugerencias de código. Especialmente los modelos de Claude Sonnet 4.5, Claude Opus 4.5 y Gemini 3 Pro.
- **Antigravity IDE**,

En todos los casos, las respuestas generadas por la IA han sido revisadas críticamente y validadas antes de su integración en el proyecto.

2.3. Beats-interaction

El microservicio **beats-interaction** ha utilizado la Inteligencia Artificial siempre como **herramienta de apoyo**. A continuación se detallan los distintos casos de uso y las herramientas empleadas.

2.3.1. Generación de tests unitarios y de integración

La IA se ha utilizado para **generar suites de tests unitarios y de integración**, empleando **Vitest** como framework de pruebas, con el objetivo de:

- Cubrir **todos los caminos posibles** (escenarios positivos y negativos).
- Detectar casos límite y ramas de código complejas que podrían pasarse por alto manualmente.
- Asegurar una alta cobertura de validaciones, reglas de negocio y condiciones de error.

Los tests generados por la IA nunca se han utilizado de forma directa. En todos los casos, han sido:

- Analizados críticamente.
- Ajustados al dominio real de la aplicación.
- Adaptados a la arquitectura del microservicio y a las convenciones del proyecto.
- Refinados manualmente para garantizar precisión y coherencia con el comportamiento esperado.

Este enfoque ha permitido construir baterías de pruebas muy exhaustivas, como puede verse en entidades complejas como **Playlist**, donde se validan múltiples combinaciones de estados, referencias y reglas de negocio.

Ejemplo de prompt utilizado:

```
Genera tests unitarios con Vitest para el servicio de Playlist teniendo en cuenta:
- Validaciones de nombre y descripción
- Límites de longitud
- Usuario no autenticado
- Usuario que excede su cuota
- Casos de error controlados
- Escenarios de éxito
- Cobertura de todas las ramas condicionales
- Crea una suite de tests por cada función

Este es el servicio de playlist: {{código del servicio de playlist}}.
```

2.3.2. Generación y mejora de vistas del frontend

La IA se ha utilizado para:

- Crear una **primera versión** de algunas vistas del frontend, evitando empezar desde cero.
- Mejorar estilos CSS para que encajen con los colores, tipografía y distribución general de la aplicación.
- Proponer estructuras más limpias y coherentes de componentes React.

Posteriormente, estas vistas han sido **refactorizadas manualmente** para ajustarlas al diseño final y a los componentes reutilizables del proyecto.

Ejemplo de prompt utilizado:

```
Genera una primera versión de una vista React para mostrar playlists de un
usuario, usando una estructura clara y estilos acordes a una aplicación tipo
Spotify/SoundCloud y siguiendo los colores de la vista principal.

{{código css de la vista principal}}
```

2.3.3. Resolución de bugs y problemas técnicos

Durante el desarrollo del microservicio, la IA se ha usado como apoyo para:

- Analizar errores complejos.
- Detectar posibles causas de fallos en tests o rutas.
- Proponer soluciones alternativas a problemas concretos.

En todos los casos, las respuestas de la IA se han tratado como **hipótesis**, nunca como soluciones definitivas, y han sido validadas manualmente.

Ejemplo de prompt utilizado:

```
Tengo este error en un test de integración con Vitest.
Dime posibles causas y soluciones.

{{adjunto error}}
{{adjunto el código del test y de la ruta}}
```

2.3.4. Herramientas de IA utilizadas

Para el desarrollo del microservicio **beats-interaction**, se han utilizado distintas herramientas de Inteligencia Artificial generativa, principalmente como apoyo durante el desarrollo y la fase de pruebas.

La herramienta utilizada de forma **principal** ha sido **ChatGPT**, empleada para:

- Generación de tests unitarios y de integración.
- Creación de primeras versiones de vistas del frontend.
- Mejora de estilos y estructura de componentes React.
- Análisis de errores y resolución de bugs.

De manera **puntual**, también se han utilizado otros modelos como **Claude** y **Gemini**, principalmente para:

- Contrastar respuestas en problemas concretos.
- Obtener explicaciones alternativas sobre errores complejos.
- Comparar enfoques diferentes ante un mismo problema técnico.
- Consultar sobre la integración con Space y la especificación técnica de la ficha técnica de la API

El uso de múltiples herramientas ha permitido reducir el riesgo de errores derivados de posibles alucinaciones de un único modelo y reforzar la validación de las soluciones propuestas. En todos los casos, las respuestas generadas por estas IAs han sido revisadas críticamente y adaptadas al contexto real del proyecto antes de su aplicación.

