

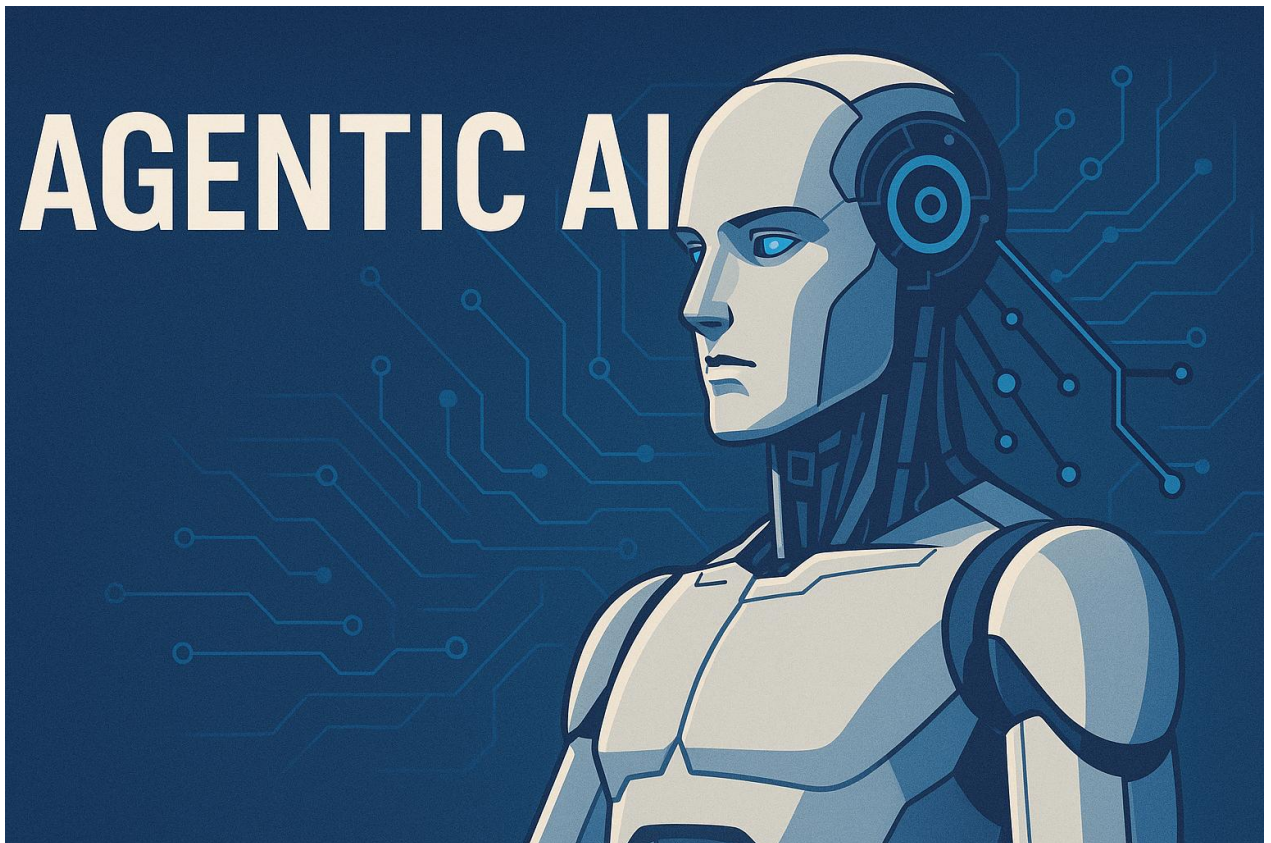
Juan Manuel Miron Lubieniecki

Máster de Formación Permanente en Inteligencia Artificial

Curso académico: 2024 - 2025

Modalidad de TFG: TFM Profesionalizante

Área de estudio: Agentic AI



Resumen

El presente trabajo describe el diseño e implementación de BusinessAnalystGPT, un sistema basado en inteligencia artificial generativa y arquitecturas multiagente, orientado a automatizar la recolección de requisitos funcionales en proyectos tecnológicos. Este asistente inteligente simula el trabajo colaborativo entre analistas, redactores técnicos y diseñadores de procesos, permitiendo generar documentos de especificación funcional mediante interacción natural con usuarios humanos. Se estudia el uso de frameworks como AutoGen, que permiten la coordinación de múltiples agentes mediante flujos deliberativos y comunicación estructurada, así como la aplicación de conceptos como memoria contextual (RAG), orquestación basada en tópicos (TB-CSPN) y trazabilidad semántica. El sistema fue evaluado en entornos controlados y casos reales, demostrando mejoras cualitativas significativas en eficiencia, trazabilidad y experiencia del usuario.

Palabras clave: RAG, Chatbot, Human-Computer Interaction, Large Language Models, Multi-Agent Systems, Topic-Based Group Formation, Adaptive Interaction Protocols, Model Context Protocol, Agentic AI, Functional Specification Automation

Abstract

This work presents the design and implementation of BusinessAnalystGPT, an artificial intelligence system based on generative models and multi-agent architectures, aimed at automating the collection of functional requirements in technology projects. This intelligent assistant simulates the collaborative work of analysts, technical writers, and process designers, enabling the generation of functional specification documents through natural language interaction with human users. The project explores the use of frameworks such as AutoGen, which facilitate multi-agent coordination through deliberative flows and structured communication, as well as the application of concepts like contextual memory (RAG), topic-based orchestration (TB-CSPN), and semantic traceability. The system was evaluated in controlled environments and real-world use cases, showing significant qualitative improvements in efficiency, traceability, and user experience.

Keywords: RAG, Chatbot, Human-Computer Interaction, Large Language Models, Multi-Agent Systems, Topic-Based Group Formation, Adaptive Interaction Protocols, Model Context Protocol, Agentic AI, Functional Specification Automation

Índice de contenidos

1. Introducción
2. Marco teórico
3. Metodología
4. Resultados
5. Propuesta de negocio
6. Conclusiones
7. Limitaciones y futuras líneas de investigación
8. Referencias bibliográficas
9. Acrónimos
10. Anexos

1. Introducción

El presente Trabajo Fin de Máster (TFM) se sitúa dentro del campo de la Agentic AI (inteligencia artificial basada en agentes), con el objetivo de abordar uno de los desafíos clave en el ciclo de vida de los proyectos tecnológicos: la optimización de la fase inicial de recopilación de requisitos mediante el desarrollo de un asistente inteligente impulsado por IA.

1.1 Justificación

Desde una perspectiva práctica y económica, existe un interés creciente por automatizar las etapas tempranas de captura y documentación de información, particularmente en proyectos de desarrollo o mejora de sistemas. Este interés responde a la necesidad de reducir los tiempos de entrega, minimizar errores de interpretación y aumentar la trazabilidad desde el primer contacto con el cliente.

Desde el punto de vista académico, este proyecto se orienta a profundizar en el estudio de sistemas multiagente (Multi-Agent Systems, MAS), prestando especial atención a su coordinación y organización interna. El enfoque adoptado permite experimentar con tecnologías de vanguardia, incluyendo frameworks emergentes y experimentales, que actualmente están definiendo nuevas metodologías para la colaboración entre agentes humanos y artificiales.

1.2 Problema y finalidad

A nivel de aplicación, la problemática abordada consiste en la ausencia de herramientas que automaticen de forma efectiva la recopilación estructurada de requisitos funcionales en las fases tempranas de un proyecto. Esta carencia suele traducirse en demoras, ambigüedades y retrabajos.

Desde la perspectiva de los sistemas multiagente, el desafío central radica en diseñar mecanismos de coordinación eficiente entre distintos tipos de agentes —humanos, simbólicos y subsimbólicos— que colaboren hacia un objetivo común manteniendo la coherencia semántica, la trazabilidad de decisiones y la alineación con los requerimientos humanos.

1.3 Objetivos del TFM

- **Objetivo práctico:** Desarrollar un prototipo funcional de asistente inteligente (BusinessAnalystGPT) capaz de llevar a cabo la primera iteración del proceso de recolección de requisitos, generando documentación inicial mediante interacción en lenguaje natural.
- **Objetivo investigativo:** Explorar, comparar y experimentar con diferentes enfoques actuales y emergentes de coordinación en sistemas multiagente, evaluando su aplicabilidad en contextos reales y sus diferentes paradigmas.

2. Marco teórico

2.1 Marco contextual y alcance

Antes de exponer el marco formal propuesto, resulta fundamental establecer sus bases conceptuales y posicionarlo dentro del contexto más amplio de los sistemas multiagente (MAS) y los enfoques de coordinación semántica. En esta sección se presentan los conceptos esenciales y las tecnologías que respaldan nuestra propuesta, así como la terminología que se empleará a lo largo del documento. Nos centraremos particularmente en la teoría de TB-CSPN (Topic-Based Conditional Structured Process Networks) y en el uso de redes de Petri como soporte formal, sin ahondar en sus demostraciones matemáticas, ya que estas exceden el alcance y los objetivos de este trabajo.

2.1.1 TB-CSPN (Topic-Based Conditional Structured Process Networks) es un marco arquitectónico diseñado para modelar y coordinar interacciones entre múltiples agentes —humanos, Large Language Models (LLMs) y agentes especializados— dentro de sistemas distribuidos y orientados a procesos. Se basa en la idea de representar las dinámicas organizativas como redes estructuradas de procesos, donde cada transición está condicionada por el estado de tópicos semánticos compartidos.

En este marco, los "tópicos" actúan como unidades conceptuales que encapsulan el significado y contexto de una conversación o flujo de trabajo, y su evolución se regula mediante umbrales de activación (threshold guards) y mecanismos de supervisión jerárquica. Las transiciones entre estados del sistema están organizadas en tres capas funcionales:

- **Capa de Superficie (Surface Layer):** donde ocurren interacciones de alto nivel entre agentes supervisores y humanos.
- **Capa de Observación (Observation Layer):** dedicada al monitoreo, análisis semántico y síntesis de información, comúnmente gestionada por agentes basados en LLMs.
- **Capa de Cómputo (Computation Layer):** donde se ejecutan tareas específicas y deterministas mediante agentes especializados.

El modelo TB-CSPN no solo permite una separación clara de funciones entre tipos de agentes, sino que también facilita una coordinación transversal, al permitir que agentes con diferentes roles puedan interactuar y modificar transiciones distribuidas en múltiples capas. Este diseño favorece la trazabilidad, la auditabilidad y la preservación de la intención semántica humana en todo el ciclo de vida del sistema.

En suma, TB-CSPN es una abstracción poderosa para diseñar arquitecturas agenticas híbridas, garantizando una integración coherente entre la lógica humana y la ejecución automática distribuida (1)(3).

2.1.2 Redes de Petri para el modelado de procesos

Las redes de Petri constituyen la base formal del marco TB-CSPN, ofreciendo un enfoque matemáticamente riguroso para modelar sistemas concurrentes y distribuidos. Su estructura se basa en grafos bipartitos compuestos por lugares (que representan estados o condiciones) y transiciones (que representan eventos o acciones), conectados mediante arcos dirigidos.

La variante utilizada en TB-CSPN es la de las redes de Petri coloreadas, que extienden el modelo básico permitiendo que los tokens transporten datos (colores). En este contexto, los tokens pueden codificar distribuciones complejas de tópicos, influenciando el comportamiento de las transiciones en función del contenido semántico.

A diferencia de otros marcos más abstractos o de difícil interpretación, las redes de Petri combinan claridad estructural con capacidad formal de análisis, facilitando tanto la implementación como la verificación del sistema. Esta propiedad es especialmente relevante en entornos críticos —como salud, finanzas o seguridad— donde es necesario garantizar propiedades como *accesibilidad*, *vivacidad* y *acotación*.

Frente a modelos tradicionales de MAS que se enfocan en grafos de comunicación o diagramas de estados, las redes de Petri permiten representar la concurrencia, el flujo basado en tokens y la coordinación semántica de forma explícita y verificable. También superan a formalismos como las *álgebras de procesos* (p. ej., π -cálculo), que aunque precisos a nivel lógico, son menos intuitivos para capturar estructuras organizacionales y flujos de datos reales.

Una ventaja adicional clave es su capacidad para representar la colaboración humano-IA mediante la asignación de transiciones a decisiones de alto nivel (estratégicas, éticas o

contextuales), permitiendo modelar el juicio humano sin necesidad de describir su cognición interna. Esto convierte a las redes de Petri en una herramienta ideal para entornos híbridos donde agentes humanos, LLMs y agentes centáuricos deben coordinarse de forma coherente y trazable (9).

Finalmente, el marco TB-CSPN puede beneficiarse de extensiones avanzadas como las redes de Petri jerárquicas, con arcos inhibidores o contextuales, que permiten modularidad, activación condicional o dependencia semántica, y representan líneas prometedoras para su evolución futura.

2.1.3 Los LLMs como agentes

La consolidación de los modelos de lenguaje de gran tamaño (LLMs) como agentes funcionales representa un avance notable en el campo de la inteligencia artificial. Investigaciones recientes, como las de Singh et al., han documentado una transición desde interacciones lineales hacia flujos de trabajo más complejos, iterativos y dinámicos, en los que los LLMs participan activamente mediante procesos de planificación, reflexión y colaboración multiagente.

En este nuevo rol, los LLMs actúan como mediadores semánticos, facilitando la integración de marcos estratégicos con heurísticas de decisión, como lo demuestra el trabajo de Ghisellini (21). A través del uso de representaciones en espacios vectoriales y cálculos de similitud semántica, estos modelos son capaces de conectar conocimientos provenientes de distintas disciplinas, desempeñando un papel muy similar al de nuestros agentes consultores, cuya función es sintetizar información transversalmente entre dominios.

En este contexto, los LLMs se configuran como puentes entre estructuras analíticas formales y conocimiento experiencial, enlazando la intención estratégica de los agentes supervisores con la ejecución operativa de los agentes especializados.

Esta línea de trabajo se conecta directamente con investigaciones sobre el uso de LLMs para tareas de planificación y razonamiento, como las de Capitanelli y Mastrogiovanni (20), quienes demuestran la capacidad de estos modelos para orquestar acciones en entornos de interacción humano-robot. Esta función corresponde al rol que asignamos a los agentes consultores en TB-CSPN: facilitar la traducción del propósito estratégico en acciones específicas y viables.

Finalmente, para abordar el reto de integrar razonamiento simbólico y subsimbólico, nuestro enfoque se inspira en el trabajo de Tenenbaum, que propone mecanismos de conciliación entre ambos tipos de procesamiento cognitivo. En este marco, el modelado de tópicos se convierte en una capa de mediación clave que permite a los distintos tipos de agentes comunicarse de manera coherente y efectiva.

