Informe Técnico Proyecto: Prototipo de Agente Experto en Piercings

Integrantes:

CORY SCOTT LEVEKE ROMERO

BENJAMIN IGNACIO RUZ CRUZ

Profesor:  
GIOCRISRAI GODOY BONILLO

# 1. Introducción

Este proyecto detalla la implementación de un agente de inteligencia artificial autónomo enfocado en modificaciones corporales, específicamente piercings. El objetivo es crear un asistente experto capaz de responder a las consultas de los usuarios utilizando múltiples herramientas y manteniendo el contexto de la conversación.

A diferencia de un sistema RAG (Retrieval-Augmented Generation) tradicional que se basa en una base de conocimientos interna estática, la solución propuesta es un agente dinámico construido con el framework crewai. Este agente utiliza un LLM (Large Language Model) avanzado (gpt-4o) y un conjunto de herramientas conectadas a fuentes externas en tiempo real, como DuckDuckGo para búsquedas web y Wikipedia para información fáctica, además de la capacidad de escribir en el sistema de archivos local.

# 2. Objetivos y Capacidades del Agente

El prototipo se diseñó para cumplir con los siguientes objetivos funcionales:

* Proveer Información Experta: Actuar como un 'Asistente Experto en Modificación Corporal' con un profundo conocimiento de seguridad, cuidados y tendencias.
* Acceso a Información Dinámica: Utilizar la herramienta DuckDuckGoTool para buscar en la web "tiendas, recomendaciones o precios".
* Acceso a Información Fáctica: Emplear la herramienta WikipediaTool para buscar "definiciones o tipos de piercings".
* Ejecución de Tareas: Utilizar la herramienta file\_write\_tool ("Escritor de Archivos") para "guardar resúmenes" o cualquier contenido de texto en un archivo local cuando el usuario lo solicite.
* Mantenimiento de Contexto: Sostener una conversación coherente recordando el historial previo, gracias a la habilitación de memory=True.
* Planificación: Determinar la secuencia de herramientas necesarias para tareas complejas (ej. "busca esto Y guárdalo").

# 3. Definición del Agente y Tareas

La lógica del sistema se define mediante la configuración de Agentes y Tareas dentro de crewai.

* Agente (piercing\_expert\_agent):
  + role: 'Asistente Experto en Modificación Corporal'.
  + goal: Un objetivo de múltiples pasos que instruye al agente a conversar, usar DuckDuckGo para tiendas/precios, usar Wikipedia para definiciones, usar el escritor de archivos para guardar y planificar tareas complejas.
  + backstory: 'Eres un experto de renombre mundial en piercings y tatuajes...'.
  + tools: Se le asigna la lista de herramientas: DuckDuckGoTool, WikipediaTool, y file\_write\_tool.
  + memory: Habilitada (True) para que el agente recuerde interacciones pasadas.
  + llm: Utiliza la instancia llm configurada (gpt-4o).
* Tarea (expert\_task):
  + description: Una instrucción genérica que recibe la consulta del usuario ({query}) y le pide al agente que use su razonamiento para decidir qué herramienta necesita (charla, búsqueda web, búsqueda en wiki o escritura de archivo).
  + expected\_output: 'Una respuesta completa y útil a la solicitud del usuario'.
  + agent: Asignada al piercing\_expert\_agent.

# 4. Arquitectura de la Solución

Componentes implementados en eva.ipynb:

* LLM (OpenAI API): Se configura una instancia de ChatOpenAI apuntando al modelo gpt-4o. Se gestionan las variables de entorno OPENAI\_API\_KEY y OPENAI\_API\_BASE\_URL para asegurar la conexión.
* Framework de Agentes (crewai): Se utiliza crewai para definir la estructura operativa:
  + Agent: El piercing\_expert\_agent que piensa y decide qué herramienta usar.
  + Task: La expert\_task que define el trabajo a realizar.
  + Crew: El piercing\_crew que orquesta al agente y la tarea, y gestiona la memoria a nivel de equipo (memory=True).
* Herramientas (Tools):
  + DuckDuckGoTool: Una clase personalizada que hereda de BaseTool y envuelve DuckDuckGoSearchRun de LangChain para búsquedas web.
  + WikipediaTool: Una clase similar que envuelve WikipediaQueryRun, configurada para buscar en español (lang="es") y limitar los resultados (top\_k\_results=1).
  + file\_write\_tool: Una herramienta personalizada creada con el decorador @BaseTool.tool que permite al agente escribir content en un filename especificado.

