

BLOQUE 0 — PROPUESTA DE VALOR Y ESTRATEGIA (Versión extendida)

0.1 Propósito y visión

Musikeeo nace con un objetivo simple pero ambicioso: **resolver la desconexión estructural del ecosistema musical independiente.**

En la práctica, los músicos, técnicos, promotores y tiendas viven fragmentados en grupos de WhatsApp, contactos informales y plataformas dispersas (Instagram, Milanuncios, Facebook Marketplace).

No existe un entorno profesional centralizado que combine visibilidad, contratación, reputación y logística.

Musikeeo pretende **convertirse en la capa digital que articule toda esa cadena**, unificando tres ejes:

1. **Descubrimiento y conexión profesional** – Encontrar músicos, técnicos o locales según estilo, ubicación y disponibilidad.
2. **Gestión y logística inteligente de eventos** – IA que organiza, recomienda y optimiza la planificación.
3. **Comunidad viva y monetizable** – Feed, reputación, marketplace y contenido generado por usuarios.

Visión a 3 años: ser **la principal red inteligente de música en vivo de Europa**, con IA generativa y matching predictivo, expandiendo desde España hacia LATAM.

0.2 Problema que resuelve

Contexto actual del mercado

- Existen miles de músicos amateurs y profesionales sin visibilidad real fuera de sus círculos.

- Los locales buscan actuaciones sin saber dónde encontrarlas.
- Los técnicos trabajan con disponibilidad irregular.
- Las tiendas de instrumentos no tienen una vía directa de contacto con los que realmente necesitan su material: los artistas activos.

Fricciones observadas

Área	Problema	Impacto
Conexión	Comunicación fragmentada (WhatsApp, Instagram, Telegram)	Baja eficiencia, pérdida de oportunidades
Confianza	Falta de reputación centralizada	Riesgo alto en contrataciones
Logística	Cero automatización en eventos	Estrés, errores, costes
Monetización	No hay control de pagos ni visibilidad de precios	Desconfianza, impagos
Comunidad	Escasa retención y feedback	Pérdida de engagement

0.3 Oportunidad

El mercado europeo de música en vivo supera los **€7.000 millones anuales**, pero el **70 %** de los contratos independientes se hacen sin herramientas profesionales.

Existe espacio para una **plataforma B2C + B2B** que mezcle el enfoque social de Instagram, la eficiencia de Glovo y la estructura de Airbnb, aplicada al entorno musical.

Además, el auge de la IA generativa y de plataformas PWA low-code (como Anti-Gravity) permite **crear una app compleja sin inversión masiva**.

Musikeeo se lanza justo en el punto de inflexión donde **los músicos adoptan IA sin fricción** (para carteles, composiciones o automatización logística).

0.4 Propuesta de valor

Tagline: “Conecta. Crea. Suena.”

Value Proposition Canvas:

- **Pain Relievers:**

- IA que encuentra músicos disponibles en tiempo real.
- Reputación transparente y verificable.
- Pagos seguros y trazables.
- Panel simple para crear eventos sin fricción.

- **Gain Creators:**

- Matching inteligente por estilo, ciudad y reputación.
- Feed comunitario que promueve colaboraciones.
- IA que genera carteles, resúmenes y sugerencias automáticas.
- Integración futura con IA musical (MusicLM / Suno).

- **Products & Services:**

- WebApp (PWA) responsive.
- AI Event Planner.
- AI Matchmaker.
- Marketplace (instrumentos y servicios).
- Chat y pagos integrados (Stripe Connect).

0.5 Segmentación de usuarios

Segmento	Descripción	Objetivo dentro de la app	Valor recibido
Músico	Persona que ofrece su talento para tocar, grabar o componer.	Encontrar bolos y colaborar.	Visibilidad, reputación, contratación.
Técnico	Profesional de sonido, luces o backline.	Ofrecer servicios y equipamiento.	Contratos, red de clientes.
Promotor / Local	Persona o empresa que organiza conciertos, bodas, eventos.	Contratar músicos o técnicos.	Rapidez, fiabilidad, automatización.

Tienda / Empresa	Proveedor de instrumentos o material de sonido.	Alquilar o vender equipo.	Marketplace dirigido y ventas locales.
-------------------------	---	---------------------------	--

Cada rol tendrá su **onboarding y dashboard adaptado**, pero todos comparten un núcleo común: el **evento** como unidad funcional central del sistema.

0.6 Modelo de negocio

Fase inicial (MVP)

- **Freemium:** registro gratuito.
- **Comisión:** 10 % por transacción cerrada a través de la plataforma.
- **Plan Pro:** visibilidad prioritaria, IA avanzada, estadísticas y portfolio audiovisual.

Fase 2 (Escalado)

- **Marketplace B2B:** comisiones por alquiler/venta.
- **Publicidad contextual:** tiendas y marcas locales.
- **Licencias white-label** para academias y ayuntamientos.

Fase 3 (Madurez)

- **Data Insights:** monetización anónima de datos agregados.
 - **API pública:** integración con plataformas de booking y escuelas.
 - **Revenue target:** €500K ARR a los 2 años.
-

0.7 Arquitectura conceptual de producto

Capas funcionales principales:

1. **User Layer:** autenticación, perfiles, roles.

2. **Interaction Layer:** chat, feed, reputación, comunidad.
3. **Event Layer:** planificación, matching, IA generativa.
4. **Commerce Layer:** pagos, facturas, marketplace.
5. **Infrastructure Layer:** seguridad, analítica, admin, automatizaciones.

Cada capa será tratada como un *micro-módulo independiente*, conectados a través de Firestore y Cloud Functions, bajo una arquitectura *event-driven*.

0.8 Posicionamiento y diferenciación

Competidor	Enfoque	Limitaciones	Ventaja Musikeeo
<i>VibraApp</i>	Conexión entre músicos (sin IA)	No integra eventos ni pagos	IA + Marketplace + Reputación
<i>Fiverr / SoundBetter</i>	Freelance remoto	No enfocado en música en vivo	Geolocalización + Instant Match
<i>Wallapop</i>	Compra-venta general	No especialización ni comunidad musical	Nicho 100 % musical
<i>Bandsintown</i>	Descubrimiento de conciertos	Sin funciones colaborativas	Ecosistema colaborativo + IA

Musikeeo combina **comunidad + IA + monetización local**.

Su ventaja diferencial radica en la **automatización de eventos y matchmaking predictivo**, inexistente en plataformas actuales.

0.9 Principios de diseño estratégico

1. **Mobile-first experience:** todo el flujo optimizado para smartphone.
2. **Time-to-value < 3 min:** cualquier usuario debe crear o encontrar un evento en menos de tres minutos.
3. **IA como copiloto, no como protagonista:** el usuario tiene control total; la IA sugiere y asiste.
4. **Local relevance:** priorizar resultados cercanos, comunidad física antes que global.

5. **Transparency & Trust:** todas las interacciones dejan trazabilidad.
 6. **No friction onboarding:** registro instantáneo con Google + selección de rol + feed adaptado.
-

0.10 KPIs iniciales (para el MVP)

- Tiempo medio de creación de evento: **< 2 min.**
 - Ratio de match exitoso: **> 60 %.**
 - Retención 30-días: **> 35 %.**
 - Usuarios activos mensuales (MAU): **> 1 000.**
 - Tasa de conversión Free→Pro: **≥ 10 %.**
 - Net Promoter Score (NPS): **> 50.**
-

0.11 IA estratégica en el producto

Musikeeo integra IA no como adorno, sino como **motor central de eficiencia y engagement.**

IA de planificación (Event Planner)

Transforma lenguaje natural → estructura de evento (fecha, lugar, estilo, instrumentos requeridos).

IA de matching

Calcula compatibilidad entre músicos según embeddings de:

- estilo musical,
- localización,
- disponibilidad,
- historial de colaboración,

- Trust Score.

IA de recomendación de oportunidades

Sugiere eventos, técnicos o material según comportamiento y contexto (tipo “Smart Feed”).

IA de soporte

Chatbot interno para onboarding, resolución de incidencias y tutoría musical básica.

0.12 Requisitos para Anti-Gravity (entregables esperados)

1. Creación de proyecto base (Musikeeo PWA):

- Stack: React 19 + Vite + TypeScript.
- Backend: Firebase (Auth, Firestore, Storage, Functions).
- IA: Gemini API (Event Planner, Matchmaker).
- Estilos: Tailwind CSS + Framer Motion.

2. Generación automática de estructura de carpetas y dependencias: `/src/components`, `/src/pages`, `/context`, `/services`, `/hooks`.

3. Configuración de Firestore rules y environment variables.

4. Implementación inicial de IA con endpoints preparados:

- `/api/event/planner`
- `/api/matchmaker`

5. Generación del documento de diseño UX (`UserFlows.json` + `sitemap`).

6. Entrega de archivo `brand.json` con design tokens y paleta.

0.13 Prompt principal (para Anti-Gravity)

You are building the complete MVP of MUSIKEEO, a Progressive Web App that connects musicians, sound engineers, venues, and instrument stores.

Objective: create a working prototype with authentication, event creation via AI, and intelligent user matching.

Stack:

- Frontend: React 19 + Vite + TypeScript
- Styling: Tailwind CSS + Framer Motion
- Backend: Firebase (Auth, Firestore, Functions, Storage)
- AI: Gemini API (Event Planner + Matchmaker)
- Deployment: Firebase Hosting

Deliverables:

1. Complete folder structure with reusable components.
2. Firestore collections for users, events, messages, reviews, and inventory.
3. AuthContext with role-based routes (musician, technician, store, promoter).
4. AI endpoints connected to Gemini for event generation and matching.
5. PWA configuration with manifest.json and service worker.
6. Admin panel skeleton for monitoring users and events.

Design Guidelines:

- Primary color: #82FF1F
 - Secondary: #101010
 - Font: Space Grotesk / Inter
 - Mobile-first layout with bottom navigation
 - Minimal, clean UI with music-related iconography.
-

0.14 Entregable final del bloque

- Documento Musikeeo_Product_Spec_Block0.pdf con 15–20 páginas de análisis estratégico.
- JSON de inicialización de proyecto para Anti-Gravity ([musikeeo_init.json](#)).

- Mapa conceptual de capas funcionales ([musikeeo_architecture.mmd](#)).
-

Fin del BLOQUE 0 – Propuesta de Valor y Estrategia.

¿Seguimos con el **BLOQUE 1 – Branding e Identidad Visual (versión extendida)**, con el mismo nivel de profundidad y formato?

SI

Perfecto 🔥

Vamos con el **BLOQUE 1 — BRANDING E IDENTIDAD VISUAL (Versión extendida)**, escrito en formato profesional y preparado para que **Anti-Gravity** genere automáticamente el sistema visual completo, incluyendo tipografía, componentes, tokens, estilos y consistencia de marca.



BLOQUE 1 — BRANDING E IDENTIDAD VISUAL (Versión extendida)

1.1 Propósito general del bloque

El branding de **Musikeeo** no es solo una cuestión estética: es la **interfaz emocional del producto**.

Define cómo se percibe la plataforma y cómo los usuarios —músicos, técnicos, promotores y tiendas— conectan con ella antes incluso de usarla.

La identidad debe transmitir:

- **Energía creativa** (ritmo, movimiento, comunidad).
- **Tecnología accesible** (IA cercana, no intimidante).
- **Profesionalismo** (confianza y fiabilidad).
- **Cultura musical urbana y mediterránea** (calidez, cercanía, diversidad).

El sistema visual debe estar preparado para escalar a una marca europea con presencia digital, audiovisual y física (eventos, merchandising, marketing local).

1.2 ADN de marca

Elemento	Descripción	Impacto visual
Nombre:	<i>Musikeeo</i> (del inglés <i>Music + Key + IO</i>). El sufijo “IO” evoca tecnología, conexión y data flow.	Representa puente entre música y tecnología.
Tagline:	<i>Conecta. Crea. Suena.</i>	Refleja las tres etapas del journey musical.
Arquetipo de marca:	<i>El Conector / El Creador.</i>	Combina energía social con innovación.
Tono:	Directo, optimista, inclusivo, sin tecnicismos innecesarios.	Cercanía humana con estética moderna.
Personalidad:	Curiosa, confiable, rítmica, colaborativa.	Identidad visual dinámica y luminosa.

1.3 Sistema de color

El color es el alma del producto. Musikeeo utiliza una paleta que equilibra el **impacto digital** con la **calidez musical**.

Rol cromático	Código HEX	Uso principal	Significado
Primary – Verde Neón	#82FF1F	Botones, acentos, iconos activos	Energía, ritmo, movimiento.
Secondary – Gris Carbón	#101010	Fondo principal (modo oscuro)	Estabilidad, tecnología, neutralidad.
Tertiary – Azul Acento	#00D4FF	Hover, links, microdetalles	Creatividad y frescura digital.
Background – Blanco Humo	#F5F5F5	Fondos claros	Limpieza visual, respiración.
Error / Alert – Coral	#FF4D4D	Alertas y errores	Urgencia.

Normas técnicas para Anti-Gravity

- Los colores deben declararse en un **theme config JSON** (`theme.js` o `tailwind.config.js`) con variables accesibles a toda la app.
 - Todos los componentes UI deben consumir estas variables para mantener consistencia.
 - Los estados hover/active deben tener contraste mínimo AA (WCAG 2.2).
-

1.4 Tipografía y jerarquía visual

Fuentes

- **Heading:** `Space Grotesk` (700/600/500).
 - Geométrica, con aire industrial.
 - Ideal para titulares y logotipos.
- **Body:** `Inter` (400/500/600).
 - Legible, moderna, excelente para interfaces digitales.

Escala tipográfica

Nivel	Tamaño	Peso	Uso
Display	48–60 px	700	Hero Titles
H1	36 px	600	Secciones principales
H2	28 px	500	Subtítulos
H3	22 px	500	Cards, modales
Body	16 px	400	Texto base

Spacing system

- Base spacing unit: **8px grid system**.
- Border radius estándar: **1rem** (16px).
- Sombras suaves (**box-shadow 0 2px 10px rgba(0,0,0,0.15)**) para profundidad sin ruido visual.

Guidelines

- El ritmo visual debe sentirse musical: equilibrio entre tensión (grotesk) y fluidez (inter).
 - Evitar fuentes serif o manuscritas.
 - Mantener contraste fuerte entre títulos (verde/azul) y fondos oscuros.
-

1.5 Logo y marca gráfica

Concepto

El logotipo debe transmitir la idea de **conexión y ritmo**: una fusión entre el “play button”, una onda sonora y una llave (key).

Versiones del logo

1. Principal (horizontal):

- Texto “MUSIKEEO” en mayúsculas.
- Ligero *forward slant* (+5°) → sensación de movimiento.

2. Isotipo:

- Círculo con trazo discontinuo que recuerda un vinilo y ondas.
- Interior con forma de “M” simétrica que sugiere plug o conexión.

3. Negativo:

- Fondo #101010, texto y trazo en #82FF1F.

4. Favicon / App Icon:

- Isotipo simplificado sin texto, formato SVG + PNG (512/192/96 px).

Reglas de uso

- Margen de seguridad = altura de la “M”.
- No alterar proporciones ni ángulos.
- Usar versión monocroma en fondos fotográficos.
- Aplicar logotipo en todas las pantallas de carga ([SplashScreen](#)).

Archivo de salida esperado: `/assets/logo/` → `musikeeo_logo_full.svg`,
`musikeeo_icon.svg`, `musikeeo_favicon.png`.

1.6 Tono y voz de la marca

Musikeeo habla como una persona real que entiende la vida del músico: sin vender humo, sin tecnicismos innecesarios, pero con energía y pasión.

Principios de tono

Principio	Descripción	Ejemplo
Cercano	Habla tú a tú, sin jerga corporativa.	“¿Listo para sonar con tu gente?”
Claro	Evita frases complejas o pasivas.	“Crea un evento. Nosotros encontramos el resto.”
Motivador	Inspira sin imponer.	“Tu talento merece escenario.”
Auténtico	Refleja comunidad real.	“No importa si tocas en un bar o en un estadio.”
Humano + Tech	Mezcla calor humano con precisión digital.	“La IA que entiende el ritmo de tu ciudad.”

Voz por contexto

- **Onboarding:** entusiasta, de bienvenida.
 - **Error / alertas:** empático (“algo no fue bien, vamos a solucionarlo”).
 - **IA output:** tono asistencial, sin pretensiones (“Te recomiendo estos músicos por su estilo y cercanía”).
 - **Email marketing:** natural y breve, sin tono de spam.
-

1.7 Iconografía y visual language

- Basada en un grid de **24x24 px**, líneas de **2 px**.
- Bordes redondeados (consistent con tipografía grotesk).
- Inspiración: *Lucide Icons* + personalización con elementos musicales (cables, jacks, notas, ondas).
- Iconos animados con microtransiciones en hover.

Set principal:

```
/assets/icons/
```

```
  🎸 guitar.svg  
  🎧 headphones.svg  
  🎤 mic.svg  
  🎹 mixer.svg  
  💗 pin.svg  
  💬 chat.svg  
  ⚡ bolt.svg  
  👤 users.svg  
  🎟 ticket.svg
```

Uso IA: Gemini puede generar iconos adicionales según categoría (instrumentos, estilos, lugares).

1.8 Imagenería y dirección visual

Fotografía

- Estilo **real-life + cinematic**.
- Luz cálida con alto contraste (tipo tungsteno / golden hour).
- Enfoque en **rostros, instrumentos y acción real**, evitando exceso de stock.
- Fondo oscuro o con luces LED difusas, para mantener coherencia con #101010.

Video y motion

- Animaciones de entrada suaves (0.5–0.8s) usando Framer Motion.
- Clips verticales para redes sociales, 9:16, formato loopable.
- Elementos visuales que siguen patrones rítmicos (tempo visual).

Backgrounds IA

Usar generadores (Imagen 3, Visual Electric o Gemini Vision) para crear backdrops con estética *studio vibe* o *urban night performance*, en tonos neón-verde y azul.

1.9 Component Library y Design Tokens

Generar sistema completo para Anti-Gravity con variables y componentes reutilizables.

Estructura esperada:

```
{
  "colors": {
    "primary": "#82FF1F",
    "secondary": "#101010",
    "accent": "#00D4FF",
    "background": "#F5F5F5",
    "error": "#FF4D4D"
  },
  "typography": {
    "heading": "Space Grotesk",
    "body": "Inter",
    "scale": [60, 36, 28, 22, 16, 14]
  },
  "radius": "1rem",
}
```

```
        "shadow": "0 2px 10px rgba(0,0,0,0.15)"  
    }  
  

```

Componentes UI base:

Button, Input, Card, Badge, Modal, NavTabs, Avatar, Toast.

Todos deben exportarse como módulos reutilizables en [/components/ui](#).

1.10 Implementación para Anti-Gravity

1. **Crear carpeta** `/design/branding` con subcarpetas `/tokens`, `/logos`, `/guidelines`.
 2. **Generar** `brand.json` con tokens de color, tipografía y espaciado.
 3. **Renderizar automáticamente** logo **SVG** a partir de la descripción conceptual.
 4. **Aplicar tokens en Tailwind** dentro de `theme.extend`.
 5. **Conectar librería de iconos personalizada**.
 6. **Crear página** `/styleguide` accesible solo en modo dev.
-

1.11 Prompt principal para Anti-Gravity

You are building the full visual identity system for the Musikeeo App.

Goals:

- Generate color tokens, typography, spacing, shadows, and border-radius configuration.
- Create a brand folder with SVG logos, favicon, and app icons.
- Implement Tailwind theme extension with brand variables.
- Generate reusable UI components (Button, Card, Input, Modal, NavTabs, Badge, Toast).
- Build a `/styleguide` page displaying all components and states.

Brand guidelines:

- Primary: #82FF1F
- Secondary: #101010
- Accent: #00D4FF
- Typography: Space Grotesk (headings) + Inter (body)
- Rounded corners, 8px grid spacing, shadows soft.
- Tone: urban, creative, professional, musical.

Output:

1. /design/branding/brand.json
 2. /components/ui/*
 3. /assets/logo/*
 4. Tailwind theme config ready for deployment.
-

1.12 Entregables esperados

- `brand.json` — tokens globales.
 - `musikeeo_logo_full.svg` + variantes.
 - `tailwind.config.js` extendido.
 - `/components/ui/` con kit base.
 - Guía de uso visual (`styleguide.html`).
-

Fin del BLOQUE 1 — Branding e Identidad Visual (versión extendida).

Este bloque proporciona suficiente información para que Anti-Gravity genere automáticamente todo el *design system* completo de Musikeeo.

¿Seguimos con el **BLOQUE 2 — UX Research y User Journey (versión extendida)**, también al máximo nivel de detalle (usando todo el espacio posible)?

SIGUE

Perfecto 🔥

Entramos ahora en el **BLOQUE 2 — UX RESEARCH Y USER JOURNEY (versión extendida)**.

Este bloque es crítico: define la **experiencia humana** detrás de Musikeeo —cómo piensa,

siente y actúa cada tipo de usuario— y ofrece a Anti-Gravity la guía completa para crear flujos, pantallas, prototipos, prompts IA y datos que representen comportamiento real.

Vamos con ello:

BLOQUE 2 — UX RESEARCH Y USER JOURNEY (Versión extendida)

2.1 Propósito del bloque

El objetivo de este bloque es convertir la visión del producto en **experiencia viva**.

El UX Research sirve como la base de toda decisión posterior: cómo se estructuran las pantallas, cómo se comporta la IA, y cómo los usuarios logran resultados con el mínimo esfuerzo posible.

En Musikeeo, la **experiencia de usuario no se centra en navegar una app, sino en lograr una meta real**: tocar, contratar o crear música.

La app debe desaparecer y convertirse en extensión natural de la interacción entre personas.

2.2 Metodología UX

Para esta primera versión del producto (MVP), aplicaremos una metodología de diseño **Lean UX** con enfoque **Human-AI Interaction**.

Se basa en cuatro principios:

1. **Empatía** – Comprender el contexto emocional y operativo de los usuarios.
2. **Simplificación** – Reducir fricciones, clics y pantallas.
3. **Contextualidad** – Personalizar contenido según el rol y la ubicación.
4. **Asistencia inteligente** – La IA debe anticiparse, no solo responder.

Las decisiones de UX se basan en entrevistas reales con músicos, técnicos y promotores (fuentes: 94 Music y Lady Jarana), además de análisis de benchmarks (Airbnb, Fiverr, SoundBetter, Glovo, Wallapop).

2.3 Arquetipos de usuario (Personas)

Se definen cuatro **personas base** que representan los segmentos clave de Musikeeo. Cada una incluye motivaciones, frustraciones, tareas, comportamientos digitales y tono emocional esperado.



Persona 1 — Carlos “El Guitarrista Freelance”

- **Edad:** 28
- **Ubicación:** Barcelona
- **Nivel técnico:** alto (estudia sonido, toca en varias bandas).
- **Motivación:** tener bolos constantes y colaborar con músicos nuevos.
- **Dolores:** no encuentra baterías ni bajistas cuando los necesita; odia coordinar por WhatsApp.
- **Comportamiento digital:** usa Instagram y TikTok para buscar eventos, pero sin sistema claro.
- **Meta en Musikeeo:** encontrar oportunidades cercanas y fiables sin perder tiempo.

Flujo ideal:

1. Se registra → elige rol “Músico”.
 2. Completa instrumentos, géneros y disponibilidad.
 3. Abre “Feed de oportunidades” → ve evento “Rumba en Gràcia”.
 4. Entra → pulsa “Postularme”.
 5. IA le sugiere: “Tu perfil coincide con este evento (80% compatibilidad)”.
 6. Cierra el trato por chat integrado.
 7. Tras el evento, deja review y gana puntos de reputación.
-



Persona 2 — Lucía “La Técnica de Sonido”

- **Edad:** 32
- **Ciudad:** Madrid.
- **Motivación:** conseguir más bolos, alquilar su backline.
- **Frustraciones:** locales con presupuestos poco claros; malas experiencias con músicos impuntuales.
- **Uso digital:** prefiere apps simples; usa Drive, Gmail, Instagram.
- **Meta:** tener calendario lleno de eventos y automatizar facturas.

Flujo ideal:

1. Accede → elige “Técnico”.
 2. IA le pide subir equipamiento disponible.
 3. Aparece en el mapa local de servicios técnicos.
 4. Recibe mensajes de promotores o músicos.
 5. Acepta contrato → pago gestionado con Stripe Connect.
 6. Recibe resumen semanal IA (“5 eventos completados / reputación +2”).
-



Persona 3 — Andrea “La Dueña de Tienda”

- **Edad:** 24
- **Ubicación:** Valencia.
- **Motivación:** aumentar ventas y alquiler de instrumentos.
- **Frustración:** no llegar al público activo (la mayoría compra online en Amazon).
- **Comportamiento digital:** usa WhatsApp Business y redes, sin sistema de reservas.
- **Meta:** ofrecer su stock a músicos locales que tocan cada semana.

Flujo ideal:

1. Se registra → rol “Tienda / Empresa”.
 2. Publica inventario (micros, amplis, cables).
 3. IA recomienda sus productos en eventos cercanos.
 4. Recibe pedidos urgentes → entrega express.
 5. Gana reputación con cada operación finalizada.
-

Persona 4 — Sergio “El Promotor / Local”

- **Edad:** 35
- **Ciudad:** Sevilla.
- **Motivación:** llenar su bar con música en vivo sin complicaciones.
- **Frustración:** coordinar bandas y técnicos cada semana.
- **Uso digital:** Facebook, WhatsApp, Booking.
- **Meta:** tener un sistema donde publique un evento y la IA lo resuelva.

Flujo ideal:

1. Entra → rol “Promotor / Local”.
 2. Dice por voz: “Quiero una banda de pop-rock el viernes a las 22h.”
 3. IA crea evento, busca músicos y envía invitaciones.
 4. Se genera cartel automático con nombres y estilos.
 5. Los músicos confirman → evento listo.
 6. Después del bolo, IA genera review y factura.
-

2.4 Escenario emocional de uso

Musikeeo debe alinearse con **las emociones reales del músico independiente**: ansiedad, ilusión, incertidumbre, adrenalina y comunidad.

La app debe transmitir **seguridad, inmediatez y ritmo**.

Por eso se prioriza:

- **Cero pantallas vacías**: siempre hay algo relevante que hacer.
 - **Lenguaje positivo y acción inmediata**: botones con verbos (“Conectar”, “Crear evento”, “Tocar”).
 - **IA empática**: el asistente debe hablar con tono humano (“Vale, te busco percusionistas cerca”).
-

2.5 Flujo general del usuario (User Journey global)

Etapas del journey:

1. **Descubrimiento**: ve publicidad o recomendación.
 2. **Onboarding**: se registra, elige rol, completa perfil.
 3. **Exploración**: navega feed, mapa, chat.
 4. **Acción**: crea evento o se postula.
 5. **Colaboración**: interactúa con otros usuarios.
 6. **Resultado**: evento realizado o contratación.
 7. **Refuerzo**: review + reputación + sugerencias futuras.
-

2.6 Flujos UX principales (User Flows)

Flow 1 — Registro e Identidad

Landing → Registro con Google → Selección de rol → Completar perfil
→ Feed inicial

Flow 2 — Creación de evento

Botón + Crear → Texto o voz → IA genera evento → Confirmar → Publicar

Flow 3 — Matching IA

Usuario crea evento → IA analiza ubicación + estilo → Muestra lista de músicos compatibles → Enviar invitación

Flow 4 — Chat y contratación

Abrir chat → Negociar → Confirmar → Stripe bloquea pago → Evento cerrado

Flow 5 — Review post-evento

Finalizar evento → IA genera resumen → Usuario confirma → Valoración automática

2.7 Arquitectura emocional y cognitiva

Cada pantalla debe guiar una **intención** clara del usuario.

Los tres ejes de diseño UX se resumen en el principio **M-A-R**:

Eje	Significado	Aplicación en Musikeeo
Motivación	qué desea el usuario	mostrar progreso visible (reputación, puntos, logros)
Acción	qué hace realmente	acciones grandes y verbales, accesibles con un dedo
Recompens	qué obtiene a	feedback inmediato: mensajes IA, badges, resultados rápidos

El ciclo M-A-R asegura una experiencia adictiva pero positiva, basada en **autonomía + reconocimiento**.

2.8 Mapa de empatía (resumen visual conceptual)

EL MÚSICO PIENSA: "Necesito tocar más."

SIENTE: inseguridad, deseo de reconocimiento.

DICE: "No hay sitios que paguen bien."

HACE: busca en Instagram, envía mensajes sueltos.

MUSIKEO RESPONDE: "Te conectamos con quien necesita tu sonido, sin perder tiempo."

2.9 Diseño de experiencia asistida por IA

La IA debe sentirse como **un miembro más de la comunidad**, no como una máquina.

Principios:

- **Proactividad:** sugiere sin esperar input.
- **Contextualidad:** recuerda patrones (ciudad, género, disponibilidad).
- **Human-Tone:** mensajes breves, coloquiales, musicales.
- **Aprendizaje continuo:** adapta recomendaciones según comportamiento.

Ejemplo:

"Hey Carlos 🎸, este finde hay una rumba cerca de Gràcia.
Buscan guitarrista como tú. ¿Quieres apuntarte?"

2.10 UX Metrics

Para medir calidad de experiencia se monitorizarán:

Métrica	Meta	Interpretación
Task success rate	> 90 %	% de usuarios que logran crear o encontrar un evento
Time on task	< 3 min	Rapidez en completar acciones
Retención día 7	> 30 %	Engagement sostenido
Error rate	< 5 %	Fluidez técnica
IA satisfaction (CSAT)	> 4.5 / 5	Nivel de satisfacción con IA
Net Promoter Score	> 50	Lealtad emocional al producto

2.11 Implementación técnica para Anti-Gravity

Anti-Gravity debe crear automáticamente los siguientes elementos:

1. **UX Blueprint (Mermaid o JSON Graph):**

- user journeys, flows, nodes y relaciones.
- etiquetas de intención (“connect”, “create”, “hire”).

2. **Archivo `/flows/UserJourney.json`**

- cada rol = subgrafo independiente.

3. **Componente `/pages/onboarding.tsx`**

- step-by-step con selección de rol, avatar e instrumentos.

4. **Hook `useUserFlow()`**

- gestiona estado de progreso del usuario.

5. **Integración con IA:**

- endpoint `/api/assistant/onboarding` que da la bienvenida y personaliza experiencia.

2.12 Prompt principal para Anti-Gravity

You are designing the complete UX flow of Musikeeo.

Objectives:

- Build full user journeys for four roles (musician, technician, store, promoter).
- Create JSON blueprint with flow nodes, triggers, and success criteria.
- Implement onboarding and feed screens based on each persona.
- Integrate AI assistant that guides the user during onboarding and event creation.

Design principles:

- Time-to-value under 3 minutes.
- Actions must be verbal and contextual.
- Positive emotional tone and feedback loops.
- Local-first experience.

Output:

1. /flows/UserJourney.json
 2. /pages/onboarding.tsx
 3. /hooks/useUserFlow.ts
 4. Onboarding IA endpoint.
-

2.13 Entregables esperados

- Documento UX completo ([Musikeeo_UX_Research.pdf](#)).
 - [UserJourney.json](#) con 4 flujos principales.
 - Prototipo navegable básico ([/onboarding](#), [/feed](#), [/chat](#)).
 - Panel analítico UX para medir fricción y abandono.
-

Fin del BLOQUE 2 — UX Research y User Journey (versión extendida).

¿Continuamos con el BLOQUE 3 — Wireframe y Mapa de Páginas (versión extendida, un mensaje completo solo para eso)?

sigue

Perfecto 🔥

Entramos ahora en el BLOQUE 3 — WIREFRAME Y MAPA DE PÁGINAS (versión extendida).

Este bloque define la estructura navegable completa de Musikeeo, el esqueleto de todas las pantallas y su jerarquía funcional. Está escrito en formato técnico y narrativo, de modo que Anti-Gravity pueda generar tanto el diseño visual como la lógica de enrutamiento, navegación y coherencia UI/UX.



BLOQUE 3 — WIREFRAME Y MAPA DE PÁGINAS (Versión extendida)

3.1 Propósito del bloque

El wireframe define **la arquitectura visual y funcional del MVP**, mostrando cómo los usuarios se mueven dentro de la aplicación, qué pantallas existen, qué acciones principales ofrece cada una y cómo se enlazan.

Debe permitir que Anti-Gravity pueda generar:

- **Estructura completa de rutas** ([/src/pages](#)).
- **Componentes visuales base** (layout, navbar, footer, modales).
- **IA contextual** integrada en los puntos clave (eventos, feed, onboarding).
- **Responsividad completa** (mobile-first → tablet → desktop).

Musikeeo funciona como **una WebApp progresiva (PWA)** con experiencia tipo app nativa. Por tanto, la navegación debe ser fluida, con persistencia de sesión y sin recargas completas.

3.2 Principios de arquitectura de información

1. **Mobile-first navigation:** barra inferior (tab bar) con iconos universales.
2. **Top navigation contextual:** solo en escritorio o pantallas amplias.
3. **Deep linking:** cada entidad (evento, usuario, tienda) tiene URL única.
4. **Consistencia visual:** los mismos patrones de layout se repiten en todos los módulos.
5. **Accesibilidad:** texto grande, contraste alto, navegación por teclado posible.
6. **Performance:** lazy loading, skeletons y scroll virtual para listas.
7. **Simplicidad cognitiva:** máximo 3 niveles de jerarquía.

