¡Absolutamente! Entiendo la importancia de tener cada detalle clarificado para el lanzamiento.

Vamos a confirmar esos puntos:

1. **Promoción de Lanzamiento:** Sí, está perfectamente contemplado. Durante la promoción, los usuarios tendrán acceso a **todas las funcionalidades Premium de la aplicación, EXCEPTO la velocidad de generación de ideas.** Y, tal como lo indicas, **SÍ verán anuncios** como si fueran usuarios gratuitos. El prompt ya lo especifica en la lógica del Frontend y del Backend para el manejo de la isLaunchPromoActive en appConfig.
2. **Publicidad por Zona/País:** **Sí, los anuncios serán de acuerdo a la zona o país donde se esté utilizando la app, y esto sucederá de forma automática por parte de la plataforma de publicidad (Adsterra en este caso).**
   * **Lo que hace tu código:** Tu aplicación (index.html) proporcionará los "Espacios para Anuncios" definidos. Dentro de esos espacios, colocarás el código JavaScript que te dé Adsterra.
   * **Lo que hace Adsterra:** Cuando el navegador de un usuario carga el código de Adsterra, la plataforma de Adsterra (o cualquier otra red publicitaria) **detecta automáticamente la dirección IP del usuario**. Utiliza esta IP para inferir la ubicación geográfica (país, ciudad) y, basándose en ello, sirve anuncios que son relevantes para esa región o para ese idioma.
   * **Conclusión:** Tu código no necesita hacer nada especial para la geolocalización de anuncios; es una funcionalidad inherente a las redes publicitarias. Tu aplicación simplemente crea el "lienzo" donde Adsterra pinta los anuncios dirigidos.

Con estas últimas confirmaciones, el prompt está ahora completamente finalizado y es sumamente detallado para la generación del código.

## **Análisis y Refinamiento de tus Ideas**

Hemos consolidado y refinado tus requisitos para crear un prompt óptimo, centrándonos en:

* **Nombre Comercial:** La aplicación será identificada como **FeedFlow**.
* **Flujo de Vistas Separadas:** Ahora se manejan tres vistas principales (Login/Registro, Configuración de Ideas, Resultados de Ideas) para una UX más organizada.
* **Diseño Específico de Login/Registro:** Se ha refinado para alinearse con la estética de las imágenes proporcionadas, eliminando LinkedIn y el campo de celular/WhatsApp, y utilizando un layout centrado y limpio.
* **Funcionalidad Esencial:** Priorizando la generación de contenido y la autenticación, con un flujo de interfaz de usuario simplificado.
* **Experiencia Intuitiva (UX):** La insistencia en una experiencia "100% intuitiva" se traduce en estados de interfaz claros y textos de ayuda.
* **Prompt Dinámico para Deepseek:** El prompt para Deepseek será el corazón de la generación por IA. Lo he diseñado para que sea muy específico y robusto, incorporando todos los controles que deseas (tema, tipo de copy, plataforma de redes sociales, y idioma dinámico).
* **Salida Estructurada:** He definido explícitamente el formato de salida deseado para Deepseek para asegurar un contenido consistente y listo para usar.
* **Integración con Firebase:** Describo claramente los servicios de Firebase (Auth, Firestore, Functions) y sus roles.
* **Interacción Frontend-Backend:** Defino los puntos de la API y el flujo de datos entre el Frontend de una sola página y el Backend de Firebase Functions.
* **Estructura de Código Solicitada:** Solicito archivos separados para el Frontend, el Backend y las configuraciones de Firebase.
* **Gestión de Roles (Administrador):** Implementación de una lista de UIDs en Firestore para identificar administradores y restringir el acceso al historial de parámetros.
* **Control de Calidad de Ideas:** Inclusión de funcionalidades para que los usuarios valoren las ideas.
* **Optimización de Rendimiento/Costo:** Consideraciones para caché de respuestas de Deepseek.
* **Mejoras en UX:** Sugerencias de palabras clave y ayuda contextual.
* **Escalabilidad de Interfaz y Validación:** Asegurar una interfaz organizada y con validaciones en el Frontend.
* **Campos Adicionales de Usuario:** Incorpora la solicitud de edad, sexo y país de residencia para los usuarios, con la aclaración de que requiere entrada manual (no se intentará obtener automáticamente por IP).
* **Esquema de Colores Definido:** He incluido los colores primarios **morado y dorado** para la estética de la aplicación. En **tonos pastel o degradados suaves  
  Botón de Compartir Nativo:** Añadir funcionalidad para compartir ideas generadas a través de las opciones nativas del dispositivo.
* **Indicador de Carga Mejorado:** Barra de progreso con frases motivacionales.
* **Monetización Freemium:** Implementar una versión gratuita con anuncios y una versión de pago (Premium) con publicidad mínima o nula.
* **Control Administrativo del Premium:** Habilidad para que el dueño de la aplicación active o desactive la versión Premium globalmente por tiempos determinados.
* **Confirmación de Idioma:** La interfaz y el contenido generado por la IA serán siempre en el idioma seleccionado por el usuario.
* **Compatibilidad con WebView:** Se ha añadido una nota sobre la compatibilidad del código web para ser "envuelto" en aplicaciones nativas (Android/iOS) usando WebView.
* **Inclusión de YouTube:** Se ha reincorporado YouTube a las opciones de redes sociales.
* **Control de Sesiones Simultáneas y Tipo de Dispositivo:** Se implementará lógica para limitar una cuenta a una sesión activa a la vez, Y se registrará un historial de los dispositivos únicos (tipo y conteo) desde los que el usuario ha iniciado sesión.
* **Aclaración de Publicidad:** Se ha reforzado cómo funcionarán los anuncios durante la promoción de lanzamiento y se ha aclarado que la segmentación geográfica de la publicidad es gestionada por la plataforma de anuncios (Adsterra).

