ChatGPT

Asistente personal de voz – Modelo de negocio y documentación técnica

Resumen Ejecutivo y Visión

¿Qué problema resuelve la app? Esta aplicación móvil actúa como un asistente personal por voz que **registra y transcribe automáticamente conversaciones y sonidos del entorno**, extrayendo de ellos información útil. Resuelve el problema de **recordar y organizar la gran cantidad de información** que manejamos diariamente – desde tareas mencionadas en conversaciones, hasta ideas espontáneas o signos de estado de ánimo. La app evita que **datos importantes pasen desapercibidos**, liberando al usuario de tomar notas manualmente y ayudándole a **gestionar su productividad y bienestar mental** de forma más eficaz.

¿Por qué ahora? En la actualidad confluyen varias tendencias tecnológicas y sociales que hacen oportuna esta solución. Los **modelos avanzados de IA de voz** (como Whisper) y lenguaje (LLMs tipo GPT) han madurado lo suficiente como para permitir transcripciones precisas y análisis contextuales en tiempo real, incluso en dispositivos personales. A la vez, hay una **creciente demanda de automatización personal**: profesionales y usuarios buscan delegar tareas repetitivas o de seguimiento a asistentes digitales. Tras la pandemia, aumentó la adopción de herramientas de **teletrabajo y auto-cuidado mental**, abriendo espacio para un asistente que combine **productividad con salud mental**. En resumen, **ahora es viable y valioso** tener un asistente de voz siempre disponible que integre IA y conectividad para mejorar nuestro día a día.

¿Quién la necesita? La app se dirige a **usuarios con altos niveles de información y actividades diarias**. Por un lado, profesionales de negocio o tecnología con agendas ocupadas (gerentes, líderes de proyecto, consultores) que deseen capturar acuerdos de reuniones, ideas o recordatorios sin interrumpir su flujo de trabajo. Por otro lado, personas interesadas en **mejorar su productividad personal** (estudiantes, emprendedores) o en llevar un **registro de su vida diaria**. También es útil para quienes buscan **monitorear su bienestar emocional** – por ejemplo, registrando por voz sus pensamientos o conversaciones para luego analizarlos. En entornos empresariales, equipos de trabajo podrían utilizarlo para **documentar reuniones** automáticamente y alimentar sistemas corporativos (Jira, Confluence). En definitiva, lo necesitan usuarios que quieran **tener control y análisis inteligente de su día** sin invertir esfuerzo extra en anotar todo manualmente.

Modelo Canvas Resumido (9 Bloques)

A continuación se presenta el modelo de negocio de la aplicación en formato Canvas, con sus nueve componentes clave resumidos:

Propuesta de Valor	Segmentos de Clientes	Canales
- Captura y transcripción de conversaciones diarias de forma automática. - Clasificación inteligente (personal, trabajo, ideas, etc.) y generación de recordatorios/tareas. - Integración con herramientas (calendario, Jira, Confluence...) para acciones automatizadas. - Mejora la productividad y bienestar evitando la pérdida de información importante.	- Profesionales con alta carga de reuniones o tareas (ej. gerentes, líderes de proyecto). - Personas que buscan mejorar su productividad personal o llevar un diario de actividades. - Usuarios con interés en autocuidado mental mediante registro de pensamientos y emociones. - Equipos o empresas que deseen aprovechar la transcripción automática para documentación.	- Distribución a través de tiendas de apps móviles (App Store, Google Play). - Posible canal B2B: acuerdos con empresas o integraciones en suites de productividad. - Comunidad en línea y contenido educativo (blogs, webinars) para atraer usuarios técnicos. - Marketplace de Atlassian (para integraciones con Jira/ Confluence).
Relación con Clientes	**Fuentes de Ingresos**	**Recursos Clave**
- Auto-servicio con flujo de onboarding guiado en la app. - Soporte en línea (chatbot integrado, FAQs) y actualizaciones periódicas de la app. - Comunidad de usuarios para compartir automatizaciones y mejores prácticas. - Soporte dedicado para clientes empresariales (gestor de cuenta, SLAs).	- Modelo freemium: app gratuita con funciones básicas; suscripción premium para características avanzadas. - Licencias empresariales o suscripciones de equipo (integración con herramientas corporativas y administración centralizada). - Servicios de valor añadido (ej., reportes de productividad, coaching personalizado basado en datos). - Marketplace: comisión por integraciones o complementos premium de terceros.	- Infraestructura de IA: modelos de transcripción (Whisper) y NLP para análisis; servidores (on-premise o nube) para procesamiento. - Equipo de desarrollo multidisciplinario (Flutter, backend, IA/ML, DevOps). - Alianzas tecnológicas (OpenAI o comunidad open source Whisper; proveedores cloud; Atlassian para acceso a API de Rovo/Jira). - Base de datos segura de transcripciones y resultados (con cifrado y controles de acceso) para aprendizaje y personalización.

