UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA AUTÓNOMO DE ASISTENCIA INDUSTRIAL

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR/A: de Miguel Beyger, Ida Maria

Tutor/a: López Patiño, José Enrique

CURSO ACADÉMICO: 2025/2026

Resumen

En las empresas es común que la eficiencia pueda ser comprometida por la dispersión de la información y la dificultad para acceder a los recursos, lo que causa una divagación recurrente y mayor tiempo de resolución de problemas. Este proyecto busca una solución mediante el diseño e implementación de un sistema autónomo que trata de optimizar el soporte, la conexión a recursos y el apoyo a personas, asegurando que la inteligencia artificial proporcione soluciones directas, accesibles y validadas.

El objetivo principal es implementar un sistema privado, escalable y corporativo capaz de resolver problemas en un entorno industrial, con posibilidad de funcionar en otras áreas. El método se centrará en arquitectura *cloud* gracias a los recursos de Microsoft Azure; automatización y escalabilidad para la gestión y despliegue de recursos en la nube mediante infraestructura como código, en específico Terraform; e integración de inteligencia artificial.

Se garantizará el cumplimiento de las restricciones de seguridad y privacidad de un modelo corporativo y se verificará la viabilidad y el impacto tecnológico y económico del proyecto.

Resum

A les empreses és comú que l'eficiència puga ser compromesa per la dispersió de la informació i la dificultat per a accedir als recursos, la qual causa una divagació recurrent i major temps de resolució de problemes. Aquest projecte busca una solució mitjançant el disseny i la implementació d'un sistema autònom que tracta d'optimitzar el suport, la connexió a recursos i el suport a persones, assegurant que la intel·ligència artificial proporcione solucions directes, accessibles i validades.

L'objectiu principal és implementar un sistema privat, escalable i corporatiu capaç de resoldre problemes en un entorn industrial, amb possiblidad de funcionar en altres àrees. El mètode es centrarà en arquitectura *cloud* gràcies als recursos de Microsoft Azure; automatització i escalabilitat per a la gestió i desplegament de recursos en el núvol mitjançant la infraestructura com a codi Terraform; i integració de inteligència artificial.

Es garantirà el compliment de les restriccions de seguretat i privacitat d'un model corporatiu i es verificarà la viabilitat i l'impacte tecnològic i econòmic del projecte.

Abstract

In companies, efficiency can often be compromised by scattered information and difficulty accessing resources, which causes recurring digressions and longer problem-solving times. This project looks for a solution through the design and implementation of an autonomous system

that aims to optimize support, connection to resources, and assistance to people, ensuring that artificial intelligence provides direct, accessible and validated solutions.

The main objective is to implement a private, scalable and corporate system capable of solving problems in the industrial environment, with the possibility of operating in other areas. The method will focus on cloud architecture with Microsoft Azure resources; automation and scalability for the management and deployment of cloud resources through infrastructure as code, specifically Terraform; and the integration of artificial intelligence.

The project will verify the feasibility and technological and economic impact while guaranteeing compliance with the security and privacy restrictions of a corporate model.

RESUMEN EJECUTIVO

La memoria del TFG del Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación debe desarrollar en el texto los siguientes conceptos, debidamente justificados y discutidos, centrados en el ámbito de la tecnologías digitales y multimedia

CONCEPT (ABET)	**CONCEPTO (traducción)**	**¿Cumple? (S/N)**	**¿Dónde? (páginas)**
1. IDENTIFY:	1. IDENTIFICAR:		
1.1. Problem statement and opportunity	1.1. Planteamiento del problema y oportunidad	S	1, 2
1.2. Constraints (standards, codes, needs, requirements & specifications)	1.2. Toma en consideración de los condicionantes (normas técnicas y regulación, necesidades, requisitos y especificaciones)	S	3
1.3. Setting of goals	1.3. Establecimiento de objetivos	S	1
2. FORMULATE:	2. FORMULAR:		
2.1. Creative solution generation (analysis)	2.1. Generación de soluciones creativas (análisis)	S	4-18, 30-33
2.2. Evaluation of multiple solutions and decision-making (synthesis)	2.2. Evaluación de múltiples soluciones y toma de decisiones (síntesis)	S	19-29, 33-34
3. SOLVE:	3. RESOLVER:		
3.1. Fulfilment of goals	3.1. Evaluación del cumplimiento de objetivos	S	33
3.2. Overall impact and significance (contributions and practical recommendations)	3.2. Evaluación del impacto global y alcance (contribuciones y recomendaciones prácticas)	S	33,39