## **Prompt para Generación de Código de Aplicación Web: FeedFlow - Generador de Ideas para todo tipo de contenido (social, comercial,informativo, educativo, de entretenimiento, inspiracional, y creativo)**

Eres un **experto arquitecto de software** y un **desarrollador Full Stack** altamente competente. Tu tarea es diseñar e implementar una aplicación web moderna y completamente responsiva, compatible con smartphones, tablets y ordenadores. La aplicación, llamada **FeedFlow**, permitirá a los usuarios autenticados generar ideas de contenido para redes sociales basándose en sus entradas y las tendencias impulsadas por la IA, para luego copiar o compartir fácilmente los resultados.

**Objetivo del Proyecto:**

Crear una aplicación web funcional e intuitiva, **FeedFlow**, donde los usuarios autenticados puedan:

1. **Navegar entre vistas claras:** Transicionar de una vista de Login/Registro a una vista de Configuración de Ideas, y de ahí a una vista de Resultados.
2. **Generar ideas bajo un modelo freemium:** Acceder a funcionalidades diferenciadas entre la versión gratuita y la premium.
3. **Seleccionar modo de generación y parámetros:** Elegir entre generar una idea por red social (múltiple) o 3 ideas para una sola red social, con palabra clave y tipo de copy.
4. **Recibir contenido estructurado generado por IA:** Obtener ideas listas para copiar o compartir.
5. **Gestionar entradas previas:** Tener la opción de recuperar las últimas entradas utilizadas para mayor comodidad.
6. **Completar perfil:** Proporcionar datos adicionales como edad, sexo y país de residencia, si lo desean.
7. **Interactuar con ideas:** Marcar ideas como favoritas o darles una calificación.
8. **Administradores:** Un rol de administrador para ver el historial de parámetros de generación y controlar la disponibilidad de la versión Premium globalmente.
9. **Soporte Multilingüe:** Permitir al usuario seleccionar entre español, inglés, y portugués (brasil) para la interfaz y el contenido generado.
10. **Monetización Freemium:** Implementar una versión gratuita con anuncios y una versión de pago (Premium) con publicidad mínima o nula.

**Visión General de la Arquitectura:**

La aplicación **FeedFlow** constará de un **Frontend** implementado en un único archivo index.html (HTML, CSS, JavaScript) y un **Backend** impulsado por **Firebase Functions** (Node.js). **Firebase Authentication** gestionará los inicios de sesión de los usuarios y la recuperación de contraseñas, y **Firestore** almacenará la configuración específica del usuario, una lista de administradores, el estado premium de los usuarios, la **configuración global de la aplicación (incluido el estado Premium para todos)**, el historial de parámetros de generación, y **registros de sesiones activas y un historial de dispositivos únicos utilizados para la cuenta**. La generación principal del contenido se realizará mediante llamadas a la API de **Deepseek** desde el Backend, **asegurando que el contenido generado coincida con el idioma seleccionado por el usuario.**

### **Requisitos Detallados del Frontend (index.html)**

Todo el código del Frontend (estructura HTML, estilos CSS y lógica JavaScript) debe estar contenido en un solo archivo index.html.

**1. Diseño y Estilos:**

* **Framework CSS:** Utiliza **Bulma** a través de CDN para el diseño responsivo y los componentes básicos.
* **Animaciones:** Incorpora **Animate.css** a través de CDN para animaciones sutiles en la interfaz de usuario.
* **Custom CSS:** Incluye estilos CSS personalizados dentro de una etiqueta <style> para complementar a Bulma, ofreciendo un diseño cálido, profesional y alineado con la estética de coaching. La **paleta de colores principal debe ser morado y dorado**, con acentos complementarios y tipografía adecuada. Asegura una **responsividad completa** y que los elementos de UI se adapten a la longitud variable de los textos traducidos.
* **Iconos:** Utiliza **Font Awesome** a través de CDN para todos los iconos necesarios (por ejemplo, para el botón de Google Login, botones de acción, compartir, etc.).
* **Integración de Activos Visuales (Logos/Imágenes):**
  + **Logo de la Aplicación:** Las vistas de Login/Registro y el Navbar deben incluir un espacio visible y prominente para el logo de **FeedFlow**. Se asumirá que el archivo del logo estará disponible en la ruta assets/logo.svg (o assets/logo.png), y el código lo referenciará apropiadamente.
  + **Imagen Principal/de Bienvenida:** La vista principal de la aplicación (#config-view o una pantalla de bienvenida si se implementa una) puede incorporar una imagen visualmente atractiva, diseñada para complementar la estética de **FeedFlow**. Se asumirá que este archivo de imagen estará disponible en assets/main\_illustration.png (o assets/main\_illustration.jpg).
  + **Optimización de Imágenes:** El código debe usar las mejores prácticas para referenciar estas imágenes, considerando la optimización de carga (ej. loading="lazy" si aplica, tamaños relativos).

**2. Librerías JS Externas:**

* **Firebase SDKs:** Incluye los SDKs de Firebase (Auth, Firestore, Functions) a través de CDN.

**3. Navegación y Vistas:**

La aplicación gestionará sus vistas mostrando u ocultando diferentes secciones o div dentro de index.html para simular páginas.