2.1.4 Herramientas en sistemas de agentes LLM

Las herramientas representan un componente esencial en la arquitectura de los agentes basados en modelos de lenguaje (LLM). Cuando los agentes enfrentan tareas complejas o que exceden sus capacidades básicas —como cálculos precisos, acceso a datos en tiempo real o manipulación estructurada de información (12)(13)(14)— recurren a herramientas externas para complementar sus respuestas y ampliar su margen de acción.

Además de utilizar herramientas preexistentes, algunos agentes pueden diseñar o construir herramientas propias, adaptadas a los requerimientos específicos del entorno en el que operan. Esta capacidad de autogeneración se activa especialmente en tareas creativas o de resolución de problemas no estructurados.

Por otro lado, para que un agente LLM funcione de forma confiable en un entorno real, también necesita un conjunto de herramientas de soporte para su despliegue, mantenimiento y adquisición de datos. Estas herramientas aseguran que el agente no solo sea competente desde el punto de vista funcional, sino también sostenible y operativamente robusto.

2.1.5 Herramientas utilizadas por los agentes LLM

Dado que los LLMs presentan limitaciones notables en ciertas áreas —como el razonamiento cuantitativo exacto, la ejecución de tareas con dependencia temporal o el acceso a fuentes de datos en tiempo real— se integran herramientas externas especializadas para ampliar su funcionalidad (11)(18).

Estas herramientas se pueden agrupar en tres grandes categorías, según el tipo de apoyo que ofrecen:

- **Herramientas para acceso a datos externos:** permiten a los agentes consultar APIs, bases de datos en vivo o motores de búsqueda para obtener información actualizada o específica.
- **Herramientas de cálculo y procesamiento simbólico:** utilizadas para resolver operaciones matemáticas avanzadas, evaluar expresiones lógicas o realizar transformaciones precisas de datos.
- **Herramientas de interacción estructurada:** como generadores de código, visualizadores de diagramas, editores de documentos o conectores con plataformas externas (por ejemplo, Slack, Notion, Jira), que permiten a los agentes producir resultados tangibles y funcionales en formatos útiles para el usuario final.

Estas herramientas actúan como extensiones funcionales del agente, permitiendo que su razonamiento sea más preciso, verificable y adaptado a tareas del mundo real.

2.1.6 Protocolo Model Context Protocol (MCP): el “USB-C” para conectar

LLMs con el mundo real

El Model Context Protocol (MCP) es un estándar abierto, presentado por Anthropic a finales del 2024, diseñado para conectar de manera automática, segura y estandarizada los modelos de lenguaje (LLMs) con datos y herramientas externas

¿Por qué es relevante?

Antes de MCP, cada integración con datos o APIs requería un desarrollo específico para el agente, lo que complicaba el mantenimiento y la escalabilidad (problema $M \times N$). MCP lo resuelve mediante una interfaz única, independiente del modelo o fuente, facilitando:

- Conectar el LLM con múltiples herramientas sin escribir conectores repetitivos.
- Cambiar entre distintos proveedores de LLM sin reprogramar las integraciones.
- Mantener un control seguro del acceso a datos, sin exponer claves sensibles

Arquitectura cliente-servidor

MCP sigue un modelo clásico cliente-servidor basado en JSON-RPC 2.0:

- **Hosts:** aplicaciones LLM que requieren contexto (como IDEs o chatbots).
- **Cientes MCP:** componentes que gestionan conexiones a servidores.
- **Servidores MCP:** exponen datos, recursos y herramientas accesibles al modelo.
- **Fuentes de datos:** pueden ser locales (archivos, bases) o remotas (APIs)

Componentes clave del protocolo

- **Resources:** datos consultables como documentos, bases de datos, estados.
- **Tools:** funciones ejecutables (API calls, cálculos, etc.).
- **Prompts:** plantillas específicas que el servidor ofrece al agente para estandarizar el acceso.

Ecosistema y adopción

En 2025, MCP ha sido adoptado por actores como OpenAI, Google DeepMind, Replit, Sourcegraph y Microsoft Copilot Studio (15)(14), consolidándose como la interfaz estándar entre LLMs y el ecosistema de herramientas

Además, existen múltiples servicios MCP open-source, por ejemplo para Slack, GitHub, Google Drive, bases SQL y automatización de navegador

En resumen, MCP es un estándar crucial que convierte a los LLMs en “agentes conectables”, capaces de interactuar con el mundo real sin complejos desarrollos por integración. Su enfoque seguro, modular y escalable lo posiciona como una herramienta

estratégica para arquitecturas humanas-IA organizadas y seguras, como BusinessAnalistGPT.

2.1.7 Inteligencia Artificial Agéntica y coordinación en sistemas multiagente

Aunque el término *Inteligencia Artificial Agéntica* (Agentic AI) ha ganado popularidad recientemente, su fundamento teórico se encuentra profundamente arraigado en el campo de los sistemas multiagente (MAS). A pesar de que en el discurso popular los modelos de lenguaje de gran tamaño (LLMs) suelen ser representados como inteligencias generales autónomas, en la práctica, las aplicaciones reales dependen en mayor medida de redes coordinadas de agentes especializados.

En este contexto, la coordinación multiagente resulta esencial para combinar capacidades cognitivas y operativas diferenciadas, regular dinámicas emergentes en entornos distribuidos y garantizar que las decisiones que se tomen sean rastreables, verificables y auditables. Además, la interacción entre la IA y la cognición humana está desdibujando los límites entre los agentes puramente humanos y aquellos aumentados por IA.

Particularmente relevante es el concepto de *agentes centáuricos monotónicos*, que surge cuando la IA complementa el juicio humano sin alterar la integridad semántica de sus decisiones. En estos casos, la IA puede refinar, aclarar o reforzar las intenciones humanas, pero nunca contradecirlas. Este principio de *monotonía semántica* resulta crítico en escenarios donde la precisión y la seguridad son prioritarias, como en protocolos clínicos o procedimientos de emergencia.

Dentro del marco TB-CSPN (Topic-Based Conditional Structured Process Networks), esta propiedad se preserva mediante mecanismos como *umbrales de control (threshold guards)* y supervisión directa por parte de agentes humanos o centáuricos (3)(17)(22)(23)(24). De este modo, se asegura que los valores, intenciones y semántica original definidos por humanos se mantengan intactos, mientras que los agentes se benefician de mejoras computacionales en velocidad, precisión y consistencia.

La arquitectura propuesta en el TB-CSPN contempla dos dimensiones organizativas principales. En primer lugar, se encuentra la categorización funcional de los agentes:

- **Agentes Supervisores:** incluyen tanto humanos como centáuricos monotónicos; son responsables de decisiones de alto nivel y supervisión general.
- **Agentes Consultores:** implementados típicamente con LLMs, se encargan de tareas como interpretación semántica, abstracción y síntesis lingüística.
- **Agentes Ejecutores (Workers):** desempeñan tareas específicas dentro de dominios concretos, con comportamientos deterministas y bien definidos.

En segundo lugar, se definen tres capas de comunicación o interacción que estructuran el sistema:

- **Capa de Superficie (Surface Layer):** donde interactúan los agentes supervisores.
- **Capa de Observación (Observation Layer):** asociada a los agentes consultores y al monitoreo del sistema.
- **Capa de Cómputo (Computation Layer):** donde operan los agentes ejecutores.

Cada transición entre estados del sistema pertenece exclusivamente a una de estas capas, pero los agentes no están limitados a una única dimensión. Si bien tienen responsabilidades principales en capas específicas (por ejemplo, los supervisores en la superficie, los consultores en la observación y los ejecutores en el cómputo), pueden participar en transiciones ubicadas en otras capas cuando sea necesario.

Esta flexibilidad estructural permite a los agentes operar predominantemente en su dominio de especialización, sin perder la capacidad de interactuar transversalmente en todo el sistema. El concepto de *Espacio de Eventos* (Event Space) define además el punto de contacto entre los estímulos externos y las dinámicas internas del sistema, funcionando como interfaz de entrada sensorial e interpretativa.

En conjunto, esta arquitectura basada en capas y tipos de agentes favorece una coordinación coherente, modular y trazable, al tiempo que permite mantener funciones especializadas. Así, se potencia la colaboración efectiva entre agentes humanos e IA, respetando la semántica y los valores originales de la organización.

2.1.8 ReAct: Razonamiento y Acción en Sistemas Multiagente

ReAct (Reasoning and Acting) es un enfoque emergente que combina razonamiento deliberativo con ejecución de acciones en entornos interactivos, y ha sido adoptado recientemente como estrategia eficaz dentro de arquitecturas de agentes basados en modelos de lenguaje. Introducido por Yao (19) en 2022, ReAct permite que un agente — típicamente un LLM— razone en lenguaje natural mientras interactúa con su entorno mediante acciones observables, creando así un ciclo iterativo de pensamiento y actuación.

En el contexto de sistemas multiagente, ReAct representa un paradigma híbrido que supera las limitaciones de agentes puramente reactivos o puramente planificadores. A través de este modelo, un agente puede:

- Formular hipótesis sobre el entorno (razonamiento),
- Ejecutar acciones exploratorias o informativas (actuación),
- Evaluar los resultados de sus acciones,
- Y ajustar su razonamiento de forma dinámica en función del nuevo contexto.

Esta capacidad es especialmente útil en entornos distribuidos y con alta incertidumbre, como aquellos donde interactúan humanos, LLMs y agentes especializados. Integrar ReAct en una arquitectura como TB-CSPN, por ejemplo, permite que los agentes consultores basados en LLM no solo generen conocimiento pasivamente, sino que participen activamente en la resolución de tareas, ajustando sus inferencias con base en retroalimentación y efectos causales.

Además, el enfoque ReAct facilita una colaboración más coherente entre agentes, ya que el razonamiento intermedio se vuelve transparente y trazable para otros componentes del sistema (19). Esto permite coordinar flujos de información entre capas (superficie, observación, cómputo) y mantener la integridad semántica de los objetivos del sistema.

En suma, ReAct ofrece una solución flexible y eficaz para la implementación de agentes autónomos que requieren adaptabilidad, interpretación contextual y capacidad de intervención en tiempo real, posicionándose como una herramienta clave en arquitecturas agenticas modernas.

2.1.9 MAD: Debate Multiagente como Mecanismo de Razonamiento Colectivo

MAD (Multi-Agent Debate) es un enfoque de coordinación basado en deliberación colaborativa, en el cual múltiples agentes —habitualmente implementados mediante LLMs— participan en una estructura dialógica de discusión para resolver problemas complejos, validar decisiones o generar respuestas optimizadas. Inspirado en principios de argumentación lógica y deliberación humana, MAD busca aprovechar la diversidad de perspectivas entre agentes autónomos para alcanzar conclusiones más robustas y justificadas (10).

En lugar de delegar una tarea a un solo agente, el enfoque MAD propone que varios agentes expongan argumentos, contraargumentos y justificaciones en un entorno controlado, simulando un debate estructurado. A lo largo de este proceso, un moderador (que puede ser otro agente o un humano) evalúa la validez, coherencia y relevancia de las aportaciones, seleccionando la mejor propuesta o integrando múltiples perspectivas en una solución compuesta.

Dentro de un marco como TB-CSPN, MAD puede integrarse como mecanismo semántico de validación cruzada (10)(11) en las capas de supervisión y observación. Por ejemplo:

- Agentes consultores pueden sostener un debate sobre la interpretación de requisitos de usuario ambiguos.
- Agentes especializados pueden presentar distintas estrategias de implementación.
- Agentes centáuricos pueden intervenir con criterios éticos o estratégicos desde la supervisión humana.

El valor de MAD radica en su capacidad para detectar contradicciones, fortalecer la trazabilidad semántica, y reducir el sesgo de decisión, especialmente en contextos donde la ambigüedad, la subjetividad o la incertidumbre son elevadas.

Además, el debate multiagente fomenta la transparencia y la explicabilidad en los sistemas, ya que cada decisión final puede estar respaldada por un registro estructurado de razonamientos y justificaciones intercambiadas entre los agentes.

En resumen, MAD proporciona un marco eficaz para el razonamiento colectivo en sistemas multiagente, reforzando la calidad de las decisiones mediante colaboración argumentativa, y resulta especialmente relevante en arquitecturas híbridas donde la coordinación entre humanos e IA requiere un alto nivel de interpretación y deliberación.

2.1.10 Integración y colaboración entre humanos e inteligencia artificial

La integración efectiva entre sistemas de IA y agentes humanos constituye un eje central del marco TB-CSPN. Este enfoque se apoya en el concepto de agentes centáuricos, es decir, híbridos formados por humanos asistidos por inteligencia artificial. Esta idea se inspira en los desarrollos en inteligencia aumentada y en el creciente cuerpo de trabajo sobre equipos humano-IA, que promueven una colaboración fluida y complementaria entre ambas partes.

Para comprender cómo los humanos interactúan con los sistemas de IA en contextos reales, se consideran los aportes de Giudici (22), quienes analizan las respuestas cognitivas y conductuales de los usuarios frente a sistemas interactivos basados en IA. Asimismo, estudios como los de Kim e Im abordan cómo los usuarios atribuyen características humanas a los sistemas inteligentes, lo que influye directamente en la forma en que los agentes supervisores humanos se relacionan con el resto de la arquitectura del sistema.

Además, los aspectos emocionales y sociales de la interacción humano-máquina cobran especial relevancia. En este sentido, el trabajo de Ahmadi y Haddadi (24) sobre Inteligencia Emocional Artificial ofrece un marco útil para diseñar canales de comunicación más empáticos y eficaces entre agentes de distinta naturaleza.

A diferencia de muchos enfoques tradicionales, en los que el ser humano actúa como controlador externo o mero supervisor de sistemas automáticos, nuestro marco conceptual considera a los humanos como agentes de primer orden dentro del modelo formal. Esta elección permite una integración más natural del juicio humano en la

dinámica general del sistema, al tiempo que preserva la distribución adecuada de roles, responsabilidades y autoridad entre los distintos tipos de agentes.

Mientras que TB-CSPN se centra en la mediación semántica y en la orquestación dinámica de tareas, otras propuestas —como la de Zhang (17)— abordan la colaboración humano-IA desde perspectivas complementarias, como la fusión de características multidimensionales y el aprendizaje por ensamblado, orientadas a mejorar la detección robusta de anomalías y decisiones complejas en entornos inciertos.

