



Streamlit



Gemini



ChatGPT

Entrega Final

IA: Prompt Engineering para  
Programadores

Alumno: Sebastian Montenegro

Profesor: Javier Gimenez

Tutor: José Ignacio López Saez

Comisión: 76425

## **Problemática**

La nutrición personalizada y la gestión del bienestar son desafíos significativos en la sociedad actual. La problemática se centra en la inaccesibilidad y el alto costo de la asesoría dietética profesional, sumada a la sobrecarga de información no verificada disponible en internet.

- **Alto Costo y Escasez:** La consulta frecuente con un nutricionista es costosa y requiere tiempo, lo que excluye a gran parte de la población que necesita planes de alimentación adaptados a sus métricas, objetivos y condiciones de salud.
- **Planes Genéricos:** Las soluciones existentes suelen ofrecer planes de dieta genéricos que no consideran factores críticos como el historial de peso, las mediciones antropométricas específicas (como la circunferencia abdominal) y el contexto de vida del usuario.
- **Riesgo para la Salud:** La falta de información fiable lleva a menudo a las personas a seguir dietas extremas o inadecuadas, lo que puede deteriorar su salud a largo plazo.

## **Solución Propuesta**

La solución propuesta es el desarrollo de una herramienta de asesoría virtual que democratice el acceso a la guía nutricional de calidad.

La solución se basa en el análisis de datos biométricos y objetivos del usuario para generar un plan de alimentación y métricas de salud detallados y accionables.

## **Relevancia y Mejora**

- **Personalización Inmediata:** Proporciona un análisis de IMC y recomendaciones de macronutrientes al instante, algo que tradicionalmente requiere una consulta manual.
- **Enfoque Holístico:** Al incluir la circunferencia abdominal, el sistema puede alertar sobre riesgos de salud metabólica más allá del IMC, ofreciendo una visión más completa de la composición corporal del usuario.
- **Acceso 24/7 y Asequibilidad:** Reduce drásticamente las barreras económicas y de tiempo, permitiendo a cualquier usuario generar un plan inicial de forma rápida y gratuita (o a bajo costo).

## Propuesta de Aplicación Web con IA

### Nombre de la App: NutriSense

La función principal de NutriSense es generar un Plan de Alimentación Personalizado (3 días) y un Informe de Evaluación Corporal basado en los datos de entrada del usuario.

### Ventajas y Beneficios

- **Alta Relevancia:** Las respuestas son específicas para la persona (ej. "Para un hombre de 35 años, con sobrepeso, tu ingesta debe rondar las X calorías para lograr tu objetivo de pérdida de peso").
- **Escalabilidad:** Una vez implementado, el sistema puede atender a miles de usuarios simultáneamente sin aumento en el costo operativo humano.
- **Educación al Usuario:** La IA puede explicar el significado de métricas complejas (como la relación cintura-cadera o los rangos de IMC) en lenguaje simple.

### Prompt Inicial

El prompt es la clave de la solución, ya que define el rol, el contexto y los requerimientos de la salida de la IA.

```
def obtener_asesoramiento(datos, imc, categoria_imc):

    # Funcion Principal de asesoramiento.
    # Construcción del Prompt.
    # Se le da un ROL al modelo y se le proveen todos los CONTEXTOS y RESTRICCIONES.
    prompt = f"""
    Eres un Asesor Virtual de Nutrición profesional y amigable. Tu tarea es analizar los datos del usuario y generar un plan de alimentación de ejemplo de 3 días con un tono informativo, motivador

    --- DATOS DEL USUARIO ---
    * Nombre: {datos['nombre']}
    * Sexo: {datos['sexo']}
    * Edad: {datos['edad']} años
    * Estatura: {datos['estatura_m'] * 100} cm
    * Peso: {datos['peso_kg']} kg
    * Circunferencia Abdominal: {datos['abdomen_cm']} cm
    * IMC: {imc:.2f} ({categoria_imc})
    * Objetivos: {datos['objetivos']}

    --- RESTRICCIONES Y REQUERIMIENTOS DEL PLAN ---
    1. Calcula y menciona las calorías aproximadas que debería consumir el usuario según sus datos y objetivos.
    2. El plan debe ser un ejemplo de 3 días (Día 1, Día 2, Día 3).
    3. Cada día debe incluir Desayuno, Almuerzo, Merienda y Cena.
    4. El plan debe estar enfocado estrictamente en alcanzar sus Objetivos.
    5. Responde en español.
    6. Comienza tu respuesta con un saludo personalizado, un resumen de su IMC y una explicación sobre la importancia de la circunferencia abdominal.
    """
```

### **Justificación y Funcionamiento Esperado**

- **Elección del Prompt:** Se elige un Prompt basado en Roles ("Asesor Virtual de Nutrición certificado") para forzar al modelo a adoptar un tono experto y generar contenido responsable. Incluir la etiqueta --- DATOS DEL USUARIO --- garantiza que la IA sepa dónde comenzar el análisis y qué información procesar.
- **Funcionamiento Esperado:** Se espera que el modelo Gemini (ej., gemini-2.5-flash) utilice sus vastos conocimientos para realizar las estimaciones calóricas (TMB) y luego diseñe platos y menús realistas que se ajusten a esos rangos calóricos, manteniendo el foco en el objetivo del usuario (pérdida de peso vs. ganancia muscular, por ejemplo).

### **Factibilidad Económica**

El proyecto es altamente factible económicamente.