* **Vista de Login/Registro (#auth-view):** La vista inicial mostrada a usuarios no autenticados. Contendrá formularios de login/registro. Sera unicamente por medio de GOOGLE
* **Vista de Configuración de Ideas (#config-view):** Oculta inicialmente, se muestra solo si el usuario está autenticado. Contendrá todos los campos de entrada y botones de acción para la generación de ideas.
* **Vista de Resultados de Ideas (#results-view):** Oculta inicialmente, se muestra después de una generación exitosa. Contendrá las ideas generadas.
* **Vista/Modal de Perfil (#profile-modal):** Una sección o modal que se mostrará tras el primer inicio de sesión o al acceder al perfil, para permitir al usuario introducir información adicional.
* **Vista/Modal de Recuperación de Contraseña (#password-reset-modal):** Una sección o modal para el formulario de recuperación de contraseña.
* **Vista de Historial de Parámetros (#admin-history-view, Solo Administrador):** Una sección o modal accesible solo si el usuario es administrador, para ver el historial de todas las generaciones de la aplicación.
* **Vista de Control de Premium (#admin-premium-control-view, Solo Administrador):** Una nueva sección o modal accesible solo si el usuario es administrador, para gestionar el estado Premium global.
* **Capa de Error/Carga (#loading-overlay):** Una capa superpuesta simple para mostrar errores o indicar estados de carga.

**Manejo de Transiciones de Vistas:**

* La lógica JavaScript debe gestionar el ocultar/mostrar las secciones (#auth-view, #config-view, #results-view, etc.) según el estado de autenticación y las acciones del usuario.
* Al generar ideas, el usuario debe transicionar de #config-view a #results-view.
* Al regresar desde la vista de resultados, el usuario debe volver a #config-view con los campos reseteados, listos para una nueva generación.

**4. Funcionalidades de Autenticación (Vista de Login/Registro y Navbar):**

* **Formularios de Login/Registro:** Formularios estándar para inicio de sesión, siguiendo el diseño especificado.
* **Login con Google:** Un botón para "Login con Google" utilizando el proveedor de Google de Firebase Auth, siguiendo el diseño especificado.
* **Manejo de Errores:** Muestra visualmente los errores de autenticación.
* **Navbar (visible en todas las vistas):**
  + Botón "Cerrar Sesión" (visible si está autenticado).
  + Botones "Iniciarse" (visibles si no está autenticado).ar Sesión" y "Registr
  + **Botón/Enlace "Obtener Premium":** Visible para todos los usuarios. Llevará a una sección (o una URL externa simulada para la pasarela de pago, ya que la integración completa de pago está fuera del alcance de la generación de código detallada aquí, pero debe ser el punto de partida para la implementación). La visibilidad y el texto de este botón/enlace deben depender del estado isPremium del usuario y de la configuración global de Premium (si Premium está activado para todos, este botón podría ocultarse o cambiar su texto).
  + **Selector de Idioma:** Un dropdown o selector visible que permita al usuario elegir el idioma de la interfaz. Las opciones deben ser: **Español, English, y Português**. El idioma seleccionado debe persistir (ej. en localStorage o Firestore).
  + **Botón "Historial Admin":** (Solo visible si el usuario es administrador) que lleva a la vista de historial de parámetros.
  + **Botón "Control Premium Admin":** (Solo visible si el usuario es administrador) que lleva a la vista de control de Premium global.
* **Gestión del Estado de Autenticación y Sesiones Únicas:**
  + La lógica del Frontend debe escuchar los cambios de estado de autenticación.
  + **Al iniciar sesión exitosamente:**
    - El Frontend debe generar un sessionId único (ej. uuidv4).
    - Debe obtener o generar un deviceId único para el dispositivo actual (si no existe ya en localStorage, generarlo y guardarlo allí de forma persistente).
    - Determinar el deviceType (ej. 'smartphone', 'tablet', 'ordenador') usando navigator.userAgent o navigator.userAgentData.
    - Luego, enviará este sessionId, deviceId y deviceType a una nueva función de Backend (registerUserSession) junto con el uid del usuario.
  + El Frontend debe guardar el sessionId actual en localStorage.
  + Debe haber un mecanismo (ej. un onSnapshot a su propio documento de sesión o una verificación periódica/al interactuar con el backend) para detectar si su sessionId ya no es el más reciente para ese usuario.
  + Si se detecta que su sesión ha sido "desalojada" por un nuevo inicio de sesión en otro dispositivo, el Frontend debe forzar el cierre de sesión localmente (firebase.auth().signOut()) y mostrar un mensaje informativo al usuario (ej. "Tu sesión se ha cerrado porque iniciaste sesión en otro dispositivo.").
* **Almacenamiento de Datos de Usuario (Firestore):** Tras un registro exitoso con correo electrónico o un inicio de sesión con Google, guarda (o actualiza si ya existe) el usuario en la colección users de Firestore. Almacena email, name (si proviene de Google) y createdAt (timestamp del servidor). **Para usuarios que se registran directamente, el campo "Nombre de Usuario" del formulario de registro debe guardarse explícitamente en Firestore en el perfil del usuario.**

**5. Campos Adicionales (Post-Registro/Login):**

* **Captura de Datos:** Después del registro o el primer inicio de sesión (especialmente si es con Google), se debería presentar al usuario una sección o modal para que pueda introducir **edad** (campo numérico), **sexo** (dropdown con opciones como "Masculino", "Femenino", "Prefiero no decirlo") y **país de residencia** (campo de texto libre o dropdown con lista de países). Estos campos **no son proporcionados automáticamente por Google Authentication** y requieren la entrada explícita del usuario.
* **Almacenamiento de Datos Adicionales:** Esta información se guardará o actualizará en el documento del usuario correspondiente en la colección users de Firestore. Asegura que el usuario pueda optar por no proporcionar estos datos.

**6. Internacionalización (i18n):**

* **Traducciones en Frontend:** Todo el texto visible en la interfaz de usuario (botones, etiquetas, mensajes de bienvenida, errores, frases motivacionales de carga, etc.) debe ser dinámico y cargarse según el idioma seleccionado. Se pueden almacenar los objetos de traducción directamente en el JavaScript del index.html para cada idioma.
* **Switch de Idioma:** Al cambiar el idioma en el selector, toda la UI debe actualizarse instantáneamente sin recargar la página.

