DATA SATURDAYS MADRID

Microsoft Agent Framework

Antonio Soto

Sponsors:

alesonITC data thinking

NTT DATA

TD SYNNEX

encamina PIENSA EN COLORES

V-Valley enhancing your business

prodware

#DataSatMadrid

¡Gracias sponsors!

Platinum

Gold

Silver

Venue

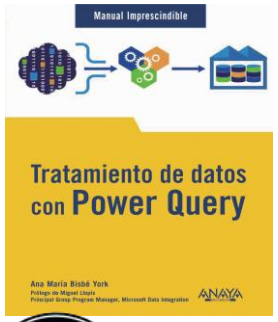
Gracias a todos los Sponsors y Colaboradores
Quiz Final + Premios + #DataBeers incluidas (18:15h)

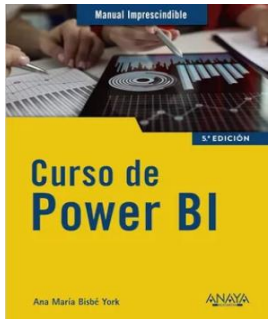
Quédate al divertido Quiz
para participar en el
Sorteo (sólo entre asistentes)
Y más ...!

Sponsor
session/stand only
NTT DATA

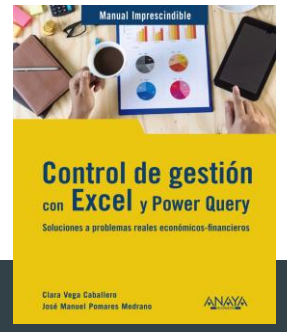
Sponsor
session/stand only
alesonITC
data thinking

Amby.net





mentorBI finanzas



Antonio Soto

Después de más de 25 años gestionando sistemas de información, principalmente en entornos de Microsoft, con especial atención a los sistemas de Business Intelligence y la gestión de datos, ahora estoy más centrado en soluciones de Machine Learning, Siempre buscando nuevos retos para ayudar a nuestros clientes a optimizar sus inversiones en su plataforma de datos. En este me centro en el desarrollo de proyectos de ML y LLM en diversos sectores y clientes.

antoniosotorodriguez@gmail.com

https://www.linkedin.com/in/antoniosql/

@antoniosql

Microsoft® MVP Most Valuable Professional

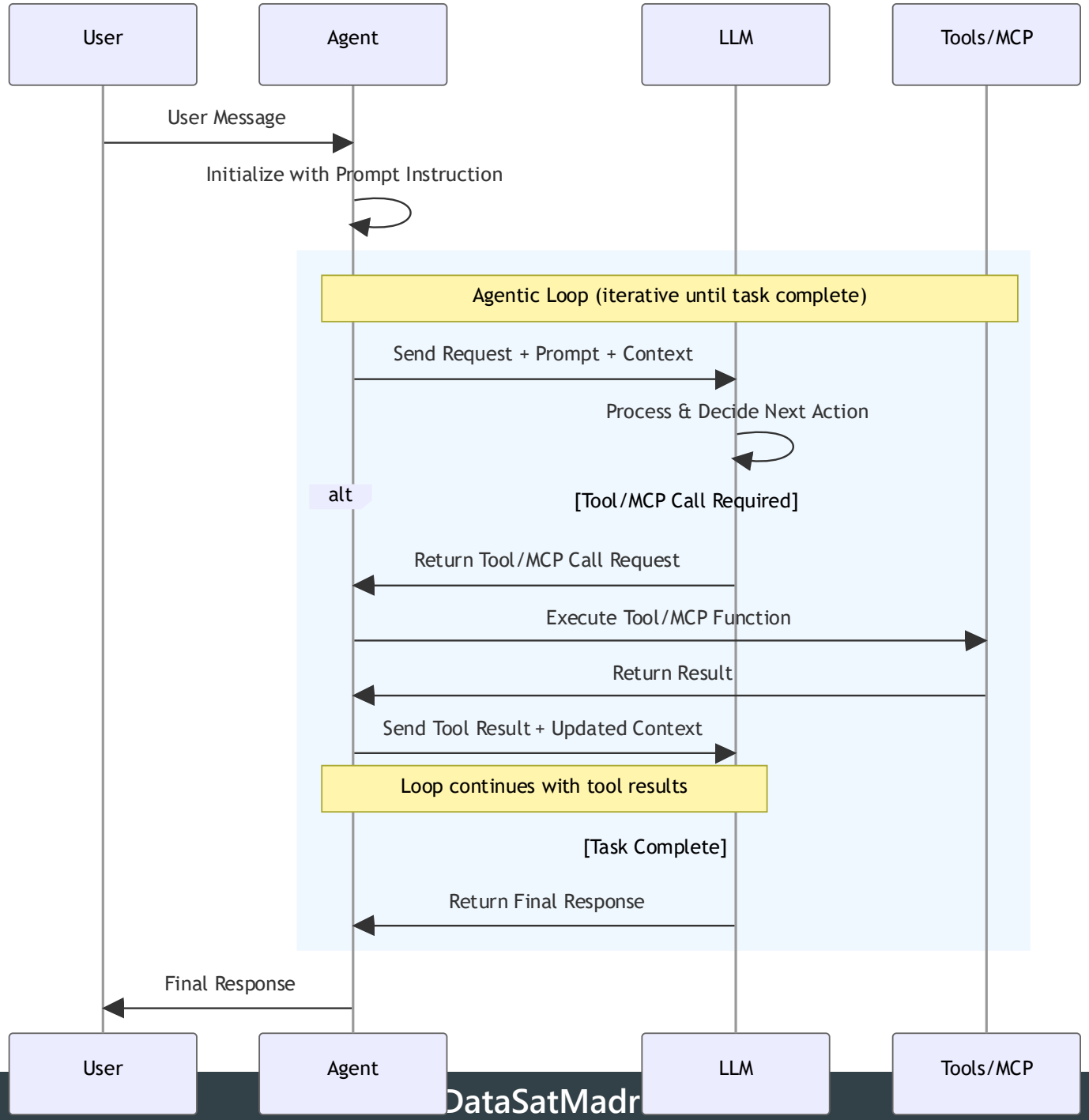
Objetivos

- Entender que es un agente
- Diferenciar cuando es necesario y cuando no nos hace falta
- Herramientas y Contextos
- Patrones de Sistemas de Múltiples Agentes

Agenda

- ¿Qué es un Agente de IA?
- Microsoft Agent Framework
- Agentes en Microsoft Agent Framework
- Model Context Protocol
- Workflows

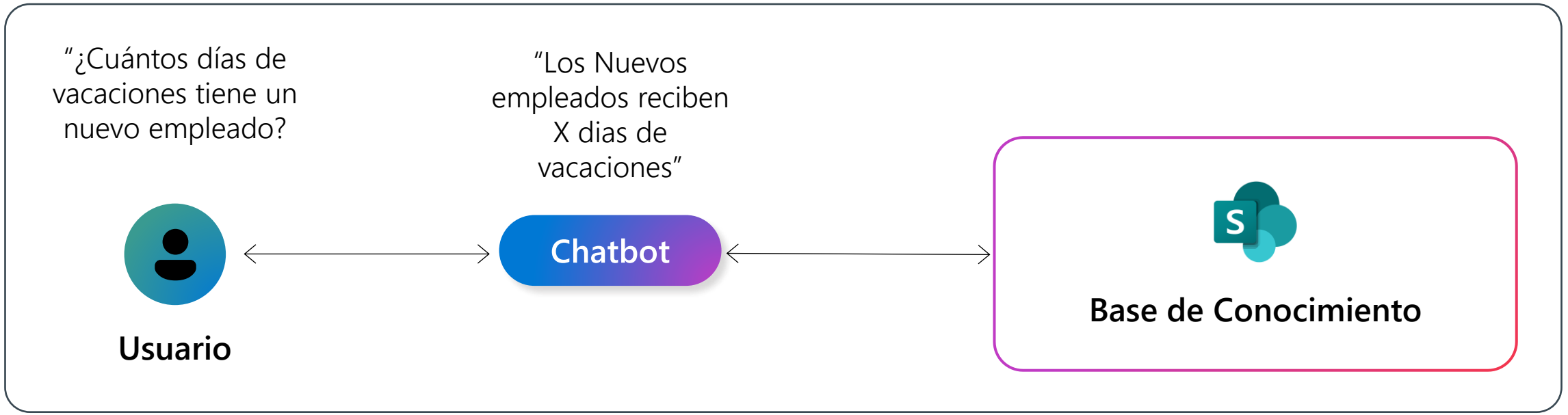
¿Qué es un Agente?