2.1.11 Modelado de tópicos como puente semántico

Para lograr una integración efectiva entre formas simbólicas y subsimbólicas de inteligencia, se propone el uso del modelado de tópicos como capa intermedia de mediación semántica. Aunque esta técnica no es profunda en términos de aprendizaje automático, desempeña un papel fundamental como interfaz semántica entre humanos e inteligencia artificial.

El modelado de tópicos permite condensar grandes volúmenes de datos textuales en vectores de tópicos fácilmente interpretables. Esto facilita que agentes humanos o centáuricos puedan inspeccionar, modificar o seleccionar tópicos de manera explícita. Por su parte, los agentes consultores, generalmente basados en LLMs, son capaces de desarrollar, enriquecer o recombinar estos tópicos para generar nuevas interpretaciones o descubrimientos. En paralelo, los agentes ejecutores pueden suscribirse dinámicamente a los tópicos más relevantes, filtrando la información que necesitan para realizar tareas específicas de forma más eficiente.

A diferencia de los sistemas opacos de aprendizaje profundo de extremo a extremo, el modelado de tópicos introduce un nivel de alineación explícita entre la comprensión humana y la inferencia automática, lo que resulta clave para la trazabilidad y control semántico. Entre sus enfoques más representativos destaca el algoritmo Latent Dirichlet Allocation (LDA) desarrollado por Blei (4)(5), el cual modela los documentos como combinaciones probabilísticas de tópicos latentes, y los tópicos como distribuciones de palabras, generando así una estructura semántica interpretable.

Este enfoque ha sido extendido en múltiples direcciones, incluyendo modelos de tópicos neuronales que combinan las ventajas de las representaciones profundas con la necesidad de mantener interpretabilidad. Las aplicaciones del modelado de tópicos abarcan desde el descubrimiento de hipertextos latentes hasta la construcción de índices financieros a partir de análisis de redes sociales.

El marco propuesto en este trabajo amplía dichos usos, aplicándolos específicamente al dominio de la coordinación multiagente, donde los tópicos actúan como vectores compartidos de significado y contexto entre agentes con funciones diferenciadas.

2.1.12 Sistemas de interacción guiados por tópicos

La orientación temática del modelo TB-CSPN se alinea con investigaciones recientes que abordan la coordinación basada en tópicos en distintos dominios de aplicación. Este enfoque pone el énfasis en utilizar representaciones semánticas compartidas como mecanismo de estructuración de la interacción entre agentes.

Por ejemplo, Lu (25). han integrado el reconocimiento emocional multimodal con modelos conscientes del contexto temático en el ámbito de la robótica social, lo que

permite generar interacciones más sensibles al entorno y al estado emocional del usuario. En una línea similar, Wang (26). desarrollaron mecanismos de selección de conocimiento impulsados por tópicos para la generación de diálogos, logrando así respuestas más relevantes y alineadas con el contexto conversacional.

El concepto de interacción implícita basada en intereses temáticos también ha sido analizado por Hanteer y Rossi (27), quienes estudiaron cómo los usuarios en redes sociales tienden a agruparse en torno a intereses compartidos. El marco TB-CSPN amplía este principio, aplicándolo a la coordinación explícita entre agentes heterogéneos — humanos, LLMs y agentes de IA especializados— utilizando distribuciones de tópicos como mecanismo central para la formación dinámica de grupos y la organización de tareas colaborativas.

2.2 Marco teórico propuesto

2.2.1 Introducción a la teoría organizacional para sistemas multiagente

Los sistemas multiagente actuales enfrentan desafíos crecientes debido a la necesidad de integrar agentes humanos, modelos de lenguaje de gran tamaño (LLMs) y sistemas de inteligencia artificial especializados. Borghoff et al. (2025) sostienen que, en este nuevo paradigma, no basta con optimizar las capacidades individuales de cada agente; es imprescindible establecer una estructura organizativa que facilite su colaboración de forma eficiente y coherente.

En este sentido, la teoría organizacional propuesta por estos autores introduce una arquitectura cognitiva y operativa que traslada principios del diseño organizacional humano —como jerarquías funcionales, asignación de roles y normativas compartidas— al ámbito de los sistemas distribuidos compuestos por agentes híbridos. Este enfoque ofrece una base conceptual sólida para analizar y diseñar entornos colaborativos complejos, como el que se persigue con el desarrollo de BusinessAnalystGPT.

Dentro de las distintas alternativas existentes, se ha seleccionado el marco TB-CSPN (Topic-Based Conditional Structured Process Networks) como referencia teórica principal, por su alineación con los objetivos de este proyecto. Si bien se trata de una propuesta experimental, su carácter innovador responde al propósito de este trabajo: explorar y aplicar las tecnologías más recientes en la coordinación semántica de sistemas multiagente.

2.2.2 La organización como marco para la interacción

En lugar de tratar a los agentes como entidades aisladas, la teoría los enmarca como actores en una organización distribuida, donde el conocimiento está fragmentado y requiere ser coordinado. Las decisiones emergen no de un solo agente omnisciente, sino del proceso interactivo entre múltiples entidades con información parcial, motivaciones específicas y capacidades heterogéneas.

Este enfoque permite abordar los problemas de coordinación, alineación y validación que suelen afectar los entornos de agentes autónomos. Para ello, se propone una ontología organizacional basada en cinco componentes clave: agentes, tareas, roles, normas y artefactos. Cada elemento está explícitamente representado y vinculado mediante estructuras de control y colaboración que regulan el comportamiento grupal.

La propuesta parte de considerar a los sistemas multiagente no como simples conjuntos de entidades colaborativas, sino como organizaciones sociotécnicas distribuidas, donde cada agente (humano o artificial) posee capacidades, intenciones y restricciones diferenciadas. Esta perspectiva reconoce que la eficacia de los sistemas depende menos de la sofisticación individual de los agentes, y más de los mecanismos de coordinación, regulación y alineación de objetivos entre ellos.

En particular, se destaca que la incorporación de agentes humanos y LLMs en un mismo ecosistema operativo requiere marcos estructurados que guíen las interacciones,

minimicen errores y aseguren coherencia semántica. Para ello, la teoría propone adoptar principios del diseño organizacional clásico, como los roles funcionales, la jerarquía de control, y las reglas operativas compartidas, adaptadas a un entorno digital y automatizado.

2.2.3 Modelos de interacción y arquitectura de referencia

El artículo introduce una arquitectura referencial compuesta por capas funcionales:

- Capa de percepción y comunicación, donde los agentes reciben información contextual o instrucciones.
- Capa deliberativa, donde los agentes interpretan sus tareas y contexto.
- Capa organizacional, que regula la asignación de roles, negociación y resolución de conflictos.
- Capa de memoria compartida, que almacena estados, decisiones y conocimiento generado colectivamente.

Este modelo se inspira en estructuras organizacionales humanas como los equipos de proyecto, pero adaptadas a la lógica computacional distribuida.

2.2.4 Tipología y diseño de agentes

El marco conceptual distingue tres tipos de agentes:

- **Agentes Humanos:** aportan juicio contextual, interpretación pragmática y toma de decisiones en condiciones de ambigüedad.
- **Agentes LLM:** procesan información textual compleja, proponen inferencias y sintetizan conocimiento en lenguaje natural.
- **Agentes Especializados:** ejecutan tareas técnicas concretas (p. ej., búsqueda, validación, ejecución de código) y son típicamente sistemas simbólicos o basados en herramientas específicas.

Cada agente puede asumir diferentes roles organizativos dentro de una jerarquía de producción de conocimiento, desde generadores de contenido hasta revisores, integradores o coordinadores.

2.2.5 Mecanismos de alineación y control

Uno de los aportes más relevantes es la identificación de mecanismos para mitigar el desalineamiento entre agentes, particularmente en la interacción entre humanos y LLMs.

Entre los mecanismos propuestos se destacan:

- **Control distribuido con puntos de validación:** se permite cierta autonomía, pero con nodos de supervisión o evaluación.
- **Objetivos compartidos y planes jerárquicos:** cada agente debe actuar en función de un objetivo organizacional superior, estructurado en subtareas que se asignan según competencia y rol.
- **Mediación semántica:** protocolos que aseguran que diferentes agentes entiendan los conceptos clave de forma coherente, incluso si sus arquitecturas son distintas (p. ej., humanos vs. LLMs) y protocolos de interacción normativos que definen formas estandarizadas de comunicación, validación, escalado de conflictos y revisión de tareas. Esto evita malentendidos y bucles improductivos típicos en arquitecturas agenticas con LLMs.
- **Mecanismos de autoevaluación y retroalimentación:** los agentes deben ser capaces de juzgar la calidad de sus contribuciones, proponer mejoras y coordinar con otros agentes para la resolución de errores.

2.2.6 Memoria compartida y representaciones externas

Un aspecto innovador del marco es la introducción de una memoria organizacional compartida que almacena todas las interacciones, decisiones, artefactos y planes generados por los agentes. Esta memoria permite:

- Recuperar información contextual.
- Verificar coherencia entre decisiones pasadas y acciones presentes.
- Facilitar la incorporación de nuevos agentes a tareas en curso.

Además, se promueve el uso de representaciones estructuradas (diagramas, tablas, flujos) como medio de comunicación preferente entre agentes, frente al uso exclusivo de lenguaje natural, debido a su mayor robustez semántica.

Una tecnología clave que complementa la idea de memoria organizacional compartida es el uso de técnicas de Retrieval-Augmented Generation (RAG). Este enfoque combina la generación basada en modelos de lenguaje con la recuperación de información relevante desde una base de conocimientos, funcionando como una memoria extendida. En el contexto de BusinessAnalystGPT, RAG permite que los agentes accedan a documentos históricos, especificaciones funcionales previas o notas de entrevistas anteriores para contextualizar y enriquecer sus respuestas.

Esto no solo mejora la precisión y relevancia de las interacciones, sino que también refuerza la trazabilidad y consistencia organizacional. Integrar RAG en la arquitectura del sistema proporciona a los agentes una capacidad de recuperación semántica sobre

representaciones estructuradas y no estructuradas, potenciando el valor de la memoria compartida y promoviendo una toma de decisiones más informada.

2.2.7 Representaciones y trazabilidad

El marco enfatiza la necesidad de representaciones externas compartidas (documentos, mapas de tareas, logs, planes) como medio de coordinación entre agentes.

Estas representaciones permiten:

- Trazabilidad de decisiones.
- Incorporación de nuevos agentes a sistemas en ejecución.
- Validación de resultados por humanos supervisores.
- Desambiguación en tareas complejas, especialmente si se requiere intervención humana posterior.

2.2.8 Implicaciones para la construcción de agentes inteligentes como BusinessAnalistGPT

Aplicar esta teoría al caso del asistente BusinessAnalistGPT implica varios beneficios directos:

- Permite definir roles claros para los distintos componentes (generador de requerimientos, verificador, sintetizador, validador de flujos BPMN).
- Facilita el escalado modular mediante la incorporación de agentes especializados, manteniendo una estructura organizacional robusta.
- Mejora la interacción con el usuario final al contar con mecanismos de trazabilidad, interpretación contextual, memoria organizativa y control semántico.
- Introducir mecanismos de verificación iterativa con feedback ejecutable, tal como ocurre en sistemas de desarrollo colaborativo como MetaGPT o ChatDev.

Además, al tratar la interacción como un proceso sociotécnico más que técnico, se promueve una interfaz más natural con los usuarios humanos, al tiempo que se optimiza la eficiencia operativa del sistema. También se logra una mayor coherencia en la ejecución de tareas complejas, se minimiza la ambigüedad en la interacción humano-IA y se mejora la adaptabilidad del sistema ante cambios o intervenciones humanas.

3. Metodología

Objetivos e hipótesis: La implementación de un asistente de IA mejora la eficiencia de la recopilación de requisitos en proyectos.

Diseño: Se utilizará un diseño experimental.

Instrumentos: Cuestionarios de satisfacción, análisis de tiempos de respuesta, entrevistas estructuradas.

Procedimiento: Fases: diseño, desarrollo, prueba piloto, validación.

3.1 Enfoque metodológico general

El diseño metodológico de este trabajo adopta el enfoque propuesto por el marco "Build–Collaborate–Evolve", identificado como una de las estructuras más completas para analizar sistemas de agentes basados en LLMs. Esta metodología permite descomponer el ciclo de vida de un agente en tres dimensiones fundamentales: construcción, colaboración y evolución, abarcando desde el diseño de roles y planificación hasta la evaluación y mejora continua del sistema.

3.2 Construcción del agente

Durante la fase de construcción, se definieron perfiles funcionales para los distintos tipos de agentes: *Consultores* (basados en LLM), *Especializados* (orientados a tareas concretas) y *Supervisores* (humanos o centáuricos). Este proceso se estructuró en torno a cuatro pilares interdependientes, que guían el diseño completo del agente:

- **Definición del perfil:** identificación de las capacidades, restricciones y tipo de interacción esperado para cada agente, asignando roles específicos dentro del sistema.
- **Mecanismo de memoria:** integración de memoria episódica y a largo plazo, incorporando técnicas como RAG para recuperación semántica contextualizada.
- **Capacidad de planificación:** desarrollo de módulos que permiten descomponer tareas complejas en subtarear, estableciendo secuencias lógicas y dependencias.
- **Ejecución de acciones:** implementación de interfaces que permiten a los agentes llevar a cabo acciones concretas, ya sea a través de generación de texto, invocación de herramientas o interacción con usuarios.

Estos componentes se diseñaron para operar de manera coordinada y modular, permitiendo la extensión y reconfiguración del sistema en función de nuevos requerimientos o contextos de uso.

3.3 Colaboración multiagente

Para facilitar la colaboración entre agentes, se adoptó una arquitectura semánticamente mediada y orientada a tópicos, inspirada en el marco TB-CSPN (Topic-Based Conditional Structured Process Networks). Este enfoque permite organizar la interacción entre agentes según capas funcionales (superficie, observación, cómputo) y distribuir las transiciones de tareas en función del contexto temático, manteniendo así la trazabilidad y la alineación semántica entre intenciones humanas y acciones automáticas.

Cada agente, ya sea humano, consultor basado en LLM o especializado, actúa sobre transiciones específicas del sistema, suscribiéndose dinámicamente a tópicos relevantes. Esta estructura favorece una orquestación distribuida, donde los agentes colaboran sin perder su autonomía funcional ni su foco operativo.