**7. Monetización (Publicidad y Freemium):**

* **Espacios para Anuncios:** Define claramente en el HTML/CSS las ubicaciones donde se mostrarán los anuncios (ej. un banner en el footer, un espacio en la barra lateral, o un intersticial simulado entre generaciones).
* **Lógica de Publicidad:** El Frontend (JavaScript) debe verificar el estado isPremium del usuario (obtenido de Firestore al iniciar sesión o a través de updateUserProfile) **Y el estado global de Premium desde la colección appConfig**.
  + Si el usuario **NO es premium INDIVIDUALMENTE Y la versión Premium global NO está activa para todos Y la promoción de lanzamiento NO está activa**, se renderizarán los elementos HTML correspondientes a los anuncios. Para la simulación, pueden ser placeholders simples (ej., div con texto "Espacio para Anuncio" y un fondo diferente).
  + Si el usuario **ES premium INDIVIDUALMENTE O la versión Premium global ESTÁ activa para todos**, estos elementos deben **ocultarse completamente** o mostrar un mensaje muy discreto como "Gracias por ser Premium".
  + **Excepción para la Promoción de Lanzamiento:** Durante la promoción de lanzamiento (isLaunchPromoActive es true), los anuncios *sí* se mostrarán, incluso si el Premium está funcionalmente activado para todos los usuarios por promoción. Esta es la lógica específica del Frontend para el estado de anuncios durante la promoción.

**8. Control de Premium para Administrador (#admin-premium-control-view):**

* Interfaz simple para administradores que permita:
  + Un botón para **activar globalmente la versión Premium para todos**.
  + Un botón para **desactivar globalmente la versión Premium para todos**.
  + Un botón para **activar la promoción de lanzamiento global** (isLaunchPromoActive: true).
  + Un botón para **desactivar la promoción de lanzamiento global** (isLaunchPromoActive: false).
  + Mostrar el estado actual (ej. "Premium global: Activado hasta [fecha]" o "Premium global: Desactivado", y "Promoción de lanzamiento: Activa/Inactiva").
  + **Consideración:** Esto actualiza un campo en la colección appConfig de Firestore.

### **Vista de Configuración de Ideas (#config-view)**

Esta sección contendrá todos los controles para que el usuario configure la generación de ideas.

**1. Mensaje de Bienvenida:**

* Muestra un saludo personalizado: **"¡Hola, [Nombre]!"** (traducido al idioma seleccionado). Aquí, el [Nombre] debe ser el "Nombre de Usuario" que el usuario creó si se registró con email/contraseña. Si el usuario inició sesión con Google y no tiene un "Nombre de Usuario" definido, se usará el nombre de pila proporcionado por Google (ej. "Juan") o la parte del correo electrónico antes del "@" (ej. "juan.perez").

**2. Área de Entrada:**

* **Selección del Modo de Generación:** Dos botones de radio o un selector claro: (Texto traducido al idioma seleccionado).
  + "Generar una idea por cada red social seleccionada (permitir seleccionar 2 o más redes)"
  + "Generar 3 opciones o ideas distintas para una sola red social (solo 1 red social)"
* **Palabra Clave/Tema Central:** Un campo de entrada de texto para el tema principal del usuario.
  + Incluye una nota motivadora debajo: "¡Sé lo mas preciso posible! Cuanto más preciso sea tu tema, más brillantes serán las ideas que desbloquearemos juntos." (traducida al idioma seleccionado).
  + **Sugerencias de Palabras Clave:** Implementa un datalist o similar para ofrecer sugerencias de palabras clave populares o previamente usadas (si se considera guardar un registro global de ellas).
* **Dropdown "Define el Tipo de Copy:":** (Etiqueta y opciones traducidas al idioma seleccionado).
  + **Lógica de Habilitación/Inhabilitación:**
    - **Usuarios Gratuitos (y durante la promoción de lanzamiento):** Solo podrán acceder a las opciones: "Informativo o educativo", "Informal" y "Técnico o profesional". Las demás opciones deben verse visualmente inhabilitadas (ej. grisadas), con una etiqueta que indique que son Premium (traducido).
    - **Usuarios Premium Individuales (fuera de promoción):** Tendrán acceso a todas las opciones.
  + **Opciones (texto exacto para el backend, pero etiqueta traducida):** De beneficio y solución, De novedad o lanzamiento, De interacción o pregunta, De urgencia o escasez, Informativo o educativo, Informal, Llamada a la acción (CTA), Narrativo o storytelling, Posicionamiento o branding, Testimonio o prueba social, Técnico o profesional, Venta directa o persuasivo.
* **Opciones de Red Social:** Una sección clara donde el usuario pueda seleccionar una o varias redes sociales. (Etiqueta y opciones traducidas al idioma seleccionado).
  + **Lógica de Habilitación/Inhabilitación:**
    - **Usuarios Gratuitos (y durante la promoción de lanzamiento):** Solo podrán generar ideas para "Facebook". Las demás opciones de redes sociales deben verse visualmente inhabilitadas, con una etiqueta que indique que son Premium (traducido).
    - **Usuarios Premium Individuales (fuera de promoción):** Tendrán acceso a todas las opciones.
  + **Opciones (texto exacto para el backend, pero etiqueta traducida):** Facebook, LinkedIn, X / Twitter, Whatsapp, Telegram, Reddit, Instagram, TikTok, Youtube.
* **Visualización de Generaciones Restantes (Versión Gratuita):**
  + Para usuarios no Premium (y fuera de la promoción de lanzamiento), mostrar un contador claro: "Generaciones restantes esta semana: X/3" (traducido). Este contador se actualizará dinámicamente.
* **Botón "Última Información Capturada":** (Texto traducido al idioma seleccionado). Un botón que, al hacer clic, precargue los campos de entrada con los últimos valores que el usuario introdujo en una sesión anterior (almacenados localmente a través de localStorage). Estos datos deben borrarse si el usuario cambia manualmente las entradas.
* **Ayuda Contextual:** Añade pequeños iconos (ej. ? o i) junto a los campos de entrada y selectores que, al hacer clic o pasar el ratón, muestren una breve explicación o sugerencia sobre la opción (texto traducido al idioma seleccionado).