User
Agent
LLM
Tools/MCP
User Message
Initialize with Prompt Instruction
Agentic Loop (iterative until task complete)
Send Request + Prompt + Context
Process & Decide Next Action
alt
[Tool/MCP Call Required]
Return Tool/MCP Call Request
Execute Tool/MCP Function
Return Result
Send Tool Result + Updated Context
Loop continues with tool results
[Task Complete]
Return Final Response
Final Response
User
Agent
LLM
Tools/MCP

Agente IA

Un Agente, en aplicaciones basadas en LLM, es un software semi-autónomo que confía en LLMs para realizar tareas específicas a través de una interacción en lenguaje natural

Aplicación de Chatbot



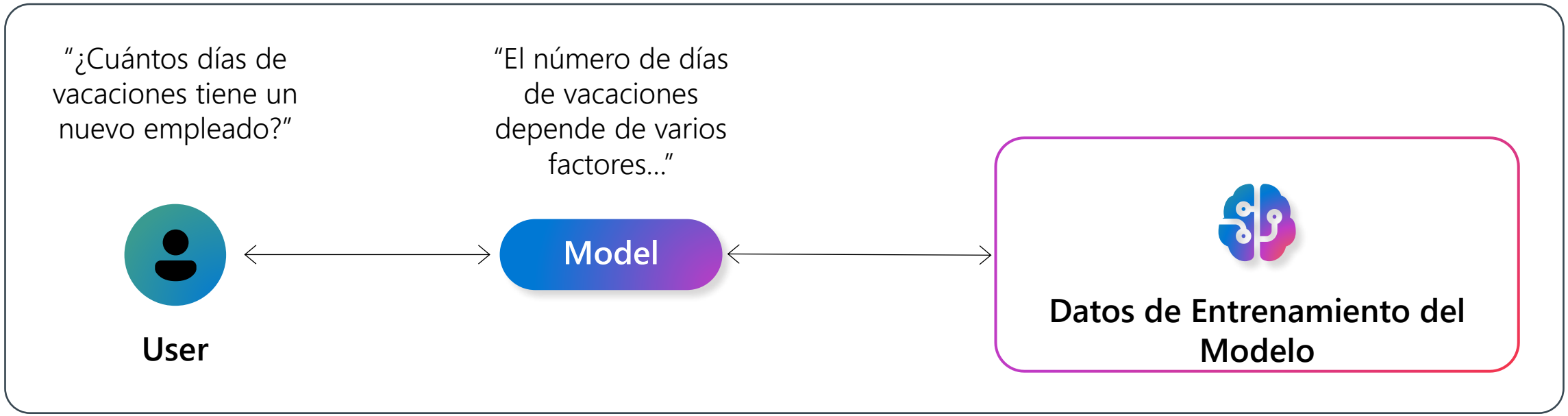
Limitaciones

Conocimiento limitado a los orígenes conectados

El Chatbot no es capaz de realizar ninguna acción

Ámbito de respuesta limitado

Aplicación IAGen sin RAG



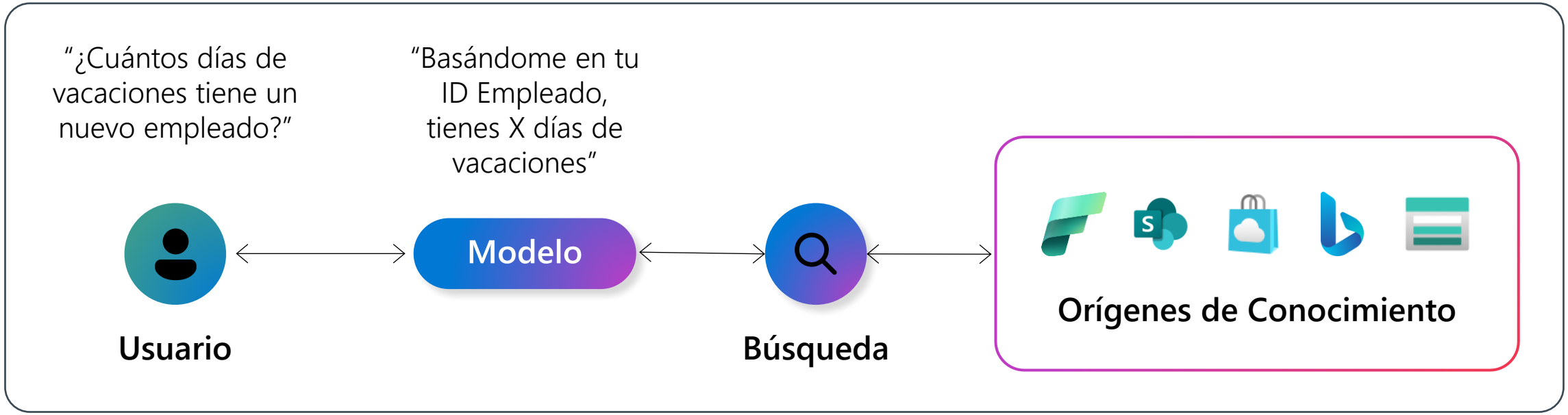
Limitaciones

Respuestas no están aterrizadas en datos relevantes para le usuario

Están limitadas a los datos de entrenamiento del modelo

Alta Probabilidad de que el modelo fabrique las respuestas

Aplicación IAGen con RAG



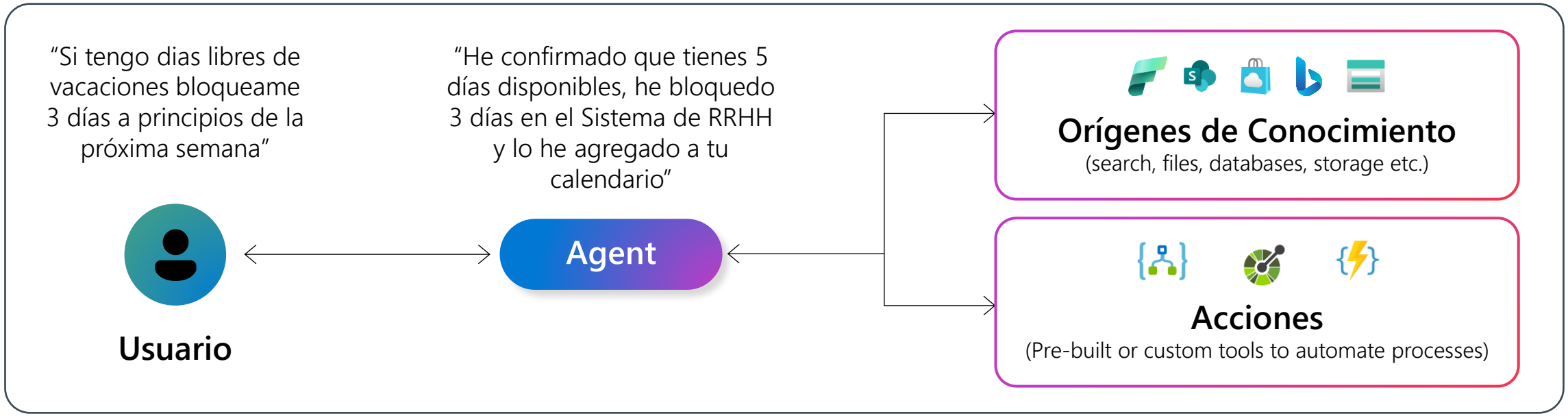
Limitaciones

Pproporciona respuestas relevantes para usuarios pero el modelo está limitado a los orígenes de datos