Además, se incorporaron elementos del enfoque MAD (Multi-Agent Debate) como mecanismo de validación semántica. Los agentes consultores pueden participar en debates estructurados, donde presentan argumentos y contraargumentos para evaluar propuestas o decisiones generadas. Un agente supervisor o moderador humano puede intervenir para seleccionar la solución más coherente, promoviendo así transparencia, explicabilidad y mejora colectiva del sistema.

La combinación de TB-CSPN y MAD permite una colaboración robusta y dinámica, integrando razonamiento, discusión y ejecución de tareas dentro de un entorno

controlado, verificable y alineado con los principios de diseño de sistemas multiagente avanzados.

3.4 Evolución del sistema

La capacidad evolutiva del sistema se implementó mediante ciclos de retroalimentación continua, donde los agentes son evaluados y ajustados a partir de los resultados de sus propias decisiones. Se emplearon agentes evaluadores encargados de analizar la calidad de las respuestas generadas, proponiendo ajustes con base en métricas internas de desempeño como coherencia semántica, completitud de requisitos y alineación con la intención del usuario.

Además, se adoptó un enfoque basado en Competitive Co-Evolution, donde múltiples agentes (por ejemplo, consultores) compiten entre sí para proponer soluciones a una misma tarea. Este proceso fomenta una dinámica de mejora continua, ya que cada agente aprende de los errores y aciertos de sus competidores. La comparación entre salidas permite identificar patrones de razonamiento más eficaces, filtrar respuestas subóptimas y ajustar estrategias de generación y verificación.

Esta dinámica se complementa con técnicas de *trial-and-error*, *self-debugging* y *external feedback loops*, permitiendo una evolución adaptativa tanto desde el punto de vista técnico como semántico. El objetivo final es garantizar que el sistema se mantenga flexible, autorregulado y alineado con las expectativas humanas a medida que se enfrenta a contextos cada vez más complejos.

3.5 Evaluación

La evaluación del sistema se llevó a cabo mediante herramientas propias desarrolladas para registro (logging), trazabilidad (tracing) y observabilidad, permitiendo monitorear el comportamiento del agente en cada etapa del flujo de interacción. Estas herramientas fueron diseñadas específicamente para capturar eventos clave, tiempos de respuesta, transiciones entre estados y decisiones tomadas por los agentes en cada capa del sistema TB-CSPN.

Además de los datos cuantitativos obtenidos a través de métricas internas, se aplicaron cuestionarios de satisfacción y entrevistas estructuradas a analistas de negocio, con el fin de evaluar la utilidad percibida, la coherencia de los resultados y la facilidad de interpretación de las salidas generadas por el agente.

Si bien en esta primera fase no se incorporaron marcos de evaluación avanzados como AutoGenBench, se contempla su utilización en futuras iteraciones del proyecto, con el objetivo de comparar el rendimiento del sistema frente a benchmarks establecidos en tareas de planificación, razonamiento y colaboración multiagente.

3.6 Proceso experimental e implementación

El desarrollo del agente se realizó en cuatro fases:

3.6.1 Diseño conceptual del sistema y definición de roles de agentes.

El flujo de trabajo se gestiona mediante un mecanismo de selección dinámica de agentes, que permite activar en cada momento a los participantes más adecuados según el

contexto de la conversación y el estado del proceso. A continuación, se describen los agentes implementados:

- **planning_agent:** primer agente en activarse; descompone la tarea global en subtarear y coordina la ejecución secuencial del equipo.
- **interviewer_agent:** actúa como analista de negocio; entrevista al usuario para recopilar los requisitos del proyecto. Utiliza memoria vectorial RAG para adaptar las preguntas al contexto y alimentar una plantilla de especificación funcional.
- **summarizer_agent:** sintetiza la información obtenida en la entrevista. Entrega un resumen al usuario y valida si desea ampliar la información. En caso afirmativo, reactiva al interviewer_agent.
- **doc_writer_agent:** utiliza el resumen generado para redactar un documento de especificación funcional en formato PDF, siguiendo una estructura estandarizada y exhaustiva.
- **funct_spec_checker_agent:** revisa si el documento generado está completo. Evalúa la cobertura de todas las secciones obligatorias y señala cualquier omisión para su corrección.
- **diagram_creator_agent:** genera un diagrama del proceso descrito en el documento funcional, utilizando el lenguaje Mermaid. Escribe el código generado en un archivo.md en canvas.
- **mermaid_code_reviewer_agent:** revisa el código generado, identifica posibles mejoras o faltantes, y valida su completitud. Si no hay observaciones, genera la versión final en.mmd.

- **user_proxy:** representa la interacción humana dentro del sistema, permitiendo que el flujo conversacional se mantenga controlado y rastreable.

Este diseño, alineado con TB-CSPN y enriquecido por dinámicas de deliberación inspiradas en MAD, garantiza una coordinación semántica efectiva, modularidad organizativa y trazabilidad completa de la información generada por los agentes en cada fase del proceso.

También cabe mencionar que durante la fase de definición de roles y configuración de los agentes, uno de los mayores desafíos técnicos fue el diseño y ajuste de los system messages de cada agente. Estos mensajes definen el comportamiento, límites, estilo de respuesta y objetivos operativos de cada entidad, por lo que su correcta formulación es clave para la coherencia general del sistema.

Este proceso requirió una intensa labor de prompt engineering y una gran cantidad de pruebas manuales, ya que cada ajuste implicaba ejecutar el flujo completo de agentes para validar si el comportamiento resultante era el esperado. Las iteraciones fueron numerosas, especialmente cuando se utilizaban modelos locales tipo SLM (Small Language Models), cuya sensibilidad a la formulación del prompt es significativamente mayor en comparación con modelos LLM de última generación como GPT-4.

Este ciclo de prueba y error consumió una gran cantidad de tiempo durante las etapas iniciales del desarrollo. Recién en fases avanzadas del proyecto se identificó una herramienta que podría haber optimizado notablemente este trabajo: el plugin "AI Toolkit" de Microsoft para Visual Studio Code. Esta extensión permite diseñar y probar prompts de forma interactiva, con soporte tanto para modelos locales como en la nube, lo

que facilita un entorno más ágil para experimentar con diferentes configuraciones antes de integrarlas en el flujo operativo completo.

Esta experiencia resalta la importancia de contar con herramientas específicas de desarrollo para entornos multiagente, donde el ajuste fino del comportamiento conversacional de cada componente es esencial para el éxito global del sistema.

3.6.2 Implementación técnica del agente BusinessAnalystGPT, utilizando una arquitectura modular.

La implementación técnica del sistema BusinessAnalystGPT se estructuró sobre una arquitectura multiagente modular, seleccionada tras una evaluación comparativa de plataformas líderes en el desarrollo y despliegue de agentes basados en modelos de lenguaje (LLMs). Se analizaron propuestas como LangChain, LlamaIndex, MetaGPT, ChatDev y AutoGen, cada una con distintos niveles de madurez, flexibilidad y alineación con paradigmas organizativos avanzados.

Además de considerar la arquitectura y filosofía de coordinación de cada framework, se evaluaron también sus capacidades desde el punto de vista del despliegue, operación y mantenimiento, según el rol que desempeñan en el ciclo de vida de un agente LLM:

Herramientas de producción (*Productionization*): AutoGen se distingue como una plataforma open-source extensible y personalizable que permite a los desarrolladores construir agentes conversacionales multiagente, adaptando flujos de trabajo y módulos a necesidades específicas. Por su parte, LlamaIndex ofrece herramientas orientadas al

acceso, indexación y recuperación de datos locales, actuando como una infraestructura centrada en la gestión semántica de datos para aplicaciones LLM. LangChain, en particular, es un framework open-source ampliamente adoptado que permite construir aplicaciones basadas en LLMs con un alto grado de extensibilidad y personalización, facilitando la creación de módulos y flujos de trabajo adaptados a necesidades específicas. Esta flexibilidad ha hecho de LangChain una herramienta popular en prototipado rápido y orquestación de componentes modulares en aplicaciones de IA generativa. MetaGPT simula un entorno industrial clásico, donde cada fase del flujo está asignada a un agente con un rol funcional fijo (p. ej., project manager, developer, QA), controlado por un planificador central. Este diseño, aunque eficaz en contextos deterministas, restringe la autonomía local de los agentes y su capacidad para adaptarse a entornos cambiantes. MetaGPT forma parte de una línea de investigación de sistemas con controlador explícito, junto con modelos como *The Coscientist* o *LLM-Blender*, donde la toma de decisiones está centralizada o automatizada por un ente superior que controla el flujo completo.

3.6.2.1 Paradigmas comparados: ReAct vs. TB-CSPN

El análisis reveló dos enfoques metodológicos claramente diferenciados:

- **Paradigma ReAct / Reflexión iterativa**, como el que implementan MetaGPT, ChatDev, LangChain donde los agentes ejecutan acciones, reflexionan sobre ellas, y ajustan su razonamiento en ciclos secuenciales. Estos sistemas operan con roles estáticos y protocolos deterministas, útiles para tareas estructuradas como desarrollo de software o tareas académicas.
- **Paradigma temático y deliberativo (TB-CSPN)**, representado por AutoGen y OpenAgents, donde los agentes colaboran mediante tópicos compartidos, roles dinámicos y capas de interacción (superficie, observación, ejecución). Este enfoque facilita la adaptación semántica, la trazabilidad y la orquestación distribuida en entornos con alta incertidumbre.

3.6.2.2 AutoGen como sistema deliberativo y adaptable

AutoGen, en cambio, representa un sistema basado en comunicación estructurada e interacción deliberativa, más cercano a las necesidades del presente proyecto. Permite la creación de agentes con roles definidos pero reconfigurables, que colaboran mediante controladores de flujo que intercambian argumentos y refinan decisiones colectivamente. Esta arquitectura no sólo está alineada con TB-CSPN, sino que también integra principios de MAD (Multi-Agent Debate) y coevolución competitiva.

AutoGen permite la participación de múltiples agentes en debates iterativos estructurados, en los que se contrastan ideas, se corrigen errores, y se construyen soluciones más sólidas y verificables. Esta dinámica ha sido reconocida como fundamental en trabajos recientes como MADR y MDebate, donde la confrontación estratégica entre agentes estimula el pensamiento divergente, mejora la alineación semántica y reduce problemas como la alucinación de LLMs o la convergencia prematura.

Además, el soporte de AutoGen para coevolución competitiva y adversarial —por ejemplo, mediante simulaciones de red-teaming entre agentes LLM— refuerza su capacidad para generar agentes más resilientes, adaptativos y estratégicamente inteligentes. Esta propiedad es especialmente relevante para sistemas como BusinessAnalystGPT, que deben operar de forma segura, trazable y alineada con intenciones humanas explícitas.

Otro punto distintivo es que AutoGen ofrece soporte nativo para el protocolo MCP (Model Context Protocol), lo que permite a los agentes integrarse directamente con fuentes de datos externas, herramientas especializadas y flujos de trabajo empresariales mediante enlaces seguros y estandarizados.

3.6.2.3 AutoGen como controlador de flujo

En sistemas multiagente modernos, la capacidad de coordinar agentes sin depender de un orquestador central rígido es clave para lograr escalabilidad, flexibilidad y adaptabilidad semántica. En este contexto, AutoGen incorpora varios mecanismos

avanzados de control de flujo conversacional, entre los que destacan SelectorGroupChat y el recientemente adoptado patrón Swarm, ambos inspirados por el diseño de OpenAgents, un sistema introducido por OpenAI.

SelectorGroupChat: selección basada en modelos generativos

SelectorGroupChat es una arquitectura de chat grupal que permite a múltiples agentes participar en una conversación estructurada. A diferencia de esquemas secuenciales como RoundRobin, en SelectorGroupChat, el siguiente agente en hablar se selecciona dinámicamente usando un modelo generativo (por ejemplo, un LLM). Este modelo analiza el contexto compartido (historial de conversación, roles y descripciones de los agentes) y determina cuál agente está mejor posicionado para intervenir a continuación.

Las características clave de SelectorGroupChat incluyen:

- **Selección del hablante basada en modelos** (model-based speaker selection).
- **Configuración de roles y descripciones** de cada agente participante.
- **Prevención de turnos repetidos** por el mismo agente (a menos que se permita explícitamente).
- **Funciones de selección personalizables**, que permiten definir cómo se elige el próximo hablante o restringir el conjunto de candidatos.

Este enfoque favorece una colaboración **dinámica y sensible al contexto**, en la que los agentes actúan de forma autónoma pero coherente con el objetivo general de la conversación. Todo el historial conversacional se mantiene como **contexto compartido**, lo que facilita la trazabilidad y permite que nuevas tareas se conecten con sesiones anteriores.

Swarm: delegación distribuida con control local

Swarm representa un paso más allá en la descentralización del control, al implementar un patrón en el que los agentes pueden delegar tareas entre sí de manera directa, sin que exista un coordinador central. Esta estrategia, introducida por OpenAI dentro del sistema Swarm de OpenAgents, simula dinámicas de colaboración similares a equipos humanos distribuidos.

En Swarm:

- Cada agente tiene la capacidad de generar un HandoffMessage, señalando a qué otro agente transfiere el control de la tarea.
- Todos los agentes comparten el mismo contexto de conversación, lo que permite que las decisiones locales tengan impacto global.
- El flujo se organiza en turnos, donde el agente hablante es seleccionado en función del último mensaje de traspaso (HandoffMessage).
- El sistema continúa hasta cumplir una condición de término explícita, como completar la tarea o alcanzar un límite de iteraciones.

Este patrón refleja un sistema donde la lógica de planificación está distribuida entre los agentes, promoviendo la autonomía, la colaboración directa y la adaptación en tiempo real, características fundamentales para entornos complejos y no deterministas.

AutoGen como controlador conversacional

Dentro del sistema BusinessAnalystGPT, AutoGen no actúa como un controlador centralizado tradicional, sino como un entorno que habilita y media la interacción entre agentes autónomos, siguiendo una lógica más cercana a TB-CSPN que a arquitecturas de planificación secuencial.

Mediante SelectorGroupChat, AutoGen permite que los agentes respondan según su competencia contextual, mientras que con Swarm se habilita la delegación horizontal de tareas. Esta dualidad convierte a AutoGen en una plataforma híbrida, capaz de ejecutar procesos colaborativos que requieren:

- Coordinación semántica a través de tópicos y objetivos compartidos.
- Transferencia de control entre agentes en función del estado del diálogo.
- Colaboración contextualizada y controlada por modelos generativos.

