# Prueba Técnica – Apartado GenAI

## PoC de Agente Conversacional para Gestión Humana

### 1. Objetivo de la PoC

La compañía requiere un agente conversacional sencillo que atienda solicitudes relacionadas con gestión humana (HR).
El agente debe ser capaz de:

- Responder preguntas generales con base en un conocimiento base HR.
- Consultar información en un archivo de Excel (`cesantías_causadas.xlsx`) y responder preguntas sobre las cesantías causadas.
- El orquestador LLM se encarga de decidir la ruta de respuesta según la intención detectada.

---

### 2. Arquitectura Propuesta (simple y demostrable)

**Componentes**

- **Streamlit (Frontend ligero):** interfaz de chat con 2 rutas visibles: *HR* y *Datos*.
- **API minimal (FastAPI o Flask):** recibe la pregunta, enruta la intención y arma la respuesta.
- **Herramienta MCP (pandas):** funciones para consultar el Excel de cesantías (totales por mes, top N, acumulados por documento).
- **Modelo LLM (Databricks):** endpoint de modelo preentrenado hospedado en Databricks.
    - La API invoca este endpoint para:
        - a) Clasificación de intención (*HR* vs *Datos*).
        - b) Redacción final (respuesta al usuario).
- **Conocimiento Base HR:** respuestas frecuentes en un diccionario/JSON.
- **Observabilidad básica:** logs a consola y métricas/monitoreo consultadas en Databricks (Jobs/Model Serving/Endpoints).

---

**Flujo**

1. Usuario pregunta en **Streamlit**.
2. **API** recibe la pregunta.
3. Llama al **endpoint LLM de Databricks** para clasificar intención.
4. Dependiendo de la intención:
    - **HR:** consulta KB y pide al LLM redactar.
    - **Datos:** usa MCP (pandas) para obtener el valor y pide al LLM redactar.
5. Devuelve la respuesta a **Streamlit**.

---

### 3. Flujo de Interacción

1. El usuario envía una pregunta en la interfaz.
2. La API llama al endpoint LLM de Databricks para intención.
3. Ejecuta KB o MCP (pandas) según corresponda.
4. La API vuelve a llamar al LLM para redactar la respuesta.
5. Streamlit muestra el resultado y guarda un log simple.

---

## 4. Elección Tecnológica

**Delievery para el PoC:**

- **UI:** Streamlit (un solo archivo `app.py` con caja de texto e historial).
- **API:** FastAPI o Flask con 2 endpoints:
  - `POST /ask` → `{question: str}` → `{answer: str, route: "HR|Datos"}`
  - `GET /health` → `ok`
- **MCP (Excel):** `pandas` + `openpyxl` con funciones:
  - `total_por_mes`
  - `topN_por_mes`
  - `acumulado_por_documento`
- **LLM:** Databricks Model Serving endpoint (modelo preentrenado recomendado por Databricks).
- **Métricas y seguimiento:** desde Databricks (paneles/Jobs/Endpoints), y logs a consola/archivo.

---

## 5. Promt Engineering (Excel de Cesantías)

Se realizaron 4 consultas de ejemplo al archivo `cesantías_causadas.xlsx`:

- **Pregunta:** "¿Cuál es el total de cesantías causadas en mayo de 2025?"
  **Acción:** `total_por_mes(2025-05)`
  **Respuesta:** 66.628.687

---

- **Pregunta:** "Dame el top 3 de documentos con mayor cesantía en junio 2025."
  **Acción:** `topN_por_mes(2025-06, n=3)`
  **Respuesta:**
  - Doc 104432 → 9.635.358
  - Doc 162000 → 6.635.673
  - Doc 179770 → 4.827.850

---

- **Pregunta:** "¿Cuánto lleva acumulado el documento 138194 en 2025?"
  **Acción:** `acumulado_por_documento(138194)`
  **Respuesta:** 9.179.031

---

- **Pregunta:** "¿Qué meses hay en el archivo y la suma por mes?"
  **Acción:** agrupación por mes.
  **Respuesta:**
  - 04/2025 → 17.695.561
  - 05/2025 → 66.628.687

- 06/2025 → 25.021.636
- 07/2025 → 14.190.298
- 08/2025 → 19.387.718

---

## 7 Contrato de la API

**POST**

```
{ "question": "total de cesantías en mayo 2025" }
```

**Response**

```
{
  "route": "Datos",
  "answer": "Total de cesantías causadas en 05/2025: 66.628.687.",
  "reason": "Intento detectado como Datos → total_por_mes(2025-05)"
}
```

## 6. Conclusión

Para una propuesta inicial se apalnca en servicios gratuitos on-cloud dado que el computo que presento a nivel local es limitado y los modelos provistos para estos ya cumplen de manera excepcional las funciones pedidas.

- Se cumple el requerimiento con la ruta más simple posible: Streamlit + API + LLM en Databricks + pandas.
- La arquitectura es modular y escalable, soportando integración de modelos más potentes y despliegue en la nube.
- La PoC demuestra viabilidad mediante consultas reales al archivo de cesantías, manteniendo simplicidad y claridad en la implementación.