Válido para escenarios de obtención de información, pero no para los basados en acciones

Las Preguntas fuera del ámbito planificado pueden no ser contestadas eficientemente

Aplicación IAGen con Agentes



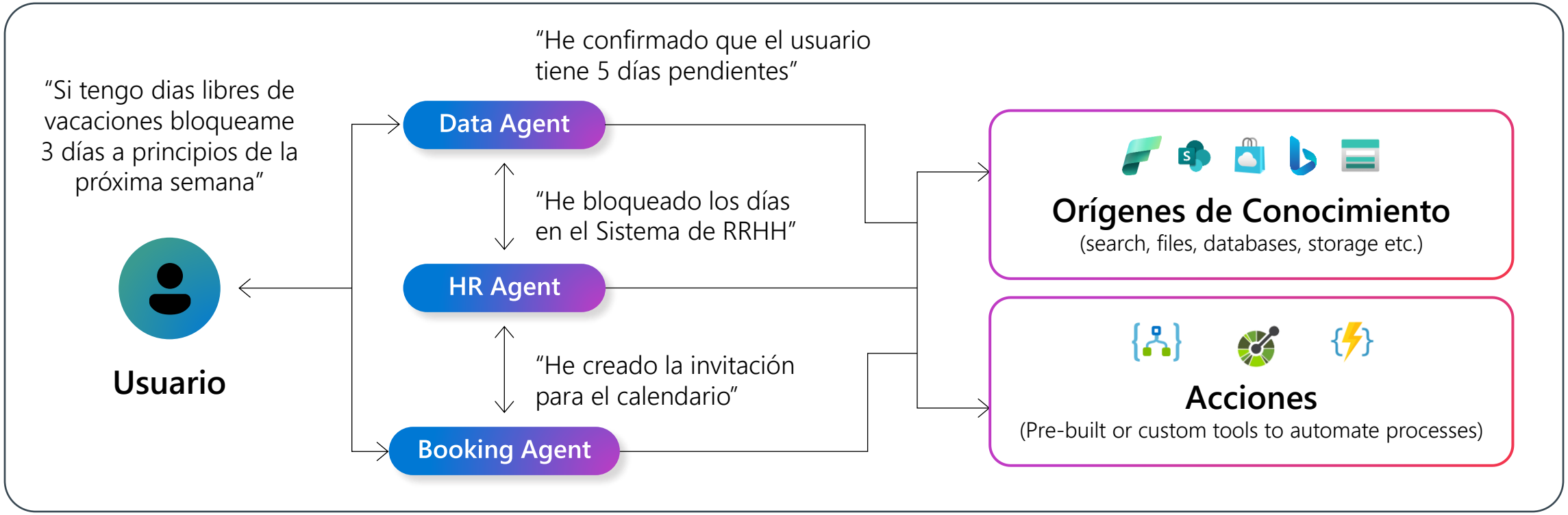
Beneficios

Agentes realizan tareas complejas

Agentes planean acciones basándose en la entrada del usuario

Agentes utilizando las bases de conocimiento, procesos de negocio y herramientas definidas

Aplicación IAGen con Múltiples Agentes



Beneficios

Agentes realizan únicamente tareas específicas asignadas

Los Agentes no se sobrecargan con prompts complejos

Agentes solo tienen acceso a herramientas y datos específicos que necesitan para completes las tareas asignadas

Consideraciones de Agentes IA

Conocimiento

Proporcionarles el contexto adecuado

Acciones

Acceso a las herramientas necesarias

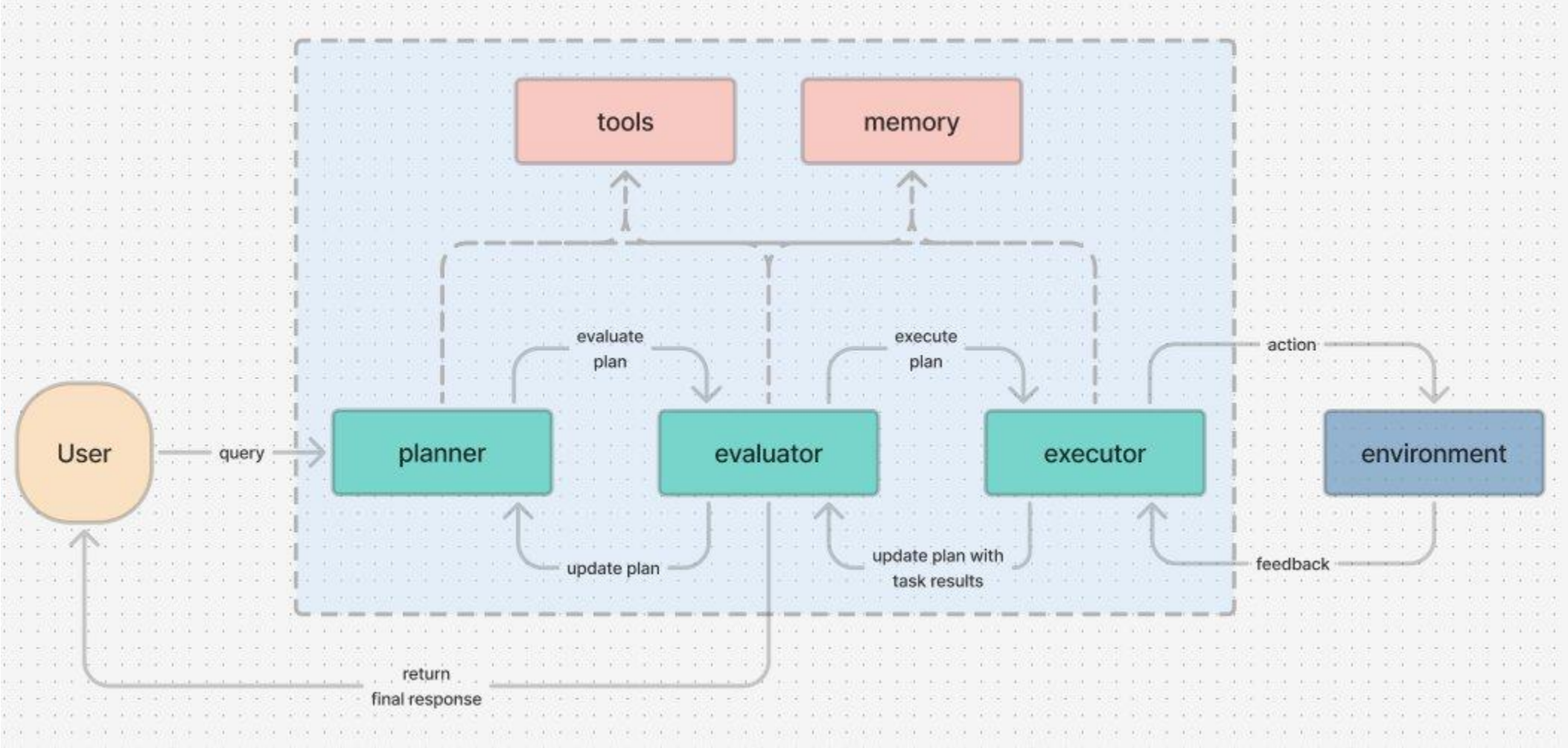
Seguridad

Acceso solo a los datos y servicios que necesitan

Evaluación

Asegurarnos de que lo hacen bien

El Interior del Agente



¿Cuándo utilizar Agentes?