Índice

Capítulo 1. Alcance y objetivos 1
1.1 Introducción y justificación 1
1.2 Objetivos del proyecto 1
1.3 Antecedentes tecnológicos 1
1.4 Metodología y planificación temporal del proyecto 2
1.5 Herramientas 2
Capítulo 2. Requisitos y restricciones 3
2.1 Requisitos funcionales 3
2.2 Restricciones técnicas y operativas 3
2.3 Restricciones de seguridad y privacidad 3
Capítulo 3. Arquitectura e ingeniería del diseño 4
3.1 Ecosistema de computación en la nube: Microsoft Azure 4
3.1.1 Definición y alcance de Microsoft Azure 4
3.1.2 Modelos de servicio (IaaS, PaaS, SaaS) 4
3.1.3 Ventajas e inconvenientes 5
3.1.4 Evaluación comparativa 6
3.2 Gobernanza y automatización: infraestructura como código con Terraform 7
3.2.1 Infraestructura como código 7
3.2.2 Fundamentos de Terraform 7
3.2.3 Ciclo de vida y plan de ejecución 7
3.2.4 Gestión de estado y backend remoto 8
3.2.5 Modularización y reutilización 8
3.3 Diseño de redes y seguridad (Modelo Zero Trust) 9
3.3.1 Diseño de la VNet y segmentación 9
3.3.2 Private Endpoints y Private Links 10
3.3.3 Control de tráfico: NSG y UDR 10
3.3.4 Identidad y control de acceso (RBAC) 11
3.4 Inteligencia artificial autónoma y orquestación de agentes 11
3.4.1 Modelos de lenguaje a gran escala: de sus orígenes al estado actual 11
3.4.2 Agente artificial 11
3.4.3 Servicio Microsoft Foundry Agent 13
3.4.4 IA Generativa vs IA Agéntica 13

3.4.5 El patrón RAG 14
3.4.6 Ecosistema de frameworks y selección de herramientas 14
3.4.7 Orquestación y microservicios 15
3.4.8 Seguridad y ética en agentes autónomos 15
3.4.9 Comunicación multimodal y tiempo real 16
3.5 Diseño de persistencia 16
Capítulo 4. Desarrollo e implementación del sistema de agentes 18
4.1 Diseño de infraestructura como código con Terraform 18
4.1.1 Estrategias de diseño y buenas prácticas 18
4.1.2 Recursos desplegados y módulos 18
4.2 Preparación del entorno de desarrollo y configuración 19
4.3 Implementación del agente inteligente 20
4.3.1 Orquestación del backend con FastAPI 20
4.3.2 Definición y ciclo de vida del agente corporativo 20
4.3.3 Herramientas y capacidades extendidas 21
4.4 Persistencia de conversaciones y gestión de hilos 22
4.5 Interfaz de usuario 22
4.5.1 Arquitectura y tecnologías web 22
4.5.2 Gestión de identidad y seguridad 26
4.5.3 Comunicación asíncrona 26
4.5.4 Implementación de voz en tiempo real 26
4.6 Observabilidad y comprobaciones 27
4.6.1 Verificación funcional con Swagger 27
Capítulo 5. Gestión financiera y evaluación de impacto 28
5.1 Cálculo de costes y estrategia financiera 28
5.1.1 Selección del nivel de soporte técnico 28
5.1.2 Análisis de costes por recurso y configuración de SKU 29
5.2 Evaluación de impacto 31
5.3 Análisis de riesgos financieros 31
5.4 Herramientas de supervisión continua 32
Capítulo 6. Conclusiones 33
6.1 Cumplimiento de objetivos y resultados obtenidos 33
6.2 Contribuciones, limitaciones y recomendaciones prácticas 33
6.3 Líneas de trabajo futuro 33
Capítulo 7. Referencias 34
Anexo A. Objetivos de desarrollo sostenible 38

