Hackathon NTT DATA 2025

Reto 1: CRM Inteligente

Optimización de Campañas de Ventas con IA

Foco Técnico del Informe:
Agente Autónomo. Uso de IA Generativa para la creación de copy y Salida Estructurada (JSON)

Información del Equipo y Tecnologías:

- BUSTMANTE GUERRA Juan Antonio
- ROSALES CASTRO Juan Sebastian
- HURTADO ASTO Yngrid Astrid Gladys
- VÁSQUEZ MORALES Carlos Camilo

Tecnologías de Desarrollo:
Python, Streamlit , OpenAI GPT-3.5 , Pandas, GitHub

Fecha:
29/11/25

ÍNDICE

1. Visión General y Estrategia del Proyecto

1.1. Contexto y Oportunidad

En el marco del Workshop NTT DATA IActiva 2025, identificamos una limitación crítica en los sistemas de gestión de clientes (CRM) tradicionales: son estáticos. Almacenan datos, pero no "piensan" ni actúan sobre ellos en tiempo real.

Nuestro equipo aceptó el reto de construir una solución que no solo almacene información, sino que transforme datos en ventas mediante el uso de Inteligencia Artificial Generativa (GenAI). El objetivo fue pasar de un modelo de "consulta de datos" a un modelo de "Agente Autónomo" capaz de tomar decisiones de marketing por sí mismo.

1.2. El Problema a Resolver

Los agentes de ventas pierden horas valiosas intentando cruzar manualmente dos mundos desconectados:

1. **Datos Históricos:** Lo que la empresa sabe (compras pasadas, edad, sector).
2. **Huella Digital Actual:** Lo que al cliente le interesa *hoy* (redes sociales, tendencias, likes).

Hacer este cruce manualmente para cada cliente es imposible de escalar. El reto exigía una herramienta que automatizara la investigación, segmentación y redacción de campañas.

1.3. Nuestra Solución: El Agente CRM Autónomo

Desarrollamos una aplicación web interactiva que integra un Agente de IA diseñado para ejecutar un "Loop de Decisiones". A diferencia de un chatbot convencional, nuestro sistema opera con autonomía bajo una arquitectura de 4 pasos:

- **Ingesta:** Lectura de perfiles desde una base de datos estructurada (CSV).
- **Enriquecimiento:** Simulación de una API de Redes Sociales para obtener el contexto actual del cliente (intereses en tiempo real).
- **Razonamiento:** Un motor de IA (OpenAI) que analiza la correlación entre el historial de compras y los intereses sociales.
- **Acción:** Generación automática de una campaña de email marketing personalizada en formato HTML y JSON.

1.4. Alcance del MVP

Para esta Hackathon, nos enfocamos en una implementación funcional que cumple con las restricciones de seguridad y usabilidad.

- **Tecnología Core:** Python, Streamlit, OpenAI API.

- **Funcionalidad Clave:** Interfaz "One-Click" donde el usuario selecciona un cliente y el agente realiza todo el trabajo cognitivo.

1.5. Generación y Gestión Segura de la API Key de OpenAI

Para que nuestro Agente CRM pueda comunicarse con el modelo GPT y generar campañas inteligentes, necesitamos una API Key, que funciona como una credencial privada que autoriza al sistema a usar la API de OpenAI.

A continuación se documenta el proceso completo, incluyendo el flujo de creación, configuración y manejo seguro de la clave, así como buenas prácticas para trabajar en equipo sin comprometer la seguridad.

1.5.1. Acceso a la Consola de OpenAI

Para generar la clave API necesaria para que el Agente CRM pueda comunicarse con los modelos de lenguaje de OpenAI, ingresamos al panel oficial de administración en: https://platform.openai.com/api-keys

Desde este panel es posible:

- visualizar claves existentes,
- crear nuevas claves secretas,
- gestionar permisos y proyectos,
- revocar claves comprometidas.