Actividades Clave	Socios Clave	Estructura de Costos
- Desarrollo y mejora continua de la app móvil (Flutter) y del backend (FastAPI, integraciones). - Entrenamiento/ajuste de modelos de IA para transcripción, clasificación de contenido e incluso detección emocional. - Monitoreo del rendimiento y experiencia de usuario (tiempos de respuesta, precisión de transcripciones) e iteración rápida. - Gestión de integraciones con terceros (APIs de Jira, Confluence, Google, n8n) y aseguramiento de cumplimiento legal (privacidad, consentimiento).	- Proveedores de tecnología: OpenAI (o frameworks open source para Whisper/LLM), servicios de voz TTS, y cloud (AWS, GCP, Azure) para infraestructura. - Atlassian y otras empresas de productividad (Google, Microsoft) para integraciones nativas y soporte en sus plataformas (ej. Atlassian Marketplace). - Comunidades open-source y startups afines (n8n, proyectos de asistentes personales) para colaboración e integración mutua. - Aliados en salud mental y productividad (expertos, instituciones) que respalden la eficacia y difundan la herramienta.	- Servidores e infraestructura cloud (GPU/CPU para procesar Whisper y modelos, almacenamiento seguro). - Costos de licencias o uso de APIs de terceros (p. ej., llamadas a APIs de OpenAI, servicios de voz, suscripción de Atlassian si aplica). - Gastos de desarrollo y operaciones (salarios del equipo, herramientas de desarrollo, CI/CD, monitorización). - Inversión en seguridad y cumplimiento (cifrado, auditorías de datos, manejo de consentimiento), marketing y soporte al cliente.

Componentes Técnicos de la Solución

Arquitectura actual de la aplicación

La arquitectura actual se basa en una **estructura modular con contenedores (Docker Compose)** que reúne los componentes principales del sistema:

- **Aplicación Flutter (móvil)** – Cliente instalado en el dispositivo del usuario (Android/iOS) que graba el audio ambiente o del micrófono. Proporciona la interfaz de usuario para controlar el asistente (iniciar/pausar grabación, ver resultados) y posiblemente activación por voz.
- **Backend FastAPI (Python)** – Servicio web que recibe los archivos de audio desde la app. Contiene la lógica para procesar esos audios: los guarda temporalmente, invoca la transcripción con Whisper y gestiona el flujo de datos resultante. Este backend expone **API REST** (por ejemplo, endpoint `/upload` para subir audio) y podría también ofrecer un canal WebSocket para transmisión en tiempo real.
- **Motor de transcripción Whisper** – Modelo de reconocimiento de voz (ASR) de OpenAI ejecutado localmente. Puede integrarse **dentro del mismo servicio FastAPI** (cargando el modelo en memoria) o desplegarse como un **microservicio separado** dedicado a transcribir. Whisper convierte el audio en texto de forma precisa, trabajando con modelos pre-entrenados (por ejemplo, modelo *base* o *tiny* para equilibrar precisión/rendimiento).
- **Workflow automation (n8n)** – Herramienta de automatización en contenedor, configurada para orquestar las acciones posteriores a la transcripción. n8n recibe datos (vía un webhook HTTP activado por el backend, o leyendo desde la base de datos/MQTT) y ejecuta flujos predefinidos: por ejemplo, si una transcripción contiene una tarea **[Trabajo]**, crear automáticamente un issue en Jira;

si contiene un evento **[Personal]**, añadir un recordatorio en Google Calendar, etc. n8n actúa como **hub integrador** sin código para conectar la app con múltiples servicios externos.

- **Broker de mensajes MQTT (EMQX)** – Sistema de mensajería en tiempo real al que la app y otros componentes se suscriben. El backend o n8n pueden publicar mensajes a **topics MQTT** para notificar a la app de resultados o alertas (p.ej., " Tarea creada en Jira" o una alerta por voz). EMQX permite una comunicación **bidireccional** casi instantánea con el dispositivo, incluso si la app está en segundo plano, y facilita la extensión futura a IoT (control de dispositivos del hogar, etc.).
- **Base de datos MongoDB** – Repositorio NoSQL para almacenar de forma persistente las transcripciones y metadatos asociados. Por ejemplo, guarda cada conversación transcrita con campos como texto, fecha/hora, categoría (personal/profesional), tonalidad emocional estimada, usuario, etc. MongoDB sirve también para almacenar configuraciones de usuario, historiales y cualquier información estructurada que necesite la app (evitando depender solo de n8n para persistencia). La elección de MongoDB se debe a su flexibilidad para almacenar documentos JSON (fácilmente podemos almacenar el resultado completo de cada análisis de audio).

Todos estos componentes están **contenedorizados y orquestados con Docker Compose** para facilitar el desarrollo y despliegue. Por ejemplo, un fichero `docker-compose.yml` levanta los servicios: app (Flutter web o emulador para pruebas), API FastAPI + Whisper, n8n, EMQX y MongoDB, de modo que se puedan ejecutar en conjunto en un entorno local o en un servidor. Esta arquitectura modular permite que cada pieza sea intercambiable o escalable (por ejemplo, podríamos escalar horizontalmente el microservicio de transcripción si la demanda crece, o sustituir n8n por otra herramienta).