- **Costo de la IA (Gemini API):** Los modelos generativos como Gemini-2.5-Flash son extremadamente económicos para tareas de texto. La facturación se realiza por tokens de entrada y salida, con costos que son fracciones de un centavo de dólar por cada plan generado. Para un proyecto inicial y moderado, los costos de la API son insignificantes comparados con el valor de la asesoría profesional que ofrece.
- **Infraestructura:** La infraestructura requerida es mínima (un servidor web básico para Python o Node.js) y se puede alojar en servicios cloud de bajo costo.
- **Conclusión:** La baja barrera de entrada y los mínimos costos operativos hacen que el proyecto sea económicamente viable para su desarrollo e implementación inicial.

### **Requerimientos Técnicos y Propuesta de Desarrollo**

Para llevar a cabo el proyecto, se requieren los siguientes paquetes y herramientas, cuya instalación se demuestra a continuación (usando Python y pip):

- Python: La base del proyecto. (Asumimos versión Python 3.8+).
- Librería Gemini: Para interactuar con la API.

**“pip install google-generativeai”**

- Gestión de Entorno: Para aislar dependencias.

**“pip install virtualenv”**

- **IPython/Jupyter:** Necesario para el entorno de Colab/Notebooks.

**“pip install ipython”**

- **Configuración de la API Key:** Esencial para la autenticación. Se debe configurar la variable de entorno GEMINI\_API\_KEY.

## **Conclusión y Conceptos Clave**

El proyecto NutriSense es una solución viable, técnica y económicamente, a la inaccesibilidad de la asesoría nutricional.

### **Concepto de Prompt y Generación de Código**

El prompt es la instrucción o consulta que se le da al modelo de IA. En este proyecto, el prompt no genera código directamente, sino que genera contenido estructurado (el plan nutricional) a partir de datos no estructurados y de conocimiento. Su utilización es clave porque:

- **Define el Contexto:** Establece el marco operativo del modelo ("Eres un Asesor Virtual...").
- **Inyecta Datos:** Transmite los datos biométricos del usuario al modelo para que la respuesta sea relevante.

Al utilizar Gemini, estamos aprovechando la capacidad de la IA para razonar sobre datos, aplicar fórmulas implícitas (TMB) y sintetizar información compleja (diseño de dieta), lo que supera con creces lo que un código basado en reglas simples podría lograr.

## **Tecnologías Utilizadas**

Para el desarrollo de **NutriSense**, se seleccionó un stack de tecnologías modernas enfocadas en la velocidad de desarrollo, la escalabilidad y la eficiencia de la IA.

### **1. Streamlit: El Framework de Frontend**

**Streamlit** es una librería de código abierto para Python que permite crear aplicaciones web personalizadas para aprendizaje automático y ciencia de datos en minutos.

- **¿Por qué Streamlit?:** A diferencia de los frameworks tradicionales (como Django o Flask), Streamlit permite transformar scripts de Python en interfaces interactivas sin necesidad de escribir HTML, CSS o JavaScript.
- **Rol en el proyecto:** Gestiona la entrada de datos (formularios), la visualización de métricas de salud (IMC) y la renderización dinámica de los planes nutricionales generados por la IA.

- **Componentes Clave:** Uso de `st.secrets` para la seguridad de credenciales y `st.form` para optimizar las peticiones a la API.

## 2. Google Gemini API (Modelo 2.5 Flash)

La "inteligencia" del asesor recae en los modelos de lenguaje a gran escala (LLM) de Google.

- **Modelo Utilizado:** `gemini-2.5-flash`.
- **Capacidades:** Se eligió la versión **Flash** por su baja latencia y alta eficiencia. Es capaz de procesar instrucciones complejas (Prompt Engineering) y devolver respuestas estructuradas en Markdown en milisegundos.
- **Rol en el proyecto:** Actúa como el motor de razonamiento que interpreta los datos biométricos (edad, peso, abdomen) y los cruza con conocimientos de nutrición para generar recomendaciones personalizadas.

## 3. Python y Ecosistema de Librerías

Python es el lenguaje núcleo debido a su dominio en el campo de la Inteligencia Artificial.

- **google-genai:** SDK oficial de Google para una integración limpia y robusta con los modelos Gemini.
- **IPython.display:** Utilizada en las fases de desarrollo (Colab) para previsualizar el formato Markdown antes de la integración final en la web.
- **Variables de Entorno:** Implementación de buenas prácticas de seguridad para el manejo de la `API_KEY`.

## 4. Git y GitHub: Control de Versiones

El flujo de trabajo se basó en el estándar de la industria para el desarrollo colaborativo.

- **Versionamiento:** Seguimiento de cambios mediante *commits* descriptivos.
- **Repositorio Central:** GitHub actúa como el puente para el despliegue continuo (CD), permitiendo que cualquier cambio en el código se refleje inmediatamente en producción.

## 5. Streamlit Community Cloud: Despliegue (Cloud)

Para la puesta en producción, se utilizó la plataforma en la nube de Streamlit.

- **Infraestructura:** Permite el alojamiento gratuito de aplicaciones de datos.
- **Seguridad:** Gestión de "Secrets" a nivel de servidor, asegurando que las claves de API nunca se expongan en el código fuente público.

## **Diagrama de Flujo de la Aplicación**

Para entender cómo viajan los datos:

1. **Input:** El usuario ingresa datos en la interfaz de **Streamlit**.
2. **Logic:** Python calcula el IMC y prepara el **Prompt** inyectando los datos del usuario y el idioma elegido.
3. **IA Processing:** El Prompt viaja a los servidores de **Google Gemini** mediante una petición segura (HTTPS).
4. **Response:** Gemini devuelve el plan nutricional en formato texto.
5. **Output:** Streamlit recibe el texto y lo renderiza automáticamente como **Markdown** estético para el usuario final.