Usar un agente (✅)

- Tareas dinámicas y no estructuradas
- Interacción conversacional
- Autonomía y toma de decisiones
- Ejemplos:
  - Atención al cliente
  - Educación personalizada
  - Generación y depuración de código
  - Investigación y síntesis

NO usar un agente (❌)

- Tareas altamente estructuradas
- Procesos simples y predecibles
- Flujos con muchas herramientas (>20)
- Coste y latencia innecesarios

Microsoft Agent Framework

Semantic Kernel

•SDK de código abierto de **Microsoft** para la creación de aplicaciones impulsadas por IA.
•Ayuda a los desarrolladores a incorporar capacidades de razonamiento natural y lenguaje natural en las aplicaciones.
•**"IA como una llamada a función"**

Autogen

Framework de código abierto de **Microsoft Research** para la creación de sistemas multi-agente, es decir, múltiples agentes LLM que interactúan entre sí y colaboran para resolver problemas.

Semantic Kernel

- Gestión de estado de nivel empresarial
- Seguridad de tipos y telemetría
- Soporte extendido de modelos

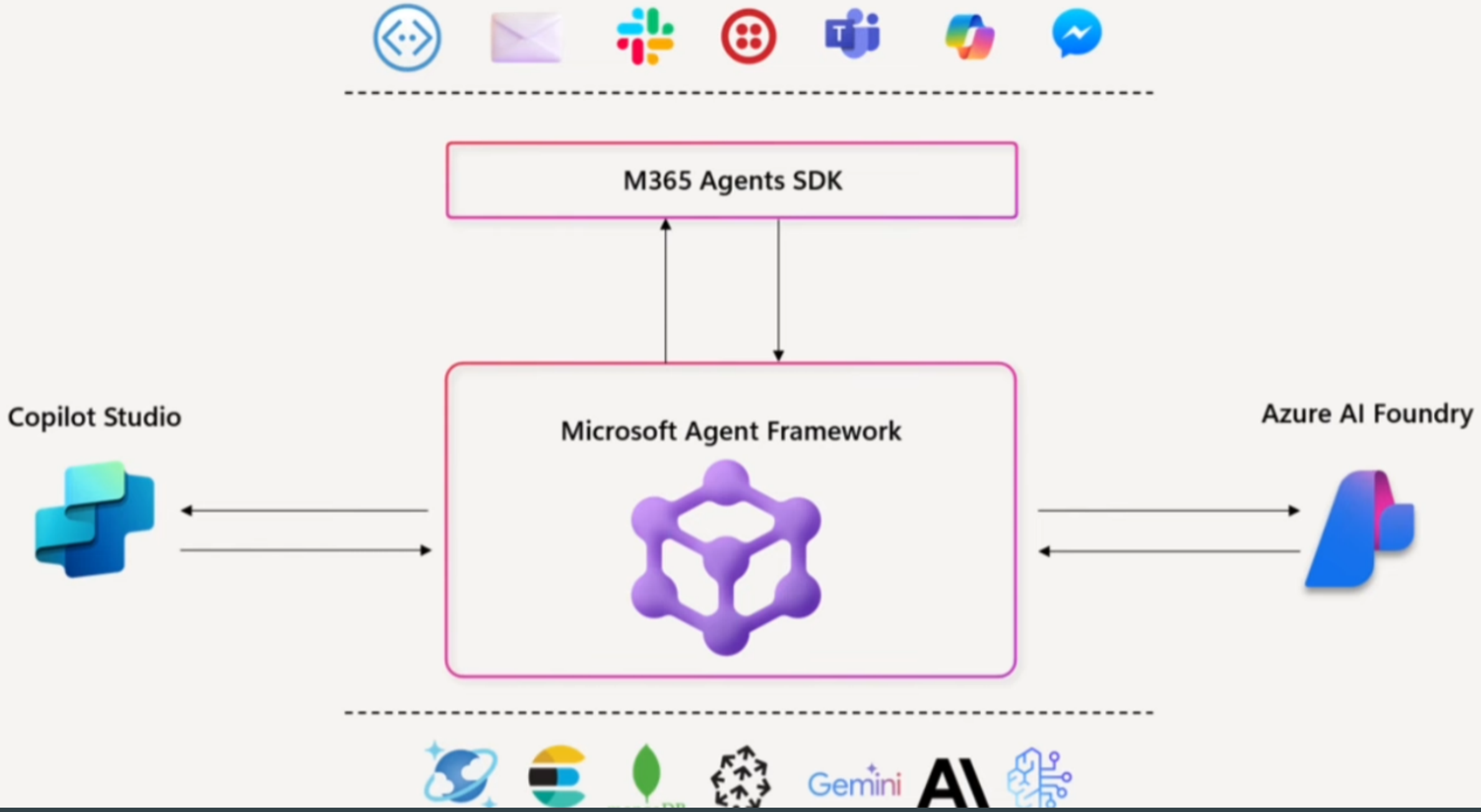
AutoGen

- Abstracciones simples para agentes
- Potentes patrones multi-agente
- Impulsado por la comunidad

Microsoft
Agent Framework

Microsoft Agent Framework

- SDK y entorno de ejecución de código abierto de Microsoft para construir, orquestar e implementar agentes de IA y flujos de trabajo multi-agente.
- Soporta tanto .NET (C#) como Python.
- Unifica Semantic Kernel y AutoGen en un único marco de trabajo.


M365 Agents SDK
Copilot Studio
Microsoft Agent Framework
Azure AI Foundry
Gemini


AI Services
Azure OpenAI
Foundry Catalog
OpenAI
Hugging Face
NVIDIA
DeepSeek
Google Gemini
Anthropic
Local models
Ollama
Foundry Local
LM Studio
Phi SLM Model Family
Memory Services
Azure AI Search
Azure CosmosDB
Elasticsearch
Redis
MongoDB
Pinecone
Qdrant
Mem0
Microsoft Agent Framework
AI and Agent Orchestration
.NET
Python
Agent Services
Foundry Agent Service
Bedrock Agents
Copilot Studio
LangGraph*
Plugins
OpenAPI
MCP
A2A
Logic Apps
UI Frameworks
ChatKit
AG-UI
Filters and telemetry
OpenTelemetry
Purview
Aspire Dashboard
Azure AI Content Safety

Agentes en Microsoft Agent Framework

Agentes

- Clase base AIAgent de la que derivan todos los tipos de agentes
- Agentes “sencillos” basados en servicios de inferencia

- Podemos crear nuestros agentes sin que estén basados en un servicio de inferencia usando AgentProtocol y BaseAgent

Underlying Inference Service	Description	Service Chat History storage supported	Custom Chat History storage supported
Azure AI Agent	An agent that uses the Azure AI Agents Service as its backend.	Yes	No
Azure OpenAI Chat Completion	An agent that uses the Azure OpenAI Chat Completion service.	No	Yes
Azure OpenAI Responses	An agent that uses the Azure OpenAI Responses service.	Yes	Yes
OpenAI Chat Completion	An agent that uses the OpenAI Chat Completion service.	No	Yes
OpenAI Responses	An agent that uses the OpenAI Responses service.	Yes	Yes
OpenAI Assistants	An agent that uses the OpenAI Assistants service.	Yes	No
Any other ChatClient	You can also use any other chat client implementation to create an agent.	Varies	Varies

Soporte de A2A

- Podemos conectar con agente que exponga vía A2A

```python
from agent_framework.a2a import A2AAgent

# Create A2A agent with direct URL configuration
agent = A2AAgent(
    name="My A2A Agent",
    description="A directly configured A2A agent",
    url="https://your-a2a-agent-host/echo"
)
```

Ejecutando Agentes

- Streaming y no streaming
- Opciones
  - Max_tokens
  - Temperature
  - Model
  - Tools
  - Response_format (ej: structured output)

Varias “Rondas” : AgentThread

- Gestión de contexto conversacional El framework permite mantener el historial de conversación entre interacciones, facilitando agentes con memoria.
- Tipos de almacenamiento
  - Persistente: historial guardado en servicios como Foundry o Azure AI.
  - En memoria: historial gestionado localmente por el agente.
- Creación de hilos (AgentThread)
  - Manual: get_new_thread()
  - Automática: al ejecutar sin especificar hilo, se crea uno temporal.
- Serialización y recuperación Los hilos pueden guardarse (serialize) y restaurarse (deserialize_thread) para continuar conversaciones previas.
- Compatibilidad entre agentes No todos los agentes pueden compartir hilos: depende del modelo subyacente (Azure, OpenAI, etc.)
- .Ejemplo práctico Un agente recuerda que la usuaria se llama “María" y responde correctamente en interacciones posteriores.
- Almacenamiento personalizado Se pueden definir ChatMessageStore personalizados para controlar cómo se guardan los mensajes.