Capítulo 1. Alcance y objetivos

1.1 Introducción y justificación

En la actualidad las organizaciones se enfrentan a un proceso de digitalización masivo, donde el volumen de datos generados aumenta de manera exponencial. El crecimiento de estos datos lleva a la dispersión de la información, lo que ha hecho que se dificulte la toma de decisiones. La gestión de grandes volúmenes de datos (*Big Data*) plantean varios retos, como por ejemplo privacidad y seguridad de los datos personales, costes de infraestructura y almacenamiento, calidad y confiabilidad de los datos, la brecha digital y la complejidad y necesidad de conocimientos especializados[1].

Bajo este contexto y con la colaboración de NTT DATA Spain, se ha realizado el proyecto "Shopfloor assistant" con un enfoque de investigación y desarrollo (I+D) aplicado al sector industrial, con el objetivo de contribuir a la modernización del sector mediante la automatización y creación de nuevos escenarios de trabajo que fomenten el aprendizaje continuo y la eficiencia operativa.

1.2 Objetivos del proyecto

El objetivo principal de este proyecto es diseñar, documentar y desplegar la arquitectura de un sistema autónomo basado en inteligencia artificial dentro de un entorno *cloud* automatizado para ofrecer soluciones optimizadas a usuarios finales. Esto se realiza mediante la implementación de un asistente industrial orientado principalmente a industria, aunque con capacidad de adaptación a otras áreas.

El sistema se basa en un agente de inteligencia artificial capaz de realizar tareas de forma autónoma e interactuar con otros componentes para logar unos objetivos. Se realiza una evaluación de la eficacia, fiabilidad, seguridad y rendimiento de la arquitectura y del agente.

Con este análisis se busca incrementar el valor de negocio, mejorando la eficiencia operativa, reduciendo los tiempos de resolución de problemas y llevando a la empresa a la innovación tecnológica.

1.3 Antecedentes tecnológicos

La inteligencia artificial tiene una larga trayectoria a lo largo de la historia. Todo comenzó con Ada Lovelace en 1842, la mujer que programó el primer algoritmo destinado a ser procesado por una máquina. A raíz de eso se creó un nuevo campo de estudio científico. A partir del año 1943 se convirtió en una realidad, cuando se descubrió el primer modelo matemático de la neurona y en 1950 Alan Turing propuso un estudio para saber si las máquinas podrían seguir un comportamiento inteligente. El año 1956 nace el término "Inteligencia Artificial" gracias a John McCanthy y ese mismo año se crea el primer programa informático de inteligencia artificial. A lo largo de los años 60 muchos científicos realizaron grandes avances como por ejemplo Eliza, el primer chatbot basado en procesamiento de lenguaje natural. Después de años sin avances por falta de fondos e interés, a finales de los 90 se creó Deep Blue, una computadora que se volvió muy popular por derrotar al campeón mundial de ajedrez. A raíz de esto la inteligencia artificial cada vez era más popular y se dedicaban más recursos para investigar todo lo que se podría llegar a lograr con la inteligencia artificial. Logros como ImageNet para la democratización de datos, redes generativas o asistentes virtuales han hecho que la IA llegue hasta donde está hoy en día.[2]

Hoy en día hay un término que destaca en la inteligencia artificial: agente de IA. Un agente inteligente es una entidad autónoma que observa a través de sensores, actúa sobre un entorno utilizando actuadores y dirige su actividad hacia el logro de objetivos a través del aprendizaje. Un sistema de múltiples agentes se compone de varios agentes que interactúan entre sí y se basan en el entorno.[3] Podemos concluir en que los agentes son el futuro de la automatización, ya que son los que añaden la capa de autonomía necesaria para resolver problemas complejos.