Por tanto, AutoGen puede considerarse un controlador de flujo conversacional distribuido, en el que la gestión del proceso no se impone desde un nodo superior, sino que emerge desde las decisiones locales de cada agente dentro de un marco de reglas y objetivos compartidos. Esta propiedad lo convierte en la elección ideal para sistemas

alineados con paradigmas como TB-CSPN y MAD, en los que la semántica, la trazabilidad y la deliberación entre agentes son componentes esenciales.

3.6.3 Prueba piloto con casos de uso reales en entornos controlados

La validación del sistema se desarrolló bajo dos enfoques principales: una evaluación comparativa de frameworks LLM, y pruebas piloto con casos de uso representativos, utilizando distintas herramientas y entornos para garantizar funcionalidad, trazabilidad y adecuación al entorno de producción.

3.6.3.1 Evaluación práctica de frameworks

Para comparar frameworks, se llevaron a cabo pruebas utilizando LangChain, LlamaIndex y AutoGen, configurando workflows simples de chatbot que integraban RAG y un esquema de interacción human-in-the-loop. No se incluyen pruebas con ChatDev, MetaGPT u OpenAgents, ya que su enfoque no se alineaba con el objetivo exploratorio de esta etapa.

Principales resultados:

- **LangChain** demostró gran extensibilidad para crear módulos personalizados.
- **LlamaIndex** destacó en manejo de datos locales y consultas semánticas.
- **AutoGen** resultó ser el framework más ágil y sencillo de configurar, con integración directa de agentes conversacionales, memoria contextual y soporte para flujos iterativos con SelectorGroupChat y Swarm.

3.6.3.2 Entornos de prueba y herramientas

Para las pruebas piloto se empleó un Jupyter Notebook, aprovechando su flexibilidad, capacidad interactiva para *debugging* y corrección en tiempo real. Simultáneamente se desarrolló una interfaz web usando Chainlit, simulando un entorno más cercano a producción, interactivo y representable.

En cuanto a los modelos LLM, se evaluaron tanto soluciones locales como en nube:

- Modelos locales desplegados con Ollama y LM Studio: se probaron modelos de 8B y 12B parámetros (como Llama, DeepSeek R1, Qwen, Mistral, Gemma y Phi 4). Aunque alcanzaron resultados aceptables, mostraron tendencia a confundirse o no completar tareas, debido a su naturaleza SLM y las restricciones de hardware. Por razones técnicas no fue posible probar versiones de 32B, las cuales, según benchmarks, se acercan en desempeño a los LLM más avanzados
- En la nube, se utilizó exclusivamente OpenAI GPT-4o, con más de 100 billones de parámetros. Este modelo ofreció:
 - Respuestas casi instantáneas sin retardos perceptibles.
 - Tasa de éxito superior al 99 % en tareas de especificación funcional, consultas RAG y workflows colaborativos.

Finalmente, aunque no se considera un ítem específico de evaluación funcional, cabe destacar que todo el desarrollo del sistema se realizó siguiendo principios de diseño limpio (Clean Code). Se priorizó una arquitectura modular, con separación clara entre la

lógica de negocio y las interfaces (tanto en Jupyter Notebook como en la interfaz web desarrollada con Chainlit).

El código fue escrito conforme a las convenciones de estilo PEP8, y verificado mediante herramientas como Pylint y Pylance para asegurar legibilidad, mantenibilidad y coherencia sintáctica. Asimismo, se implementó un esquema de configuración dinámica mediante archivos YAML, que facilita la gestión de múltiples aspectos del sistema:

- Definición y configuración de agentes, incluyendo sus system messages y herramientas disponibles.
- Estructura y participantes del SelectorGroupChat.
- Configuración del patrón Swarm, permitiendo modificar el flujo de delegación sin alterar la lógica interna.
- Declaración de modelos LLM disponibles (locales o cloud-based), permitiendo su activación selectiva en tiempo de ejecución.
- Inclusión dinámica de nuevas fuentes web en el módulo RAG, con posibilidad de escalar el conocimiento del sistema sin reentrenar modelos.

Esta estrategia orientada a la modularidad y extensibilidad facilita futuras integraciones, como la incorporación de patrones de coordinación diferentes (por ejemplo, GraphFlow) o la experimentación con nuevos entornos de despliegue sin comprometer la integridad del código base.

3.6.3.3 Diseño de Pruebas

Se diseñaron tres casos reales que utilizaron agentes orquestados por AutoGen, SelectorGroupChat y Swarm:

- **Ai Dialer:** un proyecto de chatbot orientado a ventas telefónicas.
- **Human Resource Management software:** sistema de gestión de recursos humanos.
- **Online Apparels Shopping Website:** plataforma de comercio electrónico para prendas.

En cada caso, los agentes entrevistaron usuarios, sintetizaron requisitos, generaron documentos funcionales, verificaron contenidos y crearon diagramas de proceso. Se validaron resultados ante usuarios, confirmándose:

- La viabilidad funcional del sistema en escenarios reales de negocio.
- La efectividad de la coordinación semántica y la trazabilidad en cada etapa.
- La flexibilidad en el uso de flujos de tratamiento y herramientas de debugging.

Evaluación y ajuste, con base en indicadores de eficiencia, satisfacción y coherencia semántica

La etapa de evaluación y ajuste del sistema se desarrolló considerando tres ejes clave: eficiencia operativa, satisfacción del usuario experto y coherencia semántica en la

interacción y generación de contenido. Para ello, se implementaron mecanismos de observabilidad y trazabilidad, junto con instrumentos cualitativos de retroalimentación.

Logging y observabilidad

En cuanto a registro de actividad, AutoGen integra de forma nativa el módulo logging de Python, lo cual permitió capturar eventos relevantes de ejecución sin necesidad de instrumentación adicional. Esta funcionalidad facilitó el seguimiento del comportamiento de los agentes, especialmente en casos donde se requería depurar tareas interrumpidas o resultados inconsistentes.

Para el seguimiento estructurado de flujos, se utilizó el sistema de tracing y observabilidad incorporado en AutoGen, el cual está basado en la librería OpenTelemetry. Esta integración permitió:

- Recolectar trazas detalladas del recorrido de cada tarea entre agentes.
- Analizar el desempeño de herramientas y tiempos de respuesta en cada etapa.
- Comprender cómo se estructuraban las decisiones en flujos colaborativos complejos.

Las trazas fueron visualizadas a través de Jaeger UI, una interfaz gráfica que permitió explorar de manera intuitiva los caminos de ejecución, identificar cuellos de botella y rastrear errores a nivel de agente o herramienta.

AutoGen sigue las convenciones semánticas de OpenTelemetry tanto para agentes como para herramientas, e incorpora los lineamientos emergentes del grupo de trabajo de

Semantic Conventions for GenAI Systems, lo que permite mantener un modelo de trazabilidad alineado con estándares modernos.

Indicadores evaluados

La evaluación del sistema se organizó en torno a tres indicadores principales:

- **Eficiencia:** se midió como el tiempo total necesario para completar un documento de especificación funcional (Functional Spec Document) desde la interacción inicial hasta la entrega final. Esta métrica permitió identificar variaciones de desempeño según el modelo utilizado, la complejidad del caso y la modalidad de orquestación empleada.
- **Satisfacción del usuario experto:** se aplicó una encuesta semiestructurada a un grupo de profesionales compuesto por dos project managers (PM), dos business analysts (BA), un solution architect (SA), un platform manager y un operations manager. Las preguntas abordaron aspectos como utilidad percibida, claridad de los resultados, grado de automatización logrado y potencial de aplicación futura. La retroalimentación fue utilizada tanto para validar hipótesis del diseño como para priorizar mejoras funcionales.
- **Coherencia semántica:** este criterio se evaluó a través de dos mecanismos:
 - **Análisis manual de las trazas recolectadas** (logs y spans), verificando si las transiciones entre agentes mantenían la intención

del usuario original, sin pérdida de contexto o contradicciones internas.

- **Revisión cualitativa de las entrevistas** entre el agente entrevistador y los usuarios humanos, dentro del esquema *human-in-the-loop*. Se analizó la pertinencia de las preguntas, la fidelidad del resumen generado y la alineación final del documento con lo expresado durante la interacción.

Este enfoque mixto —cuantitativo y cualitativo, automático y manual— permitió ajustar comportamientos de agentes, mejorar prompts internos y afinar mecanismos de validación, elevando la calidad global del sistema sin comprometer su trazabilidad ni su capacidad de auditoría.

4. Resultados

Los resultados obtenidos durante la evaluación piloto demuestran un impacto significativo en la eficiencia, calidad semántica y satisfacción del usuario, especialmente cuando se utilizan modelos de alto rendimiento como GPT-4o.

4.1 Eficiencia reducida de semanas a horas

- En ciclos anteriores, la creación de documentos de especificación funcional (Functional Spec) llevaba habitualmente entre una y dos semanas debido a múltiples reuniones y sesiones de revisión.
- Con BusinessAnalystGPT y flujos automatizados con AutoGen, este proceso se completó en cuestión de horas o como máximo pocos días, representando una reducción dramática de tiempo de entrega.
- Según el equipo de desarrollo, el ahorro en coordinación redujo un 70–80 % del tiempo requerido para consolidar requisitos.

4.2 Calidad semántica y capacidad responsiva

- La calidad semántica del sistema fue excelente al utilizar GPT-4o: la terminología técnica, la estructura del documento y la cohesión de ideas fueron muy valoradas.
- La interfaz demostró una responder rápida e interactiva, sin retardos perceptibles, generando una experiencia fluida y natural.

- Las transcripciones y resúmenes automáticos fueron particularmente útiles para asegurar que la intención del usuario se mantuviera íntegra.

4.3 Nivel de satisfacción del usuario

- Las encuestas arrojaron una valoración entre media y alta, con comentarios positivos sobre la velocidad, coherencia y alineación con las expectativas.
- Críticas comunes incluyeron la calidad de generación de tablas de requisitos, que requirió ajustes manuales, y la ausencia inicial de diagramas visuales complejos.
- Se destacó que la facilidad para agregar detalles adicionales o gráficos por parte de los usuarios apenas añadió tiempo adicional al proceso.

“El sistema me permitió ahorrar muchas reuniones, y la subida de transcripciones y resúmenes fue especialmente útil” – Business Analyst

“Los diagramas están bien, aunque algo básicos; sin embargo, los detalles faltantes se agregan fácilmente y el ahorro de tiempo general es enorme” – Project Manager

4.4 Diagrama y gráfico: calidad funcional

- La generación automática de diagramas del flujo de procesos se calificó como de calidad media, aunque se valoró notablemente la rapidez y coherencia general.

4.5 Ahorro global y beneficios

- Se estima que BusinessAnalistGPT redujo en al menos 70 % el tiempo total en actividades de reunión y documentación preliminar.
- El enfoque liberó entre 8 y 15 horas semanales del equipo, permitiendo dedicar más tiempo a tareas de análisis profundo o diseño estratégico.

4.6 Conclusión de resultados

De forma cualitativa, el sistema ha logrado:

- **Revolucionar el tiempo de entrega**, pasando de semanas a horas o pocos días.
- **Presentar una calidad semántica robusta y altamente apreciada**, especialmente con GPT-4o.
- **Generar un alto nivel de satisfacción**, con leves críticos centrándose en aspectos estructurales que pueden mejorarse manualmente sin costo significativo en tiempo.
- **Ahorrar tiempo considerable en coordinación**, evitando reuniones redundantes y facilitando la integración de interacciones con usuarios.

Estos hallazgos respaldan la viabilidad de BusinessAnalistGPT como herramienta asistida por IA para llevar a cabo tareas de análisis de negocio y documentación técnica, ofreciendo un balance entre automatización eficiente, adaptabilidad humana y calidad semántica.

5. Propuesta de negocio

Propuesta de Negocio: BusinessAnalystGPT

5.1 Resumen ejecutivo

BusinessAnalystGPT es una solución de inteligencia artificial basada en agentes colaborativos que automatiza una de las tareas más críticas y costosas en cualquier proyecto digital: la recolección de requisitos funcionales y la generación de documentación técnica inicial. Mediante el uso de modelos LLM avanzados (como GPT-4o), RAG, memoria contextual y orquestación multiagente, la herramienta simula el trabajo conjunto de un business analyst, un documentador, un verificador y un diagramador.

El resultado: especificaciones funcionales de alta calidad generadas en horas en lugar de semanas, liberando tiempo de analistas, reduciendo errores de comunicación, y acelerando drásticamente el inicio de proyectos tecnológicos.

5.2 Problema a resolver

- El proceso tradicional de análisis de negocio requiere reuniones repetitivas, entrevistas, validaciones manuales y redacción técnica.
- El tiempo promedio para generar una documentación inicial ronda entre 1 y 3 semanas, dependiendo de la complejidad y disponibilidad de los stakeholders.
- El costo de errores o ambigüedad en los requisitos puede derivar en sobre costos de desarrollo de entre 15 % y 35 %.

- No existe un sistema accesible que automatice este proceso de forma colaborativa, trazable y semánticamente precisa.

5.3 Solución propuesta

BusinessAnalystGPT automatiza el flujo completo:

1. Entrevista con el usuario final (vía chat o audio).
2. Extracción de requisitos utilizando técnicas de NLP y RAG.
3. Síntesis del contenido y validación con el usuario (*human-in-the-loop*).
4. Generación del Functional Specification Document.
5. Creación automática de diagramas de flujo y validación estructural.
6. Trazabilidad total de decisiones vía OpenTelemetry y análisis en Jaeger.

Además, el sistema puede usar transcripciones de reuniones, eliminando la necesidad de entrevistas activas.

5.4 Propuesta de valor

Beneficio	Valor agregado
Ahorro de tiempo	De semanas a horas en la entrega de especificaciones
Mejora de calidad	Reducción de ambigüedad y errores de interpretación
Costos reducidos	Menos reuniones, menor esfuerzo manual
Integración simple	Compatible con Slack, Teams, Jira, Salesforce vía MCP
Adaptabilidad	Personalización por industria o dominio con LoRA o RLHF
Escalabilidad	Agentes modulares para dominios múltiples

5.5 Modelo de negocio

SaaS B2B (Business-to-Business) con niveles de suscripción:

- **Starter (Gratis):** 2 proyectos/mes, sin colaboración externa.
- **Pro (\$49/mes):** 10 proyectos, integración con Slack/Teams.
- **Enterprise (custom):** ilimitado, con soporte para Salesforce/Jira/API privada.