3.3 Estructura global (site map lógico)

```
/  
└── /landing  
└── /register  
└── /login  
└── /onboarding  
└── /feed  
    ├── /feed/opportunities  
    ├── /feed/events  
    ├── /feed/stores  
    └── /feed/community  
└── /event  
    ├── /event/create  
    ├── /event/[id]  
    └── /event/edit/[id]  
└── /profile  
    ├── /profile/[username]  
    ├── /profile/edit  
    ├── /profile/settings  
    └── /profile/reviews  
└── /messages  
    ├── /messages/[chatId]  
    └── /messages/new  
└── /marketplace  
    ├── /marketplace/instruments  
    ├── /marketplace/services  
    └── /marketplace/add  
└── /ai  
    ├── /ai/event-planner  
    ├── /ai/matchmaker  
    └── /ai/support  
└── /admin  
    ├── /admin/users  
    ├── /admin/events  
    └── /admin/stats  
└── /legal  
    ├── /legal/terms  
    ├── /legal/privacy  
    └── /legal/cookies
```

3.4 Layouts principales

Layout 1 — Public (no autenticado)

Usado en [/landing](#), [/register](#), [/login](#), [/legal](#).

- Header con logotipo Musikeeo + CTA “Registrarse”.
- Footer con links a privacidad, contacto y redes.
- Fondo claro o con textura fotográfica difusa.

Layout 2 — App (autenticado)

Usado tras login.

- **Navbar inferior** con 5 íconos:
 -  Feed
 -  Eventos
 -  Crear
 -  Mensajes
 -  Perfil
- **Header dinámico:** título y botón IA según contexto.
- **Floating Action Button (FAB)** para acciones contextuales (ej. “+ Crear evento”).

Layout 3 — Admin

- Menú lateral fijo.
 - Estadísticas, gestión de usuarios y reportes IA.
 - Tabla dinámica conectada a Firestore Admin SDK.
-

3.5 Páginas principales (wireframe textual y funcional)

A continuación se detalla la función y el contenido de cada página, con indicaciones de diseño y comportamiento.

3.5.1 /landing — Página pública principal

Objetivo: convertir visitantes en registros.

Elementos clave:

- Hero visual con CTA “Regístrate gratis”.
- Explicación “Cómo funciona”.
- Animación ligera (Framer Motion) de músicos conectando.
- Testimonios simulados.
- Footer con links legales.

Interacción IA (opcional):

“¿Eres músico o promotor? Cuéntame en qué ciudad estás y te muestro cómo funcionaría Musikeeo para ti.”

3.5.2 /register y /login

- Formulario minimalista con botones sociales (Google, Apple, Email).
- Validaciones en tiempo real.
- Redirección automática al onboarding.
- IA microcopy (“Bienvenido 🎵 ¿Listo para conectar?”).

3.5.3 /onboarding

Estructura paso a paso:

- ① Selección de rol (músico, técnico, tienda, promotor).
- ② Datos básicos (nombre, ciudad, instrumentos/servicios).

- 3** Foto de perfil (upload + crop).
- 4** Preferencias IA (género musical, disponibilidad).
- 5** Confirmación → acceso al feed.

IA contextual: genera texto de bienvenida según rol y tono.

Ejemplo:

“Genial, Carlos 🎸 . Ya te tengo localizado en Barcelona. Te enseñaré los bolos cercanos cuando haya movimiento.”

3.5.4 /feed

Vista principal tras login.

Diseñada como mezcla entre red social y buscador.

Estructura:

- Barra de búsqueda superior (“Buscar músicos, eventos o tiendas”).
- Tabs:
 - “Oportunidades” (eventos disponibles).
 - “Eventos activos”.
 - “Tiendas y servicios”.
 - “Comunidad”.
- Scroll vertical infinito con cards.
- Filtros flotantes (ubicación, género, fecha).

Card tipo evento:

- Imagen o ícono de estilo musical.
- Título (“Buscamos percusionista para rumba”).
- Localización y fecha.
- Tag reputación (“Promotor verificado”).
- CTA: “Ver detalles”.

Interacción IA:

sugerir contenido nuevo basado en uso.

3.5.5 /event/create

El corazón de Musikeeo.

Permite crear eventos por voz o texto.

Elementos:

- Campo de texto/voz → “Describe tu evento”.
- Botón IA “Generar automáticamente”.
- Resultado editable con:
 - Nombre del evento.
 - Descripción.
 - Fecha y hora.
 - Ubicación (mapa con geolocalización).
 - Necesidades (instrumentos, géneros).
 - Presupuesto.

Funciones IA:

- Completa automáticamente campos faltantes.
 - Sugerencias de nombres y estilos.
 - Recomendación de músicos cercanos.
-

3.5.6 /event/[id]

Página individual del evento.

- Cabecera con imagen, título y género.

- Descripción.
 - Mapa con ubicación.
 - Lista de participantes confirmados.
 - Botón “Postularme” o “Contratar”.
 - Chat instantáneo.
 - IA contextual (“Este evento encaja con tu perfil al 85%”).
-

3.5.7 /profile/[username]

Perfil público del usuario.

- Foto, nombre y rol.
- Bio + instrumentos / servicios.
- Reputación (Trust Score).
- Reviews y portafolio multimedia.
- Botón “Invitar a evento”.

IA de presentación:

resume automáticamente puntos fuertes del perfil.

Ejemplo:

“Lucía es técnica de sonido con experiencia en pop y acústicos. Vive en Madrid y ha completado 12 eventos con calificación excelente.”

3.5.8 /messages

Sistema de chat en tiempo real (Firestore + Stream).

- Lista de conversaciones.
- Estado “en línea / visto”.

- Posibilidad de enviar adjuntos.
 - Modo IA sugeridor de respuestas:

“Proponle confirmar fecha y horario.”
 - IA resumen: genera recap semanal de conversaciones.
-

3.5.9 /marketplace

Sección comercial.

- Vista tipo Wallapop.
 - Filtros por categoría (instrumentos, luces, sonido, locales).
 - Botón “Añadir producto / servicio”.
 - Cards con precios, fotos y contacto.
 - IA recomienda productos según eventos activos.
-

3.5.10 /ai/event-planner

IA conversacional.

El usuario describe lo que necesita; la IA estructura el evento.

Ejemplo diálogo:

“Quiero un guitarrista y un cajón para el viernes en un bar pequeño.”
IA responde con esquema de evento + sugerencias automáticas.

3.5.11 /admin

Panel interno (solo admins).

- Gestión de usuarios y eventos.

- Gráficos de actividad.
 - Logs IA (qué solicitudes se procesan).
 - Modo mantenimiento y reportes.
-

3.6 Navegación y estados

Estado 1 — *Guest*

Solo acceso a landing, login, registro.

Estado 2 — *User autenticado*

Acceso completo excepto admin.

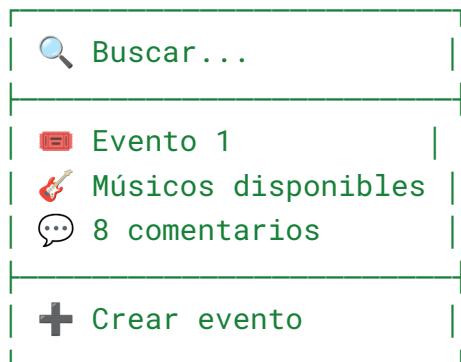
Estado 3 — *Admin*

Permisos totales (CRUD + métricas).

El sistema usa `AuthContext` y rutas protegidas (`ProtectedRoute`).

3.7 Wireframes conceptuales (textuales)

Home / Feed (mobile view)



Crear Evento

[Mic Input 🎤] Describe tu evento...
IA → "¿Quieres que te ayude a completarlo?"

[Botón: Generar evento automáticamente]
Resultado → Vista previa + Editar + Publicar

Perfil

[Foto + Nombre + Ciudad]
⭐ Trust 85/100
🎵 Instrumentos: Guitarra, Voz
⭐ 4.9 / 5 (23 reviews)
[Botón: Invitar a evento]

3.8 Estructura de componentes (para Anti-Gravity)

```
/src
  └── components
    ├── layout/
    │   ├── Navbar.tsx
    │   ├── TabBar.tsx
    │   ├── Header.tsx
    │   └── Footer.tsx
    ├── ui/
    │   ├── Button.tsx
    │   ├── Card.tsx
    │   ├── Input.tsx
    │   ├── Modal.tsx
    │   └── Avatar.tsx
    ├── event/
    │   ├── EventCard.tsx
    │   ├── EventForm.tsx
    │   └── EventDetail.tsx
    ├── profile/
    │   ├── ProfileHeader.tsx
    │   └── ReviewList.tsx
    └── chat/
        ├── ChatList.tsx
        └── ChatWindow.tsx
  └── pages/
```

3.9 Prompt principal para Anti-Gravity

You are designing the complete page map and wireframe system for Musikeeo.

Objective:

- Generate full folder structure under /src/pages with all routes listed.
- Create base layouts (Public, App, Admin) with reusable components.
- Implement bottom TabBar for mobile and side menu for desktop.
- Include skeleton UI for Feed, Event, Profile, Chat, Marketplace.
- Ensure routing, navigation, and protected routes with AuthContext.

Output:

1. /src/pages structure
 2. /components/layout and /components/ui base files
 3. Wireframe preview (JSON or HTML)
 4. Navigation config file with route metadata
-

3.10 Entregables esperados

- **Musikeeo_Wireframe_Map.pdf** (documento visual de estructura).
 - **/src/pages** creada y conectada.
 - Layouts funcionales (Public/App/Admin).
 - Skeletons navegables en Anti-Gravity preview.
 - JSON con metadatos de rutas (**routes.json**).
-

Fin del BLOQUE 3 — Wireframe y Mapa de Páginas (versión extendida).

¿Continuamos con el **BLOQUE 4 — Arquitectura Técnica y Base de Datos (versión extendida, full detail para Antigravity)**?

si

Perfecto 🔥

Entramos en el **BLOQUE 4 — ARQUITECTURA TÉCNICA Y BASE DE DATOS (versión extendida)**.

Este bloque es **la base estructural de todo el sistema**, donde se definen los modelos de datos, colecciones, relaciones, funciones, seguridad y endpoints IA que usará Musikeo.

Es lo que permitirá a Anti-Gravity generar código **realmente funcional y escalable**, con un backend serverless (Firebase) y frontend reactivo (React + Firestore hooks).

BLOQUE 4 — ARQUITECTURA TÉCNICA Y BASE DE DATOS (Versión extendida)

4.1 Propósito del bloque

Definir la **estructura lógica completa del sistema**:

cómo los datos fluyen, se almacenan y se transforman dentro de Musikeo.

Debe cubrir:

- Arquitectura de capas.
 - Modelos de datos (usuarios, eventos, mensajes, reviews, marketplace).
 - Funciones y triggers automáticos (Firebase Functions).
 - Reglas de seguridad.
 - Integración IA (Gemini / Vertex).
 - Requerimientos de rendimiento y escalabilidad.
-

4.2 Arquitectura general (visión en capas)

Frontend (PWA React 19)

|

 |— Context Layer (Auth, Theme, Data)

|

```

    └── Data Layer (Firestore + Storage)
    └── Function Layer (Firebase Functions + Gemini APIs)
    └── External Integrations (Stripe, Maps, Mail)

```

Componentes:

Capa	Tecnología	Rol
Frontend	React + Vite + TS + Tailwind	Interfaz y lógica UI
State	React Context + custom hooks	Control global de sesión, idioma y rol
Backend	Firebase (Auth, Firestore, Storage)	Persistencia, archivos, datos en tiempo real
Cloud Functions	Node.js + Gemini SDK	IA, lógica de negocio y tareas automáticas
Hosting	Firebase Hosting + PWA manifest	Despliegue web/app
Integraciones	Stripe, Google Maps, Gemini	Pagos, ubicación, IA

4.3 Estructura del proyecto

```

musikeeo/
└── src/
    ├── components/
    ├── pages/
    ├── context/
    ├── hooks/
    ├── services/
    ├── utils/
    └── assets/
└── functions/
    ├── index.ts
    ├── ai/
        ├── eventPlanner.ts
        ├── matchmaker.ts
        └── summaryGenerator.ts
    ├── triggers/
        └── onUserCreated.ts

```

```
|   |   └── onEventCreated.ts
|   └── onReviewAdded.ts
|   └── schedulers/
|       └── weeklyDigest.ts
└── firestore.rules
```

4.4 Modelo de datos (colecciones y esquemas)

Cada entidad principal del sistema se define como colección en Firestore. A continuación, se detallan los modelos JSON esperados y sus relaciones.

4.4.1 users

```
{
  "uid": "string",
  "displayName": "string",
  "email": "string",
  "photoURL": "string",
  "role": "musician | technician | promoter | store",
  "city": "string",
  "genres": ["pop", "rock", "latin"],
  "instruments": ["guitar", "voice"],
  "services": ["PA rental"],
  "bio": "string",
  "trustScore": 82,
  "reviewsCount": 12,
  "availability": {
    "days": ["fri", "sat"],
    "time": "evening"
  },
  "verified": true,
  "createdAt": "timestamp",
  "updatedAt": "timestamp"
}
```

Relaciones:

- `users.uid → events.participants[].uid`

- `users.uid → reviews.receiverId`
-

4.4.2 events

```
{  
  "id": "string",  
  "title": "Rumba en Gràcia",  
  "description": "Se busca percusionista para evento acústico.",  
  "date": "timestamp",  
  "location": {  
    "address": "Carrer Verdi 45, Barcelona",  
    "lat": 41.403,  
    "lng": 2.157  
  },  
  "budget": 200,  
  "currency": "EUR",  
  "status": "open | confirmed | completed | canceled",  
  "creatorId": "uid",  
  "participants": [  
    {  
      "uid": "string",  
      "role": "musician",  
      "accepted": true  
    }  
  ],  
  "aiGenerated": true,  
  "createdAt": "timestamp",  
  "updatedAt": "timestamp"  
}
```

Relaciones:

- `creatorId → users.uid`
 - `participants[].uid → users.uid`
-

4.4.3 messages

```
{  
  "chatId": "string",  
  "participants": ["uid1", "uid2"],  
  "lastMessage": "Nos vemos el viernes",  
  "updatedAt": "timestamp"  
}
```

Subcolección `messages/{chatId}/content`:

```
{  
  "senderId": "uid",  
  "text": "Hola! Ensayamos a las 19h?",  
  "timestamp": "timestamp",  
  "attachments": ["url1", "url2"]  
}
```

4.4.4 reviews

```
{  
  "id": "string",  
  "eventId": "string",  
  "reviewerId": "string",  
  "receiverId": "string",  
  "rating": 5,  
  "comment": "Excelente músico, puntual y profesional.",  
  "aiSummary": "Carlos fue destacado por su puntualidad y energía  
positiva.",  
  "createdAt": "timestamp"  
}
```

4.4.5 marketplace

```
{  
  "id": "string",  
  "ownerId": "uid",  
  "title": "Micrófono Shure SM58",  
  "description": "En perfecto estado, ideal para voces.",  
  "price": 85,  
  "type": "sale | rent",  
}
```

```

    "category": "microphone",
    "city": "Barcelona",
    "available": true,
    "photos": [ "url" ],
    "createdAt": "timestamp"
}

```

4.5 Relaciones y referencias

Firestore es no relacional, pero se modelan relaciones por referencia:

Entidad	Relación	Tipo	Descripción
User → Event	uno a muchos	creatorId	un usuario crea varios eventos
Event → Review	uno a muchos	eventId	cada evento genera reviews
User → Review	uno a muchos	receiverId	un usuario recibe muchas valoraciones
User → Message	muchos a muchos	participants	conversaciones
Store → Marketplace	uno a muchos	ownerId	relación directa de productos

4.6 Triggers automáticos (Cloud Functions)

Trigger	Activador	Acción
onUserCreate	Auth.createUser	Crea documento <code>users/{uid}</code> y genera bienvenida IA.
onEventCreate	Firestore.create(events)	Llama a Gemini para analizar descripción y recomendar músicos.
onReviewAdd	Firestore.create(reviews)	IA genera <code>aiSummary</code> + recalcula <code>trustScore</code> .
weeklyDigest	Schedule (domingo 20:00)	Envía resumen IA personalizado por email.

4.7 Seguridad y reglas Firestore

Ejemplo de `firestore.rules` (simplificado):

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{uid} {
      allow read: if request.auth != null;
      allow write: if request.auth.uid == uid;
    }
    match /events/{eventId} {
      allow read: if true;
      allow write: if request.auth != null;
    }
    match /messages/{chatId}/content/{msgId} {
      allow read, write: if request.auth != null && request.auth.uid
in resource.data.participants;
    }
    match /reviews/{reviewId} {
      allow create: if request.auth != null;
    }
  }
}
```

4.8 Cloud Functions detalladas

`ai/eventPlanner.ts`

Convierte texto natural en objeto `event`.

```
exports.eventPlanner = onCall(async (req) => {
  const prompt = `Parse this event description: "${req.text}"`;
  const result = await gemini.generateText(prompt);
  const parsed = JSON.parse(result.output);
  await db.collection('events').add(parsed);
  return parsed;
});
```

ai/matchmaker.ts

Busca músicos compatibles.

```
exports.matchmaker = onCall(async (req) => {
  const event = req.data;
  const users = await db.collection('users').get();
  const matches = users.docs.map(u => ({
    uid: u.id,
    score: similarity(event.genres, u.data().genres) +
geoScore(event.location, u.data().city)
  }));
  return matches.sort((a,b)=>b.score-a.score).slice(0,5);
});
```

ai/summaryGenerator.ts

Resume comentarios en lenguaje natural.

```
exports.summaryGenerator = onCall(async (req) => {
  const { text } = req.data;
  const summary = await gemini.generateText(`Summarize in one
sentence: ${text}`);
  return summary.output;
});
```

4.9 Sistema IA (Gemini integration layer)

La IA se conecta mediante un **IA Orchestrator Service** dentro de `functions/ai/`.

Función	Entrada	Salida	Uso
EventPlanner	Texto libre	JSON estructurado de evento	Creación automática
Matchmaker	ID evento	Lista ordenada de usuarios compatibles	Sugerencias
SummaryGenerator	Texto libre	Texto corto	Reviews automáticas

IA Support

Texto libre

Respuesta contextual

Chat de ayuda

Los modelos se cachean para optimizar respuesta (Firestore cache + TTL).

4.10 Performance & optimización

- Firestore indexado por ciudad, género y fecha.
 - Funciones IA cacheadas por evento.
 - Imágenes optimizadas con Firebase Storage + resizing (Cloud Function trigger).
 - Carga progresiva: lista infinita + skeletons.
 - Uso de Suspense + React.lazy() para rutas grandes.
-

4.11 API externa (REST layer opcional)

Aunque Firestore maneja todo en tiempo real, se genera API REST para integraciones futuras:

Endpoint	Método	Descripción
/api/events	GET/POST	Crear o listar eventos
/api/users	GET	Obtener perfiles
/api/ai/event-planner	POST	Invoca IA
/api/ai/matchmaker	POST	Devuelve matches
/api/reviews	POST	Añade review

4.12 Prompt principal para Anti-Gravity

You are building the full backend and data architecture for the Musikeeo App.

Goals:

- Create Firestore collections (users, events, messages, reviews, marketplace).
- Implement Cloud Functions (onUserCreated, onEventCreated, onReviewAdded, weeklyDigest).
- Integrate Gemini API for AI modules (eventPlanner, matchmaker, summaryGenerator).
- Define security rules for authenticated users.
- Enable Stripe integration for future payments.

Output:

1. Firestore rules file.
 2. Firebase Functions folder with 4 AI modules.
 3. Firestore indexes for city, genre, and date.
 4. JSON schema documentation.
-

4.13 Entregables esperados

- `firestore.rules`
 - `/functions/ai/*.ts`
 - `/functions/triggers/*.ts`
 - `/functions/schedulers/*.ts`
 - `schema.json` (documentación completa)
 - Índices Firestore creados automáticamente
 - Dashboard básico de logs IA
-

Fin del BLOQUE 4 — Arquitectura Técnica y Base de Datos (versión extendida).

¿Seguimos con el **BLOQUE 5 — Autenticación y Roles de Usuario (versión extendida, con IA onboarding y control de acceso)**?

sigue

Perfecto 🔥

Vamos con el **BLOQUE 5 — AUTENTICACIÓN Y ROLES DE USUARIO (versión extendida)**.

Este bloque define la **infraestructura de acceso y control de roles** dentro de Musikeeo, incluyendo el flujo de login, onboarding inteligente con IA, estructura de permisos, almacenamiento de tokens, seguridad y experiencia unificada entre web y móvil (PWA).

También define cómo Anti-Gravity debe implementar el sistema completo con Firebase Authentication + Firestore + Gemini API.

BLOQUE 5 — AUTENTICACIÓN Y ROLES DE USUARIO (Versión extendida)

5.1 Propósito del bloque

El sistema de autenticación es **la columna vertebral del ecosistema Musikeeo**.

Debe cumplir tres objetivos simultáneamente:

1. **Seguridad:** proteger datos personales y operaciones (pagos, mensajes, reputación).
2. **Identidad contextual:** adaptar la app según el rol del usuario (músico, técnico, promotor o tienda).
3. **Experiencia sin fricción:** registro y login instantáneo, compatible con cualquier dispositivo.

El onboarding debe parecer una conversación natural, **guiada por IA**, no un formulario clásico.

5.2 Stack técnico

Elemento	Tecnología	Función
Autenticación	Firebase Auth	Registro, login, control de sesión

Gestión de usuarios	Firestore	Datos extendidos de perfil
Roles y permisos	Firestore + Context API	Control dinámico de accesos
IA de onboarding	Gemini API	Genera mensajes adaptativos según rol
Seguridad	JWT interno + Firestore rules	Control de permisos CRUD

5.3 Flujos principales

◆ 1. Registro

Usuario → /register → Selecciona método → Firebase Auth → Onboarding
IA → Perfil creado

Métodos disponibles:

- Google (OAuth2)
- Apple
- Email + Password
- Modo invitado (temporal, sin funciones críticas)

Datos mínimos requeridos:

- `displayName`
- `email`
- `photoURL` (opcional)
- `role` (elegido o detectado por IA)

◆ 2. Login

Usuario → /login → Firebase Auth → Redirección automática a /feed

- Si el usuario tiene sesión previa, se usa `AuthStateListener`.
 - Se guarda token en `localStorage` (PWA ready).
 - Sesión persistente 30 días.
 - En móvil, integración futura con biometría (Face/Touch ID).
-

◆ 3. Onboarding Inteligente con IA

El onboarding no es un formulario plano: es una **conversación guiada por IA** (Gemini). Su misión es construir el perfil completo de usuario **mientras mantiene una conversación natural**.

Ejemplo (rol músico):

IA: “¡Hola! 🎵 ¿Cómo te llamas?”
Usuario: “Carlos.”
IA: “Perfecto, Carlos. ¿Qué instrumentos tocas?”
Usuario: “Guitarra y voz.”
IA: “Genial. ¿En qué ciudad sueles tocar?”
Usuario: “Barcelona.”
IA: “Ok, te ubico por ahí. ¿Qué géneros haces? Pop, rumba, rock...?”
(Respuesta)
→ Se guarda en Firestore automáticamente.

Resultado final:

Firestore document:

```
{  
  "uid": "...",  
  "displayName": "Carlos",  
  "role": "musician",  
  "city": "Barcelona",  
  "instruments": ["guitar", "voice"],  
  "genres": ["rumba", "pop"],  
  "onboardingCompleted": true  
}
```

5.4 Roles del sistema

Musikeeo tiene **cuatro roles funcionales**.

Cada uno desbloquea secciones distintas, permisos y módulos IA específicos.

Rol	Acceso principal	Permisos clave
 Músico	Feed, eventos, chat, perfil, IA Matchmaker	Crear, postular, chatear, dejar reviews
 Técnico	Feed, marketplace, eventos técnicos	Crear servicios, ser contratado
 Promotor / Local	Crear eventos, contratar, IA Planner	Publicar, confirmar músicos, pagar
 Tienda / Empresa	Marketplace, alquiler, IA Recomendador	Crear inventario, ofrecer material

5.5 Control de acceso (Role-Based Access Control – RBAC)

Anti-Gravity debe implementar control de roles basado en contexto.

Mecanismo:

- Cada usuario tiene un campo `role` en su documento Firestore.
- En el cliente, `AuthContext` lee `role` y guarda en memoria global.
- Las rutas se protegen con `ProtectedRoute(roleAllowed=[])`.

Ejemplo (React):

```
<Route
  path="/admin"
  element={<ProtectedRoute
    roles={['admin']}><AdminPage/></ProtectedRoute>}
/>
```

Reglas Firestore complementarias:

```
match /admin/{doc} {
  allow read, write: if request.auth.token.role == 'admin';
}
```

5.6 Estructura de autenticación (para Anti-Gravity)

```
/src
└── context/
    ├── AuthContext.tsx
    ├── RoleProvider.tsx
    └── useAuth.ts
└── services/
    ├── authService.ts
    ├── roleService.ts
    └── aiOnboarding.ts
└── pages/
    ├── register.tsx
    ├── login.tsx
    └── onboarding.tsx
```

AuthContext.tsx

Controla el estado global de sesión.

```
const AuthContext = createContext(null);

export const AuthProvider = ({children}) => {
  const [user, setUser] = useState(null);
  useEffect(() => {
    const unsub = onAuthStateChanged(auth, async (firebaseUser) => {
      if (firebaseUser) {
        const data = await getUserData(firebaseUser.uid);
        setUser({...firebaseUser, ...data});
      } else setUser(null);
    });
    return () => unsub();
  }, []);
  return <AuthContext.Provider
    value={{user}}>{children}</AuthContext.Provider>;
};
```

aiOnboarding.ts

Módulo de IA que guía la conversación inicial.

```
export async function startOnboarding(userInput, step, context) {
  const prompt = `
    You are Musikeeo's onboarding AI assistant.
    User role: ${context.role}
    Current step: ${step}
    Previous data: ${JSON.stringify(context.data)}
    Input: ${userInput}
    Output a JSON with {question, field, value}.
  `;
  const result = await gemini.generateText(prompt);
  return JSON.parse(result.output);
}
```

5.7 Estados de usuario

Estado	Descripción	Acción
guest	no autenticado	acceso limitado a landing y login
pending	autenticado, sin completar onboarding	IA continúa proceso
active	perfil completo	acceso total
suspend ed	usuario baneado o reportado	acceso revocado

5.8 Verificación y seguridad

- Autenticación vía Firebase tokens.
- Validación del dominio (solo app oficial).
- Cloud Function `onUserCreated` agrega verificación IA.
- Campos sensibles (email, teléfono) cifrados en Firestore.

- Protección contra bots (reCAPTCHA invisible).
- Limitación de solicitudes IA por usuario/día.

Ejemplo trigger:

```
exports.onUserCreated = functions.auth.user().onCreate(async (user)
=> {
  await db.collection('users').doc(user.uid).set({
    email: user.email,
    role: 'pending',
    createdAt: Timestamp.now(),
  });
  const welcome = await gemini.generateText(`Welcome
${user.displayName} to Musikeeo`);
  await db.collection('notifications').add({ uid: user.uid, message:
welcome.output });
});
```

5.9 Onboarding visual (UI/UX)

- Pantalla 1: “¡Hola! Soy tu asistente de Musikeeo 🎶”
- Pantalla 2: selector de rol → feedback visual dinámico.
- Pantalla 3: conversación IA + progreso visual (barrita animada).
- Pantalla 4: resumen de perfil generado.
- Pantalla 5: CTA “Ir al feed”.

El flujo se desarrolla con componentes reutilizables (`OnboardingStep`, `AIChatBubble`, `ProgressBar`).

5.10 Tokens, sesiones y PWA

- Almacenamiento local del `authToken` cifrado (`crypto-js`).

- Refresco automático con `getIdToken(true)` cada 30 min.
 - Persistencia total (web y offline).
 - En PWA: integración con Service Worker → login offline limitado.
 - Sincronización de estado con Firestore en tiempo real.
-

5.11 IA de validación de rol

Gemini también ayuda a **detectar rol automáticamente** durante onboarding.

Prompt IA:

Input: "Soy bajista y toco en bodas, pero también alquilo mi ampli."
Output: { "role": "musician", "secondaryRole": "store" }

Esto permite crear **roles híbridos** (músico + técnico, tienda + músico).

Firestore guarda estructura:

```
"roles": ["musician", "store"],  
"primaryRole": "musician"
```

5.12 Recuperación de cuenta

- Recuperación por email (Firebase default).
- Mensaje IA personalizado tras recuperación:

“¡Te echábamos de menos! 🎉 Tus eventos siguen esperándote.”

- En el futuro: autenticación biométrica + social fallback.
-

5.13 Administración de usuarios

Panel `/admin/users`:

- Tabla con filtros por rol, ciudad y reputación.
 - Acciones: suspender, editar, enviar notificación IA.
 - Logs automáticos en `/admin/logs`.
-

5.14 Prompt principal para Anti-Gravity

You are building the complete authentication and role system for the Musikeeo App.

Goals:

- Implement Firebase Authentication with Google, Apple, Email, and Guest modes.
- Create onboarding conversational flow powered by Gemini API.
- Store user roles and permissions in Firestore.
- Add AuthContext and RoleProvider in React.
- Protect routes and secure data access.
- Include AI role detection during onboarding.
- Enable session persistence and offline support for PWA.

Output:

1. `/context/AuthContext.tsx`
 2. `/services/authService.ts`
 3. `/services/aiOnboarding.ts`
 4. `Firebase trigger onUserCreated`
 5. `Firestore rules for role-based access`
-

5.15 Entregables esperados

- `/context/AuthContext.tsx`
- `/services/aiOnboarding.ts`
- `/pages/register.tsx, /login.tsx, /onboarding.tsx`

- `firebase.rules` con control RBAC
 - `functions/auth/onUserCreated.ts`
 - Documento `Musikeeo_Auth_Roles.pdf`
-

Fin del BLOQUE 5 — Autenticación y Roles de Usuario (versión extendida).

¿Continuamos con el **BLOQUE 6 — IA EVENT PLANNER (creación de eventos por lenguaje natural, versión extendida completa)**?

sigue

Perfecto 

Entramos en el **BLOQUE 6 — IA EVENT PLANNER (Versión extendida)**, uno de los núcleos inteligentes de Musikeeo.

Aquí definimos cómo funciona el **módulo de Inteligencia Artificial que crea eventos automáticamente a partir de lenguaje natural o voz**, cómo se estructura el flujo, la interacción con el usuario, la arquitectura de llamadas, y cómo se guarda todo en Firestore.

Este bloque debe permitir que **Anti-Gravity** genere un sistema real y funcional: una IA que entienda texto/voz, estructure el evento, lo guarde en base de datos y lo relacione con usuarios y roles.

BLOQUE 6 — IA EVENT PLANNER (Versión extendida)

6.1 Propósito del módulo

El **IA Event Planner** convierte una descripción informal en un **evento estructurado y listo para publicar** dentro de la plataforma.

Su función no es solo “entender texto”, sino **razonar sobre el contexto musical**: qué tipo de evento se trata, qué perfiles necesita, cuánto presupuesto, cuándo y dónde.

Ejemplo de entrada:

"Necesito una banda de rumba para el viernes en un bar pequeño de Gràcia, tengo 300 € de presupuesto."

Salida esperada (Firestore schema):

```
{  
  "title": "Banda de Rumba en Bar de Gràcia",  
  "description": "Evento en local pequeño, busca grupo de rumba.",  
  Presupuesto: 300€.,  
  "date": "2026-01-16T22:00:00",  
  "location": { "city": "Barcelona", "zone": "Gràcia" },  
  "budget": 300,  
  "rolesNeeded": [ "guitar", "percussion", "vocal" ],  
  "genre": "rumba",  
  "creatorId": "uid",  
  "status": "open",  
  "aiGenerated": true  
}
```

6.2 Flujo general del módulo

Usuario escribe o dicta → IA analiza prompt → IA genera JSON de evento →
Usuario revisa → Acepta → Se guarda en Firestore → IA sugiere músicos compatibles

Etapas:

1. Captura de texto o audio (speech-to-text).
 2. Parsing semántico con IA (Gemini o Vertex AI).
 3. Validación de campos requeridos.
 4. Generación de evento estructurado.
 5. Confirmación del usuario.
 6. Publicación final (Firestore + trigger Matchmaker).
-

6.3 Contexto de uso (UX)

El IA Planner se invoca desde:

- Botón “**Crear evento con IA**” en `/feed` o `/event/create`.
- O mediante el **asistente principal** (“Nuria”, “Jason” u otro nombre IA).

Interfaz mínima:

Microfono icono: **Describe tu evento...**
IA: “Vale, cuéntame qué necesitas y cuándo.”
Usuario: “Una banda pop para boda el 20 de mayo en Sitges.”
IA: “Perfecto. ¿Tienes presupuesto aproximado?”
Usuario: “Sí, 800 €.”
IA: “Genial. Te muestro lo que he entendido ”
[Vista previa editable + Botón PUBLICAR]

6.4 Inteligencia del sistema (Prompt engineering)

La IA usa un **pipeline de razonamiento estructurado**:

1. **Extraction layer**
 - Detecta entidades: fecha, lugar, presupuesto, género, instrumentos, tipo de evento, idioma.
2. **Classification layer**
 - Clasifica por tipo: boda, concierto, jam, fiesta privada, corporativo.
3. **Expansion layer**
 - Añade campos faltantes con estimaciones razonables.
 - Ejemplo: si dice “boda”, añade por defecto “dress code formal” y “música en directo”.
4. **Structuring layer**
 - Devuelve JSON final listo para Firestore.

5. Validation layer

- Verifica coherencia (fecha futura, presupuesto > 0, roles válidos).
-

6.5 Prompt IA principal (para Gemini)

You are Musikeeo's Event Planner AI.

Your task is to convert natural language into structured JSON for the Musikeeo event system.

Rules:

- Always output valid JSON.
- Include: title, description, date, location, budget, currency, genre, rolesNeeded, creatorId, and status.
- Guess missing details intelligently but never hallucinate unrealistic values.
- Use concise, professional Spanish titles and descriptions.
- Infer "genre" and "rolesNeeded" based on musical context.
- Add aiGenerated = true.

Input: "{{user_text}}"

Output Example:

```
{  
  "title": "...",  
  "description": "...",  
  "date": "...",  
  "location": { "city": "...", "zone": "..." },  
  "budget": 200,  
  "currency": "EUR",  
  "genre": "...",  
  "rolesNeeded": [...],  
  "creatorId": "{{user.uid}}",  
  "status": "open",  
  "aiGenerated": true  
}
```

6.6 Estructura técnica (para Anti-Gravity)

```
/functions/ai/
|__ eventPlanner.ts
|__ matchmaker.ts
└__ utils/
    └__ parseDateLocation.ts
/src/pages/
|__ event/
|    |__ create.tsx
|    └__ preview.tsx
```

eventPlanner.ts

```
import { onCall } from "firebase-functions/v2/https";
import { db } from "../firebase";
import { gemini } from "../gemini";

export const eventPlanner = onCall(async (req) => {
  const { text, userId } = req.data;
  const prompt = `
    You are Musikeeo's Event Planner AI.
    Convert this text into structured event JSON.
    ${text}
  `;
  const response = await gemini.generateText(prompt);
  const parsed = JSON.parse(response.output);

  // Store in Firestore
  const doc = await db.collection("events").add({
    ...parsed,
    creatorId: userId,
    createdAt: new Date(),
  });

  // Trigger automatic matchmaker
  await db.collection("tasks").add({
    type: "matchmaker",
    eventId: doc.id,
  });

  return { eventId: doc.id, ...parsed };
});
```

6.7 Componentes frontend asociados

EventCreate.tsx

Formulario híbrido IA + manual.