2.4. Analytics-and-dashboards

En el microservicio de **analytics-and-dashboards**, la Inteligencia Artificial ha jugado un papel fundamental no solo en la generación de código, sino en la **toma de decisiones arquitectónicas** desde la fase de concepción.

2.4.1. Definición de la arquitectura y stack tecnológico

Antes de escribir una sola línea de código, utilizamos **Gemini Pro** para validar nuestro enfoque. Dado que el resto de microservicios del proyecto estaban planteados en Node.js (Express), nuestra intención inicial era seguir la misma línea por coherencia. Sin embargo, al plantear a la IA los requisitos de este microservicio

(cálculo de métricas de audio, análisis de señales, procesamiento de datos), nos recomendó encarecidamente el uso de **Python**.

Esta consulta fue clave: nos permitió evitar un desarrollo en Express que habría sido mucho más complejo y menos eficiente para tareas de procesamiento de audio. Gracias a esta recomendación temprana, optamos por un stack basado en **FastAPI** y librerías científicas, lo que facilitó enormemente la implementación posterior.

Ejemplo de prompt utilizado (Gemini Pro):

```
Somos un equipo de estudiantes desarrollando una aplicación de música tipo Spotify
basada en microservicios.
Tenemos que implementar un servicio de "Analytics" que debe recibir archivos de
audio, analizar sus ondas para extraer BPM, tonalidad, energía y "danceability".
El resto de compañeros está usando Node.js con Express. ¿Deberíamos usar también
Node.js para esto o hay una alternativa mejor?
Compara las librerías disponibles en Node.js vs Python para análisis de audio
(DSP).
```

2.4.2. Apoyo en el uso de Librosa y análisis de audio

Al enfrentarnos por primera vez a la librería **Librosa**, utilizamos la IA para acelerar la curva de aprendizaje. La documentación de procesamiento de señales digitales (DSP) puede ser densa, y la IA nos ayudó a traducir conceptos teóricos en implementaciones prácticas.

Se utilizó para:

- Entender cómo cargar y procesar segmentos de audio de forma eficiente.
- Implementar algoritmos para la detección de **beats** y **onset strength**.
- Extraer características espectrales como el **centroide espectral** o el **rolloff**.

Ejemplo de prompt utilizado:

```
Necesito una función en Python usando Librosa que reciba la ruta de un archivo MP3
y me devuelva un objeto JSON con:
- El tempo estimado (BPM).
- La duración en segundos.
- Un array con los picos de energía (para visualizar la onda).
Ten en cuenta que el archivo puede ser grande, así que optimiza la carga si es
posible.
```

2.4.3. Generación de tests (Pytest)

Dado que el cambio a Python implicaba usar un framework de testing diferente al del resto del proyecto (Vitest), nos apoyamos en la IA para generar la estructura base de los tests con **Pytest**.

- Tests unitarios para las funciones de cálculo de métricas (mockeando la lectura de archivos de audio).

- Tests de integración para los endpoints de FastAPI.

Ejemplo de prompt utilizado:

```
Genera un test con Pytest para el endpoint POST /analyze de FastAPI.
El test debe:
1. Mockear la función de 'librosa.load' para no depender de un archivo real.
2. Simular el envío de un archivo 'test.mp3'.
3. Verificar que la respuesta es 200 y devuelve la estructura de métricas
correcta.
4. Verificar que si el archivo no es de audio, devuelve un 400.
```

2.4.4. Generación y mejora de vistas del frontend

Para la visualización de estos datos en el frontend, utilizamos la IA para mejorar la estética de los dashboards y gráficas, asegurando que la información técnica (como el espectrograma o las barras de energía) fuera comprensible para el usuario final.

Ejemplo de prompt utilizado:

```
Tengo un componente en React que muestra una lista de métricas (BPM, Key, Energy).
Quiero que me des estilos CSS (o Tailwind) para mostrarlos como "tarjetas"
modernas, con un icono a la izquierda y el valor grande a la derecha.
Si el valor de "Energy" es alto (>0.8), quiero que el borde de la tarjeta sea
rojo, si es medio amarillo, y bajo verde.
```

2.4.5. Herramientas de IA utilizadas

En este microservicio hemos diferenciado claramente el uso de las herramientas según el propósito:

- **Gemini Pro (Web):** Utilizado principalmente en la fase de **investigación y arquitectura**. Fue la herramienta de consulta para entender conceptos de DSP (Digital Signal Processing) y tomar la decisión de usar Python.
- **Claude (VS Code Extension):** Utilizado para la **generación de código "in-code"**, refactorización y escritura de tests dentro del propio editor.