Integraciones previstas con terceros

La propuesta incluye integraciones con varias herramientas y plataformas externas para ampliar las capacidades del asistente:

- **Atlassian Rovo (Confluence/Jira)** – Rovo es el nuevo asistente con IA de Atlassian. La app planea conectarse con Rovo y las APIs de Atlassian para, por ejemplo, **crear issues de Jira automáticamente**, extraer información de Confluence o actualizar páginas con resúmenes de conversaciones. Rovo podría ayudar a interpretar comandos de usuario relacionados con conocimiento organizacional. La integración aprovechará las APIs oficiales de Jira/Confluence (REST/GraphQL) o conectores de Rovo para enviarle la transcripción correspondiente a una reunión de trabajo y que genere actualizaciones en esos sistemas.
- **MCPs (Puntos de Conexión Modulares)** – Bajo este término se engloban integraciones configurables mediante webhooks u otros mecanismos genéricos. La idea es proporcionar en la app la opción de **conectar con "tu propio webhook" o servicio personalizado** (un **MCP** podría ser un *endpoint* definido por el usuario, o un flujo en otra plataforma de automatización). Esto hace al sistema *plug-and-play*: cada usuario o empresa puede conectar sus herramientas favoritas (por ejemplo, su propio servidor o flujo de automatización) sin cambios en el código de la app. En esencia, los MCPs permiten que el asistente desencadene acciones en **cualquier sistema externo configurado**, ya sea un script corporativo, un servicio en la nube privado, etc.
- **Google Calendar y otros servicios de calendario** – Mediante las API de Google Calendar (y potencialmente Outlook Calendar u otros), la app puede **programar eventos o recordatorios automáticamente**. Por ejemplo, si en una conversación personal el usuario dice "Te veo el lunes a las 3 PM", el asistente podría proponer crear una cita en el calendario del usuario. La integración con calendarios aseguran que los compromisos detectados en la transcripción no se queden solo en texto, sino que se reflejen en la agenda real del usuario.

- **n8n (flujos avanzados)** – Además de las acciones predefinidas, los usuarios avanzados/empresariales podrían aprovechar **flujos personalizados en n8n**. La integración prevista con n8n (ya parte del stack) permitirá exponer ciertas entradas para que cada cliente adapte el comportamiento: p. ej., un nodo de n8n podría recibir la transcripción y decidir según palabras clave qué hacer (enviar email, crear una tarea en Asana, desencadenar un webhook MCP, etc.). En futuras versiones, se podría incluso publicar un **n8n template** o colección de flujos listos para integrar con la app.
- **Otras integraciones (futuras)** – Se contempla compatibilidad con asistentes de voz existentes o agentes externos. Por ejemplo, integrarse con **Microsoft 365 Copilot** u otros copilotos empresariales para aprovechar su comprensión de contexto corporativo, o conectar con servicios como Slack/Teams (enviar un resumen de reunión transcrita a un canal). En el ámbito de bienestar, podría integrarse con aplicaciones de journaling o coaching que acepten entradas de texto. Gracias al enfoque modular (APIs abiertas, MQTT y webhooks), el ecosistema de integraciones puede crecer conforme surjan nuevas oportunidades.

Flujo de datos: desde el audio hasta la acción

El **flujo de procesamiento** de la app, de extremo a extremo, consta de varios pasos consecutivos que transforman la voz del usuario en acciones y conocimientos útiles:

1. **Captura de audio (cliente):** El usuario inicia una grabación desde la app Flutter (ya sea manualmente pulsando un botón, o en un futuro mediante activación por voz tipo *hotword*). La app utiliza el micrófono del dispositivo para captar el audio ambiente o conversación. Durante la grabación, puede funcionar en segundo plano, p. ej. grabando una reunión o simplemente el entorno del usuario.
2. **Envío al backend:** Al detener la grabación (o en bloques de audio periódicos si es streaming), la app envía el archivo de audio resultante al backend FastAPI. Esto ocurre vía **API REST HTTPS** (por ejemplo, una petición `POST /audios` con el archivo adjunto). La conexión es segura (TLS) para proteger los datos en tránsito. El backend confirma recepción y almacena el archivo en un directorio temporal (o buffer en memoria).
3. **Transcripción con Whisper:** El backend desencadena el proceso de transcripción. Si Whisper está integrado localmente, carga el modelo (si no estaba ya cargado) y ejecuta la transcripción del audio recibido. Si se usa un microservicio dedicado, el backend le enviará una solicitud (por ej. una petición HTTP interna) con el audio para obtener el texto transcrito. Whisper procesa el audio y devuelve el **texto transcrito** con alta fidelidad, manejando incluso ruido de fondo o distintas expresiones en español.
4. **Procesamiento NLP y clasificación:** Con la transcripción en texto, el backend realiza análisis adicionales. Primero, **limpia el texto** (por ejemplo, eliminando titubeos o ruido verbal si es necesario). Luego aplica **clasificación** para entender el contenido: esto puede involucrar modelos de lenguaje (LLM) o reglas definidas. Por ejemplo, se puede llamar a GPT-4 (vía API de OpenAI) con un prompt que analice la intención de la conversación o extraiga elementos clave (tareas pendientes, eventos, ideas). También se puede clasificar la conversación en categorías: personal, trabajo, proyecto X, etc., usando palabras clave o IA. Adicionalmente, se estima un **tono emocional**: mediante análisis de sentimiento sobre el texto (posiblemente complementado con paralingüística del audio en versiones futuras), se determina si la conversación fue positiva, neutra, tensa, etc. El resultado de este procesamiento es un **JSON de metadatos enriquecidos**, que incluye campos como: transcripción textual, idioma, duración del audio, categoría/etiquetas, emoción predominante, posibles acciones detectadas (ej. “tarea: enviar informe mensual”).