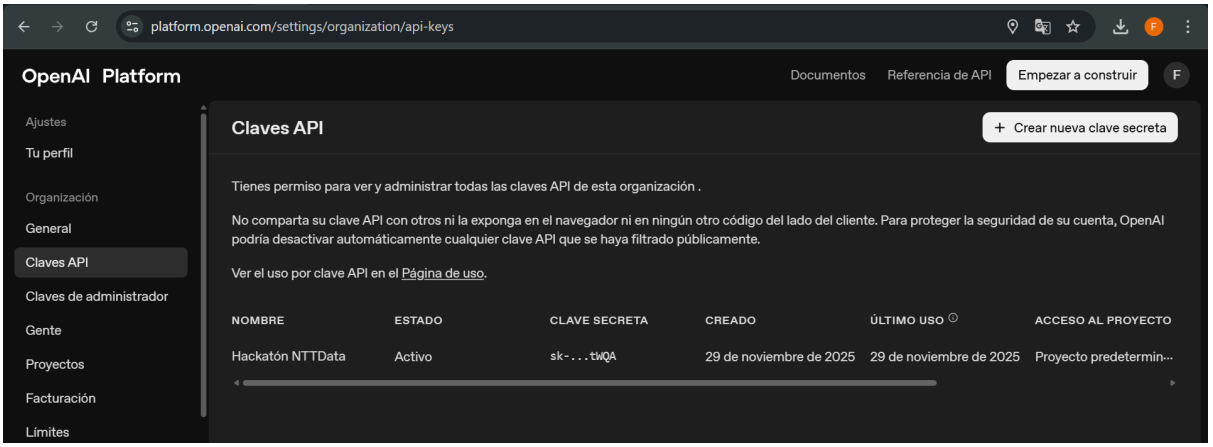

Figura 1. Panel principal de gestión de API Keys en OpenAI

1.5.2. Creación de una Nueva API Key

Una vez dentro del panel, seleccionamos el botón: “Create new secret key”

Esto abre una ventana emergente donde se configuran los parámetros iniciales para la clave.

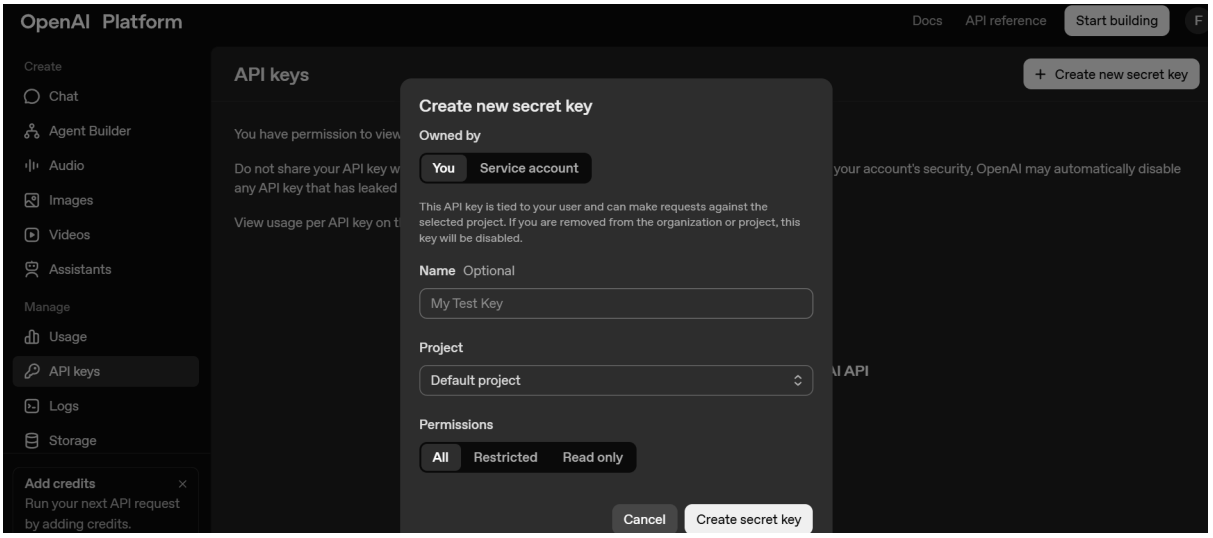

Figura 2. Ventana emergente para creación de una nueva API Key

OpenAI ofrece dos opciones para definir el propietario de la clave:

- You: la clave queda asociada directamente al usuario (ideal para desarrollo y pruebas).
- Service Account: crea una identidad técnica de bot (más utilizada en proyectos empresariales o automatizados).

