Prueba Fase 2: Trainee de GenAI y Automatizaciones

¡Felicitaciones por pasar a la segunda fase del caso! 🥳

Un Mensaje Clave para esta Fase

El objetivo principal **NO** es construir un *frontend* visualmente perfecto. El *core* de este desafío (y del rol) está en el **backend** (Cloud Run, Firestore) y, lo más importante, en la **integración E2E (End-to-End)**.

Queremos ver que puedes "conectar los cables" entre un cliente y el *backend*, ya que es un desafío clave en nuestros proyectos de automatización (donde a menudo conectamos *backends* de IA con *frontends* LCNC como Lovable o Bolt).

Enfoca tu tiempo en la funcionalidad y la integración, no en el CSS.

Tu misión es construir el MVP que diseñaste, enfocándote en el **Rol Proveedor**. Deberás entregar un repositorio de GitHub con tu solución.

Recordemos el contexto del caso

Objetivo: Desarrollar el MVP (Producto Mínimo Viable) de un nuevo **Portal de Proveedores** para optimizar y agilizar el proceso de gestión de facturas y pagos.

La aplicación debe incluir un sistema de autenticación que gestione dos roles principales:

- **Rol Administrador (Equipo Interno):**
  - Visualizar un *dashboard* con el listado de todos los proveedores registrados (RUC, Razón Social, Representante Legal, Dirección, etc.).
  - Consultar el historial de facturas subidas por cada proveedor.
  - Ver y modificar el estado de cada factura (Ej: Recibida, Por Pagar, Pagada, Vencida).
- **Rol Proveedor (Empresa Externa):**
  - Permitir el auto-registro (creación de usuario).
  - Completar y editar el perfil de su empresa (con los datos mencionados: RUC, Razón Social, etc.).
  - Acceder a una interfaz para **cargar sus facturas en formato PDF**.

🌟 **Reto Adicional (Plus Opcional):** Se valorará especialmente la **propuesta o implementación** de una función que utilice una API de IA Generativa (cualquier LLM, como Gemini u OpenAI) para:

1. **Procesar** el PDF de la factura recién cargada.

2. **Extraer** automáticamente datos clave (Ej: Monto total, RUC del emisor, fecha de vencimiento).
3. **Guardar** esta información extraída en la base de datos para facilitar la gestión del Administrador.

🛠️ Herramientas Disponibles

- Plataforma Cloud: Google Cloud Platform (GCP).
- Servicios Específicos: Firebase Authentication (vía Firebase Studio), Firestore, Cloud Storage, Cloud Run u otros que consideres necesarios.

🎯 Tu Tarea: Fase 2 - Implementación Mínima Viable

1. Backend (Cloud Run)

- Desarrollar la API (en Python o Node.js) que se despliega en Cloud Run.
- Debe tener un *endpoint* protegido que reciba la factura en PDF.
- Implementar la lógica para subir el PDF a **Cloud Storage**.
- Implementar la lógica para guardar el registro de la factura en **Firestore**, asociándola al Proveedor autenticado.

2. Autenticación

- Implementar **Firebase Authentication** para el registro y *login* del Proveedor. Tu API en Cloud Run debe validar el token de Firebase.

3. Cliente (Frontend)

- Aquí tienes flexibilidad. Queremos ver que tu API puede ser "consumida". Elige **una** de estas opciones:
    - **Opción A (Recomendada):** Usar un *framework* básico (React, Vue, etc.) para crear la vista de Login, Registro y el formulario para subir el PDF.
    - **Opción B (LCNC):** Usar una herramienta *low-code* (como Bubble, Softr, o si conoces Lovable) para construir la interfaz que consuma tu API.
    - **Opción C (Mínima):** Si el tiempo es crítico, puedes omitir el *frontend* visual y probar tu API protegida usando **Postman** (en este caso, explica en tu README cómo obtuviste el token de Firebase para probar).

4. Documentación (README.md)

- Tu repositorio de GitHub debe incluir un `README.md` claro con:
    - Instrucciones para instalar dependencias.
    - Variables de entorno necesarias.
    - Instrucciones paso a paso para desplegar tu solución en GCP.
    - (Si elegiste la Opción C) Los comandos `curl` o la colección de Postman para probar los *endpoints*.

5. Entregables

a) **<u>El enlace a su repositorio de GitHub.</u>**
   - **Todo el Código Fuente:** Tanto del Backend (Python/Node) como del Frontend (React/Vue), si eligieron esa opción.
   - **El archivo `README.md`:** Este archivo es la documentación y debe explicar:
     - Las instrucciones claras para desplegar la solución (variables de entorno, comandos, etc.).
     - La URL pública de la aplicación ya desplegada (para que el equipo técnico pueda probarla en vivo).
   - **Si eligieron la Opción C:** La colección de Postman o los comandos `curl` necesarios para probar la API protegida

b) **<u>Presentación del Elevator Pitch (máx 10 min)</u>**: Evaluaremos tu capacidad para comunicar soluciones técnicas complejas a una audiencia de negocio (no técnica). Imagina que le presentas tu proyecto al **Gerente de Operaciones** de Neo. Esta persona no sabe de GCP, APIs o Firestore. Debes incluir:
   - **El Valor (El "Por Qué"):**
     - Explica el **problema** que resuelve tu portal y el **beneficio** directo para su área.
   - **La Arquitectura (Explicada Simple):**
     - No hables de "Cloud Run" o "Firestore". Traduce los conceptos.
   - **El Producto (Demo Funcional):**
     - Muestra tu aplicación funcionando (el *frontend* que construiste o Postman).
     - Enfócate en el flujo del **usuario Proveedor**.
   - **El "Plus" de IA Explicado Simple:**
     - Explica el "Plus de IA" (la lectura del PDF) **sin usar jerga técnica**.