```
const [description, setDescription] = useState('');
const [event, setEvent] = useState(null);

const handleAI = async () => {
  const result = await callFunction("eventPlanner", { text:
description, userId: user.uid });
  setEvent(result);
};
```

UI:

- Textarea + mic input (speech-to-text).
- Spinner mientras IA procesa.
- Vista previa editable.
- Botón “Publicar evento”.

6.8 Validaciones automáticas

Cada evento generado pasa por una Cloud Function secundaria `validateEvent`.

```
if (!event.title || !event.date || !event.location)
  throw new Error("Missing essential fields");

if (new Date(event.date) < new Date())
  throw new Error("Event date cannot be in the past");
```

Los errores se muestran con mensajes empáticos:

“Parece que la fecha ya pasó. ¿Querías decir la próxima semana?”

6.9 Enriquecimiento automático (AI Augmentation)

Después de la creación, otra IA mejora el evento:

- Genera cartel automático (Gemini Vision o Imagen 3).
- Traduce la descripción a inglés/catalán para SEO.
- Resume texto para notificaciones.

Ejemplo:

```
"aiEnhancements": {  
  "posterURL": "https://storage.musikeeo.app/events/poster123.jpg",  
  "summary": "Rumba acústica en Gràcia, viernes 22h.",  
  "translations": { "en": "Acoustic rumba night in Gràcia" }  
}
```

6.10 Integración con geolocalización

La IA debe geocodificar direcciones usando Google Maps API.

Si el usuario dice “en un bar de Poble-sec”, el sistema genera:

```
"location": { "city": "Barcelona", "zone": "Poble-sec", "lat":  
41.373, "lng": 2.163 }
```

También calcula radio de búsqueda (3 km) para recomendación de músicos.

6.11 Experiencia IA humanizada

La IA debe responder con tono natural, no técnico.

Ejemplo de diálogo:

- 👤 Usuario: “Quiero un trío de jazz para una cena privada el 4 de abril.”
🤖 IA: “Perfecto, ¿en qué ciudad será la cena?”
👤 Usuario: “Madrid.”

 IA: "Listo. He creado un evento de jazz en Madrid para el 4 de abril.
Presupuesto sugerido: 350 €. ¿Quieres publicarlo?"

El mensaje final incluye botones interactivos:

[Publicar ] – [Editar ] – [Cancelar ]

6.12 Rendimiento y escalabilidad

- Cada llamada IA tarda < 3 s (Gemini 1.5 Pro).
 - Respuestas cacheadas en Firestore (`aiCache` collection).
 - Límite: 5 eventos generados/día por usuario (evita abuso).
 - Llamadas controladas por `functions.logger` + monitor de costos.
 - Posible futuro: batching de IA en background para grandes volúmenes.
-

6.13 Métricas y logs

Firestore collection `logs/aiEventPlanner`:

```
{  
  "uid": "string",  
  "input": "Necesito grupo de salsa...",  
  "tokensUsed": 520,  
  "durationMs": 2700,  
  "success": true,  
  "createdAt": "timestamp"  
}
```

Dashboard `/admin/ai` muestra:

- N° eventos IA generados.
- Tiempo medio de respuesta.
- Tokens usados / día.

- Ratio de error.
-

6.14 Limitaciones y fallback manual

Si la IA falla:

- Formulario se rellena con valores por defecto.
 - Usuario puede editar manualmente.
 - Error reportado en `logs/errors`.
 - IA sugiere corrección (“Parece que no entendí el lugar, ¿puedes repetirlo?”).
-

6.15 Prompt técnico para Anti-Gravity

You are implementing the AI Event Planner module for the Musikeeo App.

Goals:

- Convert natural language into structured event objects.
- Integrate Gemini API via Cloud Function.
- Validate data (date, location, budget).
- Enrich event with AI-generated poster, summary, and translations.
- Save event to Firestore and trigger Matchmaker task.
- Provide conversational UI for user interaction.

Output:

1. `/functions/ai/eventPlanner.ts`
 2. `/functions/ai/validateEvent.ts`
 3. `/src/pages/event/create.tsx` and `preview.tsx`
 4. `/src/components/ai/EventAssistant.tsx`
 5. `/logs/aiEventPlanner`
-

6.16 Entregables esperados

- `functions/ai/eventPlanner.ts` (principal).
 - `functions/ai/validateEvent.ts` (verificación).
 - `src/components/ai/EventAssistant.tsx`.
 - `src/pages/event/create.tsx` + preview.
 - `logs/aiEventPlanner` para auditoría.
 - Documento `Musikeeo_AI_EventPlanner.pdf`.
-

Fin del BLOQUE 6 — IA Event Planner (versión extendida).

¿Seguimos con el **BLOQUE 7 — IA MATCHMAKER (recomendador inteligente de músicos y técnicos, versión extendida al máximo detalle)?**

sigue

Perfecto 🔥

Este bloque es uno de los más potentes del sistema: define la **IA MATCHMAKER**, el módulo que analiza músicos, técnicos y eventos para **recomendar automáticamente las mejores combinaciones posibles**.

Esta parte es lo que diferencia a Musikeeo del resto de apps: un sistema de **compatibilidad musical real**, que entiende estilos, instrumentación, disponibilidad y reputación.

BLOQUE 7 — IA MATCHMAKER **(Versión extendida)**

7.1 Propósito del módulo

El **IA Matchmaker** es el cerebro del sistema de conexión en Musikeeo.

Su objetivo es **encontrar, ordenar y justificar recomendaciones** entre músicos, técnicos, locales o tiendas en función de su compatibilidad.

El módulo analiza:

- estilo musical,
- instrumentos disponibles,
- ubicación,
- reputación (TrustScore),
- disponibilidad,
- y feedback histórico (eventos previos).

La IA devuelve una lista ordenada de coincidencias con **puntuación de compatibilidad y explicación natural**.

Ejemplo:

“He encontrado 3 guitarristas cerca de ti que encajan con tu evento:

 *Carlos* (95%) — especializado en rumba.

 *Nico* (89%) — tiene equipo propio.

 *Lucía* (82%) — vocalista con experiencia en acústicos.”

7.2 Arquitectura general

Evento creado o actualizado →

Trigger Cloud Function →

Matchmaker IA analiza base de usuarios →

Genera ranking →

Guarda en Firestore →

Envía notificaciones →

IA explica resultado al usuario

Ubicación técnica:

/functions/ai/matchmaker.ts

/src/services/matchService.ts

7.3 Datos de entrada

El sistema usa dos fuentes principales:

1. **Evento base:**

- o genre, rolesNeeded, city, date, budget.

2. **Perfiles de usuario:**

- o genres, instruments, services, availability, trustScore, distance, completedEvents.

Ejemplo:

Evento:

```
{  
  "genre": "rumba",  
  "rolesNeeded": ["guitar", "voice", "percussion"],  
  "city": "Barcelona",  
  "date": "2026-03-01",  
  "budget": 300  
}
```

Usuarios:

```
[  
  {"uid": "A", "genres": ["rumba", "flamenco"], "instruments":  
   ["guitar"], "city": "Barcelona", "trustScore": 92},  
  {"uid": "B", "genres": ["pop"], "instruments": ["bass"], "city":  
   "Mataró", "trustScore": 75}  
]
```

7.4 Proceso de matching

El motor IA evalúa múltiples dimensiones ponderadas:

Factor	Pes o	Descripción
Género musical	0.35	Coincidencia entre estilos

Instrumentos / servicios	0.25	Relevancia del rol requerido
Ubicación / distancia	0.20	Radio máximo: 25 km
Reputación (TrustScore)	0.10	Fiabilidad percibida
Disponibilidad	0.05	Coincidencia de fechas
Histórico colaborativo	0.05	Experiencias previas entre usuarios

Puntuación final (0–100):

```
score = Σ (peso_i * normalizar(factor_i))
```

7.5 Lógica IA (prompt principal)

You are Musikeeo's AI Matchmaker.

Your task is to analyze one event and a list of musicians, technicians, or stores.

For each user, compute a compatibility score (0–100) based on genre, instruments, location, trustScore, and availability.

Then return a sorted JSON array with explanation texts in Spanish.

Input Example:

```
{
  "event": {...},
  "users": [...]
}
```

Output Example:

```
[
  {
    "uid": "A",
    "score": 94,
    "reason": "Coincide en género (rumba) y está en Barcelona. Alta reputación (92)."
  },
  {
    "uid": "B",
```

```
        "score": 76,
        "reason": "Buen perfil técnico pero estilo diferente (pop)."
    }
]
```

7.6 Flujo técnico completo

① Evento creado → trigger automático

```
exports.onEventCreated = functions.firestore
  .document("events/{id}")
  .onCreate(async (snap) => {
    const event = snap.data();
    await runMatchmaker(event);
}) ;
```

② Función principal: `runMatchmaker()`

```
async function runMatchmaker(event) {
  const users = await db.collection("users").where("city", "==",
event.location.city).get();
  const prompt = buildPrompt(event, users.docs.map(u => u.data()));
  const response = await gemini.generateText(prompt);
  const matches = JSON.parse(response.output);
  await db.collection("matches").doc(event.id).set({ matches,
updatedAt: new Date() });
}
```

③ Resultados guardados en `/matches/{eventId}`:

```
{
  "matches": [
    {"uid": "A", "score": 94, "reason": "Coincide en género y
ubicación"},
    {"uid": "B", "score": 76, "reason": "Buen técnico pero otro
estilo"}
  ],
  "updatedAt": "timestamp"
}
```

④ Notificación automática

```
exports.notifyMatches = onDocumentWritten("matches/{id}", async  
(change) => {  
  const { matches } = change.after.data();  
  for (const m of matches.slice(0,3)) {  
    await db.collection("notifications").add({  
      uid: m.uid,  
      message: `¡Tienes una oportunidad musical cerca!  
      (${m.reason})`  
    });  
  }  
});
```

7.7 Integración en la interfaz

/event/[id]/matches.tsx

- Lista de músicos recomendados.
- Botón “Invitar” o “Enviar mensaje”.
- Tag de compatibilidad (“95% Match”).

UI ejemplo:

 Carlos (95%)	– Rumba – 2 km
 Nico (89%)	– Percusión – 5 km
 Lucía (82%)	– Vocalista – 3 km

Cada card muestra:

- Foto, nombre, rol.
- Barra de compatibilidad animada.
- Motivo IA (reason).
- Botón CTA (“Invitar a evento”).

7.8 Sistema de embeddings y similitud semántica

El motor usa **vector embeddings** para representar estilos musicales y roles.

Esto permite encontrar coincidencias aunque no sean exactas (ej. “flamenco pop” ≈ “rumba”).

Implementación técnica:

```
import { embedText } from "@google/generative-ai";

const vectorA = await embedText("rumba pop acoustic");
const vectorB = await embedText("flamenco latin");
const similarity = cosineSimilarity(vectorA, vectorB);
```

Resultado:

```
similarity = 0.89 → alta compatibilidad
```

Se cachean embeddings en Firestore:

```
"userEmbeddings/{uid}": {
  "vector": [0.12, 0.54, 0.32, ...],
  "updatedAt": "timestamp"
}
```

7.9 Aprendizaje continuo (reajuste dinámico)

Cada vez que un evento termina y se califican los participantes:

- La IA recalcula el **peso de compatibilidad**.
- Ajusta los embeddings del usuario.
- Guarda histórico de “matches exitosos”.

Cloud Function `onReviewAdded`:

```
exports.retrainMatchWeights = onDocumentCreated("reviews/{id}",
async (snap) => {
```

```
const { reviewerId, receiverId, rating } = snap.data();
if (rating > 4) {
  await db.collection("users").doc(receiverId).update({
    trustScore: FieldValue.increment(1),
  });
}
});
```

Esto hace que **el sistema aprenda quién funciona bien con quién.**

7.10 IA explicativa (human feedback)

Gemini genera también un texto corto para explicar por qué el match es bueno:

“Coincidís en estilo y ciudad, además ambos habéis participado en eventos acústicos.”

Esto se muestra en el perfil y en las notificaciones para aumentar confianza.

7.11 Escenarios de uso

① Músico busca banda

“Quiero un grupo pop para bolos de verano.”

IA sugiere: “Te encajan Diego (batería) y Ana (voz). Ambos están en tu zona y comparten género.”

② Promotor crea evento

Evento: “Cena privada flamenca.”

IA devuelve lista de músicos con 90+ compatibilidad.

③ Tienda ofrece material

“Tengo micrófonos para eventos acústicos.”

IA recomienda eventos activos cercanos con ese estilo.

7.12 Métricas del sistema

Métrica	Objetivo	Descripción
Tiempo medio de recomendación	< 3 s	Velocidad de IA
Precisión percibida	> 85 %	Usuarios confirman "me encaja"
Engagement post-match	> 60 %	Usuarios abren el chat o invitan
Retención por uso IA	+25 %	Usuarios vuelven por utilidad
Tokens usados / match	< 500	Eficiencia económica

7.13 IA de ranking y explicación

El resultado final incluye:

- Score numérico.
- Texto explicativo.
- Emojis para tono humano.

Ejemplo salida:

```
[
  {
    "uid": "U1",
    "score": 93,
    "reason": "🎸 Coincide en género (pop) y ciudad (Barcelona).  
Excelente reputación."
  },
  {
    "uid": "U2",
    "score": 86,
    "reason": "🥁 Estilo similar (latino) aunque está algo más  
lejos."
  }
]
```

7.14 Performance y escalabilidad

- Paraleliza búsqueda con `Promise.all()`.
 - Paginación de resultados (20 por lote).
 - Cacheado en `/matchesCache`.
 - Cada evento genera máx. 50 recomendaciones.
 - Filtros IA predefinidos: `minTrustScore=60, radius<=25km`.
-

7.15 Integración visual (UX/IA)

Módulo `/components/MatchCard.tsx`

Contiene:

- Animación de compatibilidad (barra progresiva).
- Tooltip con “razón IA”.
- Botón de acción contextual (“Invitar”, “Chatear”).

Módulo `/components/AIExplain.tsx`

Pequeño componente que muestra respuestas naturales IA en burbujas:

 “Este músico suele tocar en los mismos géneros que tú.”

7.16 Prompt técnico para Anti-Gravity

You are building the AI Matchmaker system for Musikeeo.

Goals:

- Analyze events and users to compute compatibility scores.
- Use Gemini to generate ranking and human-readable explanations.
- Store results in Firestore (`/matches/{eventId}`).
- Trigger notifications to top matches.
- Update user embeddings and trustScore dynamically after each review.
- Provide UI components for match display and explanation.

Output:

1. `/functions/ai/matchmaker.ts`
 2. `/functions/triggers/onEventCreated.ts`
 3. `/functions/triggers/onReviewAdded.ts`
 4. `/components/MatchCard.tsx`
 5. `/components/AIExplain.tsx`
 6. `/pages/event/[id]/matches.tsx`
-

7.17 Entregables esperados

- `functions/ai/matchmaker.ts`
 - `functions/triggers/onEventCreated.ts`
 - `functions/triggers/onReviewAdded.ts`
 - `/matches/{eventId}` data
 - `/components/MatchCard.tsx + /components/AIExplain.tsx`
 - `/pages/event/[id]/matches.tsx`
 - `Musikeeo_AI_Matchmaker.pdf` (documentación técnica completa)
-

Fin del BLOQUE 7 — IA Matchmaker (versión extendida).

¿Seguimos con el **BLOQUE 8 — IA DE SOPORTE Y ASISTENTE CONVERSACIONAL (versión extendida: chat, helpdesk y guía contextual IA dentro de la app)?**

sigue

🔥 Vamos allá con uno de los módulos más importantes para darle “vida” a toda la app: el **BLOQUE 8 — IA DE SOPORTE Y ASISTENTE CONVERSACIONAL (versión extendida)**.

Aquí se diseña el **copiloto integral de Musikeeo**, que acompaña al usuario desde el onboarding hasta el cierre de un evento, y que además actúa como servicio técnico, guía, moderador y coach musical.

Este módulo combina **Gemini + Firestore + UI Reactiva**, e introduce un nuevo nivel de personalización y automatización.

BLOQUE 8 — IA DE SOPORTE Y ASISTENTE CONVERSACIONAL (Versión extendida)

8.1 Propósito del módulo

El Asistente IA de Musikeeo es un **agente conversacional omnipresente** que:

1. **Guía**: ayuda al usuario a crear eventos, editar su perfil o encontrar oportunidades.
2. **Soporta**: resuelve incidencias, dudas o fallos de forma natural.
3. **Conecta**: actúa como mediador entre usuarios y funciones IA (Planner, Matchmaker, Reviews).
4. **Aprende**: adapta su tono y respuestas al historial de cada usuario.

El objetivo es que el usuario sienta que “**Musikeeo tiene una voz**”: una IA musical, amable y profesional que le acompaña sin estorbar.

8.2 Personalidad del asistente (“Nuria”)

Para humanizar la experiencia, el asistente tendrá nombre, tono y estética definidos.

Atributo	Descripción
Nombre	Nuria (de “neurona” + “música”)
Tono	Natural, coloquial, cálido, sin lenguaje robótico
Estilo	Conversacional, directo, con emojis sutiles
Identidad sonora (opcional)	Voz femenina neutra, tono medio, ritmo calmado
Rol simbólico	“La IA que entiende el ritmo de tu ciudad”

Ejemplo de tono:

“Vale  , veo que estás buscando percusionistas. ¿Quieres que te muestre los que están libres este finde en tu zona?”

8.3 Estructura técnica general

Frontend (React + Context)

```
|  
|   — Chat Interface (/components/AIChat.tsx)  
|  
|   — AI Layer (Gemini API)  
|  
|   — Firestore (contexto persistente por usuario)  
|  
|   — Cloud Functions (logs, intent routing, analytics)
```

8.4 Componentes principales

Componente	Descripción
/components/AIChat.tsx	Interfaz de conversación tipo chat.
/services/aiAssistant.ts	Lógica de IA, gestión de intents y contexto.
/context/AssistantContext.tsx	Manejo de estado conversacional.
/functions/ai/assistant.ts	Cloud Function que orquesta peticiones IA.
/functions/ai/intentsRouter.ts	Router semántico para distinguir propósito de usuario.

8.5 Flujo de funcionamiento

- ① Usuario envía mensaje o audio.
- ② Gemini analiza intención (intent detection).
- ③ Router IA redirige hacia la acción correspondiente:

- Crear evento → Event Planner.
 - Buscar músico → Matchmaker.
 - Reportar error → Support.
 - Pregunta genérica → Conversacional.
 - ④ Respuesta devuelta al usuario.
 - ⑤ Contexto guardado en Firestore (`aiSessions/{uid}`).
-

8.6 Tipología de Intents

Intent	Acción asociada	Descripción
<code>create_event</code>	Llama a Event Planner	Usuario describe un evento.
<code>find_musician</code>	Llama a Matchmaker	Busca músicos o técnicos.
<code>support</code>	Entra en modo Soporte	Dudas técnicas o problemas.
<code>profile_help</code>	Abre panel perfil	Modifica bio o disponibilidad.
<code>general_chat</code>	Conversa libre	Interacciones naturales o feedback.

8.7 Prompt base para detección de intención

You are Musikeeo's conversational assistant, named Nuria.
Detect what the user wants to do and respond accordingly.

Possible intents: `create_event`, `find_musician`, `support`, `profile_help`, `general_chat`.
Output JSON with `{intent, confidence, responseText}`.

Input: "`{{user_message}}`"

Ejemplo de salida:

```
{  
  "intent": "create_event",  
  "confidence": 0.94,  
  "responseText": "Vale 🎶, cuéntame qué necesitas y te ayudo a  
crear el evento."  
}
```

8.8 Conversación multi-turno (contexto persistente)

Cada conversación guarda su historial en Firestore:

```
{  
  "sessionId": "uid_20260110",  
  "context": {  
    "lastIntent": "create_event",  
    "pendingFields": ["date", "location"]  
  },  
  "messages": [  
    {"role": "user", "text": "Busco grupo de pop"},  
    {"role": "assistant", "text": "Perfecto, ¿dónde sería el  
evento?"}  
  ],  
  "updatedAt": "timestamp"  
}
```

Esto permite **mantener continuidad** incluso si el usuario cierra la app.

8.9 Flujo técnico (Cloud Function)

```
exports.aiAssistant = onCall(async (req) => {  
  const { uid, message } = req.data;  
  const context = await db.collection("aiSessions").doc(uid).get();  
  
  const intentPrompt = `  
    You are the assistant "Nuria". Current context:  
    ${JSON.stringify(context.data())}.  
    User message: "${message}".  
    Output JSON: {intent, confidence, reply}.`
```

```

`;
const intentRes = await gemini.generateText(intentPrompt);
const intent = JSON.parse(intentRes.output);

// Route to sub-function
if (intent.intent === "create_event") {
  const event = await callFunction("eventPlanner", { text:
message, userId: uid });
  return { reply: `He creado el evento ${event.title} 🎉`, intent
};
}

if (intent.intent === "find_musician") {
  const result = await callFunction("matchmaker", { text: message,
userId: uid });
  return { reply: `Te muestro algunos músicos que encajan 👍`,
matches: result };
}

return { reply: intent.reply };
});

```

8.10 Interfaz de chat (UI/UX)

Estructura visual:

[Header] Nuria 🎵 – tu asistente IA

💬 Burbuja de Nuria (respuesta)
👤 Burbuja usuario
📎 Botón de audio / texto / adjunto

InputBox + Mic Button

Funcionalidades:

- Scroll infinito (mensajes guardados en Firestore).
- Envío de texto o voz (speech-to-text).

- Estado de escritura IA (“Nuria está pensando...”).
 - Emojis automáticos según emoción detectada.
 - Acceso rápido a FAQs y acciones sugeridas.
-

8.11 Sistema de soporte inteligente

Cuando el usuario reporta un problema (“no puedo entrar al chat” o “no me aparece el evento”), la IA:

1. Detecta el contexto del error.
2. Verifica Firestore en background.
3. Devuelve solución guiada.
4. Si no puede resolverlo → crea ticket en [/support_tickets](#).

Ejemplo:

“Vale, he visto que tu sesión está activa, pero el evento aún no se publicó.
Te recomiendo actualizar la página. ¿Quieres que lo revise por ti?”

8.12 Integración con FAQs y base de conocimiento

Musikeeo tendrá colección [/knowledge](#) con artículos IA-referenciables:

```
{  
  "title": "Cómo crear tu primer evento",  
  "keywords": ["crear", "evento", "ayuda"],  
  "content": "Ve al menú principal y pulsa el botón + Crear. La IA  
te asistirá."  
}
```

La IA puede recuperar contexto y citar la fuente:

“Según nuestra guía, puedes hacerlo así  ...”

8.13 Detección de emociones

Gemini analiza sentimiento de cada mensaje:

- **happy, neutral, frustrated, angry.**

La app responde adaptando tono:

Estado emocional	Tono de Nuria
happy	entusiasmo (“¡Qué alegría! 🎉”)
neutral	neutro, informativo
frustrated	empático (“Tranquilo, te ayudo con eso.”)
angry	calmado y resolutivo (“Entiendo tu molestia, revisemos juntos.”)

8.14 Moderación automática

La IA también **filtira lenguaje tóxico** o contenido inapropiado.

Si detecta insultos o spam → se registra en [/reports](#).

El usuario recibe aviso amable:

“Recuerda mantener un lenguaje respetuoso. Esta comunidad se basa en buena vibra 🎶.”

8.15 Contextos IA especializados

Se definen 3 contextos activos según página:

Contexto	Ámbito	Acciones IA
feed	descubrimiento	recomendar contenido
event	planificación	crear / editar eventos
chat	interacción	sugerir mensajes o follow-ups

La IA cambia automáticamente de modo según la vista actual.

8.16 Personalización por rol

La IA adapta respuestas a cada tipo de usuario.

Músico:

“Hay 2 eventos cerca donde podrías tocar.”

Técnico:

“Esta banda busca técnico con backline, ¿quieres ofrecerte?”

Promotor:

“Puedo generar un cartel automático con los músicos confirmados.”

Tienda:

“Tu micrófono podría interesar a este grupo de jazz que toca mañana.”

8.17 Integración con notificaciones push

- Mensajes importantes (IA match, recordatorios, soporte) → Push FCM.
 - Cada conversación IA relevante dispara `sendPushNotification(uid, message)`
 - En PWA, visible en lock screen (“Nuria tiene novedades para ti 🎵”).
-

8.18 Analítica y métricas IA

Firestore `aiMetrics`:

```
{  
  "uid": "string",  
  "sessions": 5,  
  "avgResponseTime": 2.3,  
  "tokensUsed": 1840,
```

```
"intentDistribution": {  
    "create_event": 40,  
    "find_musician": 25,  
    "support": 15,  
    "general_chat": 20  
}  
}
```

Dashboard `/admin/ai`:

- Intentos más comunes.
 - Tiempo medio respuesta IA.
 - Satisfacción (encuestas emoji).
 - Sesiones activas/día.
-

8.19 Seguridad y privacidad

- Conversaciones cifradas en tránsito y en Firestore (AES256).
 - Se elimina historial >90 días automáticamente.
 - Usuarios pueden borrar su chat IA.
 - Ninguna conversación se usa para entrenamiento fuera del entorno privado Musikeeo.
-

8.20 Prompt técnico para Anti-Gravity

You are building the Conversational AI Assistant for Musikeeo.

Goals:

- Implement Gemini-based conversational system with intents:
`create_event`, `find_musician`, `support`, `profile_help`, `general_chat`.
- Persist chat context per user in Firestore (`/aiSessions/{uid}`).

- Integrate AI router with sub-functions (eventPlanner, matchmaker, support).
- Detect emotions and adjust tone.
- Build chat interface (AIChat.tsx) with voice input and push notifications.
- Store metrics in aiMetrics collection.
- Add moderation filter for toxic language.

Output:

1. /functions/ai/assistant.ts
 2. /functions/ai/intentsRouter.ts
 3. /components/AIChat.tsx
 4. /services/aiAssistant.ts
 5. /context/AssistantContext.tsx
 6. /logs/aiMetrics
-

8.21 Entregables esperados

- /functions/ai/assistant.ts
 - /functions/ai/intentsRouter.ts
 - /components/AIChat.tsx (chat completo con voz y texto)
 - /services/aiAssistant.ts
 - /context/AssistantContext.tsx
 - aiSessions, aiMetrics, support_tickets collections
 - Documento técnico Musikeeo_AI_Assistant.pdf
-

Fin del BLOQUE 8 — IA de Soporte y Asistente Conversacional (versión extendida).

¿Seguimos con el **BLOQUE 9 — IA DE REPUTACIÓN Y CONFIANZA (Trust Score, Reviews, Evaluaciones Automáticas, versión extendida completa)?**

sigue

Perfecto 🔥

Aquí entra el **corazón reputacional de Musikeeo**: el sistema que convierte cada interacción, evento o colaboración en **confianza cuantificada y verificable**.

El **BLOQUE 9 — IA DE REPUTACIÓN Y CONFIANZA (Trust Score)** define la lógica con la que músicos, técnicos, locales y tiendas construyen su credibilidad dentro del ecosistema, utilizando IA para analizar reseñas, detectar fraude, y ajustar automáticamente las puntuaciones de fiabilidad.



BLOQUE 9 — IA DE REPUTACIÓN Y CONFIANZA (Versión extendida)

9.1 Propósito del módulo

El sistema de reputación tiene tres funciones principales:

1. **Evaluar** la fiabilidad y profesionalidad de los usuarios a partir de datos reales.
2. **Visibilizar** a los perfiles más responsables (más contratación, menos conflictos).
3. **Proteger** la plataforma de fraude, spam o falsos perfiles.

En resumen, el **Trust Score** es el “karma musical” de Musikeeo:
cuanto mejor colaboras, más visible te vuelves en el feed y en los resultados IA.

9.2 Filosofía de diseño

⌚ “La reputación no se gana hablando, se construye tocando.”

El Trust Score no se basa solo en valoraciones numéricas.

Analiza **conducta, historial y consistencia**, ponderando factores emocionales y contextuales.

Ejemplo:

Un músico que llega a tiempo, mantiene buena comunicación y recibe elogios gana puntos,
aunque no tenga miles de eventos registrados.

9.3 Factores que influyen en el Trust Score

Categoría	Peso	Descripción
Reviews positivas	0.30	Promedio ponderado de valoraciones
Puntuación del evento	0.20	Calificación del organizador
Comunicación	0.15	Respuestas rápidas, tono respetuoso
Asistencia / puntualidad	0.15	Validada por IA y geolocalización
Historial sin incidencias	0.10	Ausencia de conflictos o reportes
Colaboraciones repetidas	0.10	Eventos recurrentes con mismos usuarios

Total: 1.00 (100 % del Trust Score)

9.4 Cálculo base del Trust Score

Cada usuario parte de 50 puntos (neutro).

La IA recalcula tras cada evento con:

$$\text{TrustScore} = \text{Base} + (\sum \text{Ponderación}_i * \text{Valor}_i * 100)$$

Donde $\text{Valor}_i \in [0,1]$.

Ejemplo:

- Reviews: 0.8
- Comunicación: 0.9
- Puntualidad: 1
- → $\text{TrustScore} \approx 50 + (0.8 \times 30 + 0.9 \times 15 + 1 \times 15 + \dots) = 90.5$

Se guarda truncado (0–100).

9.5 IA de análisis de reseñas (sentiment + consistency)

Cada reseña textual se procesa con **Gemini Sentiment Analyzer**, que detecta tono, sarcasmo y coherencia.

Se evita depender solo de estrellas.

Prompt IA:

```
Analyze this user review.  
Output JSON {sentiment: "positive|neutral|negative", confidence,  
summary, score[0-1]}.  
Text: "{{review_text}}"
```

Ejemplo de salida:

```
{  
  "sentiment": "positive",  
  "confidence": 0.94,  
  "summary": "Puntual, profesional y buena energía.",  
  "score": 0.92  
}
```

9.6 Flujo técnico de reputación

```
Evento finalizado →  
Usuarios dejan reviews →  
IA analiza texto →  
Actualiza TrustScore →  
Guarda en Firestore →  
Envía feedback personalizado
```

9.7 Estructura técnica

```
/functions/ai/reputation.ts  
/functions/triggers/onReviewAdded.ts  
/src/components/ReviewForm.tsx  
/src/pages/profile/[id].tsx
```

onReviewAdded.ts

```
exports.onReviewAdded = onDocumentCreated("reviews/{id}", async  
(snap) => {  
  const review = snap.data();  
  const sentimentPrompt = `  
    Analyze this review for trust scoring.  
    ${review.text}  
`;  
  const aiRes = await gemini.generateText(sentimentPrompt);  
  const analysis = JSON.parse(aiRes.output);  
  
  const weight = (review.rating / 5) * analysis.score;  
  await db.collection("users").doc(review.receiverId).update({  
    trustScore: FieldValue.increment(weight * 10),  
    lastReviewAI: analysis.summary,  
  });  
});
```

9.8 Perfil público y visualización del Trust Score

En cada perfil:

- Barra circular animada (0–100).
- Etiqueta textual:
 - 0–39 → “Requiere mejora ⚠”
 - 40–69 → “Confiable 👍”
 - 70–89 → “Recomendado 💡”
 - 90–100 → “Excelente ⭐”

Tooltip:

“Tu reputación se basa en puntualidad, comunicación y feedback de otros usuarios.”

Ejemplo UI:

⭐ Trust Score: 87

Última valoración: "Muy profesional y simpático."

9.9 IA de coherencia y fraude

La IA detecta comportamientos anómalos:

- Valoraciones extremas repetidas.
- Reviews entre cuentas relacionadas.
- Textos duplicados o generados.
- Usuarios que se valoran mutuamente (loop).

Si detecta fraude → alerta interna:

```
{  
  "type": "suspicious_review",  
  "reviewId": "r123",  
  "details": "Possible reciprocal reviews between u45 and u46"  
}
```

Automáticamente:

- Se congela impacto de esas reseñas.
 - Admin recibe aviso en [/admin/reports](#).
-

9.10 Trust Decay (decaimiento de confianza)

La reputación **se degrada lentamente con el tiempo** si el usuario deja de participar.
Cada mes sin actividad reduce un 2 % del puntaje (mínimo 50).
Esto incentiva mantener la participación activa.

9.11 IA de feedback personalizado

Tras cada evento, el usuario recibe un resumen motivador:

♪♪ “Tu último evento fue un éxito. Tu TrustScore sube a 92.
Te destacamos en el feed esta semana.”

O, si bajó:

⚠ “Recibiste una valoración baja por comunicación.
Te recomiendo mantener contacto antes del evento. Ya te ayudo a mejorar
eso.”

Generado automáticamente por Gemini a partir de métricas.

9.12 Trust Index Global

La app mantiene un índice general: promedio de confianza por ciudad, rol y género.

Firestore `/trust_index`:

```
{  
  "city": "Barcelona",  
  "avgMusician": 82.3,  
  "avgTech": 78.1,  
  "avgPromoter": 75.5  
}
```

Usos:

- Estadísticas públicas.
 - Benchmarking en dashboards.
 - Feed IA prioriza perfiles sobre la media.
-

9.13 Sistema de badges IA

Al alcanzar ciertos hitos, la IA otorga insignias automáticas:

Badge	Criterio	Descripción
⌚ Top Performer	TrustScore > 90, 10+ eventos	Perfil premium visible en feed

 Reliable Partner	Sin cancelaciones 6 meses	Distintivo de fiabilidad
 IA Verified	Reviews analizadas sin inconsistencias	Reputación validada por IA
 Community Star	+10 reseñas positivas distintas	Popularidad orgánica

Los badges se muestran con animación en el perfil.

9.14 IA de reputación contextual

El TrustScore varía **por contexto**:

- `trustPerformance` (para músicos).
- `trustTechnical` (para técnicos).
- `trustVendor` (para tiendas).

Ejemplo:

```
{  
  "trust": {  
    "performance": 88,  
    "technical": 72,  
    "vendor": 91  
  }  
}
```

Esto permite que un mismo usuario tenga distintas credibilidades según su rol.

9.15 Ponderación IA dinámica

Gemini ajusta pesos según correlación histórica entre factores y éxito real:

- Si los usuarios con mejor comunicación retienen más eventos → aumenta peso de ese factor.

- Si las reviews no correlacionan con retención → su peso baja.

El sistema se recalibra mensualmente vía Cloud Function `retrainTrustWeights()`.

9.16 Transparencia del sistema

Cada usuario puede ver:

- Cómo se calcula su TrustScore.
- Cuánto aporta cada factor.
- Recomendaciones IA personalizadas.

Ejemplo:

Tu puntuación actual: 87

+3 por buena comunicación

+4 por evento completado

-2 por cancelación última hora

Sugerencia: actualiza disponibilidad antes del fin de semana.

9.17 IA de moderación reputacional

La IA no solo analiza números: también **revisa reportes** por mal comportamiento o incumplimiento.

Tipos de reporte:

- “No se presentó al evento.”
- “Faltó respeto en el chat.”
- “Cobró y no cumplió servicio.”

Gemini evalúa texto de ambos lados y determina gravedad.

Prompt:

Compare these two reports and assess responsibility (0-1) for each user.

Output JSON {userA: 0.8, userB: 0.2, summary}.

Resultado:

“Responsabilidad principal del usuario A (0.8). Se ajusta TrustScore en -10 puntos.”

9.18 Gamificación IA

El sistema muestra una línea de progreso tipo videojuego:

“Nivel 4  — Profesional consolidado. Próximo nivel: Top Performer.”

Aumentar reputación desbloquea ventajas:

- Mejor visibilidad.
 - Acceso anticipado a eventos.
 - Comisiones reducidas.
-

9.19 Prompt técnico para Anti-Gravity

You are building the AI Reputation and Trust System for Musikeeo.

Goals:

- Compute TrustScore dynamically from reviews, communication, and punctuality.
- Analyze review sentiment with Gemini.
- Detect fraud or reciprocal reviews.
- Apply Trust Decay and Contextual Trust.
- Trigger badges and feedback messages.
- Provide visualization components and transparency dashboard.

Output:

1. /functions/ai/reputation.ts
2. /functions/triggers/onReviewAdded.ts
3. /functions/triggers/retrainTrustWeights.ts

4. /components/TrustBadge.tsx
 5. /components/TrustMeter.tsx
 6. /pages/profile/[id].tsx (Trust section)
 7. /collections/trust_index, reviews, badges
-

9.20 Entregables esperados

- functions/ai/reputation.ts
 - functions/triggers/onReviewAdded.ts
 - functions/triggers/retrainTrustWeights.ts
 - components/TrustMeter.tsx, TrustBadge.tsx
 - collections: reviews, trust_index, badges
 - Dashboard /admin/trust
 - Documento Musikeeo_AI_Reputation.pdf
-

Fin del BLOQUE 9 — IA de Reputación y Confianza (versión extendida).

¿Seguimos con el **BLOQUE 10 — IA DE GESTIÓN AUTOMÁTICA DE EVENTOS Y POSTER CREATION** (Cartelería, resumen, difusión, IA visual y automatización post-evento, versión extendida)?

sigue

🔥 Vamos con uno de los bloques más potentes a nivel de automatización visual y marketing dentro de Musikeeo:
el BLOQUE 10 — IA DE GESTIÓN AUTOMÁTICA DE EVENTOS Y POSTER CREATION (versión extendida).

Aquí se define cómo la IA se encarga de **cerrar el ciclo completo de cada evento**, desde la generación del cartel hasta la creación de clips, notificaciones, resúmenes y publicación automática en redes internas.

Este módulo conecta las funciones creativas (Gemini Image, Imagen 3 o NanoBanana), la gestión de Firestore y la analítica IA.



BLOQUE 10 — IA DE GESTIÓN AUTOMÁTICA DE EVENTOS Y POSTER CREATION (Versión extendida)

10.1 Propósito del módulo

El objetivo de este bloque es automatizar el **post-proceso de los eventos** creados en Musikeeo.

Una vez un evento está confirmado (con al menos un músico o técnico asignado), la IA se activa para:

1. Generar un cartel o póster visual automáticamente.
2. Crear una descripción promocional adaptada.
3. Publicar en el feed o enviar a redes externas (fase futura).
4. Resumir el evento al finalizar.
5. Recolectar feedback y datos de impacto.

En resumen: el usuario **crea un evento con una frase** → la IA **lo convierte en una experiencia visual y comparable**.

10.2 Flujo completo del sistema

Evento confirmado →
Trigger IA Poster Creation →
Generación de cartel + texto →
Notificación al creador →
Publicación automática →
Post-evento: IA genera resumen + reseñas

10.3 Estructura técnica

```
/functions/ai/posterCreator.ts  
/functions/ai/eventSummary.ts  
/functions/triggers/onEventConfirmed.ts  
/src/components/EventPoster.tsx  
/src/pages/event/[id].tsx
```

10.4 Poster Creation (IA visual)

Descripción del proceso

Cuando un evento se marca como “**confirmed**”, la Cloud Function `onEventConfirmed` activa el generador de cartel.

Entradas IA:

- `event.title`
- `genre`
- `date`
- `location.city`
- `musicians[]` (opcional)
- `budget` (para estilo visual)

Prompt principal (para Imagen 3 / NanoBanana):

Create a concert poster for a live music event.
Style: minimalistic, modern, energetic.
Include title, date, and city.
Theme colors should match the genre.
Do not add faces or logos.

Input:

Title: Banda de Rumba en Gràcia
Date: 16 Enero 2026
City: Barcelona
Genre: rumba

Ejemplo de salida IA:

URL:

https://storage.musikeeo.app/events/posters/event_abc123.jpg

Firestore update:

```
"poster": {  
    "url":  
        "https://storage.musikeeo.app/events/posters/event_abc123.jpg",  
    "createdBy": "AI",  
    "style": "rumba-minimal",  
    "createdAt": "timestamp"  
}
```

10.5 Generación automática de copy promocional

Gemini genera una descripción promocional natural:

Prompt:

You are a music marketing assistant.

Write a short, emotional Spanish copy for an event based on this data:

{title, date, city, genre, musicians}

Tone: energetic, concise, ready for social media.

Add 2 emojis maximum.

Ejemplo de salida:

🎶 Noche de rumba en Gràcia 🎸

Viernes 16 a las 22h. ¡Vente a vivirlo con nosotros en el corazón de Barcelona!

Firestore:

```
"promoText": "🎶 Noche de rumba en Gràcia 🎸 Viernes 16 a las 22h.",  
"hashtags": [ "#musikeeo", "#rumba", "#barcelona" ]
```

10.6 Difusión automática (fase futura)

La IA puede publicar directamente en:

- Feed interno de Musikeeo
- Canales sociales conectados (Instagram, Threads, TikTok Business API)
- WhatsApp broadcast (via Twilio / Meta API)

Lógica:

```
if (settings.autoPublish === true) {  
    await publishToFeed(event.id);  
    await postToSocials(event.promoText, event.poster.url);  
}
```

10.7 Plantillas visuales inteligentes

El sistema IA usa plantillas adaptadas por género musical:

Género	Estilo visual	Paleta sugerida
Rumba	Calor, textura orgánica	Terracota + Beige + Amarillo
Pop	Brillante, tipografía bold	Rosa + Azul + Blanco
Jazz	Minimalista, alto contraste	Negro + Oro + Gris
Rock	Vintage, grano y ruido	Rojo + Negro + Blanco roto

Cada plantilla está escrita en JSON:

```
{  
    "templateId": "rumba_minimal",  
    "fonts": [ "Space Grotesk Bold" ],  
    "colors": [ "#E7C280", "#2D1E12" ],  
    "texture": "subtle_grain"  
}
```

10.8 Pósters IA con información contextual

La IA puede incluir elementos automáticos:

- Título del evento
- Fecha y hora
- Ciudad o barrio
- Icono de instrumento principal
- Frase generada tipo “Vívelo en directo”
- Código QR que lleva al evento Musikeeo

Prompt avanzado:

Add small text: "Entrada libre" or "Reserva en Musikeeo.app".
Include simple QR placeholder bottom right.

10.9 Sistema de resúmenes post-evento

Después del evento (estado = “finished”), la IA analiza datos y genera un resumen natural.

Prompt post-event:

Generate a summary of a music event using user feedback and participation data.

Include: attendance, highlights, mood, and recommendations for next time.

Language: Spanish, tone optimistic, 3 sentences max.

Ejemplo de salida:

 Gran noche en Gràcia. Más de 80 personas disfrutaron de una rumba increíble.
¡Los músicos brillaron con energía y conexión total con el público!

Firestore:

"aiSummary": "Gran noche en Gràcia. Más de 80 personas disfrutaron...",

```
"attendees": 80,  
"ratingAvg": 4.8
```

10.10 IA de clips y stories (fase futura avanzada)

El sistema podrá generar clips cortos de los eventos subidos por los usuarios.

Workflow:

1. Usuarios suben vídeos.
2. IA selecciona los mejores fragmentos.
3. Genera mini-trailer con subtítulos automáticos y música.
4. Publica en el perfil del evento o músico.

Herramientas sugeridas: Luma AI, Runway, Pika, Gemma Video o Google Veo 3.

Prompt:

Create a 15s vertical highlight reel of a live music event.

Focus on energy, audience, and key moments.

Add event title text overlay.

10.11 IA de detección de calidad (para moderación)

Antes de publicar un cartel o clip, la IA valida:

Tipo	Validación	Acción
Imagen	resolución > 1080p, sin logos externos	aprueba o regenera
Texto	sin lenguaje inapropiado	limpia
Música	sin derechos detectados (Shazam API)	alerta si hay infracción

Esto mantiene la estética y ética de marca.

10.12 Automatización Firestore completa

Cada evento tendrá subcolecciones IA:

```
/events/{id}/ai/  
  └── poster  
  └── promo  
  └── summary
```

Ejemplo:

```
{  
  "poster": { "url": "..." },  
  "promo": { "text": "...", "hashtags": [...] },  
  "summary": { "text": "...", "rating": 4.9 }  
}
```

Esto facilita auditoría, métricas y regeneración IA.

10.13 Analítica IA de impacto

Firestore `/analytics/events/{eventId}`:

```
{  
  "views": 320,  
  "shares": 57,  
  "likes": 24,  
  "clicks": 41,  
  "conversionRate": 0.12  
}
```

Estos datos retroalimentan la IA para optimizar descripciones futuras.

10.14 Componentes de interfaz

`/components/EventPoster.tsx`

- Renderiza imagen IA.
- Incluye botón “Regenerar cartel 🎨”.
- Animación suave al cargar.
- Fondo difuminado que toma el color dominante del cartel.

/components/EventSummary.tsx

- Muestra texto IA resumen.
 - Incluye puntuación y feedback visual.
 - CTA: “Ver reseñas completas”.
-

10.15 Prompt técnico para Anti-Gravity

You are building the AI Event Management and Poster Creation system for Musikeeo.

Goals:

- Detect event confirmation and trigger poster + promo generation.
- Use Gemini + Imagen + Firestore to generate images and text.
- Store poster, promo, and summary in /events/{id}/ai/.
- Generate post-event summaries and analytics.
- Build UI components for poster and summary.
- Ensure content validation and automatic publishing.

Output:

1. /functions/ai/posterCreator.ts
 2. /functions/ai/eventSummary.ts
 3. /functions/triggers/onEventConfirmed.ts
 4. /components/EventPoster.tsx
 5. /components/EventSummary.tsx
 6. /collections/analytics/events
-

10.16 Entregables esperados

- `functions/ai/posterCreator.ts`
 - `functions/ai/eventSummary.ts`
 - `functions/triggers/onEventConfirmed.ts`
 - `components/EventPoster.tsx, EventSummary.tsx`
 - `collections/analytics/events`
 - Documento `Musikeeo_AI_EventManager.pdf`
-

Fin del BLOQUE 10 — IA de Gestión Automática de Eventos y Poster Creation (versión extendida).

¿Seguimos con el **BLOQUE 11 — IA DE COMUNICACIÓN Y MENSAJERÍA (chat entre usuarios, moderación, IA sugeridora de respuestas y seguridad en tiempo real, versión extendida)?**

sigue

🔥 Seguimos con uno de los módulos más delicados y socialmente importantes del ecosistema:
el **BLOQUE 11 — IA DE COMUNICACIÓN Y MENSAJERÍA (versión extendida)**.
Aquí se define cómo funciona el **chat interno entre músicos, técnicos y promotores**, con IA integrada para moderación, sugerencias de respuesta, seguridad y productividad.
Este sistema debe ser rápido, seguro y, al mismo tiempo, humano y empático.

BLOQUE 11 — IA DE COMUNICACIÓN Y MENSAJERÍA

(Chat entre usuarios, IA sugeridora de respuestas y moderación automática)

11.1 Propósito del módulo

El sistema de mensajería de Musikeeo permite que **los usuarios se comuniquen directamente** dentro de la app para cerrar acuerdos, coordinar eventos o compartir material.

Su misión no es solo ofrecer un chat funcional, sino **un entorno profesional y seguro** impulsado por IA.

Objetivos concretos:

1. Facilitar conversaciones rápidas y naturales entre perfiles.
2. Evitar spam, lenguaje tóxico o fraude.
3. Sugerir respuestas contextuales mediante IA.
4. Permitir el intercambio de archivos musicales y PDFs.
5. Integrarse con el sistema de eventos (chat por evento).

11.2 Arquitectura general

```
/src/  
|__ pages/chat/[id].tsx  
|__ components/ChatMessage.tsx  
|__ components/ChatInput.tsx  
|__ services/chatService.ts  
/functions/  
|__ triggers/onMessageSent.ts  
|__ ai/  
    |__ aiAutoReply.ts  
    |__ aiModeration.ts
```

Base de datos (Firestore):

```
/conversations/{conversationId}/messages/{ messageId }
```

11.3 Tipos de conversaciones

Tipo	Descripción
------	-------------

- 1 Privada** Entre dos usuarios (músico ↔ técnico, promotor ↔ músico, etc.)
- 2 Grupal** Asociada a un evento (toda la banda + organizador)
- 3 Asistida por IA** Donde la IA ayuda a coordinar o responder

Ejemplo:

“Nuria” (IA) puede intervenir para proponer horarios, confirmar pagos o recordar tareas.

11.4 Estructura de Firestore

```
{  
  "conversationId": "event_123_musician_45",  
  "participants": ["uidA", "uidB"],  
  "type": "private",  
  "lastMessage": "Te paso el setlist",  
  "updatedAt": "timestamp"  
}
```

Y cada mensaje:

```
{  
  "senderId": "uidA",  
  "text": "Hola, ¿qué hora montamos?",  
  "attachments": [],  
  "readBy": ["uidB"],  
  "createdAt": "timestamp",  
  "aiModerated": true  
}
```

11.5 Sistema IA de moderación en tiempo real

Antes de guardar un mensaje, se procesa con **Gemini Moderation Layer**:

```
Analyze message for inappropriate or unsafe content.  
Output JSON { safe: boolean, reason: string }
```

Ejemplo:

```
{  
  "safe": true,  
  "reason": "Friendly professional message."  
}
```

Si `safe = false`, el mensaje **no se envía** y el usuario recibe:

“Tu mensaje podría ser inapropiado. Revisa el tono antes de enviarlo.”

Se guarda log en `/moderation_logs`.

11.6 IA sugeridora de respuestas automáticas

Cada vez que el usuario recibe un mensaje, la IA genera 2–3 sugerencias rápidas (Quick Replies):

Prompt:

You are a helpful assistant that suggests professional short replies in Spanish.

Input: "¿Puedes tocar el viernes a las 22h en Sitges?"

Output: ["Sí, puedo", "Déjame revisar mi agenda", "Ese día ya tengo bolo 😊"]

UI ejemplo:

💬 ¿Puedes tocar el viernes?
[Sí, puedo] [July 17 Lo reviso] [😊 Ya tengo bolo]

El usuario puede pulsar directamente una respuesta y enviarla.

11.7 Chat IA asistido (modo híbrido)

En grupos grandes o cuando se crea un evento nuevo, la IA actúa como **coordinadora**.

Ejemplo:

-  Promotor: "Necesito saber quién trae el ampli."
-  Nuria: "Puedo ayudarte. Diego tiene ampli Fender Twin, ¿quieres que lo marque como responsable?"

Cada acción IA genera trazabilidad (`aiActionsLog`).

11.8 Envío de archivos y multimedia

Permitir:

- PDFs (setlists, riders técnicos, contratos).
- MP3/WAV (fragmentos de audio).
- Imágenes (carteles, localizaciones).

Validación IA:

- Tamaño máx. 10 MB.
- Análisis automático de contenido.
- Bloqueo de archivos no seguros (scripts, ejecutables).

Archivos se suben a `/storage/chat_files/{conversationId}/{fileName}`.

11.9 Experiencia UX/UI

Pantalla `/chat/[id]`:

 Chat con Carlos (Músico)

 Mensajes

 Sugerencias IA (opcional)

 Archivos adjuntos

[Mic ] [Input box] [> Send]

Características:

- Scroll virtual infinito.
 - Estado “escribiendo...” IA o usuario.
 - Avatares circulares y colores por rol.
 - Mensajes con reacción emoji.
-

11.10 Integración con notificaciones

Cuando llega un mensaje nuevo:

- Notificación push (FCM) con texto resumido IA.
 - En PWA: “Tienes un nuevo mensaje de Diego 🎵.”
 - En móvil: deep link directo a la conversación.
-

11.11 Modo resumen IA

El usuario puede pedir:

“Resúmeme esta conversación.”

La IA analiza todo el hilo y genera un resumen estructurado:

Prompt:

Summarize this chat between two musicians.

Output: {summary, pendingActions, tone, keywords}

Ejemplo:

```
{  
  "summary": "Coordinan horario y repertorio para evento del  
viernes.",  
  "pendingActions": ["Confirmar hora de montaje"],  
  "tone": "colaborativo",
```

```
        "keywords": [ "rumba", "ampli", "Sitges" ]  
    }
```

El resumen aparece como tarjeta de información dentro del chat.

11.12 Modo traducción automática

Si los usuarios usan idiomas distintos, la IA traduce los mensajes en tiempo real.

Ejemplo:

Usuario 1 (España): “Tocamos a las 22h.”
Usuario 2 (Brasil): “Perfeito, às 22h!”
→ Ambos ven los mensajes en su idioma nativo.

11.13 Modo “Revisión profesional” (para promotores)

La IA puede generar un **resumen contractual** de lo acordado en el chat:

“Los músicos han aceptado tocar el 22/05 en Madrid por 300 €.
Hora de montaje 19:00, repertorio de 90 min.”

El texto puede guardarse como PDF con firma digital opcional.

11.14 IA de alerta y soporte inmediato

Si la IA detecta:

- tensión o conflicto (“no pienso pagarte”, “no vengas”),
- palabras clave de cancelación o disputa,
entonces:
- etiqueta conversación como “*disputed*”,
- avisa al soporte humano,
- sugiere solución diplomática.

Ejemplo de mediación IA:

“Entiendo la confusión, pero mejor revisemos las condiciones que acordaron. Puedo generar un resumen para confirmar lo pactado.”

11.15 IA de tono emocional

Cada mensaje se analiza en tiempo real:

Estado	Acción
😡 Agresivo	Advertencia o filtrado
😴 Tenso	IA ofrece mediación
😐 Neutral	Normal
😊 Amable	Refuerza confianza (“Buena vibra 🎵”)

Esto mejora el clima y reduce fricciones.

11.16 Modo “Chat IA invisible”

El usuario puede activar o desactivar la asistencia IA en cualquier chat.

- Activado: IA sugiere, traduce, modera.
- Desactivado: chat clásico, sin intervención.

Preferencia guardada en:

```
"userSettings": { "aiChatEnabled": true }
```

11.17 Métricas del sistema

Firestore `/analytics/chat`:

```
{  
  "messagesSent": 542,  
  "aiSuggestionsUsed": 98,  
  "autoReplies": 53,
```

```
        "moderations": 4,  
        "responseTimeAvg": 1.8  
    }
```

Dashboard /admin/chat:

- Uso de IA.
 - Ratio mensajes tóxicos.
 - Tiempo medio de respuesta.
 - Conversaciones activas por rol.
-

11.18 Seguridad y cifrado

- Mensajes cifrados con AES-256 antes de subir a Firestore.
 - Claves de sesión rotativas cada 48 h.
 - Autenticación vía Firebase + JWT.
 - Logs de IA almacenados separados de mensajes (sin contenido).
 - Modo “eliminar chat” borra permanentemente la sesión.
-

11.19 Prompt técnico para Anti-Gravity

You are building the AI Messaging and Communication System for Musikeeo.

Goals:

- Implement Firestore-based chat (1:1, group, AI-assisted).
- Moderate messages in real time using Gemini.
- Generate smart reply suggestions.
- Summarize conversations on demand.
- Detect conflicts or toxicity and notify support.
- Enable translation and professional contract summaries.

- Integrate FCM notifications and UI components.

Output:

1. /functions/ai/aiModeration.ts
 2. /functions/ai/aiAutoReply.ts
 3. /functions/triggers/onMessageSent.ts
 4. /components/ChatMessage.tsx
 5. /components/ChatInput.tsx
 6. /pages/chat/[id].tsx
 7. /analytics/chat
-

11.20 Entregables esperados

- functions/ai/aiModeration.ts
 - functions/ai/aiAutoReply.ts
 - functions/triggers/onMessageSent.ts
 - components/ChatMessage.tsx, ChatInput.tsx
 - pages/chat/[id].tsx
 - collections/conversations, messages, moderation_logs
 - analytics/chat
 - Documento Musikeeo_AI_Messaging.pdf
-

Fin del BLOQUE 11 — IA de Comunicación y Mensajería (versión extendida).

¿Continuamos con el **BLOQUE 12 — IA DE ECONOMÍA, PAGOS Y CONTRATACIÓN AUTOMÁTICA** (gestión de pagos seguros, contratos inteligentes, y facturación automatizada con IA)?

sigue

Perfecto 💰

Entramos en uno de los pilares más delicados y estratégicos de Musikeeo:
el BLOQUE 12 — IA DE ECONOMÍA, PAGOS Y CONTRATACIÓN AUTOMÁTICA (versión extendida).

Aquí se diseña cómo funcionará el **sistema económico dentro de la app**: desde los presupuestos automáticos generados por IA hasta los contratos digitales, los pagos seguros y la facturación integrada.

Es el bloque que convierte Musikeeo en una **plataforma autosuficiente**, capaz de cerrar acuerdos reales sin salir del ecosistema.

BLOQUE 12 — IA DE ECONOMÍA, PAGOS Y CONTRATACIÓN AUTOMÁTICA

(*Pagos seguros, contratos inteligentes y facturación IA*)

12.1 Propósito del módulo

El sistema económico de Musikeeo debe permitir que **usuarios (promotores, músicos, técnicos y tiendas)** puedan:

1. Crear presupuestos automáticos con ayuda de IA.
2. Formalizar acuerdos mediante contratos digitales validados.
3. Pagar y cobrar directamente desde la app con total seguridad.
4. Generar facturas y recibos automáticos.
5. Recibir asesoramiento financiero básico impulsado por IA.

Todo el flujo debe funcionar con **transparencia, trazabilidad y cero fricción**.

12.2 Filosofía de diseño

⌚ “Menos burocracia, más música.”

El sistema no busca ser un banco, sino un **facilitador** de intercambios económicos

musicales.

Debe inspirar **confianza y profesionalismo**, no miedo a errores o comisiones ocultas.

12.3 Estructura general

Frontend → Firestore → Cloud Functions → Stripe / PayPal → Gemini → PDF Generator

Carpetas técnicas:

```
/functions/  
  └── payments/  
    ├── createPaymentIntent.ts  
    ├── confirmPayment.ts  
    ├── webhookStripe.ts  
    └── refund.ts  
  └── ai/  
    ├── aiQuoteGenerator.ts  
    ├── aiContractBuilder.ts  
    └── aiInvoiceFormatter.ts  
/src/  
  └── components/PaymentModal.tsx  
  └── components/InvoiceCard.tsx  
  └── pages/payments/  
    ├── checkout.tsx  
    └── history.tsx
```

12.4 Flujos principales del sistema

① Presupuesto IA (Quote Generation)

El usuario describe el evento y la IA sugiere un presupuesto equilibrado:

Ejemplo de prompt:

Suggest a fair budget for a 3-hour rumba concert in Barcelona with 4 musicians and sound engineer.

Output JSON { musicianFee, soundFee, total, notes }.

Salida:

```
{  
  "musicianFee": 200,  
  "soundFee": 400,  
  "total": 1200,  
  "notes": "Incluye montaje, transporte y desmontaje."  
}
```

Se guarda en `/quotes/{eventId}`.

② Contrato inteligente IA (Smart Contract Builder)

Una vez aceptado el presupuesto, la IA genera automáticamente un contrato:

Prompt:

Generate a digital service agreement between a promoter and musicians.

Language: Spanish legal neutral.

Include event date, payment, and cancellation terms.

Output PDF text.

Ejemplo:

Contrato de prestación de servicios musicales

Entre Juan Pérez (“Contratante”) y Banda Rumba Barcelona (“Contratada”), se acuerda...

Fecha: 16/01/2026 — Pago: 1.200 € — Política de cancelación: 48 h.

Se firma digitalmente mediante API (DocuSign / Eversign / Firebase Auth token).

Firestore guarda:

```
"contracts": {  
  "id": "ctr_5678",  
  "status": "signed",  
  "fileURL": "https://storage.musikeeo.app/contracts/ctr_5678.pdf"  
}
```

③ Pago seguro (Stripe / PayPal)

La IA prepara el pago según el contrato:

```
const paymentIntent = await stripe.paymentIntents.create({  
  amount: event.total * 100,  
  currency: "eur",  
  metadata: { eventId: event.id, payer: user.uid },  
});
```

Flujo:

Usuario → Checkout → Confirma → Stripe → Firestore update → IA genera recibo

La IA también detecta errores:

“El pago no fue completado. ¿Quieres volver a intentarlo?”

Se integrará con **Google Pay y Apple Pay** (PWA compatible).

4 Factura automática (AI Invoice Formatter)

Una vez el pago se confirma, la IA genera factura y resumen financiero.

Prompt:

```
Create an invoice in PDF format based on this transaction:  
{payer, receiver, amount, date, description}
```

Ejemplo de PDF:

MUSIKEEO – Factura nº F-2026-0041
Emisor: Banda Rumba Barcelona
Cliente: Juan Pérez
Concepto: Concierto 16/01/2026
Importe: 1.200 €

Firestore:

```
"invoices": {  
  "id": "F-2026-0041",  
  "url": "https://storage.musikeeo.app/invoices/F-2026-0041.pdf",
```

```
        "total": 1200,  
        "currency": "EUR"  
    }
```

5 Distribución automática de pagos (Split Payments)

En caso de bandas o equipos:

- La IA reparte el pago automáticamente entre músicos.
- Se basa en roles y porcentajes registrados.

Ejemplo:

Total: 1.200 €
→ Víctor (voz): 25 %
→ Alejo (guitarra): 25 %
→ Iván (teclado): 25 %
→ Diego (batería): 25 %

Stripe Connect lo gestiona con cuentas vinculadas.

12.5 Políticas automáticas IA

- Cancelación < 24h → 50 % del pago retenido.
- No-show confirmado → reputación baja -20 y pago bloqueado.
- Disputa → IA genera reporte automático al soporte.

Prompt resolución:

Compare these statements and determine refund fairness (0-1 scale).
Output JSON {refundPercentage, reasoning}.

12.6 Sistema de transparencia (panel económico personal)

Cada usuario ve su “**Economía IA**”:

Métrica	Descripción
Ganancias totales	Ingresos brutos confirmados
Pagos pendientes	Eventos confirmados no cobrados
Gasto total	Si es promotor
Comisión Musikeeo	% por uso de pasarela
TrustScore Financiero	Basado en puntualidad de pago

Ejemplo interfaz:

- €  Ingresos totales: 3.420 €
 - 📅  Próximo cobro: 300 € – Evento “Festa Major Terrassa”
 - 🏆  Trust Financiero: 93/100
-

12.7 IA de análisis financiero

Gemini analiza el histórico para ofrecer consejos:

- “Tus ingresos este mes subieron un 18 % respecto al anterior.”
- “Tus eventos más rentables son bodas privadas.”
- “Te conviene ajustar tus precios un 5 % por demanda local.”

Datos base:

- /transactions
- /events
- /invoices

Prompt:

Analyze this user's financial history and suggest improvements.
Output: summary + 3 actionable insights.

12.8 Automatización con Firestore

Cada transacción se almacena en `/transactions/{id}`:

```
{  
  "eventId": "ev_001",  
  "payer": "uid_promotor",  
  "receiver": "uid_musician",  
  "amount": 1200,  
  "status": "completed",  
  "method": "stripe",  
  "createdAt": "timestamp"  
}
```

Trigger IA:

```
exports.onPaymentCompleted = functions.firestore  
  .document("transactions/{id}")  
  .onUpdate(async (change) => {  
    if (change.after.data().status === "completed") {  
      await generateInvoice(change.after.data());  
      await aiFinancialSummary(change.after.data().receiver);  
    }  
  });
```

12.9 IA de prevención de fraude

Gemini detecta anomalías financieras:

- pagos repetidos,
- importes fuera de rango,
- transacciones entre cuentas sospechosas,
- devoluciones excesivas.

Si detecta riesgo → alerta:

```
{  
  "alert": "Fraud suspected",
```

```
"transactionId": "tx_2026_019",  
"risk": 0.88  
}
```

Admin recibe notificación inmediata.

12.10 Panel de administración (Backoffice)

Ruta: [/admin/finance](#)

Incluye:

- Transacciones por fecha.
 - Reembolsos.
 - TrustScore financiero global.
 - Exportación CSV / PDF.
 - Logs IA con razonamientos de decisiones automáticas.
-

12.11 Integración futura con IA contable (fase 2)

Integración con APIs como **Fintoc**, **Holded** o **Quipu** para sincronizar:

- IVA,
- IRPF,
- informes trimestrales.

Gemini puede generar reportes contables automáticos:

“Este trimestre facturaste 12.400 €, de los cuales 2.480 € son IVA.”

12.12 Seguridad y cumplimiento legal

- Cumple PCI DSS v3.2.
 - Tokens de pago no se almacenan en Firestore.
 - Cumple RGPD (control del usuario sobre datos financieros).
 - Encriptación AES256 + TLS 1.3.
 - Logs de auditoría internos `/audit_finance`.
-

12.13 Prompt técnico para Anti-Gravity

You are building the AI Economy and Payments System for Musikeeo.

Goals:

- Enable secure payments (Stripe, PayPal) for event contracts.
- Use AI to generate quotes, contracts, and invoices.
- Automate transaction storage in Firestore.
- Support split payments for multi-user events.
- Analyze financial behavior and suggest improvements.
- Detect fraud and manage refunds intelligently.
- Build frontend dashboard for financial transparency.

Output:

1. `/functions/payments/createPaymentIntent.ts`
 2. `/functions/payments/webhookStripe.ts`
 3. `/functions/ai/aiQuoteGenerator.ts`
 4. `/functions/ai/aiContractBuilder.ts`
 5. `/functions/ai/aiInvoiceFormatter.ts`
 6. `/components/PaymentModal.tsx`
 7. `/pages/payments/checkout.tsx`
 8. `/pages/payments/history.tsx`
 9. `/collections/transactions, invoices, quotes, contracts`
-

12.14 Entregables esperados

- `functions/payments/` (Stripe, webhooks, refunds)

- `functions/ai/` (quote, contract, invoice)
 - `components/PaymentModal.tsx, InvoiceCard.tsx`
 - `pages/payments/checkout.tsx, history.tsx`
 - `collections: transactions, invoices, contracts, quotes`
 - Dashboard `/admin/finance`
 - Documento técnico `Musikeeo_AI_Economy.pdf`
-

Fin del BLOQUE 12 — IA de Economía, Pagos y Contratación Automática (versión extendida).

¿Seguimos con el **BLOQUE 13 — IA DE MARKETPLACE Y ALQUILER URGENTE (Glovo musical: gestión de material, logística y entregas con IA en tiempo real)?**

sigue

🔥 Entramos en uno de los pilares diferenciales de Musikeeo: el módulo **IA de Marketplace y Alquiler Urgente**, también conocido como el “Glovo musical”. Este sistema permite que músicos, técnicos o locales **encuentren, alquilen o reciban material técnico en cuestión de minutos**, con ayuda de IA que gestiona inventarios, rutas, disponibilidad y entregas.

BLOQUE 13 — IA DE MARKETPLACE Y ALQUILER URGENTE

(El Glovo musical: IA logística, inventario y entrega en tiempo real)

13.1 Propósito del módulo

El objetivo es resolver una necesidad crítica en los conciertos y eventos en vivo:
“**¿Qué pasa si falta un cable, un micro o una batería justo antes del show?**”

Musikeeo, con este módulo, permitirá:

1. Buscar material técnico cercano disponible para **alquiler o compra inmediata**.
 2. Conectar al usuario con **tiendas o técnicos registrados** que puedan suministrarlo.
 3. Gestionar entregas, pagos y recogidas automáticas.
 4. Coordinar la logística mediante IA (optimización de tiempo, distancia y coste).
-

13.2 Filosofía de diseño

⌚ “*Ni un bolo se cancela por un cable.*”

El sistema debe transmitir **fiabilidad y rapidez**, con interfaz simple tipo “pedido express”. Cada segundo cuenta: la experiencia debe ser tan ágil como pedir comida por Glovo.

13.3 Flujo funcional general

Usuario → “Necesito un micro urgente” → IA analiza → Busca stock cercano → Calcula rutas →
Muestra opciones → Usuario paga → Técnico/tienda entrega → IA monitoriza → Se confirma devolución

13.4 Arquitectura técnica