2.5. Social

En el microservicio **social** la Inteligencia Artificial se ha utilizado como apoyo durante el desarrollo. A continuación, se detallan los casos de uso y herramientas concretas utilizadas.

2.5.1. Generación de tests unitarios y de integración

El uso de la IA se centró en la generación inicial de tests unitarios y de integración, concretamente para:

- Identificar casos relevantes a cubrir en controladores y servicios.
- Proponer escenarios de éxito y error en funcionalidades como amistades, feed y mensajería.

Los tests generados sirvieron como base de trabajo y fueron revisados y refinados manualmente para adaptarlos a la lógica real del microservicio y a los requisitos del proyecto.

Ejemplo de prompt utilizado:

```
¿Qué casos debería tener en cuenta para los tests de este controlador de gestión
de conversaciones?
```

2.5.2. Generación y mejora de vistas del frontend

La IA se utilizó como asistente en la creación del esqueleto de algunas vistas del frontend relacionadas con:

- Feed de usuario.
- Lista de amigos.
- Mensajería entre usuarios.

Estas vistas iniciales permitieron evitar partir desde cero y sirvieron como base sobre la que se realizaron ajustes de estilo, estructura y comportamiento, adaptándolas al diseño final de la aplicación.

Ejemplo de prompt utilizado:

```
Genera una vista con React para un inbox de conversaciones mostrando el usuario y
el último mensaje (contenido, fecha y hora).
```

2.5.3. Documentación de la API

La IA se utilizó también como apoyo en tareas de documentación, especialmente para la generación de comentarios de **Swagger/OpenAPI** en las rutas, reduciendo el tiempo dedicado a estas tareas repetitivas.

2.5.4. Resolución de bugs y dudas técnicas

Asimismo, se empleó como herramienta de consulta para la resolución de bugs y dudas técnicas, especialmente en:

- Errores en el cliente WebSocket del frontend.
- Problemas de configuración del **Docker compose**.

Las respuestas de la IA se trataron siempre como orientaciones, siendo contrastadas y validadas manualmente antes de aplicar cualquier solución.

Ejemplo de prompt utilizado:

```
Estoy implementando un cliente WebSocket en el frontend y tengo el siguiente error
en la consola del navegador.
{{adjunto error}}
Analiza el error y dime por qué no está funcionando.
```

2.5.4. Herramientas de IA utilizadas

Durante el desarrollo del microservicio **social** se han utilizado principalmente las siguientes herramientas:

- **GitHub Copilot**, integrado en el editor para sugerencias de código.
- **ChatGPT**, como herramienta principal para la generación inicial de tests, vistas frontend y consultas técnicas.
- **Gemini**, de forma puntual para contrastar soluciones técnicas.

3. Uso de IA en documentación legal y revisión de cláusulas

Además del apoyo técnico en el desarrollo, se utilizaron herramientas de IA como asistencia en la elaboración y verificación de documentación contractual del proyecto.

- **Traducción del Customer Agreement a inglés (ChatGPT):**
  Se generó una versión en inglés del *Customer Agreement* con el único objetivo de poder evaluarlo con iCan, ya que la herramienta realiza el análisis principalmente sobre textos en inglés. La versión oficial y vinculante del contrato se mantiene en español; la traducción se utilizó únicamente como entrada para el análisis automático.

- **Detección automática de posibles cláusulas abusivas (iCan):**
  Se utilizó iCan como herramienta de apoyo para identificar cláusulas potencialmente problemáticas (p. ej., aceptación por uso, suspensión/terminación, cambios unilaterales o jurisdicción). Los resultados se interpretaron como indicativos y se revisaron manualmente, incorporando en la documentación del proyecto una justificación de los casos marcados como falsos positivos.

En todos los casos, la IA se empleó como soporte para acelerar tareas de redacción y revisión, manteniendo el criterio del equipo como referencia final.

4. Conclusión

La Inteligencia Artificial ha sido utilizada en este proyecto como una **herramienta de apoyo responsable**, alineada con buenas prácticas de desarrollo software. Su uso ha permitido:

- Aumentar la productividad.
- Mejorar la cobertura de tests.
- Reducir el tiempo de arranque de nuevas funcionalidades.
- Detectar errores y casos límite.

En todo momento se ha mantenido el **control humano** sobre el código, las decisiones de diseño y la arquitectura, garantizando que el resultado final es coherente, seguro y acorde a los objetivos de la asignatura.