1.4 Metodología y planificación temporal del proyecto

1. Familiarización con el proyecto, aprendiendo lenguajes de programación como Python, Terraform y Azure.
2. Análisis y estructuración del proyecto.
3. Implementación de proyecto de automatización de infraestructura mediante Terraform y aprendizaje del uso de la calculadora de precios de Azure para la correcta elección de recursos.
4. Implementación del proyecto del agente de inteligencia artificial en python.
5. Validación de los resultados y escritura final del trabajo de fin de grado.

Actividad	Noviembre	Diciembre	Enero	Febrero	Marzo
Familiariza-ción	x				
Análisis	x				
Terraform/ Azure		x			
Agente Python		x	x		
Validación y escritura				x	x

Tabla 1. Planificación del proyecto

1.5 Herramientas

Las herramientas y *frameworks* utilizados son:

1. *Framework* principal: Microsoft Agent Framework
   Se usa la librería agent-framework de Python para la creación de agentes que mantienen un historial de forma persistente en la nube, manejando estados de ejecución y conectando herramientas.
2. Herramientas y capacidades
   Interacción con Office 365 a través de Graph Explorer, necesario para que el agente pueda enviar correos. Se implementa mediante una Function Tool dentro del *framework* de los agentes.
   Recuperación de información mediante RAG. Azure AI Search se usa para buscar en los documentos privados de la empresa. Con Bing Grounding el agente tiene acceso a Internet en tiempo real y permite que sea información preciso y actual.
3. Automatización con Terraform
   Terraform gestiona los recursos de Azure. También se utiliza la autenticación de Terraform mediante los servicios de Managed Identities (Identidades Gestionadas) de Azure (DefaultAzureCredential) para conectarse con Graph API, AI Search y otras herramientas sin tener que usar contraseñas.

Capítulo 2. Requisitos y restricciones

2.1 Requisitos funcionales

El diseño del sistema autónomo de asistencia industrial responde a la necesidad de mejorar la ineficiencia operativa causada por la dispersión de información en entornos de fabricación. Los requisitos fundamentales se dividen en objetivos funcionales y de negocio:

- Optimización del soporte técnico: el sistema debe proporcionar un asistente de inteligencia artificial capaz de conectarse a diferentes recursos corporativos para resolver los problemas que tienen los usuarios en cuanto a la divagación de la información.
- Validación de respuestas (guardarraíles): es importante que el agente no desvaríe en las respuestas y no genere alucinaciones, por lo que hay que validar las soluciones dadas por la inteligencia artificial mediante la conexión a fuentes de información.
- Eficiencia operativa: se busca que el agente reduzca al máximo su tiempo medio de resolución de problemas (MTTR), transformando la inversión tecnológica en un aumento del valor de la compañía.
- Experiencia de usuario y rendimiento: se establecen métricas de latencia para garantizar la utilidad del asistente en tiempo real mediante llamadas simples con respuestas de menos de 3 segundos y para consultas complejas con herramientas con respuestas de menos de 10 segundos.

2.2 Restricciones técnicas y operativas

El sistema tiene que ser privado, escalable y corporativo. Para ello utiliza:

- Microsoft Azure: se usan sus recursos y servicios integrados.
- Infraestructura como código: el uso de Terraform para la gestión y despliegue de recursos garantiza que sea un código repetible y predecible, lo que permite crear entornos de desarrollo, prueba y producción parecidos y reduciendo los errores, pudiendo mejorar en cada implementación.
- Gestión de estado y colaboración: el archivo de estado de Terraform (*.tfstate*) debe almacenarse de forma remota y segura en un Azure Storage Account con versionado y bloqueo de estado para permitir el trabajo colaborativo sin corrupción de datos.
- -Escalabilidad y cómputo *Serverless*: Mediante los Azure Container Apps se consigue un modelo de consumo *serverless*. Esto permite al sistema un ajuste dinámico a los picos de carga de la planta industrial y optimiza los costes mediante el pago por uso.