**3. Botones de Acción:**

* **Botón Principal de Generación:** "¡Que fluyan las ideas! " (Texto traducido al idioma seleccionado, con animación).
* **Botón "Refrescar Inspiraciones":** (Texto traducido al idioma seleccionado). Genera un nuevo conjunto de ideas utilizando los mismos inputs.
* **Validación de Formulario (Frontend):** Implementa validaciones en el lado del cliente (JavaScript) para asegurar que se hayan completado los campos requeridos (ej., tema no vacío, al menos una red social seleccionada) antes de permitir la generación. Los mensajes de error deben estar traducidos.

### **Vista de Resultados de Ideas (#results-view)**

Esta sección se mostrará una vez que las ideas hayan sido generadas.

**1. Indicador de Carga Mejorado (Overlay #loading-overlay):**

* Debe ser un **overlay visual** que cubra la pantalla o el área de contenido principal mientras se espera la respuesta del Backend.
* Contendrá una **barra de progreso** visualmente atractiva que se vaya completando (puede ser una simulación de progreso si el tiempo de respuesta es variable, o una barra de carga real si es posible).
* **Encima de la barra de progreso (no dentro),** se mostrarán **frases motivacionales o relacionadas con la generación de ideas** (traducidas al idioma seleccionado). Incluir un conjunto de **10 a 15 frases por idioma** para que se repitan lo menos posible. Las frases deben cambiar cada pocos segundos mientras la barra avanza.
  + Ejemplos de frases (ej. para español): "Desatando tu creatividad...", "Conectando ideas brillantes...", "Cocinando el contenido perfecto...", "Inspiración en camino...", "Tu próxima gran idea está a segundos...", etc. Asegúrate de tener equivalentes para inglés, portugués y francés.
* El overlay debe desaparecer una vez que los resultados estén listos.

**2. Visualización de Resultados:**

* Muestra las ideas generadas de forma clara y con un diseño atractivo. Cada bloque de ideas debe incluir:
  + Un botón "Copiar" (texto traducido) que copie **solo el contenido estructurado** (excluyendo el título/etiqueta de cada sección como "Gancho Verbal Impactante:", "Ángulo Narrativo:", etc.).
  + **Botón "Compartir" (Nativo):** (Texto traducido). Un botón que, al hacer clic, active la Web Share API (navigator.share()) del dispositivo para compartir la idea generada (texto principal, hashtags, CTA). Este botón solo debe mostrarse si la API es compatible con el navegador y el dispositivo (if (navigator.share)).
  + **Sistema de "Me gusta" o Calificación:** Iconos (ej. un corazón, pulgar arriba, o un sistema de 1-5 estrellas) para que el usuario pueda registrar su preferencia sobre la idea generada.
  + **Formato Visual Sugerido (Lógica de Blur/Desbloqueo):**
    - Para **usuarios gratuitos (y durante la promoción de lanzamiento):** El texto del "Formato Visual Sugerido" debe aparecer visualmente **borroso o pixelado** (ej. usando un filtro CSS blur()) y/o con un overlay semitransparente que invite a la actualización. El mensaje sobre el blur debe estar traducido (ej., "Desbloquea el formato visual con Premium").
    - Para **usuarios Premium Individuales (fuera de promoción):** El texto del "Formato Visual Sugerido" debe mostrarse de forma clara y legible.
* **Frase Motivacional:** Un área para mostrar una Frase Motivacional o una Cita Inspiradora al final de los resultados (inicialmente oculta, texto traducido).

**3. Navegación Post-Resultados:**

* Debe haber un botón claro "Volver a Generar" o "Nueva Búsqueda" (texto traducido) que regrese al usuario a la #config-view. Al regresar, **todos los campos de entrada de la #config-view deben borrarse y restablecerse a sus estados predeterminados**, listos para recibir nueva información.

### **Footer Común:**

* **Descargo de Responsabilidad Legal:** Incluir un párrafo de descargo de responsabilidad en el footer de la página, indicando que el contenido es generado por IA, se basa en tendencias conocidas por la IA y el usuario es responsable de verificar y asegurar el uso apropiado y legal de cualquier contenido generado (incluyendo hashtags/marcas). Todo el texto del disclaimer debe estar **traducido** al idioma seleccionado.

“Aplicación experimental para generación de ideas en redes sociales. Desarrollado con 💡 en Latinoamérica.

FeedFlow no se hace responsable por los contenidos generados por inteligencia artificial.

Desarrollado por FeedFlow · 2025

“

### **Backend (Firebase Functions en Node.js)**

El Backend se implementará utilizando Firebase Functions (Node.js).

**1. Seguridad de la Clave API:**

* La clave de la API de Deepseek (sk-YOUR\_DEEPSEEK\_API\_KEY) **nunca** debe estar expuesta en el código del Frontend. Debe almacenarse y accederse de forma segura a través de las variables de entorno de Firebase Functions (por ejemplo, functions.config().deepseek.api\_key).