5. **Almacenamiento en base de datos:** El backend guarda este resultado en MongoDB, creando un documento por conversación. Esto permite llevar un **historial** para el usuario, al que podría acceder más tarde para búsquedas (*p.ej.*, "¿qué se dijo sobre el proyecto ABC el martes pasado?"). En este punto, el audio original puede **borrarse o cifrarse** si ya no es necesario, para reducir riesgos de privacidad; la transcripción es lo principal a conservar.
6. **Disparadores de automatización:** Tras almacenar, el backend notifica a otros componentes para que se ejecuten las acciones correspondientes. Hay dos vías principales no excluyentes:
   a. **Webhook a n8n:** el backend realiza una petición HTTP a un flujo webhook de n8n, pasándole el JSON de la transcripción. Esto activa un flujo de trabajo predefinido en n8n.
   b. **Publicación MQTT:** el backend publica un mensaje en EMQX, en un tópico específico (p. ej. `usuario123/transcripciones`). La app móvil, si está suscrita, podría recibir una notificación inmediata con un resumen, y n8n u otros servicios también pueden suscribirse a ese tópico para procesar datos.
7. **Automatizaciones (n8n y terceros):** Al activarse el flujo de **n8n**, este puede contener múltiples nodos de integración: por ejemplo, un nodo que filtra o interpreta el contenido (posiblemente usando también GPT para mayor contexto), y según ello:
8. Llamar a la API de Jira para crear/modificar una tarea con la descripción obtenida de la conversación.
9. Invocar la API de Confluence para crear una página resumen de una reunión, adjuntando la transcripción o un enlace a ella.
10. Enviar un mensaje vía API de Slack/Teams si se detectó que cierta persona debe ser notificada.
11. Llamar a un **webhook MCP** personalizado proporcionado por el usuario (por ejemplo, su propio script) con los datos.
12. Guardar ciertos datos en otra base de datos o Google Sheets para tracking.
    En paralelo o alternativamente, otros sistemas suscritos al mensaje MQTT podrían actuar. Por ejemplo, si se configuró un **agente Rovo de Atlassian**, este podría recibir la transcripción para actualizar conocimiento interno.
13. **Notificación al usuario:** Finalmente, el resultado de esas acciones vuelve al usuario de forma amigable. La app podría mostrar en pantalla una **notificación** ("3 acciones realizadas") o, dado que es un asistente de voz, incluso **leer por voz un resumen** de lo hecho: "He creado una tarea en tu Jira y agendado el evento en tu calendario". Para esto, se usaría texto a voz (TTS) integrado en el dispositivo o un servicio (por ejemplo, Coqui TTS o gTTS si es offline). La comunicación en tiempo real de estas notificaciones se facilita vía **MQTT**, ya que la app está suscrita a un tópico de respuestas (p. ej. `usuario123/notificaciones`), donde el backend/n8n publica la confirmación de las acciones completadas.
14. **Iteración con el usuario:** En caso de que el asistente necesite confirmación ("¿Deseas crear esta tarea en Jira?") o información adicional, podría solicitarlo mediante la app (voz o texto). El usuario responde, y ese audio se envía y procesa de la misma manera. De esta forma se logra una **interacción continua**, acercándose a un diálogo con el asistente.

En resumen, el flujo va **de audio -> texto -> comprensión -> acción -> retroalimentación al usuario**, cerrando el ciclo. Todo ocurre en segundos (idealmente), brindando una experiencia donde el usuario habla naturalmente y el asistente se encarga de lo demás.