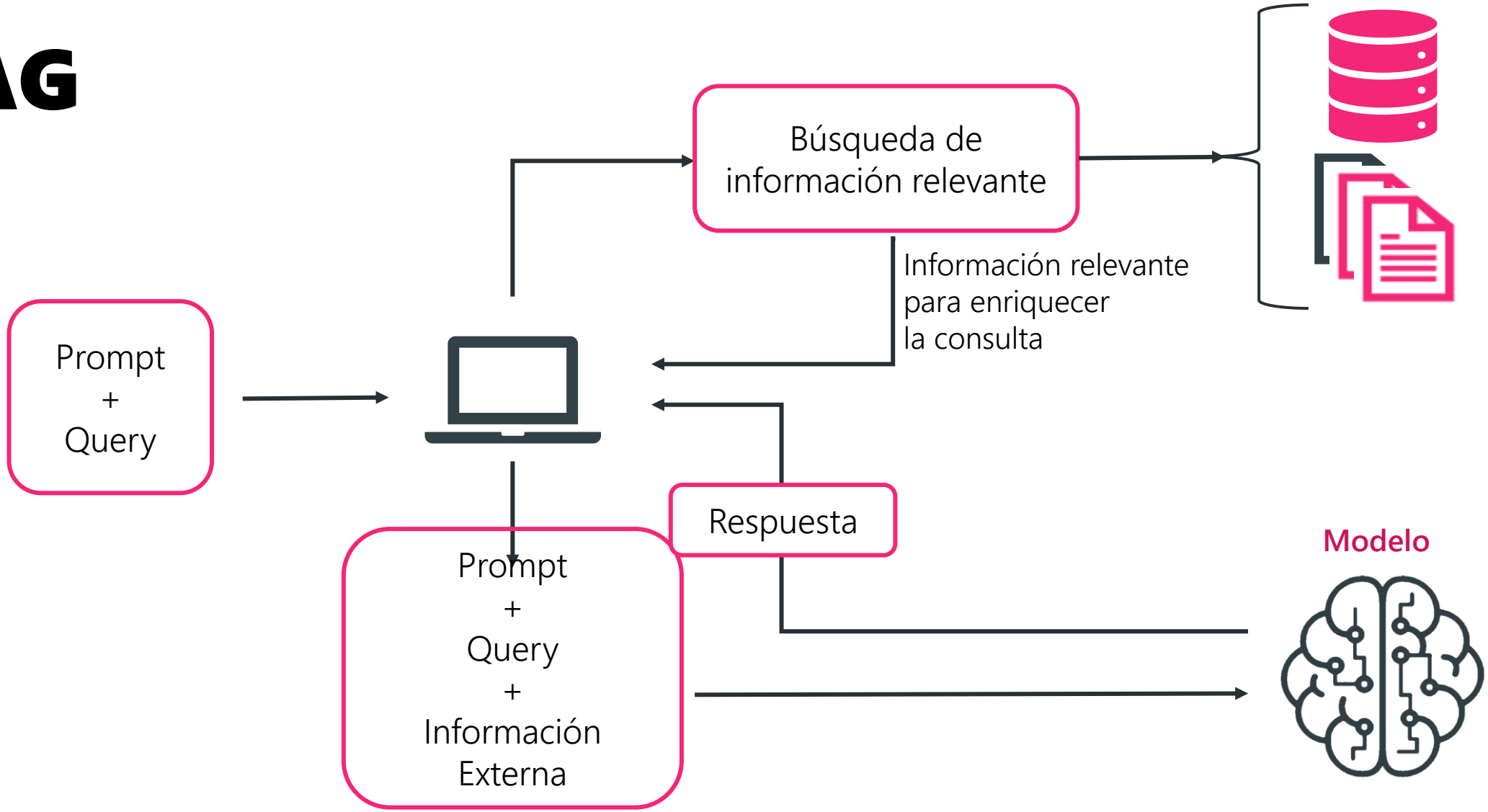
Memoria

- Capacidad del agente para mantener contexto entre conversaciones, recordar preferencias del usuario y ofrecer experiencias personalizadas.
- Tipos de memoria disponibles
  - *In-Memory Storage*: historial almacenado temporalmente durante la ejecución. Es el predeterminado
  - *Persistent Message Stores*: guarda conversaciones entre sesiones (ej. Redis, bases de datos personalizadas).
  - *Context Providers*: inyectan contexto dinámico antes de cada llamada (ej. preferencias del usuario).
  - *External Memory Services*: integración con servicios como Mem0 para capacidades avanzadas.

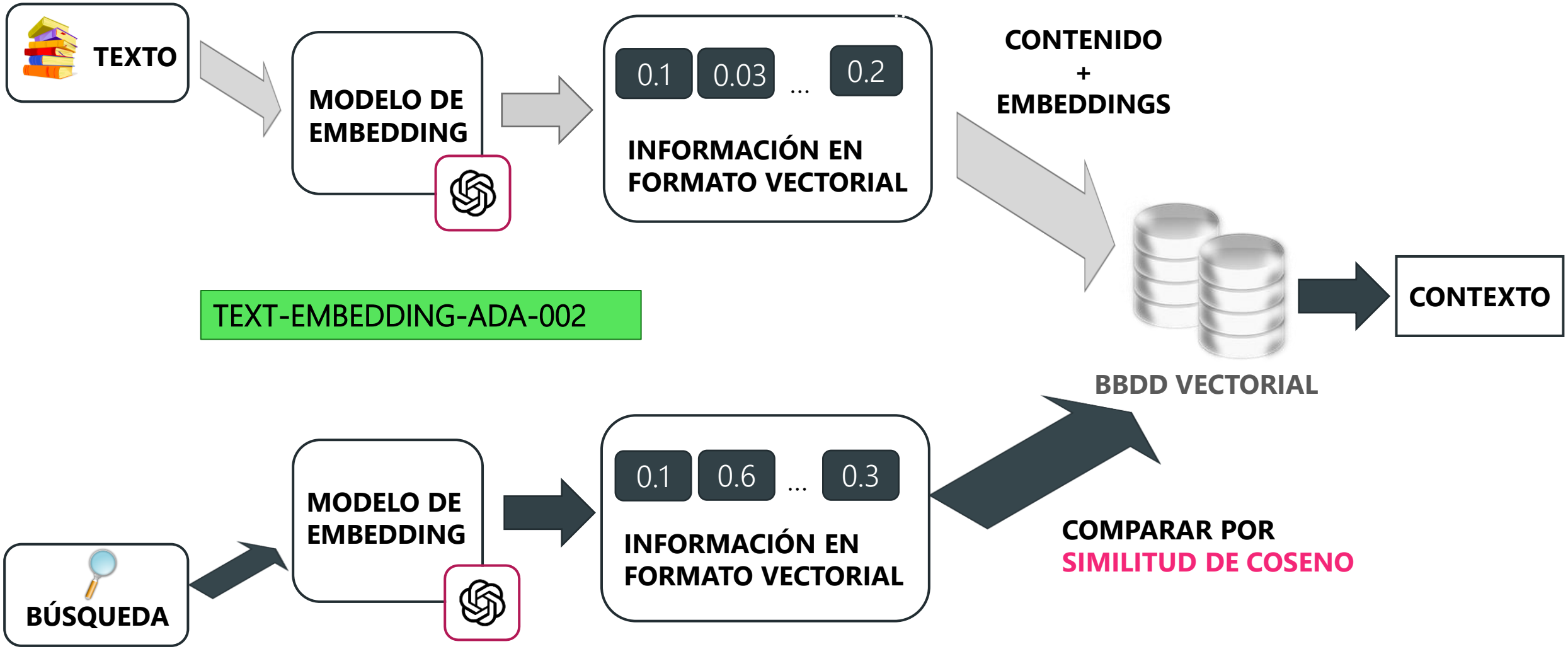
Observabilidad

- Permite monitorizar el comportamiento de los agentes, detectar errores y analizar rendimiento.
- Integración con OpenTelemetry
  - Emite trazas, logs y métricas siguiendo convenciones GenAI.
  - Compatible con OTLP endpoints y Azure Monitor.
- Opciones avanzadas
  - Configuración programática con parámetros personalizados.
  - Exportadores personalizados para trazas y métricas.
  - Soporte para Azure AI Foundry y visualización integrada.