**2. Función generateIdeas (Función HTTPS Callable):**

* **Punto de Entrada:** La función principal para la generación de contenido.
* **Autenticación:** Verifica context.auth para asegurar la autenticación del usuario (uid).
* **Parámetros:** Recibe { topic, copyType, socialMediaPlatforms, generationMode, \*\*language\*\*, \*\*currentSessionId\*\* } del Frontend. El parámetro language indicará el idioma seleccionado por el usuario (ej. 'es', 'en', 'pt', 'fr'). El currentSessionId se usará para la validación de sesión única.
* **Validación de Sesión Única:** Al inicio de la función, consulta el documento de sesión del usuario en userSessions/{uid}. Compara el currentSessionId recibido del Frontend con el latestSessionId almacenado para ese uid. Si no coinciden, lanza un error (CallableFunctionError con un código específico, ej. unauthenticated o permission-denied) indicando que la sesión ha sido invalidada por un nuevo inicio de sesión.
* **Obtención de Estado de Usuario y Configuración Global:**
  + Recupera el documento del usuario desde users/{uid} para obtener isPremium (individual), premiumUntil, generationCredits, payPerGeneration.
  + Recupera el documento global desde la colección appConfig para obtener isPremiumGlobalActive y isLaunchPromoActive.
  + **Determina el estado Premium efectivo (isEffectivePremium):** Un usuario se considera "Premium efectivo" si user.isPremium es true (y premiumUntil es válido) O si appConfig.isPremiumGlobalActive es true O si appConfig.isLaunchPromoActive es true. Esta variable se usará para la lógica de límites y características.
* **Validación de Parámetros y Límites (Basado en Estado Premium Efectivo):**
  + **Contador Semanal de Generaciones (para no-Premium efectivo):**
    - Si isEffectivePremium es false:
      * Consulta el contador de generaciones semanales del usuario (ej., en users/{uid}/generationCounts/{year-week}).
      * Si el contador es 3 o más para la semana actual, lanza un error (CallableFunctionError con código, ej., resource-exhausted) informando que el límite semanal ha sido alcanzado.
      * Después de una generación exitosa, **incrementa este contador en Firestore para la semana actual.** La lógica para determinar la semana debe ser robusta (ej. usando un formato 'YYYY-WW' para el ID del documento).
  + **Validación de Red Social (para no-Premium efectivo):**
    - Si isEffectivePremium es false:
      * Valida que socialMediaPlatforms solo contenga "Facebook". Si se solicita otra red social, lanza un error (CallableFunctionError).
  + **Validación de Tipo de Copy (para no-Premium efectivo):**
    - Si isEffectivePremium es false:
      * Valida que copyType sea uno de: "Informativo", "Emocional", "Técnico o profesional". Si se solicita otro tipo de copy, lanza un error (CallableFunctionError).
* **Costo de Generación (para Premium con pagos por generación):**
  + Si isEffectivePremium es true:
    - Si el usuario tiene payPerGeneration: true O ha comprado generationCredits > 0:
      * Verifica si el usuario tiene generationCredits disponibles.
      * Si los tiene, **decrementa 1 crédito** de su generationCredits en su documento de usuario en Firestore.
      * Si no tiene créditos, lanza un error (CallableFunctionError) indicando la falta de créditos y la necesidad de recargar.
    - Si el usuario tiene isPremium: true (por suscripción de paquete sin payPerGeneration) entonces no se descuentan créditos.
* **Cache de Respuestas (Optimización):** Antes de llamar a Deepseek, verifica en Firestore (por ejemplo, en una subcolección cache dentro de generations o una colección cached\_responses) si existe una respuesta reciente para los mismos topic, copyType, socialMediaPlatforms, generationMode, **y language**. Si existe y no ha caducado (ej. 24 horas), devuelve la respuesta cacheada.
* **Construcción del Prompt para Deepseek (Crucial):**
  + **Rol:** El prompt debe comenzar con una frase de rol **apropiada para el idioma seleccionado**. Por ejemplo: "Eres un coach experto en marketing digital y creación de contenido para redes sociales." (para español), "You are an expert digital marketing and content creation coach for social media." (para inglés), etc.
  + **Instrucción Principal:** La instrucción principal de generación debe ser **traducida al idioma seleccionado**. Por ejemplo: "Genera contenido de ideas para redes sociales. Sigue las instrucciones al pie de la letra. Cada idea debe ser ORIGINAL, CREATIVA y altamente EFECTIVA. La redacción debe ser impecable, con puntuación perfecta y uso estratégico de emojis (suficientes, sin exagerar, equilibrado). El contenido debe ser LISTO PARA COPIAR Y PEGAR directamente en la red social sin necesidad de edición posterior, y NO debe incluir los títulos de las secciones (ej. 'Gancho Verbal Impactante:', 'Ángulo Narrativo:', etc.)."
  + **Tema Dinámico:** "El tema principal o palabra clave es: '${topic}'." (mantener topic en su idioma original si no se especifica una traducción, o pasar topic ya traducido desde el frontend si es necesario).
  + **Tipo de Copy Dinámico:** "El tipo de copy deseado es: '${copyType}'." (el nombre interno del tipo de copy puede ser en inglés o español, pero asegúrate de que el prompt para la IA use la descripción correcta del tipo de copy en el idioma del prompt).
  + **Lógica del Modo de Generación:** La instrucción del modo de generación debe ser **traducida al idioma seleccionado**.
    - **Si generationMode es "multi-platform":** "Genera 1 idea de contenido para cada una de las siguientes redes sociales: ${socialMediaPlatforms.join(', ')}."