Consideraciones de seguridad y privacidad

Dado que esta aplicación maneja datos altamente sensibles (conversaciones personales, potencialmente contenido de trabajo confidencial y datos de salud emocional), la seguridad y privacidad son prioridades desde el diseño:

- **Consentimiento y legalidad de grabación:** La app debe **respetar las leyes de grabación de conversaciones**. Es esencial que solo grabe con el consentimiento del usuario y, si hay terceras personas, notificar o requerir su aprobación según la jurisdicción. Se podrían incluir advertencias legales en la instalación, y señales audibles (un tono periódico) indicando que se está grabando, para transparencia. El usuario debe tener control total, pudiendo **pausar o detener** la grabación en cualquier momento.
- **Cifrado de datos:** Toda la comunicación entre la app y el backend viaja cifrada (HTTPS/TLS). Además, los **datos almacenados** están cifrados en reposo: las transcripciones en MongoDB pueden cifrarse a nivel de campo o base de datos, de modo que si alguien accede al almacenamiento, no pueda leer el contenido sin las claves apropiadas. Igualmente, cualquier archivo de audio que se guarde temporalmente en el servidor podría cifrarse en disco o, mejor aún, mantenerse en memoria y descartar tras uso.
- **Almacenamiento temporal mínimo:** Para minimizar riesgos, el diseño procura **no almacenar audio crudo más tiempo del necesario**. Una vez transcrito el audio y procesado, el archivo original podría eliminarse automáticamente, salvo que el usuario explícitamente quiera guardarlo. Las transcripciones en texto se conservan, pero el usuario podría tener opciones de auto-borrado después de cierto tiempo (especialmente si es contenido sensible, e.g. conversaciones personales).
- **Autenticación y control de acceso:** El sistema implementará **autenticación de usuarios** tanto en la app como en las APIs. Cada transcripción y acción está asociada a un usuario ID o token, de forma que **nadie más pueda acceder a tus datos**. Si se ofrece una interfaz web o API para consultar historiales, estará protegida con login y posiblemente 2FA. Los roles de acceso se definen claramente: por ejemplo, en un entorno empresarial, quizás un administrador podría ver ciertas métricas, pero nunca el contenido de otro usuario sin permiso.
- **Privacidad en integraciones:** Al conectar con servicios externos (Jira, GCal, etc.), se usan **tokens de acceso OAuth** del usuario, y solo se opera en los ámbitos consentidos. Ningún tercero recibe más datos de los necesarios: por ejemplo, si se crea una tarea en Jira, solo se envía el extracto relevante, no toda la conversación. Además, se evaluará la política de datos de servicios como OpenAI (si se usa GPT) – por ejemplo, la API de OpenAI **no utiliza los datos enviados para entrenar** si se lo solicita explícitamente, lo cual habría que habilitar en configuración. Si la empresa o el usuario prefiere, se puede optar por modelos locales (ej. un modelo LLM on-premise) para que ninguna información sensible salga del ecosistema controlado.
- **Logs y monitoreo seguros:** Los registros del sistema (logs de backend, etc.) podrían contener fragmentos de datos (por ejemplo, errores o tiempos de procesamiento). Se configura el logging para **no volcar información sensible** o PII innecesariamente. Además, todos los logs y métricas se almacenan de forma segura y acceso limitado, eliminándolos periódicamente.
- **Cumplimiento normativo:** Desde el inicio se considera el cumplimiento de regulaciones como **GDPR** (en la UE) u otras leyes de protección de datos. Por ejemplo, la app debe obtener consentimiento explícito para procesar datos personales, y ofrecer mecanismos de **exportación y borrado** de los datos del usuario si este lo solicita (derecho al olvido). También se documenta cómo se usan los datos (política de privacidad clara). En cuanto a salud mental, si la app realiza algún tipo

of recolección de estados emocionales, podría considerarse *datos sensibles*, por lo que la protección debe ser aún mayor.

- **Seguridad de la infraestructura:** Los contenedores y servicios se endurecen (hardening) para reducir vulnerabilidades. Se aplican actualizaciones periódicas a componentes como MongoDB, EMQX, etc., y se restringen puertos y accesos innecesarios (p.ej., la base de datos no expone su puerto al exterior, solo la API la consulta). También se integran medidas como **rate limiting** en la API (evitar abuso de peticiones), y autenticación robusta en MQTT (para que solo la app suscrita con credenciales pueda escuchar su tópico).
  En suma, la seguridad y privacidad están integradas en cada capa: desde la **UX (obteniendo consentimiento)**, hasta el **cifrado de datos y control de accesos**, asegurando que el usuario pueda confiar en que su asistente personal de voz verdaderamente es *personal* y seguro.