RAG


Búsqueda de información relevante
Información relevante para enriquecer la consulta
Prompt + Query
Respuesta
Prompt + Query + Información Externa
Modelo

RAG


TEXTO
MODELO DE EMBEDDING
0.1
0.03
...
0.2
INFORMACIÓN EN FORMATO VECTORIAL
CONTENIDO
+
EMBEDDINGS
TEXT-EMBEDDING-ADA-002
BBDD VECTORIAL
CONTEXTO
BÚSQUEDA
MODELO DE EMBEDDING
0.1
0.6
...
0.3
INFORMACIÓN EN FORMATO VECTORIAL
COMPARAR POR
SIMILITUD DE COSENO

RAG

- Colección VectoreStore de Semantic Kernel
  - Azure AI Search (`AzureAISearchCollection`)
  - Qdrant (`QdrantCollection`)
  - Pinecone (`PineconeCollection`)
  - Redis (`RedisCollection`)
  - Weaviate (`WeaviateCollection`)
  - In-Memory (`InMemoryVectorStoreCollection`)
  - And more

Agentes Durables

- Extensión Durable Task: Permite crear agentes con estado y orquestaciones deterministas en un entorno serverless con Azure Functions.
- Persistencia de estado: Conserva historial de conversación y ejecución incluso tras fallos, reinicios o procesos largos.
- Características principales:
  - Serverless hosting con endpoints automáticos.
  - Hilos de agente persistentes con historial.
  - Orquestaciones deterministas (secuenciales, paralelas, con intervención humana).
  - Observabilidad y debugging mediante el dashboard de Durable Task Scheduler.
- Uso recomendado:
  - Cuando se necesita control total del código y flexibilidad.
  - Para orquestar múltiples agentes en flujos complejos y de larga duración.
  - En escenarios event-driven con triggers de Azure Functions.
- Ventajas: Escalabilidad automática (hasta miles de instancias o cero cuando no se usan), pago por invocación, continuidad de la conversación.

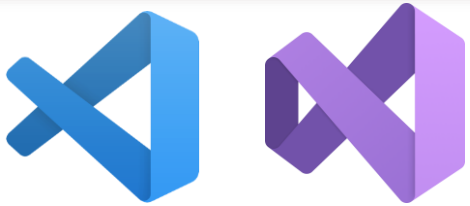
Model Context Protocol

MCP es un protocolo Abierto que estandariza como las aplicaciones proporcionan contexto a los LLMs.

Hosts

Clients

Servers






MCP Deep-Dive

MCP Client

- Llama **Tools**
- Consulta **Resources**
- Interpola **Prompts**

STDIO

SSE

MCP Server

- Expone **Tools**
- Expone **Resources**
- Expone **Prompts**

Tools

Model-controlled

Funciones invocadas por el modelo

- Obtener / Buscar
- Enviar un mensaje
- Actualizar filas BBDD

Resources

Application-controlled

Datos expuestos a la aplicación

- Ficheros
- Filas Base de Datos
- Respuestas API

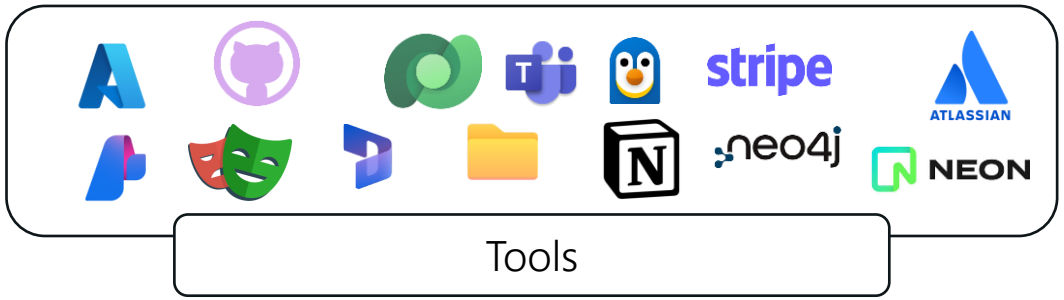
Prompts

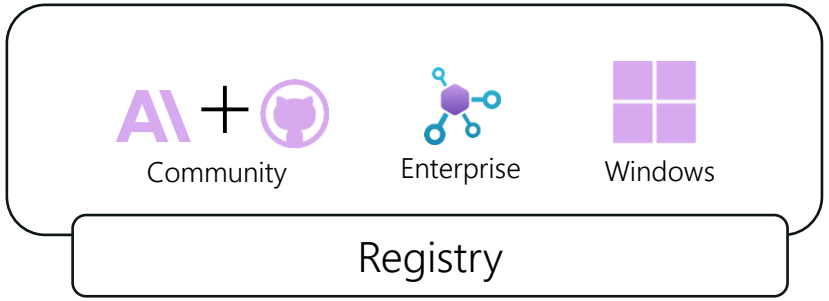
User-controlled

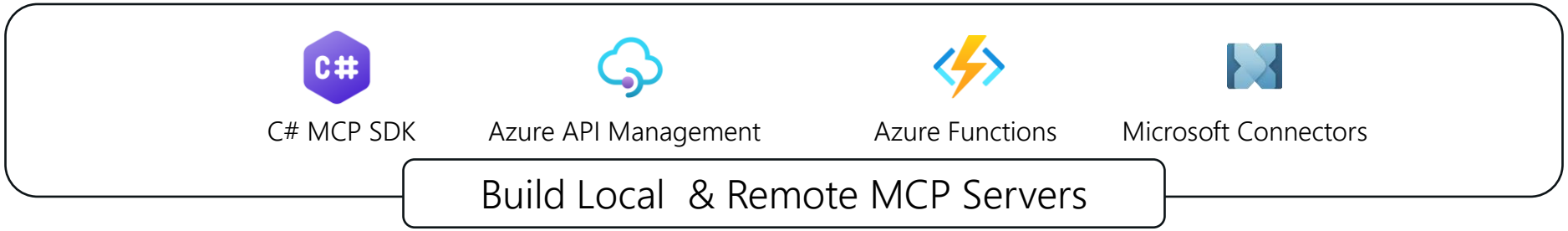
Plantillas predefinicias para interacciones con IA

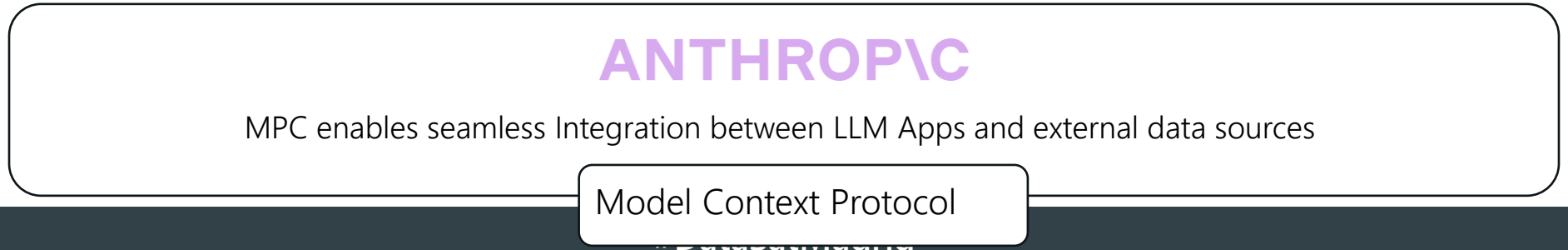
- Document Q&A
- Transcript Summary
- Output as JSON