Para el desarrollo del MVP seleccionamos You, por simplicidad y velocidad.

1.5.3. Configuración de la Clave API

En la misma ventana definimos los parámetros clave:

- Name: Hackathon NTTData (nombre opcional para identificar el uso).
- Project: Default project
- Permissions:seleccionamos All, lo que permite leer y escribir recursos durante el desarrollo

1.5.4. Visualización y Guardado Seguro de la API Key

Después de crearla, OpenAI muestra la clave solo una vez.

Esta clave tiene el formato:

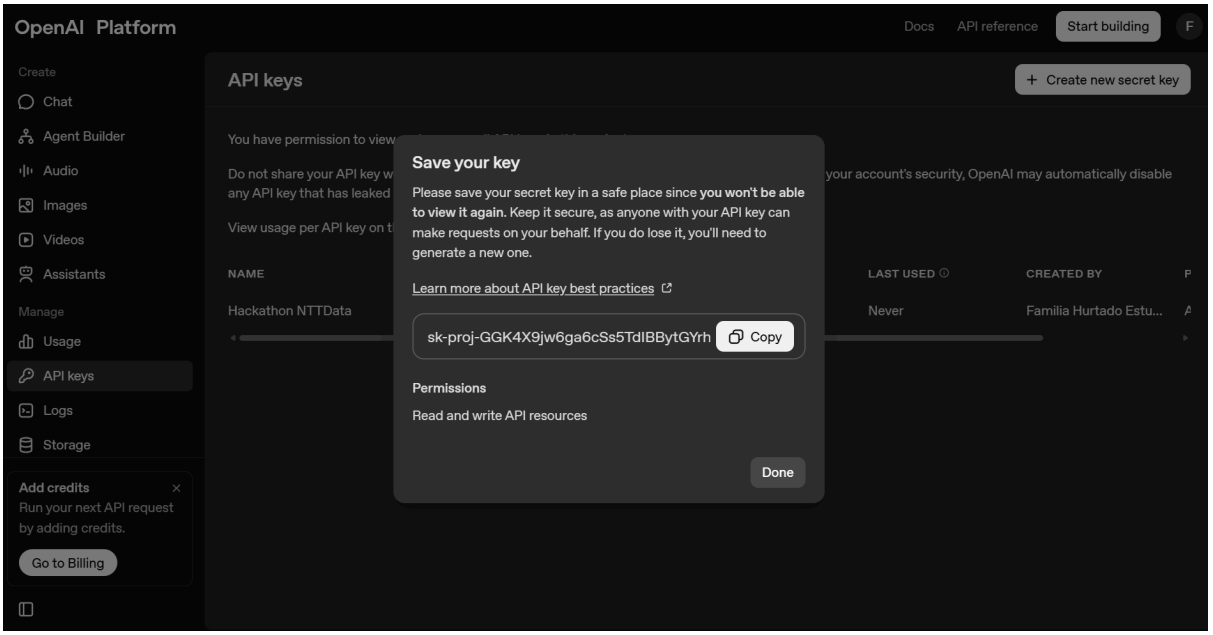

Figura 3. Clave API generada

2. Arquitectura Técnica y Lógica del Agente

2.1. Stack Tecnológico

Para garantizar un desarrollo ágil y un producto mínimo viable (MVP) robusto en el tiempo limitado del Hackathon, seleccionamos herramientas de alto impacto:

- **Lenguaje:** Python 3.10 (Estándar en Ciencia de Datos).
- **Interfaz (Frontend):** Streamlit. Elegido por su capacidad de renderizar datos complejos y componentes web en tiempo real sin la sobrecarga de frameworks tradicionales como React o Angular.
- **Motor Cognitivo:** OpenAI API (GPT-4o / GPT-3.5-Turbo).
- **Manejo de Datos:** Pandas para la ingesta estructurada del CRM.