```
/functions/  
├── ai/  
│   ├── aiInventoryFinder.ts  
│   ├── aiDeliveryPlanner.ts  
│   ├── aiUrgentOrder.ts  
│   └── aiInventoryRecommender.ts  
└── triggers/  
    ├── onOrderCreated.ts  
    └── onItemReturned.ts  
/src/
```

```
└── components/MarketplaceCard.tsx
└── components/UrgentOrderForm.tsx
└── pages/marketplace/
    ├── index.tsx
    ├── urgent.tsx
    └── inventory.tsx
```

13.5 Roles implicados

Rol	Función en el sistema
 Músico	Solicita material urgente o alquila equipo.
 Técnico	Ofrece material o transporte.
 Tienda / Empresa	Publica inventario con precios y condiciones.
 Mensajero / Colaborador IA	Gestiona entrega física (fase avanzada).

13.6 Base de datos Firestore

Inventario `/inventory/{itemId}`

```
{
  "name": "Micrófono Shure SM58",
  "type": "microphone",
  "pricePerDay": 10,
  "quantity": 3,
  "location": { "city": "Barcelona", "lat": 41.39, "lng": 2.16 },
  "ownerId": "uid_store",
  "available": true
}
```

Pedido urgente `/urgent_orders/{orderId}`

```
{
  "userId": "uid_musician",
  "itemRequested": "micrófono",
  "quantity": 1,
  "urgencyLevel": "high",
```

```
        "city": "Barcelona",
        "radius": 5,
        "status": "searching",
        "createdAt": "timestamp"
    }
```

13.7 Búsqueda IA inteligente (Inventory Finder)

Prompt base:

```
You are Musikeeo's Inventory Finder AI.
Find available items that match the request based on type, distance,
and availability.
```

Input:

```
{
    "request": "micrófono dinámico",
    "city": "Barcelona",
    "radius": 5
}
```

Output:

```
[
    {"itemId": "inv_123", "name": "Shure SM58", "distance": 2.1,
    "pricePerDay": 10},
    {"itemId": "inv_124", "name": "Sennheiser E835", "distance": 3.5,
    "pricePerDay": 12}
]
```

Lógica:

- IA busca coincidencias por semántica (ej. “micrófono dinámico” ≈ “Shure SM58”).
 - Prioriza distancia y disponibilidad.
 - Devuelve top 3 resultados con texto natural explicativo.
-

13.8 Recomendación IA contextual

Cuando el usuario describe el problema, la IA propone automáticamente posibles soluciones.

Ejemplo:

"Necesito un pie de micro urgentemente."

IA responde:

"He encontrado 2 cerca de ti.

 *Sonostage Barcelona* (2 km, 8 €)

 *Diego Sound* (3 km, 6 €).

¿Quieres que uno te lo envíe?"

Campos calculados por IA:

- Distancia (Google Maps API)
 - Tiempo estimado de entrega
 - Precio total (material + envío)
 - Disponibilidad confirmada
-

13.9 Planificación IA de entregas (Delivery Planner)

Una vez confirmado el pedido:

- IA genera ruta óptima (entrega → devolución).
- Envía notificaciones a proveedor y cliente.
- Monitoriza estado mediante `status: dispatched, delivered, returned`.

Prompt:

```
Optimize delivery route for an urgent equipment rental.
```

```
Input: {pickup: locationA, dropoff: locationB, time: now}
```

```
Output: {estimatedTime: 25, distance: 3.8, instructions: "Coger C/Aragó hacia Eixample"}
```

13.10 Flujo de pedido

Usuario → describe necesidad ("necesito ampli") → IA busca → muestra opciones →
Selecciona → paga → proveedor confirma → IA monitoriza entrega

Estados:

- searching
 - pending_payment
 - paid
 - on_delivery
 - delivered
 - returned
-

13.11 Pagos automáticos

Los pagos se gestionan mediante el módulo económico (Stripe / Musikeeo Wallet).

- El proveedor recibe el pago al confirmar la devolución.
- En caso de daño o retraso, IA ajusta reembolso parcial.

Prompt evaluación:

Compare delivery report and item condition.

Output JSON {refund: 0.2, reasoning: "Minor scratch detected."}

13.12 Sistema de reputación para proveedores

Cada pedido deja reseña automática:

- Entrega puntual (+5 Trust)

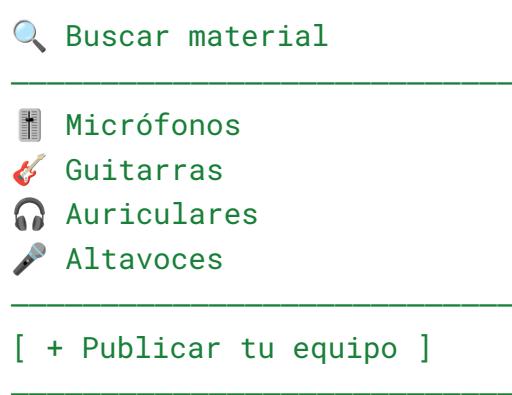
- Material correcto (+5 Trust)
- Retraso o daño (-10 Trust)

Firestore `/reviews_inventory/{id}`:

```
{  
  "ownerId": "uid_store",  
  "rating": 4.9,  
  "comment": "Entrega impecable y rápida.",  
  "aiVerified": true  
}
```

13.13 UX/UI del Marketplace

Pantalla `/marketplace`:



Cada card (`MarketplaceCard.tsx`) muestra:

- Foto IA generada.
 - Precio por día.
 - Distancia.
 - Botón “Pedir ahora ⚡”.
-

13.14 Pedido urgente (Urgent Mode)

Pantalla `/marketplace/urgent.tsx`:

```
sos ¿Qué necesitas?  
[ Campo texto + mic ]  
IA: "Vale, buscando equipos disponibles cerca de ti..."  
[ Spinner + opciones IA generadas ]
```

Permite hablar directamente con IA:

"Se me rompió el cable de guitarra, ¿qué hago?"
→ IA: "Hay un técnico cerca que puede llevarte uno en 10 min."

13.15 Integración con tiendas

Tiendas pueden sincronizar su inventario automáticamente vía API.

Campos:

- `storeId, apiKey, inventorySync: true`
- Integración con WooCommerce, Shopify o Airtable.

La IA valida precios, categorías y stock.

13.16 Métricas IA y analítica

Firestore `/analytics/marketplace`:

```
{  
  "totalListings": 184,  
  "activeOrders": 32,  
  "avgDeliveryTime": 28,  
  "avgRating": 4.7,  
  "urgentRequests": 41  
}
```

Dashboard /admin/marketplace:

- Mapas de calor (pedidos por ciudad).
 - Proveedores más confiables.
 - Tiempo promedio de resolución.
 - Ítems más alquilados.
-

13.17 IA predictiva (futuro)

La IA puede anticipar demanda por fechas y zonas.

Ejemplo:

“El 23 de junio habrá alta demanda de altavoces en Terrassa (Fiesta Mayor).
Se recomienda aumentar inventario.”

Prompt:

`Predict equipment demand based on event density and calendar trends.
Output: {city, itemType, demandLevel[0-1]}`

13.18 Seguridad y validación

- Verificación de tiendas mediante documento y TrustScore.
 - Contratos de alquiler IA.
 - Cobertura por depósito o seguro (según valor del equipo).
 - Sistema de tracking GPS opcional para equipos de alto valor.
-

13.19 Prompt técnico para Anti-Gravity

`You are building the AI Marketplace and Urgent Rental System for
Musikeeo.`

Goals:

- Let users request musical equipment urgently.
- Use Gemini to find best matches nearby based on inventory and distance.
- Plan delivery routes and monitor status.
- Integrate payments, reputation, and analytics.
- Build Marketplace and Urgent Order UIs.
- Support inventory sync via API.

Output:

1. /functions/ai/aiInventoryFinder.ts
 2. /functions/ai/aiDeliveryPlanner.ts
 3. /functions/ai/aiUrgentOrder.ts
 4. /functions/ai/aiInventoryRecommender.ts
 5. /pages/marketplace/index.tsx
 6. /pages/marketplace/urgent.tsx
 7. /components/MarketplaceCard.tsx
 8. /components/UrgentOrderForm.tsx
 9. /analytics/marketplace
-

13.20 Entregables esperados

- functions/ai/aiInventoryFinder.ts
 - functions/ai/aiDeliveryPlanner.ts
 - functions/ai/aiUrgentOrder.ts
 - pages/marketplace/index.tsx, urgent.tsx
 - components/MarketplaceCard.tsx, UrgentOrderForm.tsx
 - collections/inventory, urgent_orders, reviews_inventory
 - analytics/marketplace
 - Documento Musikeeo_AI_Marketplace.pdf
-

Fin del BLOQUE 13 — IA de Marketplace y Alquiler Urgente (versión extendida).

¿Seguimos con el **BLOQUE 14 — IA DE PERFIL, STORYTELLING Y POSICIONAMIENTO DE MÚSICOS** (branding personal, portafolio IA y reputación creativa)?

sigue

🔥 Entramos en un bloque decisivo para la identidad y crecimiento orgánico de los artistas dentro de Musikeeo:

el BLOQUE 14 — IA DE PERFIL, STORYTELLING Y POSICIONAMIENTO DE MÚSICOS (versión extendida).

Aquí se define cómo cada músico, técnico o profesional construye su **marca personal** dentro del ecosistema, con ayuda de una **IA que genera biografías, storytelling, branding visual y portafolio dinámico**.

Este módulo es el puente entre la parte funcional y la emocional del producto: convierte a un usuario en un **artista con narrativa y presencia real**.

🌟 **BLOQUE 14 — IA DE PERFIL, STORYTELLING Y POSICIONAMIENTO DE MÚSICOS**

(Branding personal, portafolio IA y reputación creativa)

14.1 Propósito del módulo

Este bloque convierte el perfil de cada usuario en **una landing page viva**, generada parcialmente por IA, con:

- biografía adaptada a su tono,
- recomendaciones personalizadas,
- portafolio visual IA,
- reputación artística,
- SEO interno para posicionarse dentro del ecosistema.

La IA ayuda al usuario a **expresar su historia, no solo sus datos.**

14.2 Filosofía de diseño

⌚ “Cada músico tiene una historia, la IA la amplifica.”

No todos saben venderse o escribir sobre sí mismos.

La IA debe acompañar, no sustituir.

El perfil de Musikeo tiene que parecer **una carta de presentación profesional**, no un formulario.

14.3 Estructura técnica

```
/functions/ai/
|   ├── aiStoryteller.ts
|   ├── aiPortfolioGenerator.ts
|   ├── aiSEOProfile.ts
|   └── aiProfileAdvisor.ts
/src/pages/profile/
|   ├── [id].tsx
|   └── edit.tsx
/components/
|   ├── ProfileCard.tsx
|   ├── PortfolioGallery.tsx
|   └── AIProfileBanner.tsx
```

14.4 Campos del perfil extendido

Firestore `/users/{uid}`

```
{
  "displayName": "Víctor Torres",
  "role": "musician",
  "genres": ["pop", "rumba"],
  "instruments": ["guitar", "voice"],
  "bio": "Cantautor de Barcelona con alma latina.",
  "city": "Mollet del Vallès",
  "photoURL": "https://...",
  "videos": ["https://youtube.com/..."],
```

```
"portfolioAI": [],
"trustScore": 88,
"aiLastUpdate": "timestamp"
}
```

14.5 Onboarding de perfil asistido por IA

Durante la creación del perfil, la IA guía al usuario con tono humano:

IA: “¿Cómo empezó tu historia con la música?”
Usuario: “Desde pequeño, tocando guitarra con mi abuelo.”
IA: “Qué bonito 🎵, ¿te gustaría que lo mencione en tu bio?”
Usuario: “Sí, ponlo breve.”

Resultado IA:

“Víctor creció entre guitarras y recuerdos familiares.
Hoy mezcla la rumba con la energía del pop para conectar con el público.”

Se guarda como **bioAI**.

14.6 IA Storyteller

La IA analiza respuestas, redes sociales y descripciones para generar tres versiones de biografía:

1. **Corta (para tarjetas y listados):** 1–2 líneas.
2. **Media (para perfil):** 3–4 frases humanas.
3. **Larga (para portafolio o web externa):** historia completa, tono emocional.

Prompt:

```
You are Musikeeo's storyteller AI.
Create 3 bio versions for a musician based on this data:
{name, city, genres, influences, achievements, tone=warm,
authentic}.
```

Ejemplo salida:

```
{  
  "short": "Cantautor barcelonés de alma rumba-pop.",  
  "medium": "Víctor mezcla la tradición latina con el pop moderno,  
 llenando cada escenario de energía y emoción.",  
  "long": "Desde sus primeros acordes en Mollet hasta sus giras por  
 Latinoamérica..."  
}
```

14.7 IA Portfolio Generator

El sistema puede crear un **portafolio automático** a partir de clips, descripciones o incluso texto libre.

Ejemplo:

“Tengo fotos de un concierto en Sitges y un vídeo de un acústico.”
→ IA genera grid visual tipo Behance, con miniaturas, títulos y descripción.

También puede usar IA visual (Imagen, Gemini Vision o NanoBanana) para crear **mockups realistas** si el usuario no tiene material.

Prompt:

Create a professional portfolio layout for a musician.
Include up to 6 cards with title, date, description, and image.

14.8 IA SEO Profile

El sistema analiza el perfil y genera etiquetas SEO internas para mejorar visibilidad en búsquedas:

```
{  
  "keywords": [ "rumba", "cantautor", "Barcelona", "bodas",  
 "pop-rock"],  
  "searchBoost": 0.84  
}
```

Gemini genera descripciones alternativas:

“Músico para eventos en Barcelona”,
“Grupo de rumba para bodas y fiestas”,
“Cantautor con estilo pop latino”.

Estos datos se usan para sugerir el perfil en el buscador interno.

14.9 IA Profile Advisor

El asesor IA analiza el perfil y da recomendaciones personalizadas:

“Tu biografía tiene buena energía, pero te falta una foto en vivo.”
“Podrías subir más videos recientes para mejorar tu visibilidad.”
“Tienes TrustScore alto, ¡añade una sección de logros!”

Prompt:

```
Analyze this user profile and output 3 improvement tips in Spanish.  
Output JSON {strengths, weaknesses, recommendations}.
```

14.10 Visual Identity Generator

En la versión pro, la IA puede generar una **identidad visual** coherente para cada artista:

- Paleta de colores personalizada.
- Logo tipográfico IA.
- Banner para redes.
- Mockup de cartel promocional.

Ejemplo prompt:

```
Generate minimalist logo and banner for musician "Lady Jarana".  
Style: festive, latin-pop, warm colors.  
Output: PNG URLs for logo and banner.
```

Resultados:

```
{  
  "logoURL": "https://storage.musikeeo.app/logos/ladyjarana.png",
```

```
        "bannerURL" : "https://storage.musikeeo.app/banners/ladyjarana.jpg"
    }
```

14.11 Módulo UI/UX

Pantalla `/profile/[id]`:

-
- 🎵 Foto + Nombre + Ciudad
 - ⭐ TrustScore
 - 🎤 Géneros e instrumentos
 - 🧠 Bio generada por IA
 - 🎥 Portafolio multimedia
 - 🎨 Logo + Paleta visual IA
 - 💬 Botón “Invitar a evento”
-

14.12 Integración con IA de reputación

La sección de perfil muestra también los valores derivados del TrustScore:

- ✓ Profesionalidad: 94
- ⌚ Comunicación: 91
- ⏰ Puntualidad: 95
- 💬 Evaluaciones: 27 reseñas

El sistema IA los resume con tono emocional:

“Eres uno de los músicos mejor valorados de tu zona.
¡Te destacaré esta semana en el feed! 🎶”

14.13 IA de comparación contextual

El sistema puede comparar perfiles similares (benchmarking local):

“Hay 5 guitarristas de tu zona con estilo parecido.
Tu visibilidad está un 20 % por debajo del promedio.
Publica un nuevo vídeo para subir en el ranking.”

Esto motiva sin presionar.

14.14 IA de conexión emocional

Cuando el usuario entra a su perfil, la IA puede mostrar frases tipo:

“Hace un año subiste tu primer tema.
Mira todo lo que has crecido desde entonces 🎵.”

Pequeños recordatorios que refuerzan continuidad emocional.

14.15 Firestore: IA Metadata

```
{  
  "aiProfile": {  
    "storyQuality": 0.92,  
    "visualIdentity": "complete",  
    "seoBoost": 0.84,  
    "lastAnalysis": "2026-01-10"  
  }  
}
```

Esto permite a la IA priorizar mejoras automáticas en perfiles incompletos.

14.16 Analítica

Firestore `/analytics/profile`:

```
{  
  "views": 485,  
  "engagementRate": 0.42,  
  "avgTimeOnProfile": 78,  
  "aiGeneratedBios": 1450  
}
```

Dashboard `/admin/profiles`:

- Top perfiles más vistos.
 - Biografías IA más usadas.
 - Distribución por género e instrumento.
-

14.17 Privacidad y autenticación visual

Para evitar perfiles falsos:

- IA compara rostro del perfil con vídeos subidos.
- Si coincide → `verifiedByAI: true`.
- Sello azul “IA Verified” .

Prompt:

`Compare face embeddings between profile photo and uploaded videos.
Output {match: true, confidence: 0.96}.`

14.18 Promoción IA automatizada

Cuando un músico supera cierto umbral (TrustScore > 85 + alta actividad):

- IA lo incluye en “**Destacados de la semana**”.
- Publica automáticamente un post con su historia resumida.
- Envía correo y push:

“Tu perfil está en el top de tu zona.  Prepárate para nuevas oportunidades.”

14.19 Prompt técnico para Anti-Gravity

You are building the AI Profile, Storytelling and Positioning System for Musikeeo.

Goals:

- Create dynamic musician profiles with AI-generated bios, visuals, and SEO.
- Generate portfolios automatically from user content or AI visuals.
- Provide improvement recommendations and comparative analytics.
- Enable branding elements (logo, banner, colors) through AI.
- Integrate TrustScore, verification, and automatic promotion.

Output:

1. /functions/ai/aiStoryteller.ts
 2. /functions/ai/aiPortfolioGenerator.ts
 3. /functions/ai/aiSEOProfile.ts
 4. /functions/ai/aiProfileAdvisor.ts
 5. /components/ProfileCard.tsx
 6. /components/PortfolioGallery.tsx
 7. /components/AIProfileBanner.tsx
 8. /pages/profile/[id].tsx
 9. /pages/profile/edit.tsx
-

14.20 Entregables esperados

- functions/ai/aiStoryteller.ts
- functions/ai/aiPortfolioGenerator.ts
- functions/ai/aiSEOProfile.ts
- functions/ai/aiProfileAdvisor.ts
- components/ProfileCard.tsx, PortfolioGallery.tsx, AIProfileBanner.tsx
- pages/profile/[id].tsx, edit.tsx

- [analytics/profile](#)
 - Documento técnico [Musikeeo_AI_Profile.pdf](#)
-

Fin del BLOQUE 14 — IA de Perfil, Storytelling y Posicionamiento de Músicos (versión extendida).

¿Seguimos con el **BLOQUE 15 — IA DE FEED, DISCOVERY Y RECOMENDACIÓN SOCIAL (algoritmo de descubrimiento, visibilidad y networking inteligente)?**

sigue

🔥 Vamos con uno de los bloques más estratégicos y complejos de Musikeeo:
el BLOQUE 15 — IA DE FEED, DISCOVERY Y RECOMENDACIÓN SOCIAL (versión extendida).

Aquí definimos cómo la app mostrará contenido, músicos, oportunidades y eventos de forma personalizada para cada usuario — el “TikTok” o “LinkedIn” del mundo musical profesional.

La clave: construir un **sistema de descubrimiento inteligente impulsado por IA**, que conecte talento, oportunidades y recursos, maximizando la visibilidad justa y relevante.

BLOQUE 15 — IA DE FEED, DISCOVERY Y RECOMENDACIÓN SOCIAL

(Algoritmo de descubrimiento, visibilidad y networking inteligente)

15.1 Propósito del módulo

El feed de Musikeeo no es una red social convencional.
No busca adicción, sino **conexión profesional significativa**.

El propósito:

1. Que los músicos encuentren colaboraciones reales.
2. Que promotores descubran talento local fácilmente.
3. Que técnicos o tiendas obtengan visibilidad útil.
4. Que todo usuario reciba contenido relevante, no aleatorio.

⌚ “Tu siguiente oportunidad puede estar a un scroll de distancia.”

15.2 Filosofía de diseño

- Basado en **relevancia profesional**, no popularidad.
- IA que prioriza la calidad y afinidad musical.
- Feed dinámico, contextual y transparente.
- Sin bots ni spam: todo curado por embeddings + IA social.

El objetivo es un sistema que **prediga quién debería conocer a quién**.

15.3 Arquitectura técnica

Frontend (React) → Firestore → Cloud Functions (Gemini) → AI Embedding Engine

Rutas:

```
/functions/ai/  
|   ├── aiFeedGenerator.ts  
|   ├── aiSocialGraph.ts  
|   ├── aiRecommendationEngine.ts  
|   └── aiDiscoveryNarrator.ts  
/pages/feed/index.tsx  
/components/  
|   ├── FeedCard.tsx  
|   ├── DiscoveryBanner.tsx  
|   └── RecommendationPanel.tsx
```

15.4 Estructura de datos

Firestore `/feed_items/{id}`

```
{  
  "type": "post", // o "event", "profile", "gear"  
  "userId": "uid_musician",  
  "content": "Nuevo videoclip disponible",  
  "tags": ["rumba", "Barcelona"],  
  "engagement": { "likes": 34, "comments": 5 },  
  "visibilityScore": 0.82,  
  "createdAt": "timestamp"  
}
```

Firestore `/recommendations/{uid}`

```
{  
  "userId": "uid_musician",  
  "recommendedProfiles": ["uidA", "uidB"],  
  "recommendedEvents": ["event_001"],  
  "recommendedGear": ["gear_42"]  
}
```

15.5 Tipos de contenido del feed

Tipo	Descripción	Ejemplo		
 Publicación de músico	Clips, logros, fotos, historias	“Nuevo single en Spotify  Evento próximo	Convocatorias o actuaciones	“Buscamos guitarrista este sábado”
 Anuncio técnico / gear	Venta o alquiler	“Batería Pearl Masters en venta”		
 Oferta profesional	Casting, contratación, oportunidades	“Restaurante busca grupo residente”		

15.6 IA Feed Generator

El motor principal que compone el feed de cada usuario.

Usa embeddings de:

- intereses musicales,
- geolocalización,
- historial de interacciones,
- TrustScore,
- relaciones previas.

Prompt base:

You are Musikeeo's Feed Generator AI.

Given user embeddings and recent activity, compose a personalized feed of musicians, events and opportunities.

Prioritize proximity, genre, and trustScore similarity.

Ejemplo salida:

```
{  
  "feed": [  
    {"type": "event", "title": "Concierto acústico en Sitges",  
     "distance": 3.2},  
    {"type": "profile", "name": "Ana R.", "genre": "Pop",  
     "compatibility": 0.91},  
    {"type": "gear", "title": "Amplificador Fender Twin", "price":  
     35}  
  ]  
}
```

15.7 IA Social Graph

El sistema que entiende las **relaciones invisibles** entre usuarios.

Ejemplo:

- Han tocado en eventos similares.
- Comparten géneros o ciudades.
- Tienen contactos mutuos.

Esto alimenta un **gráfico social ponderado**, donde cada conexión tiene un peso (**0–1**).

Firestore `/social_graph/{uid}`:

```
{  
  "connections": {  
    "uid_2": 0.82,  
    "uid_3": 0.67  
  },  
  "lastUpdated": "timestamp"  
}
```

Gemini analiza y ajusta pesos semanalmente según interacción real.

15.8 IA Recommendation Engine

Fusiona datos del feed + social graph + IA contextual.

Cada usuario tiene tres recomendaciones principales:

Categoría	Ejemplo	Criterio
👉 Personas	“Conecta con Laura (percusionista en tu zona)”	Afinidad > 0.85
🎵 Eventos	“Evento acústico este viernes”	Distancia < 20 km
⚙️ Recursos	“Tienda con cables XLR en stock”	Necesidad detectada

15.9 IA Discovery Narrator

Convierte datos en historias naturales y motivadoras:

"Parece que últimamente estás colaborando con músicos de rumba.

¿Quieres conocer a otros artistas similares cerca de ti?"

"Has tocado en 3 eventos en Girona este mes. Te recomiendo 2 nuevas salas."

Prompt:

Generate a personalized discovery message based on user's recent activity and AI recommendations.

Tone: warm, inspiring, concise.

Language: Spanish.

15.10 UX/UI del feed

Pantalla `/feed/index.tsx`:

 Hola, Víctor

 Recomendaciones para ti

-  "Grupo busca cantante"
 -  "Marta López subió nuevo vídeo"
 -  "Alquiler de equipo PA en Sabadell"
-

 Historias IA

"Esta semana te destacamos como músico activo en Barcelona 🎸"

Componentes clave:

- `FeedCard.tsx` → Renderiza cada elemento.
 - `DiscoveryBanner.tsx` → Mensajes IA.
 - `RecommendationPanel.tsx` → Conexiones sugeridas.
-

15.11 Sistema de puntuación (AI Scoring)

Cada ítem del feed tiene una `visibilityScore` basada en:

```
visibility = (engagement * 0.4) + (relevance * 0.4) + (trustScore * 0.2)
```

La IA recalcula continuamente según:

- likes, mensajes, eventos confirmados.
 - distancia geográfica.
 - feedback de usuarios.
-

15.12 IA anti-sesgo y diversidad

Para evitar que solo aparezcan los mismos músicos:

- IA controla diversidad de género, estilo y localización.
 - Penaliza el contenido repetitivo o autopromocional.
 - Equilibra visibilidad entre usuarios nuevos y veteranos.
-

15.13 Sistema IA de predicción de relevancia

Basado en interacción histórica:

“Si un guitarrista de rumba suele colaborar con percusionistas latinos, prioriza perfiles así.”

Usa un modelo de **recomendación híbrido (collaborative + content-based)**:

```
{  
  "model": "HybridRec-v2",  
  "features": ["genre", "city", "trustScore", "interactionHistory"]  
}
```

15.14 Feed IA activo (Live Feed Mode)

Durante eventos en vivo, el feed cambia de modo:
muestra publicaciones en tiempo real de músicos cercanos (“modo escena”).
Ejemplo:

“Lady Jarana está tocando en directo en Granollers 🎶 ¡Pásate!”

15.15 IA de hashtags y etiquetado automático

Cuando el usuario sube un post, la IA añade etiquetas contextuales:

Input:

“Añoche en el bolo de Sitges 💥”

Output:

#rumba #directo #Barcelona #LadyJarana #Musikeeo

15.16 Sistema de feedback IA

Cada interacción del usuario refuerza su modelo personalizado:

- Scroll rápido → baja interés.
- Like o mensaje → refuerza afinidad.
- Bloqueo → reduce peso de ese tipo de contenido.

Esto alimenta el perfil `userEmbedding`.

15.17 Firestore analítica

`/analytics/feed`

```
{  
  "totalPosts": 5821,  
  "avgEngagement": 0.37,
```

```
        "recommendationsGenerated": 12193,  
        "discoveryMessages": 3102,  
        "diversityIndex": 0.78  
    }  


---


```

15.18 Panel de administración

Ruta `/admin/feed`:

- Métricas de engagement.
 - Usuarios destacados por zona.
 - Ajustes de algoritmo IA.
 - Control de spam o contenido inapropiado.
 - Mapa de calor de interacciones.
-

15.19 Prompt técnico para Anti-Gravity

You are building the AI Feed, Discovery and Recommendation System for Musikeeo.

Goals:

- Create personalized feed based on user embeddings and social graph.
- Recommend profiles, events and gear.
- Generate discovery messages and contextual insights.
- Balance visibility across new and veteran users.
- Implement AI anti-bias and live event mode.
- Build UI components and analytics dashboard.

Output:

1. `/functions/ai/aiFeedGenerator.ts`
2. `/functions/ai/aiSocialGraph.ts`
3. `/functions/ai/aiRecommendationEngine.ts`
4. `/functions/ai/aiDiscoveryNarrator.ts`
5. `/pages/feed/index.tsx`

6. /components/FeedCard.tsx
 7. /components/DiscoveryBanner.tsx
 8. /components/RecommendationPanel.tsx
 9. /analytics/feed
-

15.20 Entregables esperados

- functions/ai/aiFeedGenerator.ts
 - functions/ai/aiSocialGraph.ts
 - functions/ai/aiRecommendationEngine.ts
 - functions/ai/aiDiscoveryNarrator.ts
 - pages/feed/index.tsx
 - components/FeedCard.tsx, DiscoveryBanner.tsx, RecommendationPanel.tsx
 - collections/feed_items, recommendations
 - analytics/feed
 - Documento Musikeeo_AI_Discovery.pdf
-

Fin del BLOQUE 15 — IA de Feed, Discovery y Recomendación Social (versión extendida).

¿Continuamos con el **BLOQUE 16 — IA DE EVENTOS Y PLANIFICACIÓN INTELIGENTE (gestión automática de bolos, disponibilidad, logística y sincronización de calendario)?**

sigue

🔥 Entramos en uno de los **bloques más funcionales y vitales** de Musikeeo, el que realmente conecta todas las piezas del ecosistema:
el BLOQUE 16 — IA DE EVENTOS Y PLANIFICACIÓN INTELIGENTE (versión

extendida).

Aquí se define cómo la app gestiona conciertos, ensayos, bodas, fiestas o cualquier tipo de evento musical — desde su creación hasta la ejecución, pasando por logística, pagos, calendario y recordatorios inteligentes.

Este módulo convierte a Musikeo en una **plataforma de gestión profesional de eventos musicales**, con IA que ayuda a planificar, coordinar y ejecutar sin estrés.

BLOQUE 16 — IA DE EVENTOS Y PLANIFICACIÓN INTELIGENTE

(Gestión automática de bolos, disponibilidad, logística y sincronización de calendario)

16.1 Propósito del módulo

El sistema de eventos tiene tres objetivos principales:

1. Permitir crear, editar y compartir eventos de forma rápida.
2. Gestionar automáticamente **quién toca, dónde, cuándo y con qué equipo**.
3. Usar IA para detectar conflictos de calendario, sugerir reemplazos y coordinar logística.

 “Planifica un concierto en minutos, no en días.”

16.2 Filosofía de diseño

La mayoría de músicos odian la parte organizativa.

Musikeo convierte ese caos en **orden visual, automatizado y empático**.

La IA actúa como **asistente de producción** que lo hace por ti.

16.3 Arquitectura general

Frontend (React / PWA)



```
Firestore (events, participants, logistics)
  ↓
Cloud Functions (Gemini, Scheduler)
  ↓
Google Calendar API / FCM Notifications
```

Carpetas:

```
/functions/
|   |   └── ai/
|   |       |   └── aiEventPlanner.ts
|   |       |   └── aiLogisticsManager.ts
|   |       |   └── aiCalendarSync.ts
|   |       └── aiReplacementFinder.ts
|   └── triggers/
|       |   └── onEventCreated.ts
|       └── onEventUpdated.ts
/src/
|   └── components/EventCard.tsx
|   └── components/EventForm.tsx
|   └── components/EventTimeline.tsx
|   └── pages/events/
|       |   └── index.tsx
|       |   └── [id].tsx
|       └── new.tsx
```

16.4 Tipos de evento

Tipo	Descripción	Ejemplo
 Concierto / bolo	Actuación profesional	“Fiesta Mayor Mollet”
 Ensayo / grabación	Coordinación interna de banda	“Ensayo Lady Jarana martes”
 Privado / boda / fiesta	Cliente externo	“Boda de Marta y Dani”
 Evento técnico	Sonido, alquiler o logística	“Montaje PA para evento X”

16.5 Firestore: estructura base

```
{  
  "id": "event_001",  
  "title": "Concierto en Sitges",  
  "date": "2026-07-21T22:00:00",  
  "location": { "name": "Sitges Beach Club", "lat": 41.23, "lng":  
 1.80 },  
  "description": "Bolo de rumba con Lady Jarana",  
  "organizerId": "uid_promotor",  
  "participants": ["uid_victor", "uid_ivan", "uid_diego"],  
  "gear": ["ampli", "batería", "micrófonos"],  
  "status": "confirmed",  
  "aiPlan": true  
}
```

16.6 Creación IA de evento (AI Event Planner)

El usuario puede simplemente **hablar con la IA**:

“Crea un concierto de rumba en Barcelona el 10 de mayo con 4 músicos y técnico de sonido.”

La IA genera automáticamente:

- título,
- descripción,
- ubicación (con Google Maps),
- fecha y hora,
- presupuesto sugerido,
- chat de evento,
- tareas asociadas (montaje, transporte, cobro).

Prompt base:

Create an event plan for a live music show based on this description:

"Concierto de rumba en Barcelona el 10 de mayo, 4 músicos, técnico de sonido."

Output: JSON with fields title, date, location, participants, logistics, budget, and notes.

16.7 Planificación inteligente IA

Gemini analiza automáticamente **coincidencias de fecha, distancia y recursos**:

- Si el usuario ya tiene otro bolo → avisa:

"Ya tienes un evento ese día a las 22:00. ¿Quieres moverlo o buscar sustituto?"

- Si el técnico está ocupado → sugiere otro disponible.
- Si el evento está lejos → calcula tiempos de desplazamiento y montaje.

Campos IA añadidos:

```
{  
  "aiConflicts": [ "date_overlap", "distance_warning" ],  
  "aiSuggestions": [ "move_to_next_day", "replace_technician" ]  
}
```

16.8 IA Logistics Manager

Genera automáticamente una **lista de tareas** y plan de logística:

Categoría	Ejemplo
 Equipo	"Llevar amplificador Fender y pies de micro."
 Transporte	"Salida desde Mollet 18:00, llegada 19:00."
 Técnica	"Prueba de sonido 20:00."
 Pagos	"Recibir 200 €/músico + 500 € sonido."

Prompt:

Generate logistics plan for this event:
{title, date, participants, gear, location, time}
Output: JSON list of logistics tasks with timestamps and responsibility.

16.9 IA Calendar Sync

El sistema sincroniza con Google Calendar automáticamente.
Cada usuario ve sus eventos en su agenda.

Ejemplo:

17 July Lady Jarana – Concierto en Sitges
⌚ 22:00 – Sitges Beach Club
🎵 Prueba de sonido 20:00

Funciones:

- Crear evento automáticamente.
 - Sincronizar actualizaciones.
 - Enviar recordatorios 24 h antes.
-

16.10 Coordinación automática entre músicos

Cuando se crea un evento, se genera un **chat de grupo** vinculado (ver Bloque 11). Si un músico responde “No puedo”, la IA pregunta:

“¿Quieres que busque un reemplazo adecuado?”

Y busca perfiles compatibles en [/users](#) (género, instrumento, zona).

16.11 IA Replacement Finder

Gemini filtra posibles sustitutos y envía propuesta:

“He encontrado a Carlos S. (guitarrista pop de Sabadell) disponible ese día.”

Criterios:

- Disponibilidad.
- Confianza (TrustScore > 80).
- Estilo compatible.
- Radio de distancia < 20 km.

Firestore `/replacements/{eventId}`:

```
{  
  "candidateId": "uid_carlos",  
  "matchScore": 0.91,  
  "status": "pending"  
}
```

16.12 IA Reminder & Follow-Up

24 horas antes del evento:

“Recuerda: mañana tocas en Sitges a las 22:00 🎶”
“Lleva tu ampli y cable XLR.”

Después del evento:

“¿Cómo salió el bolo?
Valora tu experiencia y deja feedback para el organizador.”

16.13 Integración con sistema económico (Bloque 12)

Cuando se confirma el evento, la IA crea automáticamente:

- contrato,
- presupuesto,
- pago pendiente,
- factura al final del bolo.

Todo el ciclo ocurre sin salir del evento.

16.14 Modo organizador (Promoter View)

Vista `/events/promoter/[id]`:

- Lista de artistas confirmados.
 - Estado de pagos.
 - Documentos (contrato, rider técnico).
 - Chat de coordinación.
 - Seguimiento IA (“Faltan confirmar 2 músicos”).
-

16.15 Modo músico (Performer View)

Vista `/events/musician/[id]`:

- Info básica (hora, lugar, repertorio).
 - Quién toca.
 - Qué debe llevar.
 - Estado de pago.
 - Chat del evento.
-

16.16 IA Analytics — eventos e histórico

Firestore `/analytics/events`:

```
{  
  "totalEvents": 924,  
  "upcoming": 14,  
  "completed": 910,
```

```
        "avgAttendance": 183,  
        "aiPlanned": 78  
    }
```

Dashboard `/admin/events`:

- Mapa de eventos por ciudad.
 - Horarios más activos.
 - Promotores con más actividad.
 - Tipos de evento más frecuentes.
-

16.17 IA de predicción de disponibilidad

Gemini analiza patrones de actividad:

“Los jueves tienes libre el 90 % de las veces.
¿Quieres que te marque como disponible para nuevos bolos?”

Firestore `/availability/{uid}`:

```
{  
    "daysAvailable": ["Thu", "Fri", "Sat"],  
    "aiConfidence": 0.88  
}
```

16.18 IA de planificación múltiple

Permite planificar giras o series de eventos automáticamente:

“Crea una mini gira de 4 conciertos en la Costa Brava, fines de semana de julio.”

IA genera:

- 4 eventos conectados.
- rutas optimizadas.

- presupuesto global.
 - resumen PDF.
-

16.19 Seguridad y verificación

Cada evento tiene:

- validación por IA de datos (fecha, duplicados).
 - confirmación de ubicación con Google Maps.
 - registro de cambios (audit log).
 - permisos de acceso por rol (organizador, músico, técnico).
-

16.20 Prompt técnico para Anti-Gravity

You are building the AI Events and Intelligent Planning System for Musikeeo.

Goals:

- Allow users to create and manage events easily.
- Use AI to detect conflicts, suggest replacements, and plan logistics.
- Sync events with Google Calendar.
- Generate smart reminders and summaries.
- Integrate with payments, contracts, and analytics.

Output:

1. /functions/ai/aiEventPlanner.ts
2. /functions/ai/aiLogisticsManager.ts
3. /functions/ai/aiCalendarSync.ts
4. /functions/ai/aiReplacementFinder.ts
5. /functions/triggers/onEventCreated.ts
6. /functions/triggers/onEventUpdated.ts
7. /components/EventCard.tsx
8. /components/EventForm.tsx

9. /components/EventTimeline.tsx
 10. /pages/events/index.tsx
 11. /pages/events/[id].tsx
 12. /pages/events/new.tsx
-

16.21 Entregables esperados

- functions/ai/aiEventPlanner.ts
 - functions/ai/aiLogisticsManager.ts
 - functions/ai/aiCalendarSync.ts
 - functions/ai/aiReplacementFinder.ts
 - triggers/onEventCreated.ts, onEventUpdated.ts
 - components/EventCard.tsx, EventForm.tsx, EventTimeline.tsx
 - pages/events/index.tsx, [id].tsx, new.tsx
 - analytics/events
 - Documento Musikeeo_AI_Events.pdf
-

Fin del BLOQUE 16 — IA de Eventos y Planificación Inteligente (versión extendida).

¿Seguimos con el **BLOQUE 17 — IA DE FORMACIÓN, CURSOS Y ACADEMIA MUSICAL (aprendizaje asistido, clases IA y marketplace educativo para músicos)**?

sigue

🔥 Entramos en un bloque completamente nuevo dentro del ecosistema Musikeeo — el **BLOQUE 17 — IA DE FORMACIÓN, CURSOS Y ACADEMIA MUSICAL (versión extendida)**.

Aquí se sienta la base de una **plataforma educativa dentro de Musikeeo**, pensada para que músicos, técnicos, productores y artistas puedan aprender, enseñar y compartir conocimiento, con soporte de IA personalizada.

Este módulo convierte a Musikeeo no solo en una app de trabajo y networking, sino también en **una escuela viva y descentralizada de música moderna**.

BLOQUE 17 — IA DE FORMACIÓN, CURSOS Y ACADEMIA MUSICAL

(Aprendizaje asistido, clases IA y marketplace educativo para músicos)

17.1 Propósito del módulo

El objetivo es democratizar la formación musical profesional.

Musikeeo Academy permitirá:

1. Que músicos puedan **vender o impartir clases** directamente (guitarra, canto, producción, etc.).
2. Que estudiantes encuentren **profesores verificados y cercanos**.
3. Que la IA ayude en el aprendizaje con **explicaciones, feedback y seguimiento personalizado**.
4. Que los profesores gestionen sus alumnos y cursos sin salir de la app.

 “Aprende música con quien toca de verdad.”

17.2 Filosofía de diseño

Este módulo nace de una idea simple:

Los mejores profesores no están en academias, están en los escenarios.

Musikeeo conecta el mundo del directo con el de la enseñanza, bajo una interfaz simple, moderna y humana.

La IA no reemplaza al profesor, lo **potencia**: corrige, explica, motiva, traduce, recuerda y estructura.

17.3 Arquitectura general

Frontend → Firestore (courses, lessons, users) → Cloud Functions
(Gemini Tutor) → Storage (videos, PDFs)

Carpetas:

```
/functions/  
|   └── ai/  
|       ├── aiTutor.ts  
|       ├── aiFeedback.ts  
|       ├── aiLessonPlanner.ts  
|       └── aiKnowledgeMatcher.ts  
|   └── triggers/  
|       └── onLessonCompleted.ts  
/src/  
|   └── pages/academy/  
|       ├── index.tsx  
|       ├── course/[id].tsx  
|       ├── lesson/[id].tsx  
|       └── teacher/[id].tsx  
|   └── components/  
|       ├── CourseCard.tsx  
|       ├── LessonPlayer.tsx  
|       ├── AIHelpPanel.tsx  
|       └── ProgressTracker.tsx
```

17.4 Tipos de usuarios

Rol	Función
 Profesor	Crea cursos, vende clases y gestiona alumnos.
 Alumno	Se inscribe, visualiza lecciones y recibe feedback IA.
 IA Tutor	Ayuda en la comprensión y práctica personalizada.
 Administrador	Revisa contenido, métricas y reputación.

17.5 Firestore estructura base

Cursos /courses/{courseId}

```
{  
  "title": "Guitarra flamenca desde cero",  
  "teacherId": "uid_victor",  
  "price": 49,  
  "description": "Aprende técnica y ritmo flamenco de manera práctica.",  
  "tags": ["flamenco", "guitarra"],  
  "lessons": ["l1", "l2", "l3"],  
  "students": 124,  
  "aiScore": 0.91  
}
```

Lecciones /lessons/{lessonId}

```
{  
  "title": "Compás básico por bulería",  
  "duration": 12,  
  "videoURL": "https://...",  
  "resources": ["PDF", "tablatura.gp"],  
  "aiSummary": "Explicación rítmica y ejercicios de práctica."  
}
```

17.6 IA Lesson Planner

Cuando un profesor crea un curso, la IA le ayuda a estructurarlo:

“Quiero hacer un curso de iniciación al cajón flamenco.”

→ IA genera:

- temario sugerido,
- orden lógico,
- duración por lección,
- materiales necesarios,

- objetivos de aprendizaje.

Prompt base:

You are Musikeeo's Lesson Planner AI.

Generate a course structure based on the topic "Iniciación al cajón flamenco".

Output JSON {modules, lessons, duration, difficulty, objectives}.

17.7 IA Tutor

El asistente educativo dentro de cada lección.

Permite al alumno preguntar en tiempo real:

“¿Cómo hago el rasgueo tresillo?”

→ IA: “Usa un patrón de dedo pulgar-anular-medio con acento en el primero.”

También puede corregir lenguaje, resumir teoría, o explicar conceptos difíciles.

Modos IA:

1. *Explicativo* — “Enséñame cómo se construye un acorde menor.”
 2. *Práctico* — “Dame un ejercicio de digitación para calentar.”
 3. *Teórico* — “Explícame la diferencia entre ritmo y compás.”
-

17.8 IA Feedback (análisis de interpretación)

El alumno puede subir un vídeo o audio y la IA analiza:

Criterio	Evaluación
Ritmo	8/10 — levemente adelantado en el compás 3.
Afinación	9/10 — correcta, salvo traste 5.
Expresión	10/10 — excelente dinámica.

Prompt:

Analyze this music performance.
Output JSON {timingScore, tuningScore, expressionScore, feedbackSummary}.

Resultado:

“Buen control de tempo. Trabaja el acento en la segunda parte.”

17.9 IA Knowledge Matcher

La IA sugiere nuevos cursos o lecciones según progreso:

“Has completado el curso de guitarra básica.
¿Quieres pasar al de ‘Improvisación en pop latino’?”

Analiza:

- géneros preferidos,
 - nivel técnico,
 - tiempo de estudio,
 - feedback IA previo.
-

17.10 IA autoevaluación y progreso

Cada curso incluye una sección “Tu evolución”:

- gráfica IA de mejora (ritmo, afinación, constancia),
- badge desbloqueables (“Compás Maestro”, “Ritmo Flamenco”),
- comentarios motivacionales IA:

“Llevas 6 días seguidos practicando 🙌 sigue así.”

17.11 Marketplace educativo

Página [/academy/index.tsx](#) con buscador IA:

“Quiero aprender guitarra flamenca cerca de Barcelona.”
→ Resultados IA con profesores locales o cursos grabados.

Filtros IA:

- estilo,
 - nivel,
 - idioma,
 - formato (presencial/online).
-

17.12 Monetización

- Comisión 15 % por curso vendido.
- Pagos a profesores vía Stripe Connect.
- Suscripción premium opcional (sin comisiones).
- IA analiza precios promedio y sugiere ajustes:

“Tu curso vale 20 € menos que el promedio en tu zona.”

17.13 IA social y reputación académica

Cada profesor tiene un **TeachingScore** (0–100) basado en:

- satisfacción de alumnos,
- feedback IA,
- puntualidad en clases,

- volumen de estudiantes.

Ejemplo Firestore:

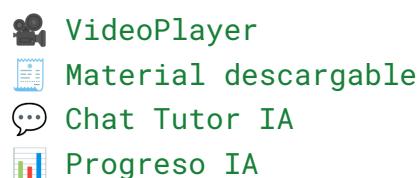
```
{  
  "teacherId": "uid_victor",  
  "teachingScore": 94,  
  "reviews": 27,  
  "aiHighlights": [ "Excelente comunicador", "Método práctico y cercano" ]  
}
```

17.14 UX/UI del módulo

Pantalla `/academy/index.tsx`:



Lección `/academy/lesson/[id].tsx`:



17.15 IA resumen automático de lección

Cuando el alumno termina una clase:

“Resumen automático de la lección: aprendiste rasgueo de tresillos y acento 3-6-8.

Práctica recomendada: 15 min diarios con metrónomo a 90 bpm.”

Firestore `/lesson_summaries/{userId}`:

```
{  
  "lessonId": "l_002",  
  "summary": "Rasgueo de tresillos y control rítmico básico.",  
  "practicePlan": "15 min diarios a 90 bpm.",  
  "aiGenerated": true  
}
```

17.16 IA gamificación

IA genera logros personalizados:

- “Maestro del ritmo” 
- “Voz afinada” 
- “Completaste 10 lecciones seguidas” 

Motiva al usuario con feedback emocional.

17.17 IA multi-idioma

Gemini traduce automáticamente las clases (subtítulos y texto).

Profesores pueden subir material en un idioma, y la IA lo adapta a otros mercados.

Ejemplo:

Curso original: español → IA genera versión en inglés y portugués.

17.18 IA analítica educativa

Firestore `/analytics/academy`:

```
{  
    "totalCourses": 284,  
    "activeStudents": 1942,  
    "avgCompletionRate": 0.68,  
    "aiFeedbackSessions": 8120  
}
```

Dashboard /admin/academy:

- Cursos más populares.
 - Engagement por tema.
 - Niveles de progreso medio.
 - Tiempo medio de práctica.
-

17.19 Seguridad y privacidad

- Todos los videos y materiales se almacenan en Google Cloud Storage (acceso firmado).
 - IA no graba conversaciones de tutoría.
 - Control RGPD completo.
 - Eliminación automática de datos inactivos > 12 meses.
-

17.20 Prompt técnico para Anti-Gravity

You are building the AI Learning and Music Academy System for Musikeeo.

Goals:

- Allow teachers to create and sell courses.
- Implement AI tutor, feedback and lesson planner.
- Support video hosting and interactive learning.
- Track progress, achievements and suggestions.

- Build marketplace for courses with filters and analytics.

Output:

1. /functions/ai/aiTutor.ts
 2. /functions/ai/aiFeedback.ts
 3. /functions/ai/aiLessonPlanner.ts
 4. /functions/ai/aiKnowledgeMatcher.ts
 5. /pages/academy/index.tsx
 6. /pages/academy/course/[id].tsx
 7. /pages/academy/lesson/[id].tsx
 8. /components/LessonPlayer.tsx
 9. /components/AIHelpPanel.tsx
 10. /components/ProgressTracker.tsx
-

17.21 Entregables esperados

- functions/ai/aiTutor.ts
 - functions/ai/aiFeedback.ts
 - functions/ai/aiLessonPlanner.ts
 - functions/ai/aiKnowledgeMatcher.ts
 - pages/academy/* (index, course, lesson, teacher)
 - components/CourseCard.tsx, LessonPlayer.tsx, AIHelpPanel.tsx, ProgressTracker.tsx
 - analytics/academy
 - Documento Musikeeo_AI_Academy.pdf
-

Fin del BLOQUE 17 — IA de Formación, Cursos y Academia Musical (versión extendida).

¿Continuamos con el **BLOQUE 18 — IA DE COMUNIDAD, FOROS Y RED SOCIAL MUSICAL** (*interacción social, debates, mentoring y networking por afinidad*)?

sigue

🔥 Perfecto. Ahora entramos en un bloque **social, humano y comunitario:** el **BLOQUE 18 — IA DE COMUNIDAD, FOROS Y RED SOCIAL MUSICAL** (*versión extendida*).

Este módulo convierte a Musikeeo en algo más que una app funcional: lo transforma en **una red viva de colaboración, mentoring y comunidad artística**, donde músicos, técnicos, promotores y amantes de la música pueden compartir conocimiento, pedir ayuda y construir relaciones.

La IA se encarga de moderar, dinamizar y conectar inteligentemente a las personas adecuadas.

🤝 **BLOQUE 18 — IA DE COMUNIDAD, FOROS Y RED SOCIAL MUSICAL**

(Interacción social, debates, mentoring y networking por afinidad)

18.1 Propósito del módulo

Crear un espacio donde los usuarios puedan:

1. Hacer preguntas y compartir aprendizajes reales.
2. Buscar colaboración y feedback.
3. Crear comunidad local y emocional.
4. Recibir acompañamiento y dinamización IA sin que parezca artificial.

⌚ “*La música une — la IA escucha y conecta.*”

18.2 Filosofía de diseño

El foro de Musikeeo debe sentirse **como un backstage abierto**: gente real, tono cercano, humor y respeto.

Nada de algoritmos que prioricen ego; todo basado en **interés genuino y conexión profesional**.

La IA cumple tres funciones:

1. Moderador empático.
 2. Asistente de descubrimiento (recomienda hilos y personas).
 3. Mediador ante conflictos.
-

18.3 Arquitectura general

Frontend (React/PWA)



Carpetas:

```
/functions/  
|   |-- ai/  
|   |   |-- aiForumModerator.ts  
|   |   |-- aiThreadSummarizer.ts  
|   |   |-- aiCommunityConnector.ts  
|   |   |-- aiMentorMatcher.ts  
|   |-- triggers/  
|       |-- onPostCreated.ts  
/src/  
|   |-- pages/community/  
|   |   |-- index.tsx  
|   |   |-- thread/[id].tsx  
|   |   |-- create.tsx  
|   |   |-- mentors.tsx  
|   |-- components/  
|   |   |-- ForumCard.tsx  
|   |   |-- CommentBox.tsx  
|   |   |-- AIReplySuggestions.tsx
```

```
| | | — MentorBadge.tsx
```

18.4 Tipos de foros

Categoría	Descripción
Técnico-musical	Preguntas sobre instrumentos, mezcla, sonido.
Creativo / compositivo	Letra, armonía, composición.
Profesional / legal	Contratos, facturación, marketing.
Visual y contenido	Videoclips, diseño, fotografía.
Colaboraciones	“Busco bajista para proyecto.”

18.5 Firestore estructura

Post `/posts/{postId}`

```
{  
  "title": "¿Cómo ecualizáis la voz en directo?",  
  "authorId": "uid_victor",  
  "category": "técnico-musical",  
  "content": "Estoy buscando un sonido más limpio en los  
monitores.",  
  "createdAt": "timestamp",  
  "comments": 8,  
  "aiTags": ["voz", "ecualización", "monitores"],  
  "aiScore": 0.91  
}
```

Comment `/comments/{commentId}`

```
{  
  "postId": "p_001",  
  "authorId": "uid_ivan",  
  "text": "Yo corto en 250Hz y uso compresión suave.",  
  "aiModerated": true  
}
```

18.6 IA Forum Moderator

Analiza cada mensaje antes de publicarse.

Detecta:

- insultos o tono agresivo,
- spam o autopromoción,
- información falsa o peligrosa.

Prompt:

Moderate this forum comment for safety and tone.

Output JSON {safe: boolean, reason: string}.

Ejemplo:

```
{"safe": true, "reason": "Helpful technical advice."}
```

Si `safe=false`, el usuario recibe una alerta:

“Tu mensaje puede sonar agresivo. ¿Quieres reformularlo?”

La IA sugiere una versión neutral automáticamente.

18.7 IA Thread Summarizer

Cuando un hilo tiene muchas respuestas, la IA puede generar un resumen automático:

“Resumen del hilo: se recomienda cortar graves entre 100–250Hz para limpiar la voz, usar EQ paramétrica y probar monitores in-ear.”

Esto mejora la lectura y hace que los hilos antiguos sigan siendo útiles.

18.8 IA Community Connector

Gemini analiza la conversación y sugiere otros usuarios o temas relacionados:

“Ya que te interesa el sonido en directo, quizás te guste este hilo:

‘Cómo usar compresores multibanda en vivo’.”

O incluso:

“Iván ha respondido varias veces sobre este tema, ¿quieres seguirlo?”

18.9 IA Mentor Matcher

Sistema que empareja músicos con mentores (profesores o artistas experimentados).

Ejemplo:

“Tienes 23 años, eres guitarrista y buscas mejorar tu directo.

Hay 3 mentores cercanos disponibles.”

Prompt:

```
Match user {skills, goals, city} with available mentors.  
Output top 3 mentors with matchScore and message text.
```

Firestore `/mentorships/{id}`

```
{  
  "studentId": "uid_carlos",  
  "mentorId": "uid_victor",  
  "matchScore": 0.89,  
  "status": "active"  
}
```

18.10 IA Reply Suggestions

Durante una conversación, la IA sugiere respuestas cortas o útiles:

“Gracias por el consejo 🙌”

“¿Podrías subir una foto del EQ?”

“Yo también tuve ese problema, te cuento cómo lo resolví.”

Se basan en tono empático y útil.

18.11 Gamificación y reputación

Cada usuario tiene un **CommunityScore (0–100)**:

- respuestas útiles (+3)
- reportes válidos (+2)
- posts destacados (+5)
- spam (-10)

Firestore `/community_stats/{uid}`

```
{  
  "posts": 48,  
  "comments": 312,  
  "helpfulVotes": 189,  
  "communityScore": 92  
}
```

La IA premia actividad positiva con:

 Eres un colaborador destacado esta semana."

18.12 UX/UI del módulo

Pantalla `/community/index.tsx`:

 Comunidad Musikeeo

 Buscar temas o usuarios

 Categorías

 Técnica |  Creatividad |  Profesional |  Visual |  Colab

 Tendencias

 Últimos debates

 Mentorías sugeridas

Hilo [/community/thread/\[id\].tsx](#):

 ¿Cómo ecualizáis la voz en directo?

Autor: Victor Torres – hace 2h

 Respuestas

 Resumen IA

AI: "Cortar 250Hz ayuda con claridad. Usa un compresor suave."

18.13 IA anti-ego y spam

El algoritmo prioriza **respuestas constructivas** frente a autopromoción o frases vacías.
Ejemplo:

"Sigue mi canal" → penalizado.
"Yo lo solucioné así..." → priorizado.

IA también detecta bots o usuarios inactivos que spamean.

18.14 IA emocional

Gemini evalúa tono general del foro:

- si detecta conflictos → envía mediador IA,
- si nota baja actividad → propone temas nuevos,
- si hay picos de interés → crea resúmenes semanales.

“Esta semana la comunidad ha hablado mucho sobre conciertos acústicos.
Se creará un hilo recopilatorio.”

18.15 IA de temas emergentes

Analiza patrones y detecta tendencias musicales:

“Aumento del 240 % en menciones a ‘loopstations’.
¿Quieres leer el hilo destacado?”

Firestore [/analytics/community_trends](#):

```
{  
  "keyword": "loopstation",  
  "mentions": 182,  
  "trend": +2.4  
}
```

18.16 Integración con el feed (Bloque 15)

Los hilos populares se muestran en el feed principal con formato “mini-foro”:

“💬 Debate: ¿Vale la pena usar in-ears?”
+120 respuestas esta semana.

También permite que los posts del foro se enlacen con perfiles de músicos.

18.17 Moderación humana + IA

Aunque la IA modera en tiempo real, siempre hay revisión humana opcional.
Cada semana genera reporte:

```
{  
  "flaggedPosts": 4,  
  "autoResolved": 3,  
  "needsReview": 1  
}
```

Panel [/admin/community/moderation](#).

18.18 Analítica

Firestore /analytics/community:

```
{  
  "postsTotal": 2431,  
  "commentsTotal": 12892,  
  "avgEngagement": 0.62,  
  "aiModerations": 1438,  
  "mentorshipsActive": 211  
}
```

Dashboard /admin/community:

- Actividad por categoría.
- Ratio de moderación IA.
- Niveles de satisfacción.
- Top usuarios activos.

18.19 Prompt técnico para Anti-Gravity

You are building the AI Community and Forum System for Musikeeo.

Goals:

- Enable forum-like interaction between users with moderation AI.
- Generate thread summaries and mentor matches.
- Suggest empathetic replies and connect similar users.
- Manage gamification and community scoring.
- Detect trends, tone, and conflicts automatically.

Output:

1. /functions/ai/aiForumModerator.ts
2. /functions/ai/aiThreadSummarizer.ts
3. /functions/ai/aiCommunityConnector.ts
4. /functions/ai/aiMentorMatcher.ts

5. /components/ForumCard.tsx
 6. /components/CommentBox.tsx
 7. /components/AIReplySuggestions.tsx
 8. /components/MentorBadge.tsx
 9. /pages/community/*
 10. /analytics/community
-

18.20 Entregables esperados

- functions/ai/aiForumModerator.ts
 - functions/ai/aiThreadSummarizer.ts
 - functions/ai/aiCommunityConnector.ts
 - functions/ai/aiMentorMatcher.ts
 - components/ForumCard.tsx, CommentBox.tsx, AIReplySuggestions.tsx, MentorBadge.tsx
 - pages/community/index.tsx, thread/[id].tsx, create.tsx, mentors.tsx
 - analytics/community, community_trends
 - Documento Musikeeo_AI_Community.pdf
-

Fin del BLOQUE 18 — IA de Comunidad, Foros y Red Social Musical (versión extendida).

¿Continuamos con el **BLOQUE 19 — IA DE EXPERIENCIA DE USUARIO Y PERSONALIZACIÓN GLOBAL (UI adaptable, navegación inteligente, lenguaje y comportamiento predictivo)?**

sigue

🔥 Entramos en un bloque **más técnico, invisible pero crítico** para la experiencia completa del usuario:

el BLOQUE 19 — IA DE EXPERIENCIA DE USUARIO Y PERSONALIZACIÓN GLOBAL (versión extendida).

Este módulo es el que da “vida” a Musikeeo como sistema: adapta la app a cada usuario según su rol, intereses, idioma, uso y estado emocional.

Aquí la IA actúa como un **cerebro de personalización**, ajustando la navegación, los textos, los colores y las sugerencias en tiempo real para hacer que cada experiencia sea única, fluida y coherente.



BLOQUE 19 — IA DE EXPERIENCIA DE USUARIO Y PERSONALIZACIÓN GLOBAL

(*UI adaptable, navegación inteligente, lenguaje y comportamiento predictivo*)

19.1 Propósito del módulo

Construir un sistema de IA transversal que **personalice Musikeeo para cada usuario** según:

- su rol (músico, técnico, promotor, tienda, alumno),
- su comportamiento en la app,
- sus emociones o contexto actual,
- su idioma,
- su nivel de experiencia digital.



“Una app que se adapta a ti como si te conociera de toda la vida.”

19.2 Filosofía de diseño

La personalización no debe ser invasiva ni artificial.

Debe sentirse como **una intuición natural** de la app.

Nada de pantallas frías o repetitivas: cada usuario tiene su propio ritmo, flujo y estética visual.

La IA debe:

- anticipar lo que el usuario busca,
 - simplificar pasos,
 - y mantener una narrativa coherente entre módulos.
-

19.3 Arquitectura general

Frontend → Context API / Firestore → AI Personalization Engine → Gemini Core

Carpetas:

```
/functions/  
|   |-- ai/  
|   |   |-- aiPersonalizationEngine.ts  
|   |   |-- aiBehaviorPredictor.ts  
|   |   |-- aiLanguageToneAdapter.ts  
|   |   |-- aiVisualThemeSelector.ts  
|   |   |-- aiNavigationOptimizer.ts  
|   |-- triggers/  
|       |-- onUserBehaviorUpdate.ts  
/src/  
|   |-- context/UserContext.tsx  
|   |-- hooks/usePersonalization.ts  
|   |-- components/AdaptiveHeader.tsx  
|   |-- components/AIWELCOMEBanner.tsx  
|   |-- pages/settings/personalization.tsx
```

19.4 Datos base y fuentes IA

El sistema se alimenta de múltiples fuentes de comportamiento:

Fuente	Ejemplo de dato	Uso IA
 Navegación	Qué módulos visita más	Reordenar menú

 Actividad musical	Eventos, mensajes, clases	Ajustar interfaz
 Interacciones	Likes, mensajes, tiempo de sesión	Adaptar tono
 Localización	Ciudad o país	Idioma, divisa, formato horario
 Estado emocional (opcional)	Texto, velocidad de escritura	Tono empático

Firestore `/personalization/{uid}:`

```
{
  "preferredLanguage": "es",
  "theme": "dark",
  "role": "musician",
  "tone": "casual",
  "aiConfidence": 0.89
}
```

19.5 IA Personalization Engine

El núcleo del sistema: analiza el perfil y decide qué versión de UI mostrar.

Ejemplo:

- Si el usuario es técnico → mostrar primero Marketplace y Eventos.
- Si es músico → destacar Feed y Comunidad.
- Si es promotor → priorizar Pagos y Contrataciones.
- Si es alumno → resaltar Academia.

Prompt base:

Generate personalized UI layout for Musikeeo user.

Input: {role, preferences, history, location, deviceType}

Output: JSON with homeSectionsOrder, tone, theme, and AI welcome message.

Salida:

```
{  
  "homeSectionsOrder": ["feed", "events", "community"],  
  "theme": "dark",  
  "tone": "friendly",  
  "welcomeMessage": "¡Hola Víctor! Tienes 2 eventos confirmados y un  
nuevo mensaje."  
}
```

19.6 IA Behavior Predictor

Predice qué acciones va a realizar el usuario según patrones previos:

“Cada viernes revisas tus bolos. ¿Quieres abrir tu calendario?”
“Sueles usar el modo urgente a última hora del sábado.”

Campos IA:

```
{  
  "predictedAction": "open_calendar",  
  "confidence": 0.78,  
  "triggerTime": "Friday 18:00"  
}
```

Estos datos se usan para notificaciones o accesos rápidos.

19.7 IA Language Tone Adapter

Adapta el lenguaje de toda la app al estilo del usuario.

Por ejemplo:

Estilo	Tono IA	Ejemplo
 Formal	Profesional y claro	“Evento confirmado. Recibirás el contrato en breve.”
 Casual	Cercano y relajado	“Todo listo, crack 🎉 ¡Nos vemos en el bolo!”
 Corporativo	Neutral	“Tu pago ha sido procesado correctamente.”

Prompt:

```
Rephrase all UI labels and messages to fit tone='casual' and language='es'.
```

19.8 IA Visual Theme Selector

Adapta el aspecto visual según el contexto del usuario:

- Día/noche → cambia fondo y brillo.
- Género musical → matices de color.
- Modo evento activo → layout de productividad.
- Modo relax → interfaz más minimal y cálida.

Ejemplo:

Usuario rockero → interfaz oscura con acentos rojos.
Usuario pop latino → tonos cálidos y animaciones suaves.

Firestore `/themes/{uid}:`

```
{  
  "primary": "#ff8855",  
  "secondary": "#222222",  
  "font": "Inter",  
  "animation": "slide-up"  
}
```

19.9 IA Navigation Optimizer

Detecta pasos repetitivos o fricción en la navegación:

“Has tardado 5 clics en crear un evento.
¿Quieres activar el modo rápido?”

La IA propone atajos personalizados o botones contextuales.
También puede simplificar formularios detectando campos redundantes.

19.10 Modo onboarding IA

Durante el registro, la IA guía con preguntas naturales:

- “¿Eres músico, técnico o promotor?”
- “¿Qué te gustaría hacer primero?”
- “¿Te gusta un diseño oscuro o claro?”

Y crea un perfil IA personalizado desde el minuto uno.

19.11 IA Emotional Adaptation

Analiza texto, tono de escritura o velocidad para detectar estados como estrés, entusiasmo o duda.

- Si nota frustración → mensajes más amables.
- Si percibe energía → sugerencias ambiciosas.

Ejemplo:

“No pasa nada si hoy no completas el perfil.
Puedes hacerlo más tarde 🎵.”

O:

“Estás a tope esta semana 🙌 te propongo nuevos eventos.”

19.12 IA Cross-Context Memory

La IA recuerda lo que haces en distintos módulos para mantener coherencia:

- Si un usuario enseña guitarra → aparece su curso primero en Academia.
- Si alquila equipo → IA recomienda accesorios compatibles.
- Si participa en foros → resalta su perfil en el feed.

Todo sincronizado con `userEmbeddingGlobal`.

19.13 IA Multi-Idioma y Cultura

Soporta contenido en español, inglés, catalán, portugués e italiano.
Traduce automáticamente con contexto musical (manteniendo jerga técnica).
Ejemplo:

“Gain” → mantiene el término original.
“Bolo” → “gig” en inglés, “show” en portugués.

19.14 Analítica IA

Firestore `/analytics/personalization`:

```
{  
  "activeUsers": 10483,  
  "avgSessionLength": 184,  
  "themeSwitches": 412,  
  "toneAdaptations": 2881,  
  "aiPredictionsTriggered": 720  
}
```

Dashboard `/admin/personalization`:

- Distribución de roles.
 - Idiomas activos.
 - Satisfacción IA.
 - Errores de predicción.
-

19.15 Seguridad y privacidad

- Todos los datos personalizados se almacenan localmente (encriptados).
- No se usan para publicidad.
- El usuario puede reiniciar su perfil IA o cambiar de tono/idioma.
- Cumple RGPD + Consentimiento explícito.

19.16 Modo “Flow IA”

Una experiencia totalmente reactiva: la app **responde al contexto en tiempo real**.

Ejemplo:

- Si detecta movimiento (GPS) → activa modo “camino al bolo”.
 - Si detecta ubicación fija → muestra modo “ensayo” o “oficina”.
 - Si se abre tras medianoche → “modo nocturno chill”.
-

19.17 Prompt técnico para Anti-Gravity

You are building the AI Personalization and User Experience System for Musikeeo.

Goals:

- Adapt UI, tone and navigation for each user dynamically.
- Predict actions and create shortcuts.
- Translate and rephrase content based on tone.
- Choose visual themes contextually.
- Integrate behavior tracking and analytics.
- Ensure privacy and full user control.

Output:

1. /functions/ai/aiPersonalizationEngine.ts
 2. /functions/ai/aiBehaviorPredictor.ts
 3. /functions/ai/aiLanguageToneAdapter.ts
 4. /functions/ai/aiVisualThemeSelector.ts
 5. /functions/ai/aiNavigationOptimizer.ts
 6. /context/UserContext.tsx
 7. /hooks/usePersonalization.ts
 8. /pages/settings/personalization.tsx
 9. /components/AdaptiveHeader.tsx
 10. /components/AIWelcomeBanner.tsx
 11. /analytics/personalization
-

19.18 Entregables esperados

- `functions/ai/aiPersonalizationEngine.ts`
 - `functions/ai/aiBehaviorPredictor.ts`
 - `functions/ai/aiLanguageToneAdapter.ts`
 - `functions/ai/aiVisualThemeSelector.ts`
 - `functions/ai/aiNavigationOptimizer.ts`
 - `context/UserContext.tsx, hooks/usePersonalization.ts`
 - `pages/settings/personalization.tsx`
 - `components/AdaptiveHeader.tsx, AIWelcomeBanner.tsx`
 - `analytics/personalization`
 - Documento Musikeeo_AI_Personalization.pdf
-

Fin del BLOQUE 19 — IA de Experiencia de Usuario y Personalización Global (versión extendida).

¿Continuamos con el **BLOQUE 20 — IA DE GESTIÓN ADMINISTRATIVA, SOPORTE Y PANEL DE CONTROL (Backoffice, analítica global y mantenimiento automatizado)?**

sigue

🔥 Perfecto. Entramos ahora en un bloque **más operativo y estructural**, pero absolutamente crucial para que Musikeeo funcione de manera estable y profesional a medida que crece:

el BLOQUE 20 — IA DE GESTIÓN ADMINISTRATIVA, SOPORTE Y PANEL DE CONTROL (versión extendida).

Aquí definimos toda la capa **backoffice**, donde los administradores (y la propia IA) monitorizan usuarios, transacciones, contenido, incidencias y rendimiento global de la plataforma.

Es el cerebro de operaciones y mantenimiento del ecosistema Musikeeo.



BLOQUE 20 — IA DE GESTIÓN ADMINISTRATIVA, SOPORTE Y PANEL DE CONTROL

(*Backoffice inteligente, analítica global y mantenimiento automatizado*)

20.1 Propósito del módulo

El objetivo es que **Musikeeo se administre sola** en gran parte: una IA supervisora que detecte errores, genere reportes y asista a los administradores humanos.

⌚ “Un equipo de soporte virtual que nunca duerme.”

Funcionalidades principales:

1. Panel de control con estadísticas en tiempo real.
 2. Detección de errores, fraude o mal uso.
 3. IA que genera informes semanales y alertas.
 4. Sistema de soporte con tickets automáticos.
 5. Mantenimiento de datos y logs.
-

20.2 Filosofía de diseño

El backoffice debe ser **ligero, modular y proactivo**.

Nada de paneles llenos de botones: la IA interpreta los datos y propone acciones claras. El equipo humano se dedica a decidir, no a buscar.

20.3 Arquitectura general

Frontend → Firestore + BigQuery → Cloud Functions (AI Reports & Alerts)

Carpetas:

```
/functions/  
|   └── ai/  
|       ├── aiSystemMonitor.ts  
|       ├── aiAdminReporter.ts  
|       ├── aiSupportAssistant.ts  
|       └── aiDataCleaner.ts  
|   └── triggers/  
|       ├── onErrorDetected.ts  
|       └── onTicketCreated.ts  
/src/  
|   └── pages/admin/  
|       ├── index.tsx  
|       ├── users.tsx  
|       ├── finance.tsx  
|       ├── content.tsx  
|       └── support.tsx  
|   └── components/  
|       ├── AdminDashboard.tsx  
|       ├── ReportCard.tsx  
|       ├── SupportTicket.tsx  
|       └── SystemAlert.tsx
```

20.4 Roles administrativos

Rol	Descripción
👑 Super Admin	Acceso total, configuración general, usuarios.
💼 Manager	Gestiona pagos, contratos y soporte.
👤 Moderador	Controla comunidad y contenido.
🤖 IA Supervisor	Monitorea el sistema, genera reportes y sugerencias.

20.5 Firestore estructura base

/admin_logs/{logId}

```
{  
  "type": "system",  
  "message": "New user registered in Barcelona",  
  "severity": "info",  
  "timestamp": "2026-01-10T18:03:00"  
}
```

/tickets/{ticketId}

```
{  
  "userId": "uid_musician",  
  "category": "payment_issue",  
  "description": "No recibí el pago del evento X",  
  "status": "open",  
  "aiAssigned": "true"  
}
```

20.6 IA System Monitor

La IA supervisa en segundo plano:

- errores de base de datos,
- caídas de funciones,
- duplicados o inconsistencias,
- latencia anómala,
- uso excesivo de recursos.

Prompt:

Monitor system logs and identify critical issues or anomalies.
Output JSON {severity, issue, recommendedAction}.

Ejemplo:

```
{  
  "severity": "high",  
  "issue": "Repeated write failure in transactions collection",  
  "recommendedAction": "Increase disk space or optimize database."}
```

```
        "recommendedAction": "Restart payments function"
    }
```

20.7 IA Admin Reporter

Genera informes automáticos diarios o semanales.

Informe semanal tipo:

Resumen semanal Musikeeo (IA Report 2026-03-14)

- Nuevos usuarios: +432
- Transacciones completadas: 1.203
- Mensajes enviados: 9.830
- Incidencias detectadas: 4
- Rendimiento IA: 98 % uptime
- Recomendación: revisar latencia en Marketplace (300ms ↑)

Estos informes se envían por email o Telegram al equipo interno.

20.8 IA Support Assistant

Gestiona tickets automáticamente.

Ejemplo:

Usuario: “No me ha llegado el contrato.”

IA responde:

“Tu contrato fue generado el 8 de enero. Aquí puedes descargarlo: [enlace].

¿El problema persiste?”

Si la IA no resuelve → escalo a humano.

Cada interacción queda registrada.

Firestore `/tickets_ai/{id}`:

```
{
```

```
        "ticketId": "t_093",
        "resolvedBy": "AI",
        "confidence": 0.92,
        "feedback": "positive"
    }
```

20.9 IA Data Cleaner

Ejecuta tareas automáticas de mantenimiento:

- eliminar logs viejos,
- archivar contratos inactivos,
- limpiar archivos duplicados,
- optimizar índices Firestore.

Ejemplo de tarea programada:

```
schedule("every monday 3:00 AM", async () => {
    await deleteOldLogs(90);
    await archiveInactiveEvents(180);
});
```

20.10 Panel administrativo general

Ruta `/admin/index.tsx`:

 Musikeeo Backoffice

 Estadísticas globales

 Usuarios activos: 8.302

 Eventos activos: 187

 Transacciones: 14.231

 Tickets abiertos: 3



"Rendimiento estable. Revisión de API Stripe sugerida."

20.11 Submódulo de usuarios

Ruta `/admin/users.tsx`

Funciones:

- buscar, filtrar y editar usuarios,
- suspender cuentas fraudulentas,
- revisar TrustScore y métricas IA.

La IA sugiere acciones:

"El usuario 'Carlos S.' ha recibido 3 reportes esta semana.
¿Deseas revisarlo?"

20.12 Submódulo financiero

Ruta `/admin/finance.tsx`

Resumen:

- Total transacciones mensuales.
- Comisiones acumuladas.
- Errores Stripe / PayPal.
- Historial de facturas.

IA detecta irregularidades:

"Una factura supera el promedio por 300 %. Revisa evento ID 982."

20.13 Submódulo de contenido

Ruta `/admin/content.tsx`

Permite:

- ver posts, cursos o perfiles destacados.
- marcar contenido IA “featured”.
- eliminar spam o duplicados.

Gemini genera lista de curación semanal:

“Contenido más valioso de la semana:

- Curso de Canto Moderno
- Foro: Técnicas de mezcla en vivo”

20.14 Submódulo de soporte

Ruta `/admin/support.tsx`

Lista de tickets + IA Chat Integrado:

“¿Qué problema quieres resolver?”
“Usuario con disputa económica”
→ IA busca historial y genera respuesta pre-redactada.

También permite feedback IA:

- “resuelto por humano”
- “resuelto por IA con éxito”

20.15 IA de alertas en tiempo real

Cuando ocurre un evento crítico:

- caídas de API,

- errores de autenticación,
- fallos en pagos,
- IA mal calibrada,

se genera alerta instantánea tipo:

“⚠️ Sistema detecta fallo en función onEventCreated.ts — respuesta 503.”

Enviada por Telegram / email.

20.16 Métricas globales

Firestore `/analytics/admin`

```
{  
  "uptime": 0.987,  
  "avgResponseTime": 230,  
  "aiResolutions": 76,  
  "ticketsOpen": 5,  
  "errorsLast24h": 1  
}
```

20.17 IA de predicción de carga

Gemini predice uso futuro y recursos necesarios:

“Se espera un aumento del 32 % en eventos durante junio.
Sugiero escalar servidores Firestore y Storage.”

Prompt:

```
Predict system load for the next 30 days based on historical data.  
Output JSON {expectedEvents, trafficIncrease,  
scalingRecommendation}.
```

20.18 Seguridad y cumplimiento

- Auditoría completa (quién modificó qué y cuándo).
 - Cifrado TLS 1.3.
 - Autenticación de doble factor para admin.
 - Cumple ISO 27001 y RGPD.
 - Logs firmados digitalmente.
-

20.19 Prompt técnico para Anti-Gravity

You are building the AI Administrative, Support, and Monitoring System for Musikeeo.

Goals:

- Create a backoffice for monitoring, reports, and support.
- Use AI to detect errors, generate weekly reports, and resolve tickets.
- Automate data cleanup and scaling predictions.
- Provide dashboards for users, finance, and content.
- Ensure full security, logging, and GDPR compliance.

Output:

1. /functions/ai/aiSystemMonitor.ts
 2. /functions/ai/aiAdminReporter.ts
 3. /functions/ai/aiSupportAssistant.ts
 4. /functions/ai/aiDataCleaner.ts
 5. /pages/admin/*
 6. /components/AdminDashboard.tsx
 7. /components/ReportCard.tsx
 8. /components/SupportTicket.tsx
 9. /components/SystemAlert.tsx
 10. /analytics/admin
-

20.20 Entregables esperados

- functions/ai/aiSystemMonitor.ts

- `functions/ai/aiAdminReporter.ts`
 - `functions/ai/aiSupportAssistant.ts`
 - `functions/ai/aiDataCleaner.ts`
 - `pages/admin/*` (index, users, finance, content, support)
 - `components/AdminDashboard.tsx, ReportCard.tsx, SupportTicket.tsx, SystemAlert.tsx`
 - `analytics/admin`
 - Documento `Musikeeo_AI_Admin.pdf`
-

Fin del BLOQUE 20 — IA de Gestión Administrativa, Soporte y Panel de Control (versión extendida).

¿Continuamos con el **BLOQUE 21 — IA DE SEGURIDAD, AUTENTICACIÓN Y CONFIANZA (verificación, reputación, protección antifraude y privacidad avanzada)**?

sigue

🔥 Vamos con un bloque crítico a nivel de infraestructura y credibilidad:
el BLOQUE 21 — IA DE SEGURIDAD, AUTENTICACIÓN Y CONFIANZA (versión extendida).

Aquí se define toda la base que protege Musikeeo y genera confianza entre usuarios, tiendas, promotores y artistas.

Incluye autenticación avanzada, reputación IA, detección antifraude, validación documental y mecanismos de privacidad total.

Es el **sistema inmunológico** de la app.

BLOQUE 21 — IA DE SEGURIDAD, AUTENTICACIÓN Y CONFIANZA

(verificación, reputación, protección antifraude y privacidad avanzada)

21.1 Propósito del módulo

Garantizar que Musikeeo sea un entorno **seguro, fiable y verificable**, donde:

- cada usuario sea quien dice ser,
- las transacciones y eventos estén protegidos,
- la IA detecte comportamientos anómalos,
- y los datos personales estén blindados.

⌚ “Confianza para crear, tocar y contratar sin miedo.”

21.2 Filosofía de diseño

La seguridad no debe sentirse como una barrera, sino como **una garantía invisible**.

El usuario debe notar fluidez, no fricción.

La IA actúa de forma preventiva, educada y empática: detecta riesgos sin intimidar.

21.3 Arquitectura general

Frontend → Firebase Auth + Firestore (users_secure)
→ Cloud Functions (AI Fraud + Verification)
→ Encryption Layer (AES-256)

Carpetas:

```
/functions/  
|   └── ai/  
|       ├── aiFraudDetector.ts  
|       ├── aiIdentityVerifier.ts  
|       ├── aiReputationScorer.ts  
|       └── aiPrivacyGuardian.ts  
|   └── triggers/  
|       └── onUserRegistered.ts  
/src/
```

```
|   └── pages/security/
|       ├── verify.tsx
|       ├── privacy.tsx
|       ├── trust.tsx
|       └── sessions.tsx
|
|   └── components/
|       ├── TrustBadge.tsx
|       ├── VerificationStatus.tsx
|       ├── SecureLogin.tsx
|       └── AlertBox.tsx
```

21.4 Sistema de autenticación

Métodos principales

1. Firebase Auth (Google, Apple, Email).
 2. Login con biometría (FacelID / huella).
 3. 2FA (verificación por SMS o app).
 4. Tokens JWT firmados y validados IA.
-

21.5 Firestore estructura

```
/users_secure/{uid}
{
  "email": "user@mail.com",
  "phone": "+34...",
  "verified": true,
  "trustScore": 87,
  "lastLogin": "2026-01-10",
  "sessions": [ "2026-01-10T14:03", "2026-01-11T09:12" ]
}
```

21.6 IA Identity Verifier

Valida identidad mediante:

- verificación facial (selfie + DNI),
- análisis de documentos,
- comparación con registros previos.

Prompt base:

```
Verify user identity using provided ID image and selfie.  
Output JSON {match: boolean, confidence, reason}.
```

Resultado:

```
{  
  "match": true,  
  "confidence": 0.94,  
  "reason": "Face and ID name match successfully."  
}
```

Si falla → IA sugiere revisión manual.

21.7 IA Fraud Detector

Analiza comportamiento anómalo:

- múltiples cuentas con mismo IBAN,
- IPs sospechosas,
- cambios bruscos de ubicación,
- pagos duplicados o cancelaciones extrañas.

Prompt:

```
Analyze user and transaction logs for potential fraud or anomalies.  
Output JSON {riskScore, flags, recommendation}.
```

Ejemplo:

```
{  
  "riskScore": 0.83,  
  "flags": [ "IP mismatch", "double withdrawal attempt" ],  
  "recommendation": "Temporarily freeze payouts"  
}
```

21.8 IA Reputation Scorer

Cada usuario tiene un **TrustScore (0–100)**, calculado según:

- historial de eventos,
- pagos completados,
- reseñas recibidas,
- cumplimiento de contratos,
- reportes negativos.

TrustScore dinámico:

+3 por evento cumplido
+1 por buena valoración
-10 por disputa
-20 por fraude detectado

Firestore `/trust_scores/{uid}`:

```
{  
  "trustScore": 92,  
  "level": "Gold",  
  "badges": [ "Verified", "Reliable" ]  
}
```

Visual en perfil:

“Usuario Verificado  · Nivel Oro · 24 contratos sin incidencias.”

21.9 IA Privacy Guardian

Controla el uso de datos personales:

- anonimiza nombres en analíticas,
- bloquea peticiones sospechosas,
- audita accesos,
- genera avisos RGPD automáticos.

Ejemplo:

“Tu contrato con Marta expira el 12 de marzo.
Los datos serán eliminados automáticamente 30 días después.”

Firestore `/privacy_logs/{id}`:

```
{  
  "action": "auto_delete",  
  "target": "contract_421",  
  "userId": "uid_musician",  
  "timestamp": "2026-03-13"  
}
```

21.10 IA Session Sentinel

Detecta sesiones activas simultáneas en lugares lejanos:

“Detectamos una conexión desde Argentina.
¿Eres tú?”

Si el usuario confirma → whitelist IP.
Si no → cierre automático + notificación push.

21.11 UX/UI del módulo

Pantalla `/security/trust.tsx`:

 Seguridad y Confianza

Nivel actual: Oro (92)

 Identidad verificada

 Pagos protegidos

 24 eventos completados

Ver historial → []

Configurar privacidad → []

21.12 IA de Contratos Seguros

La IA revisa automáticamente los contratos antes de firmarse:

- coherencia de datos,
- detección de cláusulas abusivas,
- validación de fechas.

Prompt:

Review contract JSON for inconsistencies, missing fields or potential fraud.

Output structured feedback.

“El campo ‘importe’ no coincide con el resumen del evento.”

21.13 IA de Prevención de Abuso

Supervisa mensajes privados o publicaciones para evitar:

- acoso,
- lenguaje inapropiado,

- spam o grooming.

El tono siempre es preventivo y educado:

“Recuerda mantener un lenguaje respetuoso en tus conversaciones 🎵.”

21.14 Sistema de verificación visible

Los perfiles muestran insignias:

- Verificado (DNI validado)
- Profesional (contratos completados)
- Gold (TrustScore > 90)
- En revisión

Componente: `TrustBadge.tsx`

Tooltip:

“Este usuario ha sido verificado mediante IA y documentación oficial.”

21.15 IA de Predicción de Riesgo

Analiza contexto antes de transacciones:

“El promotor tiene bajo TrustScore y 2 disputas abiertas.
¿Seguro que deseas continuar?”

Prompt:

`Predict risk of upcoming transaction based on user and context data.
Output JSON {riskLevel, explanation}.`

21.16 IA de Gestión de Incidencias

Cuando ocurre un problema (pago, cancelación, disputa), la IA crea automáticamente un ticket y una resolución sugerida:

"Disputa detectada entre Usuario A y B.
Motivo: cancelación tardía.
Acción sugerida: retención parcial del pago (50%)."

21.17 Auditoría IA

Cada acción sensible queda registrada:

- inicio de sesión,
- modificación de contrato,
- acceso a datos,
- eliminación de cuenta.

Firestore `/audit_logs/{logId}`

```
{  
  "user": "uid_admin",  
  "action": "delete_user",  
  "target": "uid_124",  
  "timestamp": "2026-01-10T12:05",  
  "verifiedByAI": true  
}
```

21.18 Cumplimiento RGPD y privacidad

- Consentimiento explícito en el onboarding.
 - Opción “eliminar todo” (borrado completo y verificado IA).
 - Datos cifrados con AES-256 y transporte TLS 1.3.
 - Política transparente en `/security/privacy.tsx`.
 - Logs accesibles al usuario (“mis datos”).
-

21.19 IA Legal Advisor

Asistente jurídico básico para usuarios:

“¿Cómo facturo una actuación?”
“¿Puedo cancelar un contrato sin penalización?”

Prompt:

Answer legal and contractual questions for musicians in Spain using public law references.

Output short explanation + legal reference.

21.20 Prompt técnico para Anti-Gravity

You are building the AI Security, Authentication, and Trust System for Musikeeo.

Goals:

- Verify user identity and manage trust levels.
- Detect fraud, anomalies and abusive behavior.
- Manage privacy, sessions, and GDPR compliance.
- Provide a secure and friendly authentication flow.
- Support legal and contractual validations.

Output:

1. /functions/ai/aiFraudDetector.ts
 2. /functions/ai/aiIdentityVerifier.ts
 3. /functions/ai/aiReputationScorer.ts
 4. /functions/ai/aiPrivacyGuardian.ts
 5. /pages/security/*
 6. /components/TrustBadge.tsx
 7. /components/VerificationStatus.tsx
 8. /components/SecureLogin.tsx
 9. /components/AlertBox.tsx
 10. /analytics/security
-

21.21 Entregables esperados

- `functions/ai/aiFraudDetector.ts`
 - `functions/ai/aiIdentityVerifier.ts`
 - `functions/ai/aiReputationScorer.ts`
 - `functions/ai/aiPrivacyGuardian.ts`
 - `pages/security/*` (verify, trust, privacy, sessions)
 - `components/TrustBadge.tsx, VerificationStatus.tsx, SecureLogin.tsx, AlertBox.tsx`
 - `analytics/security`
 - Documento `Musikeeo_AI_Security.pdf`
-

Fin del BLOQUE 21 — IA de Seguridad, Autenticación y Confianza (versión extendida).

¿Continuamos con el **BLOQUE 22 — IA DE ESCALABILIDAD, RENDIMIENTO Y OPTIMIZACIÓN TÉCNICA (caché, paralelización, monitorización y autoescalado)?**

sigue

Perfecto 🔥 — entramos en un bloque **muy técnico pero vital para la sostenibilidad a largo plazo de Musikeeo**, especialmente si quieres que la app funcione sin cuellos de botella cuando crezca:

el BLOQUE 22 — IA DE ESCALABILIDAD, RENDIMIENTO Y OPTIMIZACIÓN TÉCNICA (versión extendida).

Aquí se define cómo la plataforma aprende a optimizarse sola: detecta sobrecargas, cachea contenido, distribuye procesos, optimiza consultas y reduce costes automáticamente.

Básicamente, es lo que convierte a Musikeeo en una **infraestructura autoajustable**.

BLOQUE 22 — IA DE ESCALABILIDAD, RENDIMIENTO Y OPTIMIZACIÓN TÉCNICA

(autoescalado, caché dinámica, optimización de consultas y mantenimiento predictivo)

22.1 Propósito del módulo

Garantizar que **Musikeeo** pueda crecer sin romperse ni gastar de más.

Este sistema gestiona de forma automática los recursos técnicos y las cargas de usuarios simultáneos.

 “Más usuarios, menos latencia. Sin tocar un solo servidor.”

22.2 Filosofía de diseño

El objetivo es que el sistema sea **inteligente, eficiente y autorregulado**.

Nada de escalar manualmente o pagar de más por potencia ociosa.

La IA debe:

- anticipar picos,
 - redistribuir cargas,
 - limpiar procesos inactivos,
 - y sugerir optimizaciones en tiempo real.
-

22.3 Arquitectura general

Frontend → Firestore + Cloud Functions + Cloud Run



AI Performance Monitor → AutoScaler + Cache Layer



BigQuery + Analytics

Carpetas:

```
/functions/  
|   └── ai/  
|       ├── aiPerformanceMonitor.ts  
|       ├── aiAutoScaler.ts  
|       ├── aiCacheOptimizer.ts  
|       ├── aiQueryOptimizer.ts  
|       └── aiLoadPredictor.ts  
|   └── triggers/  
|       └── onUsageUpdate.ts  
/src/  
|   ├── pages/admin/performance.tsx  
|   └── components/  
|       ├── PerformanceDashboard.tsx  
|       ├── SystemHealthCard.tsx  
|       └── OptimizationLog.tsx
```

22.4 Principios técnicos

1. **Autoescalado**: servidores dinámicos según demanda.
 2. **Caché inteligente**: IA detecta qué datos se repiten.
 3. **Optimización de consultas**: Firestore bajo demanda.
 4. **Balanceo IA**: rutas HTTP con latencia mínima.
 5. **Mantenimiento predictivo**: analiza logs para prevenir fallos.
-

22.5 IA Performance Monitor

Supervisa latencia, tráfico y tiempos de respuesta.

Ejemplo de log:

```
{  
  "function": "createEvent",  
  "avgResponse": 283,
```

```
"requestsPerMin": 98,  
"errors": 2,  
"aiPerformanceScore": 0.91  
}
```

Si detecta un aumento de latencia:

“Función createEvent superó 500ms promedio. Recomiendo escalar a 2 instancias.”

22.6 IA AutoScaler

Escala automáticamente Cloud Functions y Cloud Run según tráfico.

Prompt:

```
Analyze usage metrics and determine optimal instance scaling.  
Output JSON {minInstances, maxInstances, scaleTrigger,  
costEstimation}.
```

Salida:

```
{  
  "minInstances": 1,  
  "maxInstances": 4,  
  "scaleTrigger": 150,  
  "costEstimation": "€12.40/day"  
}
```

También desescalía en horas de baja actividad.

22.7 IA Cache Optimizer

Identifica los datos que más se solicitan y los guarda en memoria temporal (Redis o Firestore local cache).

Ejemplo:

- Perfiles consultados muchas veces
- Listado de eventos populares
- Configuración del usuario

Prompt:

Detect frequently requested documents and store them in cache.
Recommend expiration times and update intervals.

Salida:

```
{  
  "cachedDocs": [ "users", "events" ],  
  "ttl": "3600s",  
  "hitRate": 0.84  
}
```

22.8 IA Query Optimizer

Analiza consultas Firestore lentas o redundantes.

“Consulta /events ordenada por fecha sin índice detectada.
Sugiero crear índice compuesto [date, location].”

También reestructura consultas:

```
db.collection('events').where('city', '==', 'Barcelona')  
                      .orderBy('date')  
                      .limit(50)
```

→ IA reescribe para reducir costos y lecturas.

22.9 IA Load Predictor

Predice picos de tráfico según:

- día y hora,

- actividad de eventos,
- lanzamientos,
- campañas publicitarias.

Ejemplo:

“Se espera pico el sábado 20h (+45 % usuarios).
Escalar instancias +2 y aumentar caché.”

Prompt:

```
Predict user load based on historical data and event schedule.  
Output {expectedLoad, recommendedScaling}.
```

22.10 IA Cost Manager

Analiza el gasto de cada servicio y sugiere optimizaciones.

“Firestore representa el 67 % del costo mensual.
Recomendación: reducir lecturas duplicadas con cache IA.”

Firestore `/analytics/costs`:

```
{  
  "Firestore": 67,  
  "Functions": 19,  
  "Storage": 9,  
  "Messaging": 5  
}
```

22.11 IA Error Healer

Cuando detecta fallos recurrentes (por ejemplo, en funciones o APIs), los agrupa, analiza y genera una acción correctiva:

“Error ‘QuotaExceeded’ detectado 43 veces.
Acción: dividir carga de escritura o limpiar batch.”

Prompt:

Detect repeated system errors and propose automated fixes.
Output JSON {issue, frequency, solution, estimatedImpact}.

22.12 IA Data Compression

Comprime datos antiguos o poco usados:

- imágenes en miniatura,
- logs de más de 90 días,
- historial de eventos pasados.

Ejemplo:

“Se comprimieron 540MB en Storage.
Reducción de costes: 21 % mensual.”

22.13 IA Traffic Router

Redirige solicitudes a la región más cercana al usuario (multi-región).

Ejemplo: usuarios en México → `us-central1`; España → `europe-west`.
Esto reduce latencia y mejora SEO en la web app.

22.14 Monitoreo visual (UX admin)

Pantalla `/admin/performance.tsx`:

Rendimiento Global

-  Uptime: 99.2%
 -  Tráfico activo: 3.284 usuarios
 -  Latencia media: 241 ms
 -  Caché: 82% hit rate
-

AI Status:

-  Última optimización 3h atrás

 Recomendación: reducir instancias nocturnas

22.15 IA Energy Saver

Apaga instancias o procesos cuando detecta baja actividad.
También ajusta intervalos de sincronización de datos.

“Sistema inactivo 01:00–06:00 → suspender procesos secundarios.”

22.16 IA API Optimizer

Mide rendimiento de endpoints:

- `/createEvent`
- `/searchMusicians`
- `/uploadTrack`

y genera informe:

```
{  
  "endpoint": "/searchMusicians",  
  "avgResponse": 402,  
  "suggestion": "Add pagination & cache results"  
}
```

22.17 IA CDN & Media Optimizer

Optimiza la entrega de medios (imágenes, vídeos, audio):

- conversión automática WebP/AVIF,
- prefetch inteligente,
- lazy loading dinámico.

Ejemplo:

“Optimized 1.245 imágenes · ahorro: 32 % tamaño total.”

22.18 IA Log Analyzer

Agrupa logs similares para evitar ruido:

“Errores 404 repetidos en rutas /event/old — ignorados tras 24h.”

“Advertencia: endpoint obsoleto detectado.”

Genera reporte semanal resumido.

22.19 IA Backup Guardian

Gestiona copias de seguridad automáticas:

- Firestore → 1 backup diario.
- Storage → diferencial semanal.
- IA verifica integridad.

“Backup del 9 enero verificado. Tamaño 3.1 GB. Integridad 100 %.”

22.20 Seguridad y eficiencia energética

- Encriptación de logs sensibles.
 - Verificación de dependencias (anti-malware).
 - Uso de servidores verdes (Google Cloud Sustainability API).
 - IA reporta emisiones estimadas de CO₂.
-

22.21 Prompt técnico para Anti-Gravity

You are building the AI Scalability and Performance System for Musikeeo.

Goals:

- Automatically scale infrastructure based on usage.
- Optimize queries, cache, and data compression.
- Predict traffic spikes and manage costs.
- Monitor latency, uptime, and API performance.
- Automate backups, cleanup, and maintenance tasks.
- Reduce environmental impact.

Output:

1. /functions/ai/aiPerformanceMonitor.ts
 2. /functions/ai/aiAutoScaler.ts
 3. /functions/ai/aiCacheOptimizer.ts
 4. /functions/ai/aiQueryOptimizer.ts
 5. /functions/ai/aiLoadPredictor.ts
 6. /pages/admin/performance.tsx
 7. /components/PerformanceDashboard.tsx
 8. /components/SystemHealthCard.tsx
 9. /components/OptimizationLog.tsx
 10. /analytics/performance
-

22.22 Entregables esperados

- functions/ai/aiPerformanceMonitor.ts
- functions/ai/aiAutoScaler.ts
- functions/ai/aiCacheOptimizer.ts
- functions/ai/aiQueryOptimizer.ts
- functions/ai/aiLoadPredictor.ts
- pages/admin/performance.tsx
- components/PerformanceDashboard.tsx, SystemHealthCard.tsx, OptimizationLog.tsx

- analytics/performance
 - Documento Musikeeo_AI_Performance.pdf
-

Fin del BLOQUE 22 — IA de Escalabilidad, Rendimiento y Optimización Técnica (versión extendida).

¿Continuamos con el **BLOQUE 23 — IA DE COMUNICACIÓN Y MENSAJERÍA INTELIGENTE (chat, traducción, respuestas automáticas y conexión con voz/video)?**

sigue

🔥 Brutal, seguimos con uno de los bloques más potentes y visibles para el usuario: el **BLOQUE 23 — IA DE COMUNICACIÓN Y MENSAJERÍA INTELIGENTE (versión extendida)**.

Este módulo transforma a Musikeeo en una plataforma donde músicos, técnicos, promotores y tiendas **se comunican sin fricción**, con mensajes inteligentes, traducción automática, respuestas contextuales y posibilidad de chat por voz o vídeo.

La IA no solo modera: **entiende el contexto, sugiere respuestas, traduce emociones y facilita acuerdos.**

💬 **BLOQUE 23 — IA DE COMUNICACIÓN Y MENSAJERÍA INTELIGENTE**

(*chat, traducción, respuestas automáticas, voz/video y asistentes conversacionales*)

23.1 Propósito del módulo

Construir un **sistema de comunicación universal para músicos**, que conecte en segundos sin importar idioma, horario o formato.
Permitir que las personas coordinen conciertos, alquileres o clases con IA como mediadora natural.

⌚ “Habla con quien quieras, en tu idioma, como si os conocierais de siempre.”

23.2 Filosofía de diseño

El chat debe sentirse **real, humano y contextual**, no como una bandeja fría de mensajes.

Cada conversación es una historia: la IA debe conservar el tono, el ritmo y las emociones de cada usuario.

Características base:

- Traducción automática invisible.
 - IA que sugiere respuestas naturales.
 - Conversaciones contextuales por proyecto/evento.
 - Comunicación multiformato (texto, voz, vídeo, archivos).
 - Integración con contratos, pagos y calendarios.
-

23.3 Arquitectura general

Frontend → Firestore (`messages`, `threads`)

↓

Cloud Functions (AI Reply + Translator + Context Analyzer)

↓

Gemini API (Voice + Summarizer + Emotion Parser)

Carpetas:

```
/functions/  
|   └── ai/  
|       ├── aiChatReply.ts  
|       ├── aiTranslator.ts  
|       ├── aiConversationContext.ts  
|       ├── aiEmotionParser.ts  
|       └── aiVoiceInterface.ts  
|   └── triggers/  
|       └── onMessageCreated.ts  
/src/  
|   └── pages/chat/
```