2.3 Restricciones de seguridad y privacidad

El proyecto adopta un modelo de seguridad Zero Trust, donde no se asume confianza por defecto y se realiza la verificación de identidad en cada intento de acceso. Se desarrollará en profundidad este aspecto en el apartado 3.3.

Capítulo 3. Arquitectura e ingeniería del diseño

3.1 Ecosistema de computación en la nube: Microsoft Azure

Para el desarrollo de los siguientes puntos, se ha tomado como referencia principal los datos del informe sobre el Almacenamiento en la Nube, elaborado por la Asociación de Consultores Técnicos[4].

3.1.1 Definición y alcance de Microsoft Azure

La nube es un concepto que consiste en el acceso a través de Internet a todo tipo de servicios y recursos informáticos ofrecidos por una red de servidores destinados a ese fin. Tales servidores pueden encontrarse en cualquier lugar.

Los usos habituales de la nube son almacenamiento, acceso a recursos almacenados, utilización de servicios como el correo electrónico, uso a distancia de aplicaciones, uso compartido de información, etc.

Los servicios en la nube pueden implementarse y ofrecerse de tres formas distintas:

- Nube pública: ofrece sus servicios de forma pública, en general a través de Internet. Un proveedor es el dueño de la infraestructura y es quien gestiona y ofrece los servicios. A menudo, los servicios se ofrecen a cualquier usuario con acceso a Internet, y es habitual también que sea gratis, al menos para servicios básicos, aunque existe la posibilidad de contratar más funcionalidades. Ejemplos de proveedores son Amazon, Microsoft o Google, entre otros.
- Nube privada: ofrece servicio a organizaciones pudiendo ser estas dueñas de la nube alojarla en sus propios centros de datos o que la nube sea propiedad de un tercero que se encargue de gestionarla y que se aloje fuera de la organización; por ejemplo, en los centros de datos de esos proveedores de terceros. También son posibles otras combinaciones tales como una nube privada de la empresa pero alojada en un centro de datos externos.
- Nube híbrida: ofrece servicios mixtos en la nube, que combinan nubes privadas, públicas o de otros tipos en un número y proporción variables. Las nubes híbridas son el tipo más flexible, potente y versátil. Ofrecen ventajas como: infinidad de combinaciones diferentes en cuanto a infraestructura, funcionamiento, requisitos y utilización, además de escalabilidad casi ilimitada[4, pp. 2–4].

3.1.2 Modelos de servicio (IaaS, PaaS, SaaS)

Los servicios específicos que ofrece la nube son muchos. Existen tres modelos básicos para la agrupación de estos servicios:

- Software as a Service (SaaS): modelo de servicios en la nube que permite a los usuarios utilizar programas o aplicaciones hospedados y ofrecidos por un proveedor en la nube. Se accede a ellos a través de Internet. Un ejemplo de SaaS es Microsoft Office 365 , Gmail de Google o aplicaciones como Salesforce.
  Los servicios SaaS suelen estar dirigidos a un «usuario final», que son particulares o trabajadores de empresas que se limitan a utilizar el producto informático.
  Como el programa o aplicación se encuentra en la nube, no es necesario que el usuario lo instale en su equipo. Puede usarse de manera remota después de acreditar algún tipo de credenciales que identifiquen el usuario ante el proveedor en la nube.
- Platform as a Service (PaaS): modelo de servicios en la nube que ofrece infraestructura, medios y recursos necesarios para el desarrollo o implementación de aplicaciones en la nube. En algunos casos, ofrece el entorno donde desarrollar las aplicaciones SaaS, que serán accesibles y utilizables a través de Internet posteriormente.
  Incluye entornos de programación y compilación, además de recursos básicos como almacenamiento, infraestructura de red, bases de datos, servidores web, etc. También

ofrece entornos donde probar la aplicación para distintos sistemas operativos y plataformas, y herramientas o utilidades punteras cada vez más indispensables en las aplicaciones informáticas.
El proveedor tiene la responsabilidad de controlar y administrar los recursos e infraestructuras, configuraciones o personalizaciones del entorno, aunque es el cliente el que tiene control de las propias aplicaciones que se desarrollen en ese entorno.