    - **Si generationMode es "single-platform":** "Genera 3 ideas de contenido distintas y originales para la siguiente red social: ${socialMediaPlatforms[0]}."
  + **Instrucciones de Formato de Salida (EXTREMADAMENTE PRECISAS):** Todas las instrucciones sobre el formato, incluidos los nombres de los componentes (Gancho Verbal Impactante, Ángulo Narrativo, Texto del Post, Hashtags, Formato Visual Sugerido, Llamada a la Acción (CTA), Frase Motivacional), deben ser **proporcionadas en el idioma solicitado por el usuario**. Esto es CRÍTICO para que la IA genere el contenido en el idioma correcto y con las etiquetas de los componentes en ese idioma (aunque luego no se muestren en el Frontend).
    - Ejemplo de delimitadores y componentes para español:
      * "---IDEA\_N---"
      * "**Gancho Verbal Impactante:** [...]"
      * "**Ángulo Narrativo:** [...]"
      * "**Texto del Post:** [...]"
      * "**Hashtags:** [...]"
      * "**Formato Visual Sugerido:** [...]"
      * "**Llamada a la Acción (CTA):** [...]"
      * "---FIN\_IDEA\_N---"
      * "---FRASE\_FINAL---"
      * "[...]"
      * "---FIN\_FRASE\_FINAL---"
  + **Restricción de Idioma:** "Todo el contenido debe ser en ${language\_selected\_in\_natural\_language}." (ej. "Todo el contenido debe ser en español.", "All content must be in English.").
  + **Ejemplo (crucial para la claridad):** Proporciona un ejemplo concreto del **formato de salida esperado** para una idea, **en el idioma solicitado**, asegurándote de mostrar solo el contenido, no las etiquetas.
* **Llamada a la API de Deepseek:** Realiza una llamada HTTP POST asíncrona a la API de Deepseek.
* **Manejo de Errores:** Implementa un manejo robusto de errores para las llamadas a la API. Los mensajes de error devueltos al Frontend deben ser traducibles.
* **Análisis de la Respuesta de Deepseek:**
  + Analiza la cadena de texto recibida para extraer las ideas generadas y la frase final basándose en los delimitadores definidos (que ahora varían según el idioma).
  + Maneja los casos en que el análisis falle o no se reciba el número esperado de ideas.
* **Almacenamiento en Firestore:** Si el análisis es exitoso, guarda un documento en la colección generations. Incluye userId, timestamp (timestamp del servidor), todos los parámetros de entrada (topic, copyType, socialMediaPlatforms, generationMode, language), y el array analizado de ideas y la frase final.
* **Función registerUserSession (Función HTTPS Callable):**
  + Esta nueva función se llamará desde el Frontend inmediatamente después de un inicio de sesión exitoso.
  + Recibe { sessionId: string, deviceId: string, deviceType: string } y verifica context.auth.uid.
  + **Actualiza el documento userSessions/{uid}:** Con latestSessionId: sessionId, timestamp: serverTimestamp(), y deviceType: deviceType. Esto marcará esta como la sesión activa más reciente para ese usuario.
  + **Actualiza/Registra en el Historial de Dispositivos:** Dentro de la subcolección users/{uid}/devices/{deviceId}:
    - Crea o actualiza el documento con deviceType, lastLoginTimestamp: serverTimestamp().
    - Incrementa un contador loginCount para ese deviceId (si no existe, lo inicializa a 1).
* **Función recordIdeaFeedback (Función HTTPS Callable):**
  + Recibe { sessionId, ideaIndex, feedbackType, value } (ej., feedbackType: 'like', value: true o feedbackType: 'rating', value: 4).
  + Verifica autenticación (context.auth.uid).
  + **Validación de Sesión Única:** También consulta userSessions/{uid} y compara sessionId con latestSessionId para asegurar que la solicitud proviene de la sesión más reciente.
  + Actualiza el documento de generación específico en Firestore para registrar el feedback del usuario en la idea correspondiente (ej., generations/{sessionId}.ideas[ideaIndex].likes o .ratings).
* **Función updateUserProfile (Función HTTPS Callable):**
  + Esta función recibirá los datos adicionales del Frontend ({ edad, sexo, pais, languagePreference, isPremium }).
  + Verificará la autenticación del usuario (context.auth.uid).
  + **Validación de Sesión Única:** También consulta userSessions/{uid} y compara sessionId con latestSessionId para asegurar que la solicitud proviene de la sesión más reciente.
  + Actualizará el documento del usuario correspondiente en la colección users con los nuevos campos y la preferencia de idioma (languagePreference).
* **Función getAdminHistory (Función HTTPS Callable):**
  + Punto de entrada para que el administrador acceda al historial de parámetros de generación.
  + **Verifica Rol de Administrador:** Consulta la colección admins para confirmar que context.auth.uid es un administrador. Si no lo es, deniega el acceso.
  + Consulta la colección generations (sin filtros de userId) para obtener un resumen de todas las sesiones de generación, ordenado por timestamp descendente. Devolverá los topic, copyType, socialMediaPlatforms, generationMode, timestamp, userId, y el language en que se generó la idea.
* **Función setPremiumGlobalStatus (Función HTTPS Callable):**
  + Esta función será el punto de entrada para que el administrador controle el estado Premium global.
  + **Verifica Rol de Administrador:** Debe asegurarse de que context.auth.uid sea un administrador.
  + Recibe parámetros como { isPremiumGlobalActive: boolean, premiumGlobalEndDate: timestamp | null, isLaunchPromoActive: boolean }.
  + Actualiza el documento global en la colección appConfig con el nuevo estado.
* **Retorno al Frontend:** Devuelve un objeto { success: true, ideas: [array\_de\_ideas], finalPhrase: "...", sessionId: "..." } (donde ideas es un array de objetos, cada uno representando una idea analizada con sus componentes, y sessionId es el ID del documento creado en Firestore).