MCP Microsoft Ecosystem







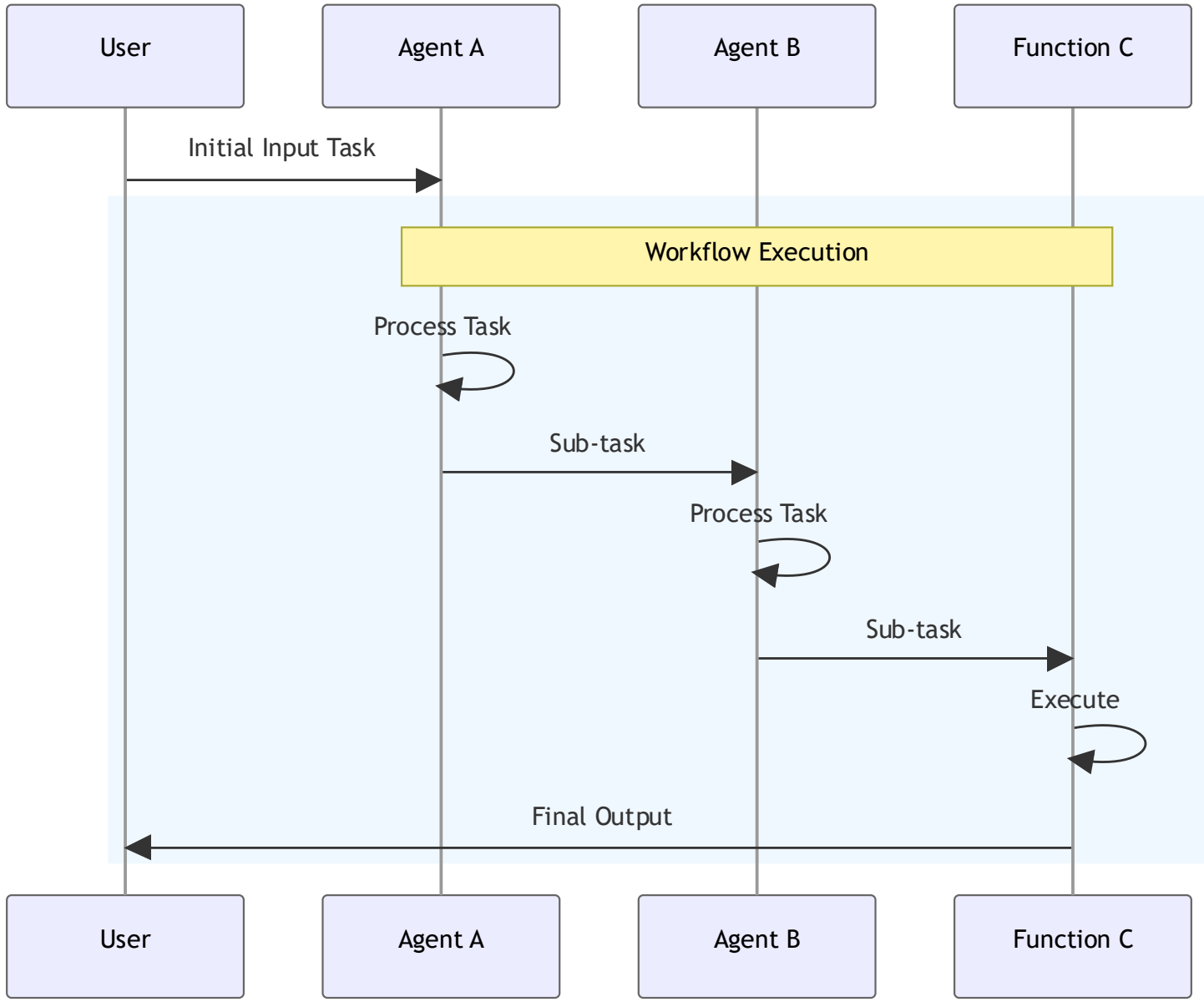


Cómo pasar de mi API a MCP

- Herramientas de conversión…. Problemas:
  - Los LLMS son malos seleccionando si la lista de herramientas es muy grande
  - Es probable que las descripciones de tus APIS no estén listas para que las consuma MCP
  - Las API están diseñadas para gestión de recursos y automatización, no para los humanos
- Solución Híbrida
  - Autogenera el código MCP
  - Quita herramientas
  - Evalúa / rescribe las descripciones
  - Agrega nuevas herramientas
  - Crea tus evaluaciones

Workflows

- Executors
- Edges
- Workflows
- Events



Como agrego los agentes

- Usar agentes directamente como participantes
  - Con SequentialBuilder o ConcurrentBuilder, donde los participantes son ChatAgent o AgentExecutor

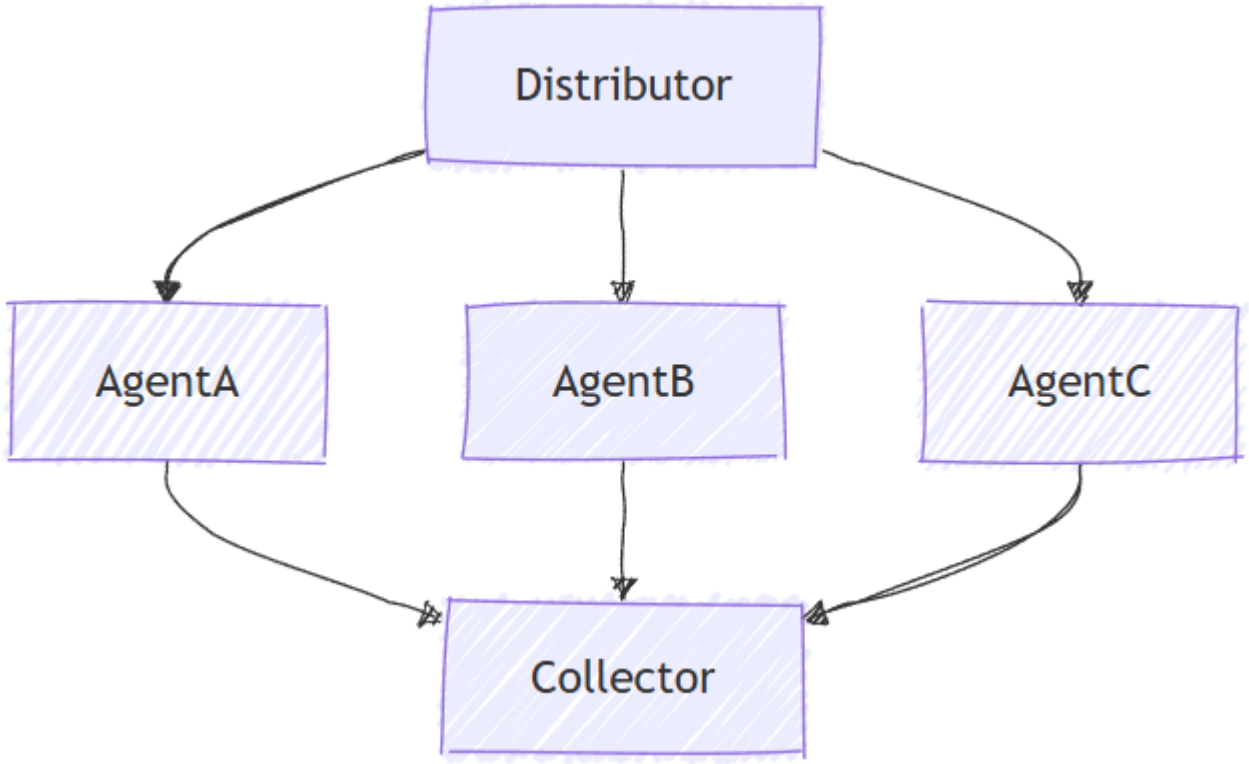
- Crear ejecutores personalizados que envuelvan agentes con AgentExecutorRequest/Response