Flujo de Ejecución:

1. Un usuario proporciona una entrada (query) al método piercing\_crew.kickoff().
2. La expert\_task recibe esta consulta.
3. El piercing\_expert\_agent analiza la tarea, su goal y el historial (si existe).
4. El agente razona y decide si necesita una herramienta.
5. Si es necesario, invoca la herramienta (ej. DuckDuckGoTool para "tiendas en Santiago").
6. Si es una consulta de seguimiento (ej. "¿Sobre qué ciudad te acabo de preguntar?"), utiliza su memoria interna.
7. Si la tarea es compleja (ej. "Busca... y guarda..."), el agente planifica la secuencia: primero WikipediaTool, luego file\_write\_tool.
8. El agente genera una respuesta final y el crew la devuelve al usuario.

# 5. Implementación y Pruebas

El prototipo se validó con un conjunto de pruebas funcionales diseñadas para verificar cada capacidad clave del agente:

* Prueba 1 (Uso de DuckDuckGo): Se ejecutó una consulta para buscar "recomendaciones de tiendas de piercing seguras en Santiago de Chile".
* Prueba 2 (Uso de Memoria): Se realizó una consulta de seguimiento: "¿Sobre qué ciudad te acabo de preguntar?", para verificar que el agente recordaba la Prueba 1.
* Prueba 3 (Uso de Wikipedia): Se solicitó una definición fáctica: "¿Qué es un piercing 'helix' según Wikipedia?".
* Prueba 4 (Planificación y Múltiples Herramientas): Se planteó una solicitud compleja: "Busca en Wikipedia los cuidados para un piercing helix y guarda esa información en el archivo 'cuidados.txt'". Esta prueba valida la capacidad del agente para secuenciar la búsqueda de información y la escritura de archivos.

La evaluación de los resultados se realiza mediante la inspección directa de las variables response\_ impresas en la consola.

6. Justificación Técnica

* Autonomía y Planificación (crewai): El uso de crewai permite que el agente vaya más allá de una simple respuesta. Puede razonar, planificar y ejecutar múltiples pasos (como se ve en la Prueba 4) para resolver una consulta compleja.
* Extensibilidad (Herramientas): La arquitectura basada en BaseTool es altamente extensible. Permite al agente interactuar con el mundo exterior (web, wikipedia) y el sistema local (escritor de archivos), superando las limitaciones de un LLM base.
* Información Actualizada: A diferencia de un RAG basado en documentos estáticos, el uso de DuckDuckGoTool y WikipediaTool asegura que el agente pueda recuperar información en tiempo real (ej. nuevas tiendas, precios, definiciones actualizadas).
* Contexto Conversacional: La habilitación de memory=True tanto en el Agente como en el Crew es crucial para una experiencia de usuario natural, permitiendo consultas de seguimiento (Prueba 2).

# 7. Conclusiones

La implementación en eva.ipynb demuestra exitosamente la creación de un agente autónomo y capaz utilizando crewai y langchain\_openai.

El prototipo valida que el agente puede:

* Seleccionar la herramienta adecuada (DuckDuckGo, Wikipedia) según la consulta del usuario.
* Mantener el contexto de la conversación utilizando su memoria.
* Ejecutar tareas en el sistema de archivos local.
* Planificar y ejecutar secuencias de múltiples herramientas para tareas complejas.

Este enfoque basado en agentes y herramientas dinámicas proporciona una alternativa flexible y potente a los sistemas RAG tradicionales, especialmente para casos de uso que requieren información en tiempo real y la ejecución de acciones.