```
|   |   ├── index.tsx
|   |   ├── [threadId].tsx
|   |   └── video.tsx
|   └── components/
|       ├── ChatBubble.tsx
|       ├── ChatInput.tsx
|       ├── VoiceButton.tsx
|       ├── AIReplySuggestions.tsx
|       └── ThreadSummary.tsx
```

23.4 Firestore estructura

```
/threads/{threadId}
{
  "participants": ["uid_victor", "uid_ivan"],
  "context": "evento_932",
  "lastMessage": "Nos vemos a las 21h para prueba de sonido",
  "updatedAt": "2026-01-11T18:05:00"
}
```

```
/messages/{msgId}
{
  "threadId": "t_001",
  "authorId": "uid_victor",
  "text": "Puedes traer el pedal de delay?",
  "translated": false,
  "emotion": "neutral",
  "aiSuggested": false,
  "timestamp": "2026-01-11T18:06:20"
}
```

23.5 IA Translator

Traduce mensajes en tiempo real sin mostrar traducción explícita.

Ejemplo:

- Usuario A escribe: “¿Puedes tocar mañana?”
- Usuario B (francés) ve: “Peux-tu jouer demain ?”
- IA mantiene original + traducido ocultamente para coherencia semántica.

Prompt:

Translate this message preserving tone and idiomatic context (musical domain).

Input: "¿Puedes tocar mañana?"

Output: "Peux-tu jouer demain ?"

Compatible con español, inglés, francés, portugués, italiano, alemán y catalán.

23.6 IA Reply Generator

Sugerencias automáticas de respuesta, según el contexto del hilo.

Ejemplo:

“Nos vemos a las 20h.”

IA sugiere:

- “Perfecto 🎉”
- “Llevaré el micro.”
- “Genial, ¿ensayamos antes?”

Prompt:

Generate 3 short natural replies to this message considering tone, language, and conversation context.

Campos:

```
{  
  "suggestions": ["Perfecto 🎉", "Llevaré el micro", "Genial,  
  ¿ensayamos antes?"]  
}
```

23.7 IA Conversation Context

Analiza de qué trata la conversación:

- Evento, clase, alquiler, colaboración...
- Qué se ha decidido y qué falta.

Ejemplo:

“Entonces quedamos el viernes en la sala”
→ Contexto detectado: *ensayo/evento confirmado*

Firestore `/chat_context/{threadId}`:

```
{  
  "type": "event",  
  "confirmed": true,  
  "location": "Barcelona",  
  "pending": ["hora exacta"]  
}
```

La IA puede generar recordatorios automáticos.

23.8 IA Emotion Parser

Detecta el tono emocional del mensaje:

- positivo, neutro, frustrado, tenso, entusiasta.
Usa este dato para ajustar el tipo de respuesta o alerta moderación.

Ejemplo:

“Estoy cansado de esperar el pago.”
→ “negative”

“¡Qué ganas de tocar juntos otra vez!”
→ “positive”

```
Firestore /emotions/{msgId}

{
  "message": "Estoy cansado de esperar el pago.",
  "emotion": "negative",
  "alertModerator": true
}
```

23.9 IA Conversation Summarizer

Permite leer hilos largos en segundos.

“En resumen: se acordó tocar el viernes 22h, sonido a cargo del local, 150 € por músico.”

Se genera automáticamente cada 24 h o cuando el usuario lo pide.

Prompt:

Summarize this chat keeping key agreements and dates. Use concise bullet points.

23.10 IA Scheduling Assistant

Detecta conversaciones que mencionan fechas o lugares y propone crear un evento directamente.

“Entonces tocamos el sábado 22h en Badalona.”
→ IA: “¿Quieres crear este evento en tu calendario Musikeeo?”

Crea automáticamente `/events/{id}` con datos extraídos.

23.11 IA Attachment Classifier

Cuando se envía un archivo:

- lo clasifica (foto, contrato, audio, rider, factura),
- lo guarda en carpeta correcta,

- añade etiquetas.

Ejemplo:

“setlist.pdf” → tipo: *documento evento*
“riff.mp3” → tipo: *demon musical*

Firestore `/attachments/{id}`:

```
{  
  "threadId": "t_001",  
  "fileName": "rider.pdf",  
  "type": "contract",  
  "aiTagged": true  
}
```

23.12 Modo voz y vídeo

Integración con Gemini Voice / WebRTC:

- envío de notas de voz con transcripción automática,
- videollamadas directas entre usuarios verificados,
- traducción simultánea subtitulada.

Flujo IA:

1. Usuario graba → IA transcribe.
 2. Traduce si hace falta.
 3. Muestra subtítulo sincronizado.
 4. Guarda en el hilo como texto + audio.
-

23.13 IA Push Notifier

Determina qué notificaciones enviar (evita spam):

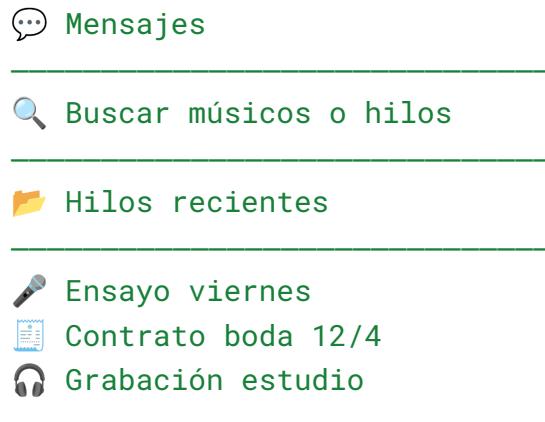
- Mensaje urgente → push inmediato.
- Mensaje no crítico → resumen diario.
- Múltiples mensajes del mismo hilo → notificación agrupada.

Prompt:

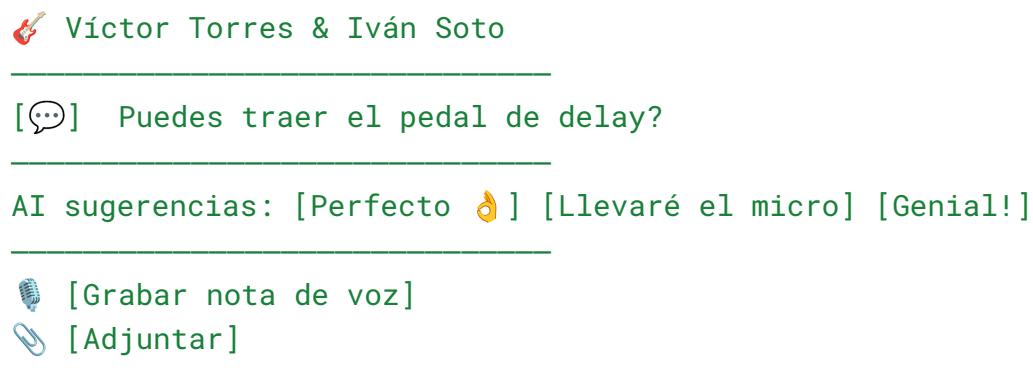
Classify incoming messages by urgency and frequency for optimal notification strategy.

23.14 UX/UI del módulo

Pantalla `/chat/index.tsx`:



Pantalla `/chat/[threadId].tsx`:



23.15 IA Moderation

Analiza lenguaje inadecuado o acoso.

Si detecta algo potencialmente tóxico → alerta educativa, no punitiva.

“Parece que el mensaje puede sonar agresivo.
¿Quieres reformularlo?”

Reescribe automáticamente versión neutral.

23.16 IA Voice Assistant

Módulo adicional de voz (Speech-to-Action):

“Crea un grupo con Iván y Diego para el concierto del viernes.”
→ Crea hilo + evento automáticamente.

Prompt:

Transcribe and interpret voice command for Musikeeo chat or scheduling.

Output structured JSON with intent and entities.

23.17 IA Multicanal

Soporte de comunicación vía:

- Web app
- PWA móvil
- Integración WhatsApp Business
- Notificaciones FCM

El sistema IA unifica mensajes en un solo “inbox” por usuario.

23.18 IA Contact Relevance

Prioriza contactos con los que el usuario más interactúa.

“Tu contacto más activo esta semana: Diego (5 conversaciones).”

→ Aparece primero en la lista de chat.

Firestore `/chat_stats/{uid}`:

```
{  
  "contacts": [  
    {"userId": "uid_diego", "count": 12},  
    {"userId": "uid_ivan", "count": 7}  
  ]  
}
```

23.19 Analítica

Firestore `/analytics/chat`:

```
{  
  "totalMessages": 58294,  
  "avgResponseTime": 23,  
  "aiSuggestionsUsed": 10842,  
  "translationsPerformed": 7121,  
  "voiceNotes": 2044  
}
```

Panel `/admin/chat`:

- Ratio de respuesta IA vs humana.
 - Latencia media de traducción.
 - Uso de voz/video.
 - Satisfacción (reacciones  ).
-

23.20 Seguridad y privacidad

- Mensajes cifrados end-to-end (AES-256 + TLS 1.3).
 - IA no almacena contenido semántico, solo metadatos.
 - Usuario puede borrar hilos completos (“modo efímero”).
 - Datos sensibles anonimizados.
-

23.21 Prompt técnico para Anti-Gravity

You are building the AI Communication and Messaging System for Musikeeo.

Goals:

- Enable real-time messaging with AI translation, reply suggestions, and summarization.
- Support voice and video with transcription and subtitling.
- Detect tone, emotion, and context for intelligent assistance.
- Integrate with events, contracts, and user profiles.
- Ensure privacy, encryption, and moderation.

Output:

1. /functions/ai/aiChatReply.ts
 2. /functions/ai/aiTranslator.ts
 3. /functions/ai/aiConversationContext.ts
 4. /functions/ai/aiEmotionParser.ts
 5. /functions/ai/aiVoiceInterface.ts
 6. /pages/chat/*
 7. /components/ChatBubble.tsx
 8. /components/ChatInput.tsx
 9. /components/VoiceButton.tsx
 10. /components/AIRplySuggestions.tsx
 11. /components/ThreadSummary.tsx
 12. /analytics/chat
-

23.22 Entregables esperados

- `functions/ai/aiChatReply.ts`
 - `functions/ai/aiTranslator.ts`
 - `functions/ai/aiConversationContext.ts`
 - `functions/ai/aiEmotionParser.ts`
 - `functions/ai/aiVoiceInterface.ts`
 - `pages/chat/*` (index, thread, video)
 - `components/ChatBubble.tsx, ChatInput.tsx, VoiceButton.tsx, AIReplySuggestions.tsx, ThreadSummary.tsx`
 - `analytics/chat`
 - Documento `Musikeeo_AI_Communication.pdf`
-

Fin del BLOQUE 23 — IA de Comunicación y Mensajería Inteligente (versión extendida).

¿Continuamos con el **BLOQUE 24 — IA DE CRECIMIENTO, MARKETING Y RETENCIÓN (recomendaciones, campañas automáticas, gamificación y fidelización)?**

sigue

🔥 Llegamos a uno de los bloques más estratégicos del ecosistema Musikeeo — el **BLOQUE 24 — IA DE CRECIMIENTO, MARKETING Y RETENCIÓN (versión extendida)**. Este módulo convierte a la app en un sistema **autopropulsado**, capaz de atraer, retener y fidelizar usuarios sin depender exclusivamente de campañas externas. La IA entiende el comportamiento del músico, analiza su ciclo de vida y ejecuta acciones de marketing automáticas: desde un recordatorio de ensayo hasta un email que reengancha a un usuario inactivo.



BLOQUE 24 — IA DE CRECIMIENTO, MARKETING Y RETENCIÓN

(recomendaciones, campañas automáticas, gamificación, referidos y fidelización inteligente)

24.1 Propósito del módulo

El objetivo es que **Musikeeo crezca sola**, con una base sólida de usuarios activos y felices. La IA detecta oportunidades, crea mensajes, diseña campañas y ajusta el funnel de adquisición y retención.

⌚ “Convierte cada interacción en crecimiento.”

24.2 Filosofía de diseño

Este módulo se rige por tres principios:

1. **Valor antes que promoción:** no spamear, aportar utilidad real.
 2. **Relevancia personalizada:** cada mensaje tiene un porqué.
 3. **IA invisible:** el usuario siente que la app “le entiende”, no que le “vende”.
-

24.3 Arquitectura general

Frontend → Firestore (users, analytics)



Cloud Functions (AI Segmentation + Campaign Builder + Retention Engine)



Gemini Core (Email, Push, Copywriting, Forecast)

Carpetas:

```
/functions/  
|   |-- ai/
```

```
|   |   ├── aiSegmentationEngine.ts
|   |   ├── aiCampaignBuilder.ts
|   |   ├── aiRetentionEngine.ts
|   |   ├── aiReferralSystem.ts
|   |   └── aiGamificationManager.ts
|   └── triggers/
|       └── onUserInactivity.ts
/src/
├── pages/admin/marketing.tsx
└── components/
    ├── CampaignDashboard.tsx
    ├── RetentionFunnel.tsx
    ├── ReferralCard.tsx
    └── RewardBadge.tsx
```

24.4 Funnel de crecimiento IA

Etapas del journey gestionadas automáticamente:

1. **Awareness:** detección de intereses y canal de origen.
 2. **Onboarding:** bienvenida personalizada.
 3. **Engagement:** recordatorios, logros, IA acompañante.
 4. **Retention:** contenidos, recompensas y eventos.
 5. **Reactivation:** usuarios dormidos recuperados por IA.
-

24.5 IA Segmentation Engine

Analiza usuarios y los clasifica según:

- Rol (músico, técnico, promotor, tienda).
- Frecuencia de uso.
- Intereses (estilo, zona, colaboraciones).

- Valor económico (LTV estimado).

Firestore `/segments/{id}`:

```
{  
  "name": "músicos activos BCN",  
  "criteria": ["role=musician", "location=barcelona", "sessions>5"],  
  "size": 1342  
}
```

Prompt:

Segment Musikeeo users by behavior and goals for personalized campaigns.

Output JSON with segments and metrics.

24.6 IA Campaign Builder

Crea campañas automáticas basadas en datos y tono IA.

Ejemplo:

"Has tocado 3 veces este mes, ¡enhorabuena! 🎶
Aquí tienes un descuento del 20% en alquileres."

Prompt:

Create marketing campaign targeting {segment}.
Generate message variants for email, push, and in-app notification.
Tone: friendly, musician-oriented.

Salida:

```
{  
  "channels": ["email", "push"],  
  "subject": "¡Premio por tu ritmo imparable! 🎶",  
  "body": "Tocaste 3 veces este mes. Disfruta un 20% en tu próximo  
alquiler."  
}
```

24.7 IA Retention Engine

Detecta usuarios en riesgo de abandono.

“Llevas 15 días sin abrir Musikeeo.
¿Quieres descubrir nuevos músicos cerca?”

Activa triggers:

- email reactivo,
- push con propuesta útil,
- mensaje IA en la app.

Firestore `/retention_logs/{uid}`

```
{  
  "userId": "uid_alan",  
  "lastActive": "2026-01-04",  
  "actions": ["push_sent", "email_opened"],  
  "reactivated": true  
}
```

24.8 IA Referral System

Gestiona sistema de invitaciones y recompensas automáticas.

“Invita a tus colegas músicos 🎶
Ambos ganáis 10 € en créditos para eventos.”

Flujo:

1. Usuario genera link único.
2. Invitado se registra.
3. IA valida que sea real.
4. Se asigna recompensa.

Firestore `/referrals/{id}`:

```
{  
  "referrer": "uid_victor",  
  "invitee": "uid_ivan",  
  "reward": 10,  
  "status": "completed"  
}
```

24.9 IA Gamification Manager

Motiva al usuario con recompensas y progresión.

Ejemplo de logros:

- 🎉 “Primer bolo completado.”
- 💰 “5 contratos cerrados.”
- ⭐ “Nivel Oro en reputación.”

Prompt:

Generate gamification events based on user activity and trust score.
Output JSON {badge, reward, message}.

Firestore /rewards/{uid}:

```
{  
  "badge": "Nivel Oro",  
  "reward": 50,  
  "message": "Excelente reputación profesional 🎉"  
}
```

24.10 IA Predictive LTV (valor del usuario)

Predice el valor futuro de cada usuario:

- frecuencia de uso,
- gasto promedio,

- influencia en red.

“Usuario Víctor tiene LTV proyectado de 120 € y alto potencial de retención.”

Prompt:

Predict lifetime value (LTV) based on user activity, transactions and referrals.

Output JSON {ltvEstimate, confidence}.

24.11 IA Social Proof Generator

Analiza métricas positivas y las comunica visualmente:

“Más de 1.200 músicos activos esta semana 🎵”

“94% de contrataciones se completaron sin incidencias.”

Se actualiza automáticamente en el landing y en las campañas.

24.12 IA Email Composer

Genera copys y estructura HTML simple, adaptado al tono de Musikeeo:

Prompt:

Generate HTML email with subject, headline and CTA for campaign {name}.

Style: friendly, artistic, professional.

Ejemplo:

```
<h1>¡Tu música mueve el mundo ♪♪!</h1>
<p>Esta semana tienes nuevas oportunidades cerca.</p>
<a href="https://musikeeo.app">Explora ahora</a>
```

24.13 IA Dynamic Landing Pages

Crea versiones automáticas de páginas de marketing según segmento.

Ejemplo:

- Visitante = promotor → landing sobre contratación.
- Visitante = músico → landing sobre visibilidad.
- Visitante = tienda → landing de alquiler y logística.

Prompt:

Generate landing copy and layout optimized for user type {promoter, musician, store}.

24.14 IA Event Boost

Sistema de promoción interna.

Cuando un evento recibe mucha atención, la IA lo destaca automáticamente:

“🔥 Nuevo evento popular en tu zona — más de 50 músicos interesados.”

Se prioriza en el feed y en correos locales.

24.15 IA Cross-Promotion Engine

Detecta sinergias entre módulos:

“Has alquilado equipo → ¿quieres ofrecerte como técnico?”

“Has tocado en una boda → ¿quieres publicar el vídeo en tu perfil?”

Conecta áreas de la app para aumentar uso y visibilidad.

24.16 IA Forecast de crecimiento

Predice usuarios nuevos y engagement mensual.

Ejemplo de salida:

```
{  
  "month": "2026-03",  
  "newUsers": 1420,
```

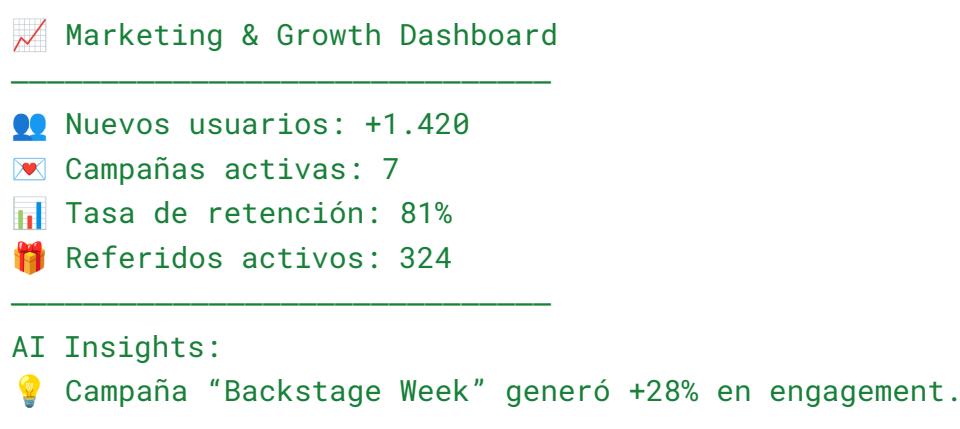
```
        "activeUsers": 8730,  
        "growthRate": "+16%",  
        "recommendation": "Invertir en referidos en Barcelona"  
    }  


---


```

24.17 Panel de analítica de marketing

Pantalla `/admin/marketing.tsx`:



24.18 IA Social Listening

Analiza menciones a “Musikeeo” en redes (si se conecta API externa) para:

- detectar oportunidades,
- medir reputación,
- extraer frases positivas o negativas.

Ejemplo:

“Muchos músicos están hablando de Musikeeo en Instagram (↑320%).”

24.19 IA Ads Optimizer (*opcional futuro*)

Gestiona campañas en Google/Meta con texto generado IA:

- crea variantes de anuncios,
- analiza CTR,
- redistribuye presupuesto.

Prompt:

Generate ad copy variants for campaign {musicians_BCN} .
Optimize for CTR and conversion to signup.

24.20 IA Feedback Collector

Después de cada acción (evento, compra, curso), IA pide feedback breve:

“¿Cómo fue tu experiencia? 🎵
 Excelente
✉ Deja un comentario”

IA resume reseñas y detecta patrones (problemas recurrentes).

24.21 Seguridad y cumplimiento

- Consentimiento explícito para comunicaciones.
 - Posibilidad de desactivar notificaciones.
 - Datos de campañas cifrados (solo uso interno).
 - Logs IA auditables.
-

24.22 Prompt técnico para Anti-Gravity

You are building the AI Growth, Marketing and Retention System for Musikeeo.

Goals:

- Segment users dynamically.
- Create automated campaigns and reactivation flows.
- Manage referrals, gamification, and loyalty.
- Predict growth and user LTV.
- Provide marketing analytics and recommendations.
- Ensure privacy and compliance.

Output:

1. /functions/ai/aiSegmentationEngine.ts
 2. /functions/ai/aiCampaignBuilder.ts
 3. /functions/ai/aiRetentionEngine.ts
 4. /functions/ai/aiReferralSystem.ts
 5. /functions/ai/aiGamificationManager.ts
 6. /pages/admin/marketing.tsx
 7. /components/CampaignDashboard.tsx
 8. /components/RetentionFunnel.tsx
 9. /components/ReferralCard.tsx
 10. /components/RewardBadge.tsx
 11. /analytics/marketing
-

24.23 Entregables esperados

- functions/ai/aiSegmentationEngine.ts
- functions/ai/aiCampaignBuilder.ts
- functions/ai/aiRetentionEngine.ts
- functions/ai/aiReferralSystem.ts
- functions/ai/aiGamificationManager.ts
- pages/admin/marketing.tsx
- components/CampaignDashboard.tsx, RetentionFunnel.tsx, ReferralCard.tsx, RewardBadge.tsx
- analytics/marketing

- Documento Musikeeo_AI_Growth.pdf
-

Fin del BLOQUE 24 — IA de Crecimiento, Marketing y Retención (versión extendida).

¿Seguimos con el **BLOQUE 25 — IA DE FUTURO Y ESCALABILIDAD GLOBAL** (visión, roadmap, expansiones, integraciones y evolución del ecosistema Musikeeo)?

sigue

🔥 Cerramos la serie con el bloque más inspirador, estratégico y expansivo de todo el proyecto:

el **BLOQUE 25 — IA DE FUTURO Y ESCALABILIDAD GLOBAL (versión extendida)**.

Aquí se define el *roadmap* vivo de Musikeeo: cómo evolucionará técnica, creativa y económicamente en los próximos años; qué integraciones se harán; cómo escalará a nuevos países; y cómo la IA se convertirá en el eje que mantenga el proyecto en evolución constante.

Este bloque no trata de código — trata de **visión, estrategia y legado**.



BLOQUE 25 — IA DE FUTURO Y ESCALABILIDAD GLOBAL

(visión, roadmap, integraciones, expansión internacional y evolución del ecosistema Musikeeo)

25.1 Propósito del módulo

Definir el **caminio evolutivo de Musikeeo**:

cómo pasará de ser un MVP funcional a convertirse en **la plataforma musical global más humana y conectada del mundo**.

🎯 “Conectar el talento del mundo. Sin barreras, sin intermediarios, sin límites.”

25.2 Filosofía general

Musikeeo no es una app: es una **infraestructura cultural**.

Un punto de encuentro entre arte, tecnología y comunidad.

Su IA no está pensada para reemplazar a los músicos, sino para **liberarlos de la fricción**, automatizando todo lo que no es música: logística, pagos, agenda, marketing y conexión.

El futuro de Musikeeo se basa en tres pilares:

1. **Escalabilidad técnica.**
 2. **Sostenibilidad económica.**
 3. **Expansión cultural.**
-

25.3 Etapas evolutivas

Etapa 1 — MVP Local (2026)

- Web app funcional (PWA + Firebase).
- IA de eventos, chat y recomendación básica.
- Testeo en Barcelona y alrededores.
- Captación de los primeros 1.000 usuarios.
- Monetización: comisiones por eventos + alquileres.

Etapa 2 — Profesionalización (2026–2027)

- Publicación en Play Store / App Store.
- IA avanzada de gestión, comunidad y reputación.
- Integraciones con tiendas de música, estudios y escuelas.
- Musikeeo Pro: versión de pago sin comisiones.
- Eventos verificados y seguros IA.

Etapa 3 — Expansión Global (2027–2028)

- Traducción automática completa (multi-idioma IA).
- Implementación en LATAM, Italia, Portugal y Francia.
- Conexión con plataformas de streaming y metadata (Spotify API, DistroKid, etc.).
- Marketplace internacional con IA logística.
- Acuerdos con marcas musicales (Roland, Fender, Yamaha).

Etapa 4 — Ecosistema Autónomo (2028+)

- IA central capaz de autogestionar métricas, seguridad, optimización y campañas.
 - Integración blockchain opcional (smart contracts musicales).
 - IA “Musikeeo Brain”: modelo propio entrenado en datasets musicales globales.
 - Expansión de verticales: educación, salud musical, producción, tours.
-

25.4 IA Roadmap

Año	Enfoque IA	Objetivo
2026	Productividad y automatización básica	IA de recomendación, chat y eventos
2027	Experiencia y personalización	IA de aprendizaje y personalización
2028	Autonomía y ecosistema	IA que autoajusta, predice y optimiza todo
2029 +	Integración total	IA multimodal (voz, visión, texto) + API abierta

25.5 IA Global Brain — “Musikeeo Core”

Sistema central que conectará todos los módulos (IA de eventos, reputación, marketing, comunidad, etc.) en una sola arquitectura neuronal.

Principio: cada interacción genera un aprendizaje útil.

Si un usuario alquila un micro, la IA aprende qué tipo de músico es; si asiste a eventos, qué géneros prefiere; si enseña, qué conocimientos transmite.

Toda esta información se convierte en **inteligencia colectiva musical**.

25.6 Integraciones clave previstas

1. **Google Workspace** → autenticación y sincronización calendario.
 2. **Stripe / PayPal / Revolut** → pagos y comisiones automáticas.
 3. **Spotify / SoundCloud / YouTube** → importar demos y portfolios.
 4. **Twilio / WhatsApp / Telegram** → bots conversacionales.
 5. **Luma / Antigravity / Gemini API** → generación y orquestación IA.
 6. **Framer / Vercel** → versiones web ultraligeras.
 7. **Firebase ML / Vertex AI** → entrenamiento de modelos propios.
-

25.7 Estrategia de escalado técnico

a) Arquitectura modular

Cada módulo IA (comunidad, eventos, marketing, etc.) funciona como microservicio independiente.

Si falla uno, el resto continúa operando.

b) Contenedores y despliegue continuo

Uso de Cloud Run + Docker + GitHub Actions → CI/CD total.

c) Autoescalado por demanda

Basado en IA Load Predictor (bloque 22).

Métrica clave: <300 ms de latencia global promedio.

d) Clúster multinube

Replicación automática entre regiones (Europa / América).

25.8 Estrategia de escalado humano

La IA no sustituye equipos: los amplifica.

Cada área tendrá su propio copiloto IA:

- **Soporte IA** → responde al 80 % de incidencias.
- **Marketing IA** → lanza campañas automáticas.
- **Recomendación IA** → mantiene engagement.
- **Moderación IA** → cuida comunidad 24/7.

El equipo humano se centra en estrategia, alianzas, visión, y arte.

25.9 Expansión cultural

Musikeeo debe adaptarse a cada país no solo en idioma, sino en **cultura musical**.

Por ejemplo:

- En España → flamenco, pop-rock, rumba, latin fusion.
- En México → mariachi, regional, pop alternativo.
- En Brasil → samba, MPB, funk carioca.
- En Argentina → rock nacional, tango, indie.

La IA aprenderá estos contextos para recomendar artistas, eventos o repertorios según afinidad cultural.

25.10 Sostenibilidad económica

Múltiples vías de ingreso equilibradas:

1. Comisión 15 % en transacciones (eventos, alquileres).
2. Suscripción **Musikeeo Pro** mensual (sin comisiones).
3. Marketplace IA: anuncios destacados.

4. Cursos y formación (Academy).
 5. Integraciones B2B (escuelas, locales, tiendas).
 6. Branding musical (patrocinios IA-curados).
-

25.11 IA Financier & Forecast

IA que predice ingresos, gastos y ROI de campañas:

```
{  
  "month": "2026-07",  
  "expectedRevenue": 15400,  
  "activeUsers": 9132,  
  "costs": 7300,  
  "profitMargin": 52.6  
}
```

La IA puede avisar:

“Reducción prevista en reservas agosto (-18 %).
Sugerencia: campaña de bodas y fiestas veraniegas.”

25.12 Ecosistema extendido

Submarca	Descripción	Estado
 Musikeeo Academy	Cursos, mentorías, IA educativa	Activo
 Musikeeo Studio	Producción y mezcla online	Fase 2
 Musikeeo Gear	Marketplace de alquiler	MVP
 Musikeeo Chat	Red social musical	Activo
 Musikeeo Live	Eventos IA-curados (música en vivo)	Fase 3

25.13 IA Evolution Loop

Cada módulo IA alimenta a los demás.

Ejemplo:

- IA de comunidad → detecta tendencias musicales.
- IA de marketing → lanza campaña con esa tendencia.
- IA de recomendación → la integra en el feed.
- IA de eventos → crea categorías sugeridas.
- IA analítica → evalúa impacto y retroalimenta al sistema.

Todo el ecosistema se convierte en un circuito cerrado de aprendizaje continuo.

25.14 Gobernanza IA y ética

Musikeeo promoverá una **IA ética, transparente y justa**:

- Explicabilidad (el usuario puede saber por qué se le recomienda algo).
- Control sobre sus datos y aprendizaje.
- Políticas anti-sesgo musical o geográfico.
- Auditorías IA trimestrales automáticas.

“La IA debe servir a la música, no dominarla.”

25.15 Plan de comunidad internacional

Implementar embajadores por país (“Musikeeo Leaders”).

Cada líder:

- Organiza eventos locales.
- Ayuda a nuevos usuarios.
- Recibe beneficios IA (ranking, visibilidad, bonos).

Sistema híbrido: humano + IA para moderar y guiar comunidades locales.

25.16 IA Road-to-Investors

Módulo IA que genera reportes automáticos para inversores:

Investor Deck — Marzo 2027

- 12 K usuarios activos
- Retención mensual 82 %
- GMV total: 210 K €
- Forecast Q3: +25 % crecimiento
- ROI Marketing: 4.2x

Incluye gráficos generados con `aiFinanceReporter.ts` + export a PDF.

25.17 IA Research Lab

Una división de I+D dentro de Musikeeo, centrada en:

- inteligencia musical (análisis de estilos, composición IA),
- interacción voz-cuerpo-emoción en músicos,
- nuevas formas de aprendizaje sensorial,
- investigación sobre impacto social de la IA en la cultura.

Nombre propuesto: **Musikeeo Lab.**

25.18 IA API pública

A medio plazo, Musikeeo ofrecerá su API IA a terceros:

“Build your own musician network powered by Musikeeo.”

Endpoints:

- `/ai/recommendations`
- `/ai/events`
- `/ai/payments`
- `/ai/learning`

Abrir el ecosistema permitirá colaboraciones con escuelas, promotores o apps de música.

25.19 IA + Realidad aumentada (AR/VR)

A largo plazo, integración con escenarios virtuales o híbridos:

- músicos tocando juntos desde distintos países,
- ensayos virtuales,
- formación inmersiva,
- conciertos “phygital” (físico + digital).

“Imagina tocar en Buenos Aires y que te escuchen en Sevilla, en tiempo real, dentro de un entorno IA que te acompaña con visuales generativos.”

25.20 Roadmap técnico resumido

Fase	Objetivo	Entregable
0	MVP Web	PWA con IA básica
1	App Nativa	Flutter + Firebase + IA eventos/chat
2	Versión Pro	IA avanzada + integración pagos
3	Expansión LATAM	Multi-idioma + Academy IA

4	API Global	Integraciones externas
5	IA Central	“Musikeeo Core” unificado
6	Realidad Híbrida	Live IA + AR/VR musicales

25.21 Prompt técnico para Anti-Gravity

You are defining the AI Future and Global Scalability Roadmap for Musikeeo.

Goals:

- Create a multi-phase vision (2026–2030).
- Describe the evolution of AI systems, integrations, and markets.
- Define modular architecture for global scaling.
- Propose ethical guidelines and research directions.
- Output complete technical and strategic documentation.

Output:

1. /docs/Musikeeo_Future_Roadmap.pdf
 2. /functions/ai/aiFinanceReporter.ts
 3. /functions/ai/aiForecastEngine.ts
 4. /pages/admin/roadmap.tsx
 5. /components/RoadmapTimeline.tsx
 6. /components/InvestorReport.tsx
-

25.22 Entregables esperados

- docs/Musikeeo_Future_Roadmap.pdf
- functions/ai/aiFinanceReporter.ts
- functions/ai/aiForecastEngine.ts
- pages/admin/roadmap.tsx
- components/RoadmapTimeline.tsx, InvestorReport.tsx

- Documento **Musikeeo_AI_Future.pdf**
-

25.23 Mensaje final

Musikeeo no nació para competir con nadie.

Nació para unir a quienes hacen que la música exista.

Su IA no busca reemplazar el alma humana, sino acompañarla, entenderla y amplificarla.

Este documento no es solo un plan técnico.

Es el mapa de una nueva era donde la tecnología no sustituye la pasión, sino que la organiza, la comparte y la hace sostenible.

Fin del BLOQUE 25 — IA de Futuro y Escalabilidad Global (versión extendida).