Iteraciones Futuras del Proyecto

Mirando más allá de la versión actual, existe una ambiciosa hoja de ruta con **nuevas características y mejoras** que enriquecerán al asistente personal de voz:

- **Detección emocional avanzada:** Incorporar análisis de la voz para detectar emociones y estados de ánimo de forma más precisa. Además del sentimiento derivado del texto, analizar elementos paraverbales del audio (tono, ritmo, volumen) para inferir si el hablante está estresado, alegre, cansado, etc. Esto permitiría crear un "diario emocional" y ofrecer retroalimentación al usuario sobre su bienestar mental a lo largo del tiempo. Por ejemplo, alertar si se detecta un tono de ánimo decaído constante, sugiriendo al usuario descansar o buscar apoyo.
- **Control por voz del asistente:** Implementar un modo manos libres total, donde el usuario pueda **invocar y manejar la app mediante comandos de voz**. Por ejemplo: *"Asistente, comienza a grabar"*, *"Asistente, ¿qué pendientes tengo hoy?"*. Esto requerirá integrar un sistema de wake-word (tipo "Hey VoiceBuddy") y procesamiento de comandos en lenguaje natural. Una vez activo, el asistente podría responder por voz utilizando TTS, haciendo la interacción mucho más natural y sin necesidad de mirar la pantalla.
- **Integración con agentes de IA autónomos:** Evolucionar la app de ser un pipeline fijo a incorporar **agentes inteligentes** capaces de tomar decisiones más autónomas. Por ejemplo, tras transcribir una conversación, un agente (estilo AutoGPT adaptado al usuario) podría preguntar: *"Noté que mencionaste un viaje, ¿quieres que busque vuelos?"* y realizar acciones en varios pasos proactivamente. Estos agentes podrían conectarse a APIs diversas según la necesidad, actuando como un **asistente proactivo multi-modal** que no solo reacciona, sino que sugiere y ejecuta tareas complejas encadenadas.
- **Análisis visual y de contexto:** Sumando a la voz, en el futuro se considera utilizar la **cámara u otros sensores** del dispositivo para dar más contexto al asistente. Por ejemplo, reconocimiento de texto en pizarras o pantallas (OCR) durante una reunión para adjuntarlo al resumen, o detección de personas en la sala y reconocimiento facial (respetando privacidad) para saber quién dijo qué. En entornos domésticos, la cámara podría vigilar por seguridad (como se mencionó teóricamente, detectar un intruso y alertar). Integrar esta visión por computador abriría la puerta a funcionalidades como: recordar dónde vio por última vez un objeto el usuario, o complementar las notas de voz con fotos relevantes.
- **Control del hogar e IoT:** Gracias a la integración con MQTT y asistentes, el app podría extenderse a **automatización del hogar inteligente**. Por ejemplo, a través de comandos de voz o detección de eventos, podría encender/apagar luces, ajustar termostato, o leer notificaciones importantes en voz

alta en la casa. Con la capa de n8n/MQTT ya en pie, sería factible conectar dispositivos domóticos (compatibles con MQTT o con IFTTT) para que el asistente personal sea también un mayordomo digital en la vida del usuario.

- **Mejora de la modularidad y extensibilidad del agente:** En futuras versiones, se planea que cada componente de IA sea **intercambiable o actualizable** sin rehacer todo el sistema. Por ejemplo, poder **enchufar diferentes modelos de transcripción** (si aparece uno más liviano o preciso que Whisper) o distintos motores de NLU para la clasificación. La arquitectura basada en contenedores y APIs ya favorece esto; se formalizará en documentación para que otros desarrolladores puedan crear *plugins* o módulos personalizados. Incluso la comunidad podría desarrollar integraciones (p. ej. un módulo de exportar a Notion, o un analizador especializado para conversaciones terapéuticas) que se sumen fácilmente.
- **Interfaces y plataformas adicionales:** Actualmente es una app móvil, pero a futuro se considera ofrecer **una versión de escritorio o web** para acceder a las transcripciones y configuración desde el PC, sincronizando todo vía la nube. También explorar integración con **wearables** (un smartwatch que capture breves notas de voz) o con dispositivos AR/VR donde el asistente pueda acompañar al usuario en realidad mixta. En cuanto a idiomas, extender soporte multilingüe (Whisper ya soporta muchos, pero también ajustar las clasificaciones a diferentes idiomas y culturas).
- **Escalabilidad masiva con contenedores, Kubernetes y Terraform:** A medida que el producto pase de prototipo a servicio comercial con muchos usuarios, se prepara la infraestructura para escalar. Se contempla desplegar todos los componentes en **Kubernetes (K8s)**, lo que permite alta disponibilidad y escalado automático (p. ej., múltiples pods de FastAPI/Whisper según carga, balanceo de carga, etc.). Kubernetes facilitará además separar entornos (dev/staging/prod) y despliegues multi-cloud. Con **Terraform** se manejará la infraestructura como código: aprovisionamiento de máquinas virtuales, clusters K8s, redes, balanceadores, almacenamiento, de forma reproducible y versionada. Esto hará más sencillo crecer (por ejemplo, levantar instancias en nuevas regiones geográficas para reducir latencia a usuarios internacionales, simplemente ajustando el código de Terraform). También se evaluará usar servicios serverless para partes específicas: p. ej., una función Lambda que procese audio corto si conviene costo/rendimiento. En resumen, la visión es una plataforma **escalable y robusta** que pueda servir tanto a 100 usuarios como a 100 mil, manteniendo tiempos de respuesta bajos y confiabilidad.