2.2. Diseño del "Loop de Decisiones"

El corazón de nuestro agente no es un script lineal, sino un ciclo de razonamiento lógico diseñado para imitar el proceso de un experto en marketing humano. La arquitectura se divide en cuatro fases secuenciales:

1. **Fase de Ingesta (CRM Data):** El sistema lee el perfil "frío" del cliente desde el archivo clientes.csv. Aquí obtenemos datos estáticos como edad, sector y compras históricas.
2. **Fase de Enriquecimiento (Mock Social API):** Implementamos un módulo de simulación (mock_social.py) que actúa como una API externa. Este módulo inyecta

"señales vivas" (intereses actuales, último post en redes) para darle contexto fresco al agente.
    - *Decisión Técnica:* Usar un Mock (simulación) nos permitió cumplir con la restricción de "No Scraping Real" sin sacrificar la lógica de integración de datos externos.
3. **Fase de Razonamiento (GenAI Core):** A diferencia de soluciones básicas, no le pedimos a la IA "escribe un correo". Diseñamos un **Prompt Estructurado** que obliga al modelo a pensar antes de actuar:
    - *Chain of Thought:* "Analiza -> Segmenta -> Decide -> Redacta".
    - *JSON Enforcement:* Forzamos la salida en formato JSON estricto para garantizar que los datos sean procesables por el sistema y no texto libre desordenado.
4. **Fase de Acción (Renderizado):** El frontend toma el JSON generado y lo transforma dinámicamente en dos vistas: una tarjeta de campaña visual (HTML) y un objeto de datos estructurados para futuras integraciones.

2.3. Estrategia de Resiliencia (Manejo de Errores)

Un buen sistema no debe fallar, incluso cuando las dependencias externas lo hacen. Implementamos una lógica de **"Modo Rescate"** (Fallback Mechanism):

- **Problema:** Las APIs de IA pueden sufrir latencia, errores de conexión o límites de cuota (Error 429).
- **Nuestra Solución:** Envolvemos la llamada a la IA en un bloque try-except. Si el "Cerebro Principal" falla, el sistema activa automáticamente un **generador de contingencia** que utiliza plantillas pre-aprobadas y lógica local (Python random choice).
- **Resultado:** El usuario *siempre* obtiene una respuesta. La experiencia de usuario (UX) nunca se rompe, lo cual es crítico en entornos empresariales reales.

3. Experiencia de Usuario, Impacto de Negocio y Futuro

3.1. Experiencia de Usuario (UX): Transparencia y Simplicidad

El diseño de nuestra interfaz se rigió por un principio fundamental: **"Complejidad en el Backend, Simplicidad en el Frontend"**.

- **Interacción de Clic Único:** Cumpliendo con el requisito de "inputs mínimos", eliminamos la necesidad de que el usuario escriba prompts complejos. El vendedor solo selecciona un cliente y activa el agente.
- **Visibilidad del Proceso (Caja de Cristal):** A menudo, la IA se percibe como una "caja negra". Para generar confianza en el usuario, implementamos indicadores visuales en tiempo real (st.status) que muestran el **razonamiento paso a paso** del

agente (Ingesta → Análisis → Segmentación). Esto permite al humano supervisar la lógica sin intervenir en ella.

- **Visualización Dual:** La interfaz ofrece dos vistas simultáneas:
    1. **Vista Ejecutiva:** Un renderizado visual de cómo se verá el correo final en la bandeja del cliente.
    2. **Vista de Datos:** El código crudo para que los desarrolladores validen la estructura de la información generada.

3.2. Impacto de Negocio: ¿Por qué esto genera valor?

La implementación de este **Agente CRM Inteligente** ataca directamente los puntos de dolor de los equipos de ventas modernos:

1. **Hiper-Personalización a Escala:** Pasamos de campañas genéricas ("Estimado Cliente") a mensajes que resuenan con los intereses reales del usuario (ej. mencionar su interés en "Startups" o "Wellness"). Esto incrementa drásticamente la tasa de apertura y conversión.
2. **Reducción de Tareas Repetitivas:** Un vendedor humano tardaría entre 15 y 20 minutos en investigar a un cliente en redes, cruzar datos y redactar un correo. Nuestro agente reduce este ciclo a **menos de 5 segundos**.
3. **Estandarización de Calidad:** Al usar un Prompt estructurado y un modelo avanzado (GenAI), aseguramos que todas las comunicaciones mantengan un tono profesional y persuasivo, eliminando errores humanos de redacción.