- Infrastructure as a Service (IaaS): modelo de servicios en la nube que ofrece infraestructuras de servidores físicos, redes o almacenamiento accesibles a través de Internet. También, otros aspectos no físicos tales como el procesamiento en centro de datos, creación de copias de seguridad o cortafuegos. Todos son proporcionados y administrados por el proveedor de servicios en la nube. Sus clientes (empresas, departamentos técnicos, gestores de proyecto, arquitectos de TI) escogen y utilizan los recursos que necesitan. Suelen tener una tarifa variable contratada que depende del consumo de los servicios. En lugar de adquirir, administrar, mantener, actualizar y ampliar o renovar las infraestructuras que necesite, el cliente simplemente las alquila[4, pp. 5–8].

3.1.3 Ventajas e inconvenientes

Las ventajas más relevantes que ofrece la utilización de la nube son:

- Acceso desde cualquier lugar.
  Todos los recursos, programas, archivos accesibles están disponibles desde cualquier lugar para el aprovechamiento total a través de una conexión a Internet.
- Uso multiplataforma.
  El proveedor de la nube automatiza y se ocupa por sí solo de adecuar el acceso al servicio que sea desde cualquier dispositivo o equipo, sin que el usuario o cliente deba preocuparse por indicar el tipo de dispositivo usado.
- Trabajo colaborativo.
  Existe la capacidad para permitir que equipos o grupos de personas participen de manera simultánea en un proyecto, incluso si estos están geográficamente dispersos o acceden desde dispositivos o equipos diferentes.
  Esta característica de la nube es especialmente importante en el mundo empresarial, ya que gracias a la globalización es posible crear mayor capital, ventaja competitiva, escalabilidad y crecimiento.
- Software centralizado y sin instalación.
  El proveedor de la nube se encarga de instalar, mantener, gestionar, proteger, actualizar y renovar los programas y aplicaciones que ofrece a sus clientes.
- Desarrollo, prueba e implementación de aplicaciones optimizadas.
  La nube favorece de manera significativa la rapidez y facilidad de creación, prueba e implementación de nuevas aplicaciones, lo que fomenta la innovación.
- Ahorro en hardware y software.
- Escalabilidad y adaptabilidad.
  La escalabilidad se define como la capacidad de un sistema para crecer y responder de manera eficiente ante un aumento en la demanda de recursos sin perder la calidad de servicio. En la era de la inmediatez tecnológica, la falta de esa capacidad supone pérdidas económicas y daño reputacional para la empresa. Gracias a la nube podemos obtener una adaptación dinámica y en tiempo real.
- Ahorro en mantenimiento técnico y de seguridad.
  Es al proveedor de la nube a quien le corresponde lo relacionado con el mantenimiento técnico y la protección del hardware y el software que ofrece.
- Pago por uso.
  En la nube, uno paga por lo que consume y existe una adaptación a las necesidades del cliente.
- Protección frente a ataques y desastres.

Ningún servicio es invulnerable. Por eso es necesario disponer de sistemas de protección, como servidores de apoyo, generadores, copias de seguridad actualizadas, etc. para fortalecer el sistema y hacerlo resiliente y capaz de responder rápidamente ante contratiempos. La nube ofrece estas medidas de protección y recuperación.

- Procesamiento y análisis de alto nivel.
  Muchas aplicaciones exigen una enorme capacidad de procesamiento, como el análisis de modelos de inteligencia empresarial o Big Data. Es necesario contar con servidores y centros de datos de alto nivel. La nube permite que estos servicios sean accesibles a todo tipo de empresas, organizaciones y particulares[4, pp. 9–13].