En conjunto, estas iteraciones futuras apuntan a convertir la app en un **asistente integral, inteligente y ubicuo**, manteniéndose a la vanguardia de la tecnología de voz y de las expectativas de los usuarios en productividad y bienestar.

Indicadores Clave de Éxito (KPIs)

Para evaluar el desempeño de la aplicación y orientar mejoras continuas, se han definido varios KPI enfocados en **rendimiento, precisión y uso**:

- **Tiempo promedio por conversación transcrita:** Este indicador mide la **latencia** desde que el usuario termina de hablar hasta que la transcripción (y acciones asociadas) están disponibles. Un objetivo podría ser, por ejemplo, que una conversación de 1 minuto sea procesada en <10 segundos. Monitorear este tiempo bajo distintas condiciones (longitud del audio, carga del sistema) permite optimizar – por ejemplo, si Whisper tarda demasiado, se consideraría usar un modelo más ligero o más recursos. Un tiempo de respuesta bajo es crucial para la experiencia; este KPI también se evaluará con distintas calidades de red (ver último punto).

- **Precisión del análisis de intención/emoción:** Si bien Whisper ofrece alta exactitud en transcripción, aquí nos centramos en **qué tan bien el sistema interpreta** ese texto para extraer las acciones o emociones correctas. Se puede medir mediante validaciones con usuarios: porcentaje de tareas correctamente identificadas, sentimiento correctamente catalogado vs. evaluación humana, etc. Un KPI podría ser "Precisión de clasificación de categoría >90%" o "Concordancia del análisis emocional con encuestas del usuario". Esto guía mejoras en los modelos NLP; por ejemplo, si se detecta baja precisión en distinguir lo personal de lo laboral, podría ajustarse el prompt de GPT o añadir palabras clave.
- **Frecuencia de uso y retención de usuarios:** Indicadores de **adopción y engagement**. Aquí se consideran métricas como: **DAU/WAU** (usuarios activos diarios/semanales), número promedio de conversaciones grabadas por usuario al día, porcentaje de usuarios que continúan usando la app después de X semanas (retención a 1 mes, 3 meses, etc.). Un alto abandono tempranero indicaría que algo no aporta valor o hay fricción. También se puede rastrear cuántas automatizaciones se activan por usuario (muestra cuánto explotan las integraciones). Estos datos ayudan a iterar en la UX y en educar al usuario sobre las capacidades de la app para aumentar su uso recurrente.
- **Tiempos de respuesta en distintas condiciones de red:** Dado que el uso puede ser móvil, evaluar cómo funciona el sistema con **red celular lenta, intermitente o sin conexión**. KPI ejemplos: "Tiempo de transcripción con 4G vs 3G vs offline (cachear hasta reconexión)". Se buscará que la app degrade elegantemente: por ejemplo, si no hay conexión momentánea, que al menos grabe local y espere a enviar luego (sin perder datos). Medir cuántos eventos offline se logran sincronizar con éxito más tarde, o el tiempo adicional que añade una red lenta, servirá para optimizar (tal vez reduciendo el audio bitrate antes de enviar, etc.). La meta es que el asistente siga siendo útil en movilidad, evitando largas esperas o fallos cuando la conectividad es subóptima.

Además de estos KPIs principales, se monitorizarán otros como **satisfacción del usuario** (medida quizás por NPS o valoraciones internas de "¿fue útil esta recomendación?"), **tasa de error del sistema** (conversaciones que no se pudieron transcribir o procesar correctamente) y **uso de recursos** (costo por usuario, para vigilar sostenibilidad). En conjunto, estos indicadores de éxito permitirán afinar tanto la calidad técnica del asistente personal de voz como su ajuste a las necesidades reales de los usuarios.

Notas de Investigación y Contexto

Para situar esta aplicación en el panorama actual, se recopilaron algunas referencias sobre soluciones similares, requisitos legales y tendencias de mercado:

- **Benchmark con apps o herramientas similares:** Existen productos que tocan aspectos parciales de esta idea. Por ejemplo, la **app oficial de OpenAI (ChatGPT)** ahora permite entrada de voz y conserva el historial de conversaciones, sirviendo como asistente general, aunque no está orientada a registrar automáticamente todo el día ni a integraciones con tareas. Herramientas como **Otter.ai** y **Fireflies.ai** transcriben reuniones y generan resúmenes y tareas, enfocadas al ámbito profesional (reuniones en Zoom, etc.), lo que es cercano a la parte de productividad de nuestra app – aunque típicamente requieren programar la grabación de una reunión específica, no capturan conversaciones espontáneas del entorno. En el espacio de **diarios personales**, destacan aplicaciones como *Rewind.ai* (para Mac), que registra todo lo que el usuario hace en su dispositivo e incluso escucha el micrófono para permitir buscar después cualquier momento; o *Personal.ai*, que crea un modelo personal a partir de tus conversaciones y notas, permitiendo consultarle datos de tu vida. Estas indican un interés por el "**second brain**" impulsado por IA. Ninguna, sin embargo,

combina a la vez **productividad corporativa, registro personal y ángulo de salud mental** como se propone aquí. Esto representa una oportunidad de diferenciación, aunque también significa que la app compite indirectamente con múltiples categorías (asistentes virtuales tipo Siri/Alexa, apps de productividad como Notion o Todoist con voz, e incluso apps de bienestar). Mantenernos al tanto de avances en **tecnología de voz (Whisper, Deepgram)** y en **asistentes inteligentes (Copilots de Microsoft, Rovo de Atlassian)** será clave para integrarse o al menos medirse contra ellos en prestaciones.