3.3. Escalabilidad: Del MVP a la Producción

Aunque este prototipo utiliza datos simulados (Mock) y archivos planos (CSV) para cumplir con las reglas del Hackathon, la arquitectura es totalmente modular y escalable:

- **Módulo de Datos:** El archivo clientes.csv es fácilmente reemplazable por una conexión SQL a bases de datos empresariales como Salesforce, HubSpot o SAP.
- **Módulo Social:** La función mock_social.py está diseñada como un "placeholder". En un entorno real, esta función se conectaría a APIs oficiales (como Facebook Graph API o LinkedIn API) o a herramientas de *Social Listening* corporativas.
- **Modelo de IA:** La flexibilidad de nuestra clase OpenAI permite cambiar entre modelos (GPT-3.5 para velocidad/costo o GPT-4o para razonamiento complejo) simplemente ajustando una variable de configuración, adaptándose al presupuesto de la empresa.

4. Lecciones Aprendidas y Desafíos Técnicos

Durante el desarrollo de este prototipo bajo la presión del hackathon, nuestro equipo se enfrentó a desafíos reales de ingeniería de software que se convirtieron en valiosas lecciones de arquitectura:

4.1. Resiliencia y Manejo de Fallos

Uno de los mayores aprendizajes fue que **"la nube no es infalible"**. Nos encontramos con errores de cuota (Error 429) y latencia en la API.

- **La Lección:** Un sistema robusto no debe romperse cuando falla un servicio externo.
- **La Solución:** Implementamos un patrón de diseño de **"Fallback" (Respaldo)**. Si el motor de IA principal no responde, el sistema degrada su funcionalidad de manera elegante, activando un generador local de respuestas simuladas. Esto garantiza que el usuario final *siempre* obtenga un resultado, protegiendo la experiencia de uso.

4.2. Rigurosidad en Versionado de Modelos (API vs. Marketing)

Aprendimos la diferencia crítica entre los nombres comerciales de los productos ("GPT-5", "GPT-4 Turbo") y los identificadores técnicos estrictos de la API (gpt-4o, gpt-3.5-turbo).

- **La Lección:** La precisión en la configuración del código es vital. Intentar invocar modelos con nombres no oficializados provoca errores de "Modelo no encontrado (404)". La documentación oficial debe ser siempre nuestra fuente de verdad, por encima de las tendencias del mercado.

4.3. Importancia del Output Estructurado (JSON)

Inicialmente, la IA generaba respuestas muy conversacionales ("Claro, aquí tienes la campaña..."). Esto hacía imposible integrar la respuesta con el Frontend.

- **La Lección:** Para construir software sobre LLMs, el texto libre es ineficiente. Aprendimos a forzar al modelo a actuar como una "API de datos", devolviendo estrictamente formato **JSON**. Esto nos permitió tratar la respuesta de la IA como si fuera una base de datos tradicional, facilitando el renderizado en pantalla.

5. Ingeniería de Prompts (Anatomía del Agente)

El éxito del agente no radica solo en el modelo utilizado, sino en la calidad de las instrucciones dadas. Diseñamos un **"System Prompt"** avanzado utilizando técnicas de *Prompt Engineering* para garantizar consistencia y calidad.

A continuación, desglosamos la estructura del prompt utilizado en el código logic/agente_ia.py:

5.1. El Prompt Maestro (Código Fuente)

```
# 1. El Prompt
inmediato = F"""
Eres un Agente Experto en Marketing.
CLIENTE:{cliente['Nombre']}, Sector:{cliente['Sector']}
INTERESES:{", ".unirse(información_social['intereses'])}

TAREA: Generar campaña de venta en JSON.

FORMATO JSON ESPERADO:
{{
    "segmento": "...",
    "producto_sugerido": "...",
    "asunto": "...",
    "mensaje": "..."
}}
"""
```