Los inconvenientes y riesgos que presenta la nube son:

- Falta de seguridad o privacidad.
  Al trasladar la información a servidores ajenos, el cliente cede el control físico y lógico sobre sus datos. Esto podría generar incertidumbre sobre quién accede a la información y con qué fines. Los proveedores se convierten en objetivos críticos.
- Necesidad de una conexión a Internet.
  La operatividad de la nube está ligada a la conexión a Internet. Aunque existen soluciones offline, suelen ser de pago y ofrecen funcionalidades limitadas.
- Problemas legales o jurisdiccionales.
  La ubicación de la nube implica que los datos pueden residir en servidores sujetos a legislaciones extranjeras. Esto genera riesgos de inseguridad jurídica.
- Conflictos relacionados con la propiedad intelectual.
  La externalización de datos puede derivar a plagios o robo de información.
- Incompatibilidad entre proveedores.
  Muchos proveedores diseñan sus sistemas con falta de interoperabilidad como estrategia comercial. Esto dificulta la migración de datos y aplicaciones hacia otros competidores, forzando una dependencia tecnológica.
- Pérdida de control[4, pp. 14–15].

3.1.4 Evaluación comparativa

Para la correcta elección del servicio en la nube hay que tener en cuenta diferentes características que satisfagan las necesidades de un cliente. Entre ellas se encuentran:

- Nivel de seguridad.
- Opciones y escalabilidad.
- Precio.
- Estabilidad y confiabilidad.
- Servicio de Asistencia y Atención al Cliente.
- Portabilidad.
- Responsabilidades y cumplimiento de requisitos.
- Resiliencia, capacidad de recuperación.
- Facilidad de uso[4, pp. 16–24].

Teniendo en cuenta los conceptos y criterios técnicos definidos por Gutiérrez (2018) en su informe para la asociación ACTA[4], se elige que la solución óptima sería la utilización de Microsoft Azure como plataforma de implementación. Tiene la capacidad de integrar herramientas y recursos del ámbito corporativo, garantizando el cumplimiento del marco normativo europeo y la protección de datos mediante sus centros de datos regionales, lo que permite resolver de forma efectiva los conflictos jurisdiccionales y de privacidad. Azure también ofrece escalabilidad dinámica bajo un modelo de nube híbrida que permite optimizar la inversión financiera de la empresa mediante un sistema de pago por uso y asegura la continuidad del negocio ante picos de demanda o posibles fallos de conectividad gracias a su infraestructura resiliente. Finalmente, se ha elegido Azure por su gran capacidad a la hora de gestionar todo de forma conjunta y sencilla, ofreciendo una tecnología muy fiable y robusta, especialmente para el uso de herramientas de inteligencia artificial, como por ejemplo Microsoft Foundry, con una garantía de funcionamiento del 99.9 %.

3.2 Gobernanza y automatización: infraestructura como código con Terraform

3.2.1 Infraestructura como código

La infraestructura como código (IaC) usa la metodología de DevOps y el control de versiones con un modelo descriptivo para definir e implementar la infraestructura. Un modelo IaC genera el mismo entorno cada vez que se implementa[5]. Es un componente de entrega continua, que es el proceso de automatizar la prueba, la configuración y la implementación de una compilación en un entorno de producción. Es un requisito obligatorio para las grandes organizaciones ya que de esta manera se consigue tener un entorno de producción actualizado y se minimiza el tiempo de implementación y el tiempo de corrección de incidentes de producción.

La integración continua (CI) inicia el proceso de entrega continua (CD). El pipeline de lanzamiento transita a través de cada entorno una vez las pruebas se hayan completado correctamente[6].

En el apartado anterior se ha justificado el uso de Microsoft Azure como plataforma de computación en la nube en este proyecto, por lo que deberíamos usar herramientas compatibles como son Terraform o Bicep. La diferencia entre Terraform y Bicep es que Terraform tiene una naturaleza multiplataforma y su estado de gestión, lo que permite administrar recursos en diversos proveedores de la nube de forma unificada y Bicep solo se puede usar con Azure. La elección de Terraform demuestra madurez tecnológica y un ecosistema extendido, ya que utiliza el lenguaje HCL (HashiCorp Configuration Language), el cual facilita la portabilidad de configuraciones y ofrece una mayor flexibilidad para organizaciones con nube híbrida, permitiendo una gestión más robusta del ciclo de vida de los recursos mediante su archivo de estado compartido(*.tfstate*)[7].