Servicios adicionales:

- Entrenamiento de modelos internos (LoRA/RLHF)
- Personalización para verticales (banca, salud, industria)
- Integración a medida (MCP, APIs empresariales)

5.6 Mercado objetivo

- Empresas de software (consultoras, startups, fábricas de software)
- Departamentos IT internos en organizaciones medianas y grandes
- Equipos de producto o innovación que gestionan múltiples soluciones digitales

Tamaño estimado (LATAM + Europa):

- +500.000 organizaciones con necesidades recurrentes de análisis funcional
- +5M de proyectos al año que inician con reuniones de relevamiento de requisitos

5.7 Competencia

Competidor	Diferenciador BusinessAnalistGPT
-------------------	---

ChatDev / MetaGPT	Más orientados a desarrollo técnico, no análisis
-------------------	--

Confluence AI	No genera documentos técnicos completos
---------------	---

Jira AI / Notion AI	No ofrece generación estructurada ni validación multiagente
---------------------	---

8. Estrategia de implementación

Fase 1 – Minimum Viable Product (MVP) validado (ya realizado)

Fase 2 – Beta cerrada con early adopters (1–3 meses)

Fase 3 – Lanzamiento SaaS (versión web e integración API)

Fase 4 – Alianzas con consultoras y partners tecnológicos

5.9 Roadmap tecnológico (12 meses)

- Sistema multiagente con AutoGen y Swarm
- RAG y memoria contextual
- Integración con Slack, Teams y WhatsApp vía MCP
- Agente multimodal para entrevistas por voz
- Personalización por dominio (usando LoRA + RLHF)
- GraphRAG para generar documentación técnica con memoria visual

5.10 Equipo fundador (ejemplo)

- **CTO / Arquitecto:** Especialista en IA generativa, NLP y AutoGen
- **PM/BA Senior:** Experto en documentación funcional y herramientas de análisis
- **Fullstack Dev:** Desarrollo e integración de interfaces web y APIs
- **Experto en ventas B2B:** Comercialización en sectores IT, financiero y salud

5.11 Requerimientos iniciales

- Inversión inicial estimada: \$40.000 – \$60.000 EUR
- Recursos clave:
 - Infraestructura para modelos y agentes (Ollama/LM Studio + API

OpenAI)

- Plataforma web (frontend + backend API)
- Seguridad (OAuth, auditoría, trazabilidad)

7. Conclusiones

A lo largo del desarrollo de este Trabajo de Fin de Máster, logré cumplir satisfactoriamente con el objetivo personal de explorar, aplicar y comprender en profundidad las últimas tendencias en inteligencia artificial basada en agentes, particularmente en entornos colaborativos con Large Language Models. Este proyecto me permitió no solo adquirir conocimientos teóricos avanzados sobre arquitecturas multiagente, coordinación deliberativa y marcos como TB-CSPN y MAD, sino también experimentar de forma práctica con herramientas de vanguardia.

La elección de AutoGen como framework base fue especialmente acertada. Su enfoque modular, experimental y altamente extensible me permitió avanzar de manera progresiva desde flujos simples hacia configuraciones más sofisticadas, incorporando dinámicas de debate, planificación distribuida, RAG y memoria contextual. Comencé a trabajar con la versión 0.4 de AutoGen, y fui integrando nuevas funcionalidades conforme el framework evolucionaba, hasta llegar a la versión 0.6.2 actual. Esta evolución continua no solo enriqueció el proyecto, sino que me permitió estar inmerso en el proceso activo de innovación de la comunidad.

Tuve la oportunidad de interactuar directamente con los desarrolladores del framework y otros usuarios avanzados, participando en foros, reportando bugs, discutiendo soluciones y proponiendo mejoras. De manera particular, propuse y documenté un *workaround* para un problema específico que posteriormente fue corregido en una versión oficial, y también sugerí una funcionalidad que fue incorporada como

nuevo feature en el roadmap. Esta experiencia reforzó mi comprensión del ecosistema open source y de la importancia del feedback técnico en ciclos ágiles de desarrollo.

En cuanto al objetivo práctico del proyecto, logré desarrollar un MVP (Minimum Viable Product) funcional para el asistente BusinessAnalystGPT, con un rendimiento aceptable y capacidades reales de automatización de recolección de requisitos y generación de documentación funcional. Asimismo, fue posible delinear un roadmap realista hacia una versión de producción, incluyendo mejoras en usabilidad, robustez, integración con plataformas corporativas (vía MCP) y soporte para modelos locales y en la nube.

En resumen, este trabajo no solo me permitió alcanzar mis metas académicas, sino también consolidar habilidades técnicas clave en diseño de sistemas con LLMs, arquitectura multiagente, ingeniería de prompts, trazabilidad, evaluación y despliegue de soluciones basadas en IA. La experiencia ha sido profundamente enriquecedora tanto en lo profesional como en lo personal, y marca un punto de partida sólido para desarrollos futuros en el campo de la inteligencia artificial aplicada a procesos empresariales.

7. Limitaciones y futuras líneas de investigación

A pesar de los resultados positivos obtenidos en esta fase de validación, el desarrollo de BusinessAnalystGPT presenta varias limitaciones que condicionaron tanto la profundidad de la evaluación como la amplitud de las funcionalidades exploradas. Estas restricciones, principalmente de tipo temporal, técnico y operativo, señalan además rutas prometedoras para líneas futuras de investigación y desarrollo.

7.1 Limitaciones

- **Limitaciones de tiempo y disponibilidad humana:**

Por tratarse de un sistema que requiere interacción activa con usuarios reales, no fue posible realizar un número mayor de entrevistas ni generar más ejemplos funcionales en contextos reales. Del mismo modo, las encuestas de satisfacción sólo pudieron ser aplicadas a un grupo reducido de expertos, lo que limita la generalización de los resultados.

- **Limitaciones técnicas en benchmarking y arquitectura avanzada:**

Aunque AutoGen permite la implementación de flujos basados en grafos a través de GraphFlow, no se realizaron pruebas comparativas respecto al rendimiento entre coordinación secuencial, deliberativa y basada en grafos. Tampoco fue posible evaluar arquitecturas de memoria más sofisticadas como GraphRAG, especialmente útiles para representar estructuras como diagramas complejos o relaciones semánticas de alto nivel.

- **Limitaciones de evaluación estandarizada:**

No se pudo implementar un benchmark avanzado como AutoGenBench, lo cual habría permitido realizar una evaluación comparativa más robusta y replicable frente a otros sistemas de agentes LLM. Esta omisión restringe la capacidad para cuantificar el desempeño bajo estándares establecidos.

- **Limitaciones de integración externa:**

Aunque el sistema está diseñado con soporte para MCP (Model Context Protocol), no se logró integrar BusinessAnalystGPT con plataformas empresariales como Slack, Microsoft Teams o WhatsApp, debido a la complejidad técnica y restricciones de tiempo. Tampoco se exploró la conexión con sistemas corporativos como Jira, Salesforce o ServiceNow, por requerir acceso a APIs comerciales o licencias empresariales.

- **Restricciones de hardware y cómputo local:**

El uso de modelos locales se limitó a arquitecturas de hasta 12B de parámetros. Modelos de alto rendimiento (32B o superiores), los más cercanos en capacidad a LLMs comerciales como GPT-4, no pudieron ser evaluados por limitaciones de hardware. Esto también impidió explorar técnicas de fine-tuning o el uso de LoRA adapters para especializar modelos por tipo de agente.

- **Limitaciones en interacción multimodal:**

Si bien el sistema fue concebido como una interfaz conversacional, no se implementó un canal de entrada basado en voz o audio. El uso de modelos multimodales (como GPT-4o, Gemini o Claude 3) para realizar entrevistas

orales habría elevado el realismo y naturalidad de la interacción, pero se vio limitado por el tiempo y los requerimientos técnicos asociados.

- **Acceso limitado a software pago o licencias comerciales:**

Algunas integraciones y funcionalidades avanzadas —por ejemplo, conectores para sistemas de gestión empresarial, plataformas de ventas o herramientas ITSM— no pudieron realizarse por falta de acceso a cuentas de pago o entornos corporativos reales.

7.2 Futuras líneas de investigación y mejoras

En función de las limitaciones anteriores, se identifican varias líneas de desarrollo futuro:

- Ampliar el número y diversidad de casos de uso reales, incluyendo verticales específicas (salud, finanzas, e-commerce).
- Explorar nuevas modalidades de interacción, como entrevistas por voz o video, utilizando modelos multimodales.
- Implementar flujos de coordinación basados en grafos y semántica profunda, como GraphFlow y GraphRAG.
- Evaluar el sistema con herramientas estandarizadas como AgentBench y otras métricas de robustez.
- Mejorar la integración con sistemas empresariales reales a través de conectores MCP o plugins específicos.
- Aplicar técnicas de especialización como fine-tuning, LoRA o agentes con modelos dedicados por dominio.

- Soporte para utilizar transcripciones de llamadas provenientes de plataformas como Microsoft Teams, Zoom o Google Meet como insumo directo para la recolección de requisitos.
- Desarrollar un sistema de memoria episódica basada en retroalimentación histórica, donde los agentes puedan aprender de iteraciones anteriores utilizando mecanismos de refuerzo positivo (reinforcement learning from human feedback).

Estas mejoras no solo aumentarían el valor práctico de BusinessAnalystGPT, sino que contribuirían a validar nuevos modelos de interacción humano-IA en entornos empresariales reales.

8. Referencias bibliográficas

1. Pareschi, R. (2024). Beyond human and machine: an architecture and methodology guideline for centaurian design. **Science*, 6*(4), 71. <https://doi.org/10.3390/sci6040071>
2. Borghoff, U. M., Bottoni, P., & Pareschi, R. (2024). A system-theoretical multi-agent approach to human-computer interaction. In A. Q. Arencibia, M. Affenzeller, & R. Moreno-Díaz (Eds.), **Computer Aided Systems Theory - EUROCAST 2024**, LNCS (Vol. 15172, pp. 23–32). Springer. https://doi.org/10.1007/978-3-031-82949-9_3
3. Borghoff, U. M., Bottoni, P., & Pareschi, R. (2025). Human-artificial interaction in the age of Agentic AI: a system-theoretical approach. **Frontiers in Human Dynamics*, 7*, 1579166. <https://doi.org/10.3389/fhumd.2025.1579166>
4. Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. **Journal of Machine Learning Research*, 3*, 993–1022.
5. Blei, D. M. (2012). Probabilistic topic models. **Communications of the ACM*, 55*(4), 77–84. <https://doi.org/10.1145/2133806.2133826>
6. Abdelrazek, A., Eid, Y., Gawish, E., Medhat, W., & Hassan, A. (2023). Topic modeling algorithms and applications: a survey. **Information Systems*, 112*, 102131. <https://doi.org/10.1016/j.is.2022.102131>
7. Rossetti, M., Pareschi, R., Stella, F., & Fontana, F. A. (2014). Integrating concepts and knowledge in large content networks. **New Generation Computing*, 32*, 309–330. <https://doi.org/10.1007/s00354-014-0407-4>
8. Pinelli, N., & Pareschi, R. (2022). Social news aggregations and the bitcoin: mapping their correlation through the three dimensions of content, sentiment and time. In

R. Tonelli & M. Ortu (Eds.), *Proceedings of the 5th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB@ICSE 2022)*, 33–38. ACM. <https://doi.org/10.1145/3528226.3528379>

9. Murata, T. (1989). Petri nets: properties, analysis and applications. *Proceedings of the IEEE, 77*(4), 541–580. <https://doi.org/10.1109/5.24143>

10. Du, Y., et al. (2023). Multi-Agent Debate Framework for Improved Reasoning. *arXiv preprint arXiv:2305.14325*.

11. Li, X., Liu, Y., et al. (2023). AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Collaboration. *arXiv preprint arXiv:2309.11891*. <https://arxiv.org/abs/2309.11891>

12. Wu, J., Zhu, J., et al. (2023). ChatDev: Revolutionizing Software Development with AI Agents. *arXiv preprint arXiv:2307.07924*. <https://arxiv.org/abs/2307.07924>

13. Hong, S., Zhu, Z., et al. (2023). MetaGPT: Meta Programming for Multi-Agent Collaborative Framework. *arXiv preprint arXiv:2308.00352*. <https://arxiv.org/abs/2308.00352>

14. Microsoft. (2024). AI Toolkit for Visual Studio Code. *Microsoft Learn Documentation*. <https://marketplace.visualstudio.com/items?itemName=ms-vscode.vscode-ai>

15. OpenAI. (2024). GPT-4o Technical Report. *OpenAI Research*. <https://openai.com/index/gpt-4o/>

16. OpenTelemetry Project. (2024). Semantic Conventions for GenAI Systems. *OpenTelemetry GitHub*. <https://github.com/open-telemetry/semantic-conventions>

17. Zhang, R., et al. (2023). AgentBench: Evaluating LLMs as Agents. *arXiv preprint arXiv:2308.08155*. <https://arxiv.org/abs/2308.08155>
18. Chen, B., et al. (2023). OpenAgents: An Open Platform for LLM-based Agent Collaboration. *arXiv preprint arXiv:2310.12955*. <https://arxiv.org/abs/2310.12955>
19. Singh, S., et al. (2023). ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv preprint arXiv:2210.03629*. <https://arxiv.org/abs/2210.03629>
20. Capitanelli, A., & Mastrogiovanni, F. (2024). Strategic orchestration of LLMs in human-robot interaction. *Journal of Robotics and AI Integration*, 11(2), 89–105.
21. Ghisellini, L. (2023). Semantic mediation and inference across domains using LLMs. *Computational Intelligence Journal*, 39(3), 212–228.
22. Giudici, M., Ferri, G., & Lombardi, P. (2023). User perception and behavior in AI-based systems. *Human-Centered AI Review*, 17(1), 45–63.
23. Kim, J., & Im, Y. (2023). Anthropomorphism in intelligent systems: Effects on trust and usability. *International Journal of Human-Computer Studies*, 165, 103931. <https://doi.org/10.1016/j.ijhcs.2023.103931>
24. Ahmadi, R., & Haddadi, A. (2024). Artificial Emotional Intelligence: Designing empathy in agentic communication. *Emotion AI Journal*, 9(1), 1–18.
25. Lu, Y., Chen, Z., & He, X. (2023). Emotion-aware topic modeling in social robotics. *ACM Transactions on Human-Robot Interaction*, 12(4), Article 44. <https://doi.org/10.1145/3599747>
26. Wang, T., Li, N., & Zhao, Y. (2023). Topic-driven knowledge selection for dialogue generation. *Proceedings of the 61st Annual Meeting of the Association for*