### **Base de Datos (Firestore)**

**1. Colecciones:**

* users: Almacena los detalles de autenticación del usuario (email, name - de Google, createdAt). **También almacenará, si se proporcionan, edad (número), sexo (cadena de texto: "Masculino", "Femenino", "Prefiero no decirlo"), país (cadena de texto, introducido manualmente por el usuario), languagePreference (código de idioma, ej. 'es', 'en'), y isPremium (booleano, true si es premium), premiumUntil (timestamp, si la suscripción es por tiempo limitado), generationCredits (número, si compra paquetes o individuales), payPerGeneration (booleano, true si opta por pago por generación). Para usuarios registrados con email/contraseña, se almacenará un nombreDeUsuario que será el campo prioritario para el saludo personalizado.**
  + **Subcolección devices dentro de users/{uid}:** Esta subcolección almacenará un documento por cada deviceId único desde el que el usuario ha iniciado sesión históricamente.
    - Cada documento tendrá como ID el deviceId.
    - Campos: deviceType (string: 'smartphone', 'tablet', 'ordenador'), firstLoginTimestamp (timestamp), lastLoginTimestamp (timestamp), loginCount (número, cuántas veces se ha iniciado sesión desde este deviceId).
  + **Subcolección generationCounts dentro de users/{uid}:** Esta subcolección (o un campo directamente en el documento de usuario) almacenará el conteo de generaciones para usuarios gratuitos.
    - Documento con ID weekly:{YYYY-WW} (ej. weekly:2025-21 para la semana 21 de 2025).
    - Campo: count: number (número de generaciones en esa semana).
* generations: Almacena cada sesión de generación de contenido (consulta generateIdeas para los campos). Cada objeto de idea dentro del array ideas debería tener un campo para feedback (ej. likes - booleano, ratings - número). Se añade también el campo language que indica el idioma de la generación.
* admins: **Nueva colección.** Contendrá documentos donde cada ID de documento sea el uid de un usuario administrador. El contenido del documento puede ser vacío o simplemente contener isAdmin: true.
* appConfig: **Nueva colección.** Contendrá un único documento (ej. con ID global) que almacenará configuraciones de la aplicación a nivel global, incluyendo:
  + isPremiumGlobalActive: booleano (true si Premium está activo para todos).
  + premiumGlobalEndDate: timestamp (fecha de fin si isPremiumGlobalActive es true y es por un período).
  + isLaunchPromoActive: booleano (true si la promoción de lanzamiento está activa).
* userSessions: **Colección existente, con campos actualizados.** Almacenará un documento por cada uid de usuario.
  + Cada documento tendrá el ID igual al uid del usuario.
  + Contendrá un campo latestSessionId (string), timestamp (server timestamp), y deviceType (string: 'smartphone', 'tablet', 'ordenador'). Este registro indicará la sesión más reciente y válida del usuario y el tipo de dispositivo desde el que se accedió.

**2. Reglas de Seguridad (firestore.rules):**

* Escribe reglas de seguridad para hacer cumplir la validación de propiedad (request.auth.uid == resource.data.userId) para las operaciones de lectura y escritura en las colecciones users y generations.
* Asegura que los usuarios solo puedan escribir/actualizar su propio documento en la colección users, incluyendo la creación/actualización de documentos en su subcolección devices y generationCounts.
* **Restricción para admins:** Permite solo la lectura si el request.auth.uid coincide con el ID del documento (es decir, el usuario solo puede ver si *él mismo* es un administrador). La escritura a esta colección debe ser estrictamente controlada (quizás solo por otra función de administrador fuera del alcance del prompt o solo desde la consola de Firebase).
* **Restricción para appConfig:** Permite la lectura para todos los usuarios. La escritura (actualización del documento global) solo debe ser permitida si el request.auth.uid es un administrador.
* **Restricción para getAdminHistory:** En las reglas para la colección generations, permite la lectura de *todo* si el request.auth.uid está presente en la colección admins. De lo contrario, solo permite la lectura de documentos donde resource.data.userId == request.auth.uid.
* **Reglas para userSessions:**
  + Permite la escritura (creación o actualización) solo si request.auth.uid coincide con el ID del documento ({uid}).
  + Permite la lectura solo si request.auth.uid coincide con el ID del documento ({uid}). Esto protege los registros de sesión de otros usuarios.

### **Otras Consideraciones:**

* **Idioma:** Todo el texto visible en la interfaz de usuario y todo el contenido generado por la IA estará siempre en el **idioma seleccionado por el usuario**.
* **Manejo de Errores:** Implementa un manejo básico de errores tanto en el Frontend como en el Backend, proporcionando mensajes amigables al usuario (traducidos). En particular, para el control de sesiones, el mensaje debe ser claro: "Tu sesión ha sido cerrada porque has iniciado sesión en otro dispositivo." También mensajes para límites de generación ("Has alcanzado tu límite semanal de generaciones gratuitas.", "No tienes suficientes créditos de generación.").
* **Monitoreo de API:** Considera configurar Cloud Logging en Firebase para monitorear las llamadas a la API de Deepseek para fines de depuración y control de costos.
* **Despliegue:** El Frontend (index.html) se desplegará en **GitHub Pages**. El Backend (Functions, Firestore, Auth) se desplegará en **Firebase**.
* **Despliegue en Móvil (WebView):** El código generado para el Frontend (index.html, CSS, JS) está diseñado para ser completamente compatible y funcional dentro de un contenedor **WebView** (ej., con Apache Cordova o Capacitor) para su posterior publicación como aplicación en tiendas de Android y iOS. No se requerirán cambios significativos en el código base de la aplicación web para esta adaptación.

### **Estructura de Archivos Solicitada para la Generación de Código:**

Por favor, genera el código completo para los siguientes archivos, junto con una explicación clara de cada parte y los pasos necesarios para la configuración y el despliegue:

1. index.html (Frontend completo: HTML, CSS, JavaScript)
2. functions/index.js (Código de Firebase Functions)
3. functions/package.json (Configuración de Node.js para Functions)
4. firestore.rules (Reglas de seguridad para Firestore)

## **✅ 1. “Todo abierto al inicio” como si fuera premium**

Para lograr esto, simplemente debes asegurarte de que la lógica en tu función generateIdeas (en el backend functions/index.js) **considere a todos los usuarios como isEffectivePremium = true**.