- **Requisitos legales y éticos:** Al trabajar con grabación de audio y datos personales, hay varios aspectos legales a cumplir. En muchos países, **grabar conversaciones sin permiso** puede ser ilegal; por tanto, el usuario debe ser consciente de las implicaciones y usar la app de forma responsable. De nuestra parte, incluir términos de uso donde el usuario se compromete a no vulnerar la privacidad de terceros es importante. Además, leyes de privacidad de datos como **GDPR** en Europa imponen obligaciones: por ejemplo, informar claramente qué datos se recopilan (audio, transcripciones, emociones derivadas), con qué propósito y base legal (p. ej., consentimiento del usuario), y cómo puede ejercer derechos sobre ellos (acceso, rectificación, borrado). Si el servicio escala globalmente, habrá que contemplar otras normativas (CCPA en California, LGPD en Brasil, etc.). A nivel ético, manejar con cuidado la **detección emocional** es fundamental: habría que evitar interpretaciones erróneas o acciones no deseadas (imagina que el asistente deduzca algo delicado emocionalmente; debe tratarse con respeto y sin invadir la autonomía del usuario). También, la seguridad es parte de la ética: una filtración de conversaciones privadas sería muy grave, por lo que la arquitectura de seguridad antes descrita no es solo una opción técnica sino una exigencia ética para proteger a los usuarios.
- **Datos de mercado y adopción:** La adopción de asistentes de voz sigue en pleno crecimiento. Por ejemplo, estudios recientes señalan que el número de asistentes de voz en uso a nivel mundial se **duplicó de ~4.2 mil millones en 2020 a ~8.4 mil millones proyectados para 2024** [1], impulsado por la proliferación de smartphones y altavoces inteligentes. Cada vez más personas utilizan la voz para interactuar con sus dispositivos en tareas cotidianas. En el ámbito empresarial, gigantes como Microsoft y Atlassian están invirtiendo en asistentes inteligentes integrados (Copilot, Rovo), lo que valida la necesidad de soluciones que ayuden a manejar la sobrecarga de información. En cuanto a salud mental, la pandemia también vio un auge de apps de bienestar digital; si bien la mayoría son chatbots conversacionales (tipo Wysa, Replika), hay evidencia de que **los asistentes de voz pueden reducir la sensación de soledad y aportar compañía**, especialmente en poblaciones como adultos mayores [2]. Combinando estas tendencias, nuestra app se ubica en la intersección de mercados en crecimiento: *productividad personal (voice productivity)* y *bienestar asistido por IA*. La clave será educar al mercado en que es posible y beneficioso **"grabar tu vida" de forma segura para ayudarte luego**, algo que todavía es novedoso. También debemos observar la aceptación cultural: algunos usuarios pueden sentirse intranquilos siendo grabados incluso con consentimiento; el éxito requerirá generar **confianza** y demostrar valor tangible (¡que vean resultados útiles rápidamente!). En definitiva, el momento parece propicio: la tecnología lo permite, y las necesidades de los usuarios (organización, menos estrés, más apoyo personal) están presentes. Con un posicionamiento claro y cumpliendo con las expectativas de privacidad, este asistente personal de voz tiene el potencial de ganar tracción en un mercado emergente y escalar comercialmente.

```
###   Prompt para Cursor (Versión 1.0 Backend + App Flutter)

Quiero que este archivo represente la versión 1.0 del backend de mi app móvil
```

basada en Flutter para grabar y enviar audios. Esta app se conectará a un backend FastAPI que transcribirá los audios usando Whisper, los clasificará y los enviará a un broker MQTT o n8n para su análisis. El prompt debe:

1. Crear una API REST que reciba archivos de audio y los guarde temporalmente.
2. Llamar a un servicio de transcripción (puede ser un microservicio o script local).
3. Generar un JSON de metadatos: duración, fecha, tono emocional (simulado), etc.
4. Preparar los puntos de integración con n8n vía webhook o MQTT.
5. Incluir comentarios para posibles mejoras (control de privacidad, identificación por usuario, etc.)

Además, sugiero que este archivo tenga tags y se vincule con la visión general del proyecto, como se documenta en Confluence.

---

[1] [2] Older adults' intention to use voice assistants: Usability and emotional needs - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC10663927/