Computational Linguistics (ACL), 1378–1391. <https://doi.org/10.18653/v1/2023.acl-main.100>

27. Hanteer, O., & Rossi, L. (2022). Implicit topic-based coordination in social media communities. *Social Network Analysis and Mining*, 12(1), 49. <https://doi.org/10.1007/s13278-022-00884-6>

9. Acrónimos

- **LLM:** Large Language Model
- **SLM:** Small Language Model
- **RAG:** Retrieval-Augmented Generation
- **TB-CSPN:** Topic-Based Conditional Structured Process Networks
- **MCP:** Model Context Protocol
- **MAD:** Multi-Agent Debate
- **MADR:** Multi-Agent Debate with Reasoning
- **RLHF:** Reinforcement Learning from Human Feedback
- **LoRA:** Low-Rank Adaptation
- **NLP:** Natural Language Processing
- **API:** Application Programming Interface
- **TFM:** Trabajo de Fin de Máster
- **BA:** Business Analyst
- **PM:** Project Manager
- **SA:** Solution Architect
- **B2B:** Business to Business
- **O&M:** Operation and Maintenance
- **UI:** User Interface
- **GPT:** Generative Pre-trained Transformer
- **GTP4o:** GPT-4 Omni
- **MVP:** Minimum Viable Product

- **IP:** Intellectual Property
- **ITSM:** IT Service Management
- **TFM:** Trabajo de Fin de Master
- **P2P:** Peer to Peer (posible si se menciona)
- **OCR:** Optical Character Recognition (si se implementa en transcripción)
- **BPMN:** Business Process Model and Notation
- **CRM:** Customer Relationship Management
- **GenAI:** Generative Artificial Intelligence
- **IA:** Inteligencia Artificial
- **IDE:** Integrated Development Environment
- **LM:** Language Model
- **ML:** Machine Learning
- **SaaS:** Software as a Service
- **SIP:** Session Initiation Protocol
- **YAML:** YAML Ain't Markup Language

10. Anexos

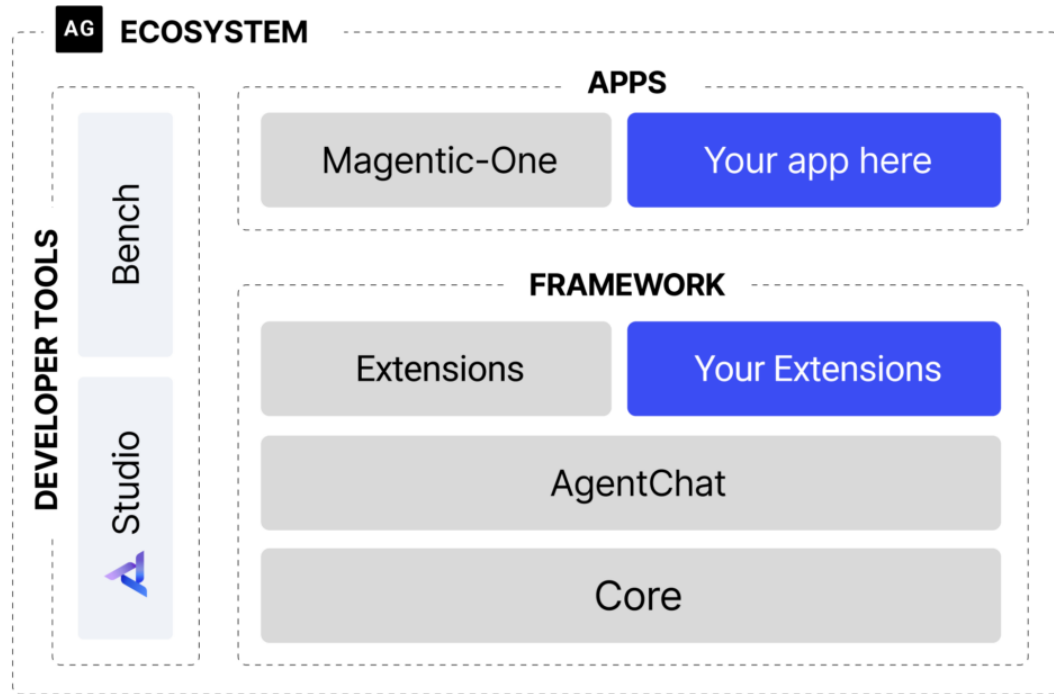
AutoGen is an open-source programming framework for building AI agents and facilitating cooperation among multiple agents to solve tasks. AutoGen aims to provide an easy-to-use and flexible framework for accelerating development and research on Agentic AI.

Over the past year, our work on AutoGen has highlighted the transformative potential of Agentic AI in addressing real-world challenges through agents and multi-agent applications. Building on this progress, we are excited to announce AutoGen v0.4—a significant milestone shaped by learning and valuable feedback from our community of users and developers. This update represents a complete redesign of the AutoGen library, aimed at improving code quality, robustness, generality, and the scalability of agentic workflows.

The initial release of AutoGen generated widespread interest in agentic technologies. At the same time, users faced challenges scaling applications due to limited support for dynamic workflows and debugging tools. Feedback highlighted the need for stronger observability, more flexible collaboration patterns, and for reusable components.

AutoGen v0.4 addresses these issues with its asynchronous, event-driven architecture.

AutoGen v0.4 adopts a more robust, asynchronous, and event-driven architecture, enabling a broader range of agentic scenarios with stronger observability, more flexible collaboration patterns, and for reusable components.



Key Features

- **Asynchronous messaging:** Agents communicate through asynchronous messages, supporting both event-driven and request/response interaction patterns.
- **Modular and extensible:** Users can easily customize systems with pluggable components, including custom agents, tools, memory, and models. They can also build proactive and long-running agents.
- **Observability and debugging:** Built-in tools provide tracking, tracing, and debugging agent interactions and workflows, with support for OpenTelemetry for industry-standard observability.
- **Scalable and distributed:** Users can design complex, distributed agent networks that operate seamlessly across organizational boundaries.
- **Built-in and community extensions:** The extensions module enhances the framework's functionality with advanced model clients, agents, multi-agent

teams, and tools for agentic workflows. Support for community extensions allows open-source developers to manage their own extensions.

- **Cross-language support:** This update enables interoperability between agents built in different programming languages, with current support for Python and .NET and additional languages in development.
- **Full type support:** Interfaces now enforce type checks at build time, ensuring robust and cohesive code quality.

Ejemplo de parte del chat con el interviewer:

User Prompt*

I want to create a chatbot that will make calls to people by phone and ask if they want to buy a product, if the person say yes the bot will direct the call to an human agent, if the person say no, the chatbot will try to convince the person making special offers 3 times, if the customer still says no, it will thank the person and say good bye

Assistant prompt*

To begin understanding your project, let me ask a few questions:

1. *What platform or systems is this chatbot expected to integrate with?* For example, will this chatbot be part of an existing CRM system, or will it require integration with a call service platform?

User Prompt*

yes, should integrate with the CRM to get phone numbers and use a dialler from a call service patform to make the calls

Assistant prompt*

Got it!

2. *What CRM system and call service platform will the chatbot integrate with?* For example, Salesforce, HubSpot, Twilio, or other platforms? This will help define compatibility and integration requirements.

User Prompt*

Should integrate with D365 and salesforce for CRM, y soportar SIP y whatsapp para llamadas de voz

...

...

User Prompt*

Could you give me a summary

Executive Summary

- **Overview**: Develop a chatbot system to make phone calls, interact with customers, and determine their interest in purchasing a product. If the customer is interested, the call is transferred to a human agent; otherwise, the chatbot tries to persuade with up to three special offers.

- **Business Benefits**: Streamlines sales efforts, increases productivity, and provides data-driven insights into customer behavior.

Scope

- ****In Scope****:

- Integration with Dynamics 365 (D365) and Salesforce CRMs.
- Supporting SIP and WhatsApp for voice calls.
- Advanced voice recognition using ML models.
- Pre-determined offers manually entered into the chatbot system.
- Configurable audience targeting by age, annual salary, and city.
- Analytics on call success rates, human transfers, and negative customer responses.
- Summary of conversations for human agents during call transfers.
- Configurable call length.
- Instant response time.
- Issue logging sent to support for technical failures.
- Happy path and unhappy path user stories.

- ****Out of Scope****:

- Integration with systems not supporting SIP or WhatsApp voice calls.
- Use of Large Language Models (LLMs).

- ****Systems & Platforms****: Dynamics 365 CRM, Salesforce CRM, SIP-based systems, WhatsApp voice calling.

Requirements

- ****Functional Requirements****:

- Call initiation, interaction, and escalation based on customer responses.

- Language supported: English.
- Configurable call duration and audience targeting.
- Voice recognition to analyze customer responses (yes/no).
- Manual entry of pre-determined offers.
- **Non-Functional Requirements**:
 - Instant response during interactions.
 - Analytics reporting for success rates, transfers, and negatives.
 - Logs and issue reporting for failures.
 - Cloud infrastructure setup.

Functional Solution

- **Overview**: A chatbot-led sales call system integrated with CRM and call platforms, leveraging ML models.
- **Impact Assessment**: Improved sales conversion and reduced manual effort.
- **Assumptions**:
 - Cloud infrastructure will be available on time.
 - Access to a dataset for training ML models.
 - Offers are up-to-date and accurately entered.
- **Dependencies**:
 - Cloud infrastructure.
 - Dataset for ML models.
- **Entities**:
 - Chatbot, CRM systems, call platform, support team.

Business Processes

- **AS IS**: Manual outbound sales calls with no automation.
- **TO BE**: Automated chatbot handling initial customer interaction with escalation to human agents when necessary.

Stories

- **Stories List**: Covers happy path (successful interactions) and unhappy path (silent customers, irrelevant responses).
- **User Stories**: Focus first on a successful call lifecycle, then cover failures.

Risk Assessment

- **Risk Matrix**:
 - Reliance on cloud infrastructure readiness.
 - Availability of high-quality training dataset.
 - Technical issues with APIs for SIP/WhatsApp.

Ejemplo de parte de un Log:

```
response = await self._model_client.create(messages=select_speaker_messages)
```

```
Model selected a valid name: planning_agent (attempt 1)
```

```
DEBUG:autogen_agentchat:Model selected a valid name: planning_agent (attempt 1)
```

```
Selected speaker: planning_agent
```

```
DEBUG:autogen_agentchat:Selected speaker: planning_agent
```

```
DEBUG:autogen_agentchat:Model selected a valid name: planning_agent (attempt 1)
```

Selected speaker: planning_agent

DEBUG:autogen_agentchat:Selected speaker: planning_agent

----- TextMessage (user) -----

Write the functional specifications for a new product

Write the functional specifications for a new product

----- TextMessage (planning_agent) -----

To write the functional specifications for a new product, we need to break down the task into manageable subtasks. Here's the plan:

1. **interviewer_agent**: Conduct interviews to understand the product requirements, stakeholders, and use cases.
2. **summarizer_agent**: Summarize the findings from the interviews into clear and concise product requirements.
3. **func_spec_checker_agent**: Review the summarized requirements to ensure they are clear, actionable, and complete from a functional specification standpoint.
4. **diagram_creator_agent**: Create visual diagrams or workflows to represent the product's functionality and design based on the specifications.
5. **mermaid_code_reviewer_agent**: Review the generated diagrams or workflows and ensure the visual representation is accurate and aligns with the functional specifications.

Let's begin with step 1: **interviewer_agent**.

1. **interviewer_agent**: Gather product requirements, stakeholders, and use cases information. Please reach out with findings when complete.

----- TextMessage (planning_agent) -----

To write the functional specifications for a new product, we need to break down the task into manageable subtasks. Here's the plan:

1. **interviewer_agent**: Conduct interviews to understand the product requirements, stakeholders, and use cases.
2. **summarizer_agent**: Summarize the findings from the interviews into clear and concise product requirements.
3. **funct_spec_checker_agent**: Review the summarized requirements to ensure they are clear, actionable, and complete from a functional specification standpoint.
4. **diagram_creator_agent**: Create visual diagrams or workflows to represent the product's functionality and design based on the specifications.
5. **mermaid_code_reviewer_agent**: Review the generated diagrams or workflows and ensure the visual representation is accurate and aligns with the functional specifications.

Let's begin with step 1: **interviewer_agent**.

1. **interviewer_agent**: Gather product requirements, stakeholders, and use cases information. Please reach out with findings when complete.

```
/run/media/jun/Jun 6T/Code/TFM_BA_GPT/venv/lib/python3.13/site-  
packages/autogen_agentchat/agents/_assistant_agent.py:1118: UserWarning: Resolved  
model mismatch: gpt-4o-2024-08-06!= gpt-4o-2024-11-20. Model mapping in  
autogen_ext.models.openai may be incorrect. Set the model to gpt-4o-2024-11-20 to  
enhance token/cost estimation and suppress this warning.
```

```
model_result = await model_client.create(
```

```
/run/media/jun/Jun 6T/Code/TFM_BA_GPT/venv/lib/python3.13/site-  
packages/autogen_agentchat/teams/_group_chat/_selector_group_chat.py:270:  
UserWarning: Resolved model mismatch: gpt-4o-2024-08-06!= gpt-4o-2024-11-20.  
Model mapping in autogen_ext.models.openai may be incorrect. Set the model to gpt-4o-  
2024-11-20 to enhance token/cost estimation and suppress this warning.
```

```
response = await self._model_client.create(messages=select_speaker_messages)
```

```
Model selected a valid name: interviewer_agent (attempt 1)
```

```
DEBUG:autogen_agentchat:Model selected a valid name: interviewer_agent (attempt  
1)
```

```
Selected speaker: interviewer_agent
```

```
DEBUG:autogen_agentchat:Selected speaker: interviewer_agent
```

```
/run/media/jun/Jun 6T/Code/TFM_BA_GPT/venv/lib/python3.13/site-  
packages/autogen_agentchat/teams/_group_chat/_selector_group_chat.py:270:  
UserWarning: Resolved model mismatch: gpt-4o-2024-08-06!= gpt-4o-2024-11-20.  
Model mapping in autogen_ext.models.openai may be incorrect. Set the model to gpt-4o-  
2024-11-20 to enhance token/cost estimation and suppress this warning.
```

```
response = await self._model_client.create(messages=select_speaker_messages)
```

Model selected a valid name: interviewer_agent (attempt 1)

DEBUG:autogen_agentchat:Model selected a valid name: interviewer_agent (attempt 1)

Selected speaker: interviewer_agent

DEBUG:autogen_agentchat:Selected speaker: interviewer_agent

----- MemoryQueryEvent (interviewer_agent) -----

[MemoryContent(content='• \nFunctional Requirements Specification \n• \n1.

