# Evaluación Parcial N°2 - Desarrollo de un Agente Funcional

Asignatura: Ingeniería de Soluciones con IA (ISY0101)

Estudiante: Benjamín Aravena Gallardo

Proyecto: Agente Inteligente para la Tienda Online Huerto Hogar

Framework utilizado: LangChain + RAG

Fecha: Octubre 2025

## 1. Introducción

El presente informe corresponde a la segunda etapa del proyecto 'Huerto Hogar', una tienda online chilena dedicada a la venta de productos frescos. En esta fase, se implementa un agente funcional desarrollado con el framework LangChain y la arquitectura RAG (Retrieval-Augmented Generation), con el fin de automatizar consultas, mejorar la eficiencia en la atención al cliente y optimizar los tiempos de respuesta.

## 2. Diseño e Implementación del Agente

El agente inteligente fue diseñado bajo la arquitectura RAG, integrando herramientas de consulta, escritura y razonamiento. LangChain permite orquestar el flujo entre el modelo de lenguaje (LLM), el recuperador de contexto (Retriever) y la base de datos vectorial (FAISS o Pinecone).

• Herramienta de consulta: Permite buscar información relevante del catálogo y políticas internas.  
• Herramienta de escritura: Redacta respuestas claras y coherentes para los usuarios.  
• Herramienta de razonamiento: Analiza los datos recuperados para generar respuestas personalizadas.

## 3. Configuración de Memoria y Recuperación de Contexto

El agente incluye mecanismos de memoria a corto y largo plazo. La memoria de corto plazo mantiene coherencia en las conversaciones recientes, mientras que la memoria de largo plazo permite recuperar información de interacciones anteriores o del historial de usuario. Esto asegura continuidad en tareas prolongadas.

La recuperación de contexto se implementa mediante embeddings y consultas semánticas. El RAG permite que el modelo combine el texto generado con información factual proveniente de documentos vectorizados.

## 4. Planificación y Toma de Decisiones

El agente cuenta con un esquema de planificación basado en prioridades. Ante cada solicitud, evalúa el tipo de consulta (catálogo, pedidos, devoluciones, etc.) y elige la herramienta más adecuada. Si la información no está disponible, deriva el caso a soporte humano.

Por ejemplo, ante una consulta de disponibilidad de stock, el agente primero ejecuta la búsqueda en la base vectorial y luego utiliza el modelo generativo para elaborar una respuesta clara y natural.

## 5. Evaluación de Desempeño y Métricas

Se desarrollaron métricas visuales para evaluar la eficiencia del agente en términos de tiempos de recuperación y generación, así como la precisión contextual. A continuación, se muestra un fragmento del código utilizado:

with col2:  
 st.subheader("Retrieval Metrics")  
 fig = px.scatter(df, x='retrieval\_time', y='generation\_time',   
 size='docs\_retrieved', title="Retrieval vs Generation Time")  
 st.plotly\_chart(fig, use\_container\_width=True)  
   
 if 'relevance' in df.columns and 'context\_precision' in df.columns:  
 st.subheader("Relevance vs Context Precision")  
 fig = px.scatter(df, x='relevance', y='context\_precision',   
 title="Relevance vs Context Precision")  
 st.plotly\_chart(fig, use\_container\_width=True)

Estos gráficos permiten comparar el rendimiento del sistema en distintos escenarios, identificando relaciones entre el tiempo de recuperación y la calidad de las respuestas generadas.

## 6. Arquitectura General del Agente

El flujo general del agente es el siguiente:  
  
Cliente → Interfaz Web/App → API del Asistente → (LangChain Agent + Retriever → Vector DB → Documentos)  
→ Validación y generación → Respuesta final al usuario.

## 7. Conclusión

El agente inteligente implementado con LangChain y RAG demuestra una mejora significativa en la gestión de consultas y atención al cliente. Su arquitectura modular permite escalar la solución, integrando nuevas fuentes de datos y mejorando la personalización de las respuestas. Con la incorporación de métricas visuales, se garantiza la trazabilidad y optimización continua del rendimiento del sistema.

## 8. Referencias

LangChain. (2025). LangChain Documentation. https://python.langchain.com  
OpenAI. (2025). Retrieval-Augmented Generation (RAG) Overview. https://platform.openai.com  
Plotly. (2025). Plotly Express Documentation. https://plotly.com/python/plotly-express/