5.2. Análisis de la Estrategia del Prompt

1. **Asignación de Rol:**
    - *Instrucción:* "Eres un Agente Experto en Marketing..."
    - *Efecto:* Configura el tono, vocabulario y sesgo cognitivo del modelo para que actúe como un profesional de ventas, evitando lenguaje informal o irrelevante.
2. **Inyección de Contexto Dinámico:**
    - *Instrucción:* Uso de variables {cliente['Nombre']} dentro del texto.
    - *Efecto:* Personalización total. La IA no alucina datos genéricos, sino que trabaja estrictamente con la información provista por el CSV y el Mock Social.
3. **Cadena de Pensamiento (Chain of Thought):**
    - *Instrucción:* "1. Analiza... 2. Segmenta... 3. Crea..."
    - *Efecto:* Obligamos al modelo a razonar paso a paso antes de escribir el correo final. Esto reduce las alucinaciones y mejora la lógica de la recomendación de productos.
4. **Restricción de Formato:**
    - *Instrucción:* "RESPUESTA SOLO EN JSON".
    - *Efecto:* Es la parte más crítica para la ingeniería de software. Garantiza que la salida sea legible por máquina (Machine Readable), permitiendo que nuestra aplicación web muestre pestañas, métricas y diseños HTML sin errores de sintaxis.

6. Integración del proyecto adicional

6.1. Resumen del proyecto adicional

Adicionalmente al trabajo descrito en este informe, desarrollamos un prototipo funcional que materializa la propuesta conceptual: una aplicación web que automatiza la creación de campañas personalizadas a partir de datos CRM y señales sociales simuladas. Este prototipo fue implementado como una demostración operativa que permite seleccionar clientes, ejecutar el agente autónomo con un solo clic y obtener una campaña lista para enviar, junto con un archivo de salida estructurado en formato JSON.

6.2. ¿Por qué lo integramos al informe?

El propósito de integrar este proyecto adicional es mostrar que la idea no se quedó solo en la teoría: contamos con un **artefacto funcional** que valida los flujos planteados en el informe (ingesta → enriquecimiento → razonamiento → acción). Incluir este entregable refuerza la evidencia de que la solución es viable y reproducible en un contexto real de empresa.

6.3. Aportes principales del prototipo al caso de negocio

- **Validación práctica:** demuestra que el proceso propuesto efectivamente produce campañas personalizadas en menos de un clic.

- **Producto entregable:** genera artefactos útiles para negocio (vista visual del correo, JSON estructurado y archivos descargables).

- **Confianza operativa:** la interfaz muestra el paso a paso del agente, lo que facilita la adopción por usuarios comerciales al ofrecer transparencia.

- **Mecanismo de contingencia:** incorpora un modo de respaldo que garantiza resultados aun cuando servicios externos (p. ej. el motor de IA) no respondan.

6.4. Qué incluye el entregable y cómo se puede revisar

El entregable práctico contiene:

- Una interfaz web (demo) para seleccionar clientes y activar el agente.

- Salidas exportables: campaña en HTML (preview), JSON estructurado para integración y CSV consolidado para reportes.

- Un historial de ejecuciones que permite auditar decisiones y resultados.

Para los evaluadores: se puede ejecutar localmente en pocos pasos (instalar dependencias y abrir la app), o revisar los archivos de salida incluidos en la entrega si no desean ejecutar el prototipo.

6.5. Cómo se relaciona con las conclusiones del informe

La existencia del prototipo refuerza las principales conclusiones del informe: la IA generativa puede operar como motor de decisión, el output estructurado (JSON) facilita la integración con procesos existentes y un agente autónomo bien diseñado reduce tiempo operativo y mejora la personalización. Además, la demo ilustra las consideraciones de resiliencia y seguridad que recomendamos para el paso a producción.

CONCLUSIÓN FINAL

El proyecto **"NTT DATA CRM Inteligente"** demuestra que la Inteligencia Artificial Generativa no es solo una herramienta de chat, sino un motor de decisión empresarial.

Hemos logrado construir un **Agente Autónomo** funcional que respeta las reglas de negocio, integra fuentes de datos heterogéneas y entrega un producto final listo para enviar. No solo hemos resuelto el reto técnico de "optimizar campañas", sino que hemos presentado una visión de cómo será el futuro del trabajo: humanos y agentes de IA colaborando para alcanzar resultados excepcionales.