Document Information \no 1.1. Document Approvers & Reviewers \no 1.2. Document History \no 1.3. Reference Documents \no 1.4. Definitions, Acronyms and Abbreviations \n• \n2. Executive Summary \no 2.1. Overview \no 2.2. Business Benefits \n• \n3. Scope \no 3.1. In Scope \no 4.2. Out of Scope \no 3.3. Systems & Platforms LeanIX Meta-Data \n• \n4. Requirements \no 4.1. Functional Requirements \no 4.2. Non-Functional Requirements \n• \n5. Functional Solution', mime_type='MemoryMimeType.TEXT', metadata={'chunk_index': 3, 'mime_type': 'MemoryMimeType.TEXT', 'source': 'Functional Requirements Specification Template.doc.pdf', 'score': 0.6303571164608002, 'id': '449a8a70-4043-4372-b176-0e7f32232870'}), MemoryContent(content='Functional Requirements Specification - \nTemplate \n \nFunctional Requirements Specification \nBS-[number] [title] \nDOCUMENTATION \nDocument Status IN PROGRESS ONHOLD COMPLETED \nAuthor(s) \n[BA name] \nSponsor \n[Sponsor name] \nRelated Area \n[area related to the sponsor] \n \n1. Document Information \n \n1.1. Document Approvers & Reviewers \nName \nRole \nApprover / \nReviewer \nApproval / \nReview Date \nApproved / \nReview Version \nHead of UAT \nHead of UAT \nApprover \n \n \nHead of Business \nChange', mime_type='MemoryMimeType.TEXT',

```
metadata={'chunk_index': 0, 'source': 'Functional Requirements Specification
Template.doc.pdf', 'mime_type': 'MemoryMimeType.TEXT', 'score':
0.6152962148189545, 'id': 'ff84ede0-0003-4b59-9b71-d49e2cbffa41'}),
MemoryContent(content='to make early adjustments to save time and money.
Alternatively, iterative development lets you continue making adjustments for a higher-
quality end product.43. Do You Provide a Functional Specification Document?Business
analysts use the general functional requirement document to produce a functional
specification document that shows concrete functions. The requirements document shows
the system's what and why, while the specifications document shows the how and when.
You want both', mime_type='MemoryMimeType.TEXT', metadata={'mime_type':
'MemoryMimeType.TEXT', 'chunk_index': 51, 'source':
'https://www.requiment.com/business-analyst-requirements-gathering-interview-
questions-answers/', 'score': 0.5994978845119476, 'id': '49eb7b9e-71bc-4617-85d3-
49472082db6f'})]
```

```
[MemoryContent(content='• \nFunctional Requirements Specification \n• \n1.
Document Information \no 1.1. Document Approvers & Reviewers \no 1.2. Document
History \no 1.3. Reference Documents \no 1.4. Definitions, Acronyms and Abbreviations
\n• \n2. Executive Summary \no 2.1. Overview \no 2.2. Business Benefits \n• \n3. Scope
\no 3.1. In Scope \no 4.2. Out of Scope \no 3.3. Systems & Platforms LeanIX Meta-Data
\n• \n4. Requirements \no 4.1. Functional Requirements \no 4.2. Non-Functional
Requirements \n• \n5. Functional Solution', mime_type='MemoryMimeType.TEXT',
metadata={'chunk_index': 3, 'mime_type': 'MemoryMimeType.TEXT', 'source':
'Functional Requirements Specification Template.doc.pdf', 'score': 0.6303571164608002,
```

```
'id': '449a8a70-4043-4372-b176-0e7f32232870'})), MemoryContent(content='Functional
Requirements Specification - \nTemplate \n \nFunctional Requirements Specification
\nBS-[number] [title] \nDOCUMENTATION \nDocument Status IN PROGRESS
ONHOLD COMPLETED \nAuthor(s) \n[BA name] \nSponsor \n[Sponsor name]
\nRelated Area \n[area related to the sponsor] \n \n1. Document Information \n \n1.1.
Document Approvers & Reviewers \nName \nRole \nApprover / \nReviewer \nApproval
/\nReview Date \nApproved / \nReview Version \nHead of UAT \nHead of UAT
\nApprover \n \n \nHead of Business \nChange', mime_type='MemoryMimeType.TEXT',
metadata={'chunk_index': 0, 'source': 'Functional Requirements Specification
Template.doc.pdf', 'mime_type': 'MemoryMimeType.TEXT', 'score':
0.6152962148189545, 'id': 'ff84ede0-0003-4b59-9b71-d49e2cbffa41'})),
MemoryContent(content='to make early adjustments to save time and money.
Alternatively, iterative development lets you continue making adjustments for a higher-
quality end product.43. Do You Provide a Functional Specification Document?Business
analysts use the general functional requirement document to produce a functional
specification document that shows concrete functions. The requirements document shows
the system's what and why, while the specifications document shows the how and when.
You want both', mime_type='MemoryMimeType.TEXT', metadata={'mime_type':
'MemoryMimeType.TEXT', 'chunk_index': 51, 'source':
'https://www.requiment.com/business-analyst-requirements-gathering-interview-
questions-answers/', 'score': 0.5994978845119476, 'id': '49eb7b9e-71bc-4617-85d3-
49472082db6f'})])
```

----- ModelClientStreamingChunkEvent (interviewer_agent) -----

To----- ModelClientStreamingChunkEvent (interviewer_agent) -----

To create the functional specifications for the new product, we must gather detailed information about the project create the functional specifications for the new product, we must gather detailed information about the project requirements, scope, and overall vision requirements, scope, and overall vision.. Let us begin by exploring your product idea in depth.

Question 1:

What is the Let us begin by exploring your product idea in depth.

Question 1:

What is the overarching goal of the new product and what business problem or overarching goal of the new product and what business problem or opportunity does it aim to address?

opportunity does it aim to address?

/run/media/jun/Jun 6T/Code/TFM_BA_GPT/venv/lib/python3.13/site-packages/autogen_agentchat/teams/_group_chat/_selector_group_chat.py:270:
UserWarning: Resolved model mismatch: gpt-4o-2024-08-06!= gpt-4o-2024-11-20.
Model mapping in autogen_ext.models.openai may be incorrect. Set the model to gpt-4o-2024-11-20 to enhance token/cost estimation and suppress this warning.

response = await self._model_client.create(messages=select_speaker_messages)

Model selected a valid name: interviewer_agent (attempt 1)

DEBUG:autogen_agentchat:Model selected a valid name: interviewer_agent (attempt

1)

Selected speaker: interviewer_agent

DEBUG:autogen_agentchat:Selected speaker: interviewer_agent

DEBUG:autogen_agentchat:Model selected a valid name: interviewer_agent (attempt

1)

Selected speaker: interviewer_agent

DEBUG:autogen_agentchat:Selected speaker: interviewer_agent

----- MemoryQueryEvent (interviewer_agent) -----

[MemoryContent(content='by which it must be realised by the project's conclusion are among the items it specifies. However, it makes no mention of how it will be supplied in detail. More emphasis should be placed on setting the product's context, such as the necessity for the product or the issue it resolves. There are no specifics about how it will be implemented. Learn more in our other blog 'How to Write a Software Requirements Document (SRD).'What are entities in requirements gathering?In the context of', mime_type='MemoryMimeType.TEXT', metadata={'mime_type': 'MemoryMimeType.TEXT', 'chunk_index': 13, 'source': 'https://www.requirement.com/the-most-asked-questions-about-requirements-gathering/', 'score': 0.6315523684024811, 'id': '7aec1369-2868-4f5c-adea-50f2c8fda3f0'}), MemoryContent(content='to make early adjustments to save time and money. Alternatively, iterative development lets you continue making adjustments for a higher-quality end product.⁴³. Do You Provide a Functional Specification Document?Business analysts use the general functional requirement document to produce a functional specification document that shows

concrete functions. The requirements document shows the system's what and why, while the specifications document shows the how and when. You want both',

```
mime_type='MemoryMimeType.TEXT', metadata={'source':  
'https://www.requiment.com/business-analyst-requirements-gathering-interview-  
questions-answers/', 'mime_type': 'MemoryMimeType.TEXT', 'chunk_index': 51, 'score':  
0.6242744624614716, 'id': '49eb7b9e-71bc-4617-85d3-49472082db6f'}),
```

MemoryContent(content='Distinguishing between customer needs and project requirements is fundamental. This question tests the candidate's understanding of this critical differentiation.\n41. What Are Non-Functional Requirements, and How Are They Captured?\nNon-functional requirements, such as performance and security, are crucial for project success. This question evaluates the candidate's awareness of these critical aspects.\n42. What Is an Activity Diagram and Its Key Elements?',

```
mime_type='MemoryMimeType.TEXT', metadata={'chunk_index': 52, 'source':  
'https://www.morebusiness.com/business-analyst-interview-questions/', 'mime_type':  
'MemoryMimeType.TEXT', 'score': 0.6216502487659454, 'id': '37e306e6-0f18-42dc-  
9851-11a2042734ac'}})]
```

[MemoryContent(content='by which it must be realised by the project's conclusion are among the items it specifies. However, it makes no mention of how it will be supplied in detail. More emphasis should be placed on setting the product's context, such as the necessity for the product or the issue it resolves. There are no specifics about how it will be implemented. Learn more in our other blog 'How to Write a Software Requirements Document (SRD).'
What are entities in requirements gathering?In the context of',

```
mime_type='MemoryMimeType.TEXT', metadata={'mime_type':
```

'MemoryMimeType.TEXT', 'chunk_index': 13, 'source': 'https://www.requiment.com/the-most-asked-questions-about-requirements-gathering/', 'score': 0.6315523684024811, 'id': '7aec1369-2868-4f5c-adea-50f2c8fda3f0'}), MemoryContent(content='to make early adjustments to save time and money. Alternatively, iterative development lets you continue making adjustments for a higher-quality end product.⁴³ Do You Provide a Functional Specification Document?Business analysts use the general functional requirement document to produce a functional specification document that shows concrete functions. The requirements document shows the system's what and why, while the specifications document shows the how and when. You want both', mime_type='MemoryMimeType.TEXT', metadata={'source': 'https://www.requiment.com/business-analyst-requirements-gathering-interview-questions-answers/', 'mime_type': 'MemoryMimeType.TEXT', 'chunk_index': 51, 'score': 0.6242744624614716, 'id': '49eb7b9e-71bc-4617-85d3-49472082db6f'}), MemoryContent(content='Distinguishing between customer needs and project requirements is fundamental. This question tests the candidate's understanding of this critical differentiation.⁴¹ What Are Non-Functional Requirements, and How Are They Captured?Non-functional requirements, such as performance and security, are crucial for project success. This question evaluates the candidate's awareness of these critical aspects.⁴² What Is an Activity Diagram and Its Key Elements?', mime_type='MemoryMimeType.TEXT', metadata={'chunk_index': 52, 'source': 'https://www.morebusiness.com/business-analyst-interview-questions/', 'mime_type': 'MemoryMimeType.TEXT', 'score': 0.6216502487659454, 'id': '37e306e6-0f18-42dc-9851-11a2042734ac'}})]

----- ModelClientStreamingChunkEvent (interviewer_agent) -----

What is----- ModelClientStreamingChunkEvent (interviewer_agent) -----

What is the overarching goal of the new product, and what specific business problem or the overarching goal of the new product, and what specific business problem or opportunity does it aim to resolve? This will help establish the opportunity does it aim to resolve? This will help establish the context and purpose for creating the product.

context and purpose for creating the product.

/run/media/jun/Jun 6T/Code/TFM_BA_GPT/venv/lib/python3.13/site-packages/autogen_agentchat/teams/_group_chat/_selector_group_chat.py:270:
UserWarning: Resolved model mismatch: gpt-4o-2024-08-06!= gpt-4o-2024-11-20.
Model mapping in autogen_ext.models.openai may be incorrect. Set the model to gpt-4o-2024-11-20 to enhance token/cost estimation and suppress this warning.

response = await self._model_client.create(messages=select_speaker_messages)

Model selected a valid name: user_proxy (attempt 1)

DEBUG:autogen_agentchat:Model selected a valid name: user_proxy (attempt 1)

Selected speaker: user_proxy

DEBUG:autogen_agentchat:Selected speaker: user_proxy

DEBUG:autogen_agentchat:Model selected a valid name: user_proxy (attempt 1)

Selected speaker: user_proxy

DEBUG:autogen_agentchat:Selected speaker: user_proxy

----- TextMessage (user_proxy) -----

I want to create a bot that makes phone calls to people and offer to sell travel packages, if the person that take the phone call say yes, the bot send the call to a human

agent if the person says no, the bot wil try to convince him 3 times indicating cool thing of this travel pack

I want to create a bot that makes phone calls to people and offer to sell travel packages, if the person that take the phone call say yes, the bot send the call to a human agent if the person says no, the bot wil try to convince him 3 times indicating cool thing of this travel pack

```
/run/media/jun/Jun 6T/Code/TFM_BA_GPT/venv/lib/python3.13/site-packages/autogen_agentchat/teams/_group_chat/_selector_group_chat.py:270:
```

```
UserWarning: Resolved model mismatch: gpt-4o-2024-08-06!= gpt-4o-2024-11-20.
```

```
Model mapping in autogen_ext.models.openai may be incorrect. Set the model to gpt-4o-2024-11-20 to enhance token/cost estimation and suppress this warning.
```

```
response = await self._model_client.create(messages=select_speaker_messages)
```

```
Model selected a valid name: interviewer_agent (attempt 1)
```

```
DEBUG:autogen_agentchat:Model selected a valid name: interviewer_agent (attempt 1)
```

```
Selected speaker: interviewer_agent
```

```
DEBUG:autogen_agentchat:Selected speaker: interviewer_agent
```

```
DEBUG:autogen_agentchat:Model selected a valid name: interviewer_agent (attempt 1)
```

```
Selected speaker: interviewer_agent
```

```
DEBUG:autogen_agentchat:Selected speaker: interviewer_agent
```

```
----- ModelClientStreamingChunkEvent (interviewer_agent) -----
```

```
##### Question 2 Question 2:
```

Who will be the primary users of this bot, and what specific audience or market segment:

Who will be the primary users of this bot, and what specific audience or market segment is the target for the travel packages? For instance is the target for the travel packages? For instance, are you, are you targeting business professionals, families, or young travelers?

targeting business professionals, families, or young travelers?