Orquestaciones

- Patrones
  - Concurrent
  - Sequential
  - Group Chat
  - Handoff
  - Magentic

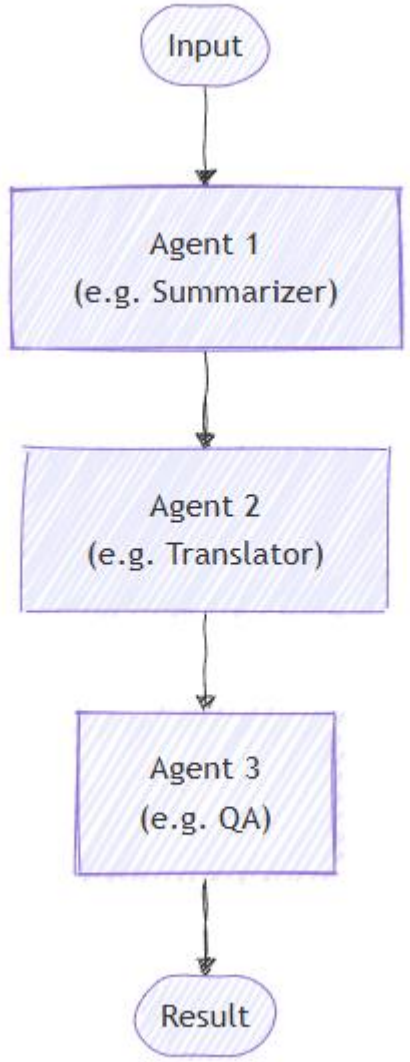
Concurrente

- Ejecución simultánea de agentes (investigador, marketer, legal, etc.).
- Cada agente procesa la entrada de forma independiente.
- Los resultados se recopilan y se agregan en un flujo final.
- Escenarios ideales: brainstorming, razonamiento en conjunto, votaciones o perspectivas diversas.



Secuencial

- Soporta agregar Custom Executors

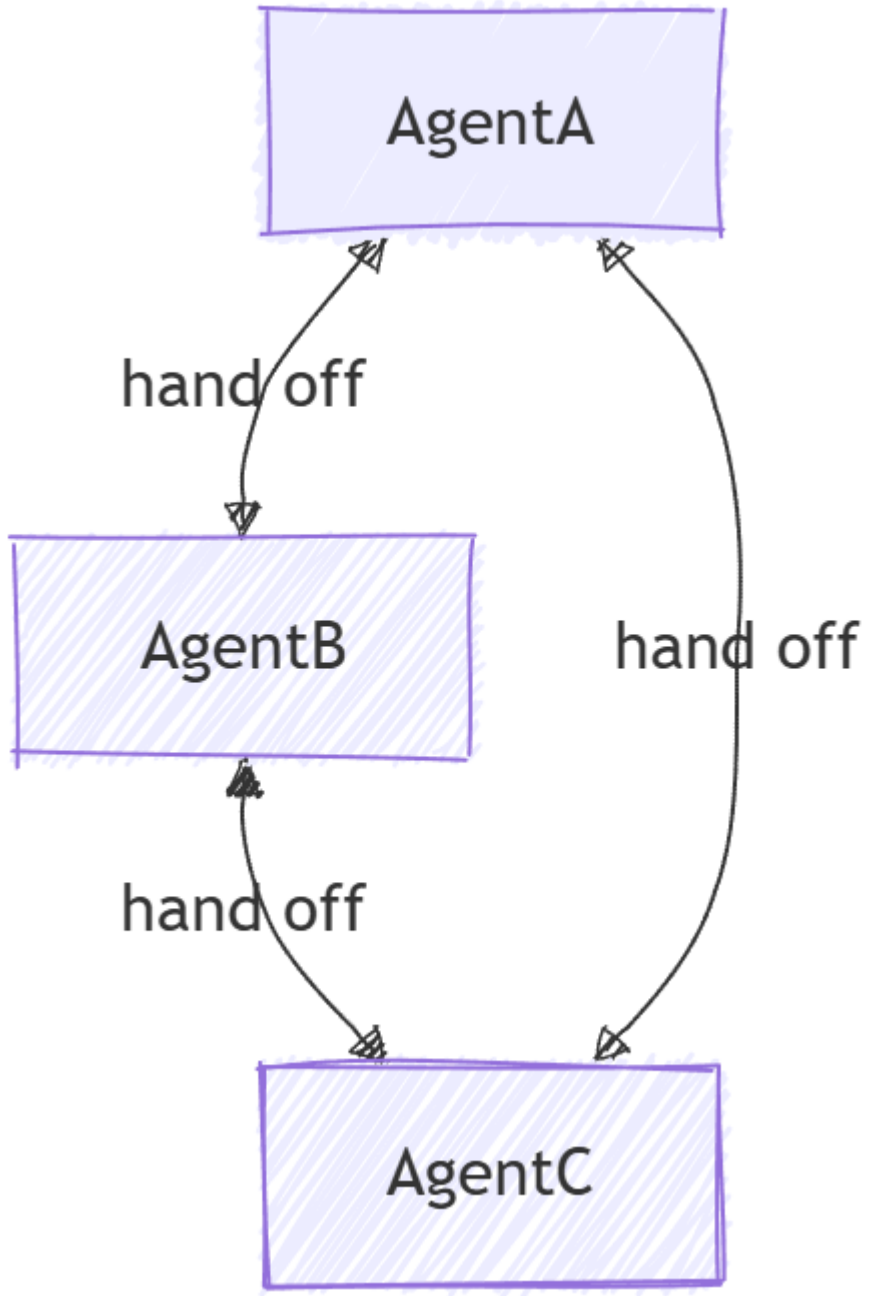


Group Chat

- Coordinación centralizada por un manager.
- Refinamiento iterativo: los agentes revisan y mejoran respuestas en rondas.
- Selección flexible de hablantes (round-robin, lógica personalizada, IA).
- Contexto compartido: todos los agentes ven el historial completo.
- **Aplicaciones ideales**
  - Creación de contenido colaborativo.
  - Resolución de problemas desde múltiples perspectivas.
  - Revisión y mejora de respuestas en equipo.

Handoff

- Patrón que permite transferir el control entre agentes dentro de un workflow, manteniendo contexto y continuidad.
- Aplicaciones típicas
  - Escenarios donde un agente prepara información y otro la procesa.
  - Flujos con múltiples etapas: clasificación → redacción → envío.
  - Integración de agentes con roles especializados.



Magentic

- Estructura del flujo
  - *Manager Magentic*: coordina agentes especializados según el contexto y progreso.
  - *Agentes participantes*: cada uno con capacidades específicas (ej. investigación, codificación).
  - *Contexto compartido*: todos los agentes acceden al historial y estado del flujo.
- Características clave
  - Selección dinámica del agente más adecuado en cada ronda.
  - Iteración y refinamiento de soluciones.
  - Detección de estancamientos y reinicio del plan si es necesario.
  - Soporte para revisión humana del plan (*human-in-the-loop*).

Hosting

Hosting

- Facilitar la integración de agentes de IA en aplicaciones ASP.NET Core mediante librerías de hosting.
- Core Hosting Library
  - Microsoft.Agents.AI.Hosting es la base para registrar y configurar agentes.
  - Extiende IHostApplicationBuilder para añadir agentes y workflows.
- Podemos publicar workflows como agentes
- Se incluyen dos librerías para diferentes escenarios de integración
  - A2A
  - OpenAPI

¿Preguntas?

DATA SATURDAYS MADRID

Microsoft Agent Framework

Antonio Soto

Sponsors:

alesonITC data thinking

NTT DATA

TD SYNNEX

encamina PIENSA EN COLORES

V-Valley enhancing your business

prodware

#DataSatMadrid