3.2.2 Fundamentos de Terraform

Terraform es un software de infraestructura como código desarrollado por HashiCorp y de código abierto que sirve para configurar e implementar la infraestructura en la nube. Codifica la infraestructura en archivos de configuración que describen el estado deseado para la topología. Terraform permite la administración de cualquier infraestructura, como nubes públicas, nubes privadas y servicios SaaS mediante distintos proveedores de Terraform.

En este proyecto se usan los siguientes proveedores:

- AzureRM: administra recursos y funcionalidades estables de Azure, como máquinas virtuales, cuentas de almacenamiento e interfaces de red.
- AzureAD: administra los recursos de Microsoft Entra, como grupos, usuarios, entidades de servicio y aplicaciones.

Los proveedores de Azure de Terraform permiten administrar toda la infraestructura de Azure mediante la misma sintaxis y grupo de herramientas. Podemos configurar funcionalidades básicas de la plataforma, configurar proyectos y canalizaciones de Azure DevOps para automatizar implementaciones de aplicaciones o implementar los recursos de Azure necesarios para aplicaciones.

La sintaxis de archivos de configuración basados en plantillas de Terraform permite configurar recursos de Azure de forma repetible y predecible. La automatización de la infraestructura ofrece ventajas como reducir los errores humanos, implementar la misma plantilla varias veces para crear entornos de desarrollo, prueba y producción idénticos y reducir el costo de entornos de desarrollo y prueba al crearlos a petición[8].

3.2.3 Ciclo de vida y plan de ejecución

Terraform se basa en un flujo de trabajo predecible que permite a los administradores del sistema gestionar la infraestructura como código de forma segura y verificable. Este proceso se realiza con el Plan de Ejecución.

Una vez escrito el código que implementa la infraestructura, donde se recomienda el uso de archivos de variables (*.tfvars*) para separar la lógica de los recursos de los datos y así facilitar la reutilización del código en distintos entornos, el comando *terraform init* prepara el espacio de trabajo para que Terraform pueda aplicar la configuración.

El comando *terraform plan* actúa como una etapa de validación obligatoria y ofrece una vista previa de los cambios que se han implementado antes del cambio real en el proveedor de la nube. Su función principal es determinar qué acciones son necesarias para que la infraestructura real coincida con el estado deseado descrito en los archivos de configuración y con los valores cargados desde el archivo *.tfvars*. Este análisis preventivo se fundamenta en:

- Lectura del estado actual: Terraform consulta el archivo de estado (*terraform.tfstate*) y la infraestructura desplegada para identificar la situación inicial.
- Comparativa: contrasta el código fuente con la realidad para detectar qué recursos faltan, cuáles sobran o qué ha cambiado. Los cambios utilizan una nomenclatura que permite obtener información de forma visual: creación (+), modificación (~), destrucción (-), reemplazo (-/+).

Por último, el comando *terraform apply* ejecuta los cambios propuestos por *terraform plan* y crea, actualiza o destruye recursos de forma permanente en el proveedor de la nube.

El ciclo de vida de Terraform transforma la gestión de la infraestructura en un proceso determinista. El plan de ejecución sirve tanto como herramienta de previsualización como para tener un mecanismo de auditoría y control de riesgos para entornos de producción y despliegue continuo[9].

3.2.4 Gestión de estado y backend remoto

La gestión del estado es el componente fundamental que permite a Terraform realizar un seguimiento de los recursos desplegados y su relación con el código fuente. Terraform utiliza el archivo *terraform.tfstate* para mapear los recursos de la infraestructura real con la configuración declarativa del usuario.

El propósito principal del estado de Terraform es almacenar enlaces entre objetos de un sistema remoto e instancias de recursos declaradas en la configuración. Cuando terraform crea un objeto remoto en respuesta a un cambio de configuración, registra su identidad en una instancia de recurso específica y posteriormente, puede actualizarlo o eliminarlo en respuesta a futuros cambios de configuración.

Aunque el formato de los archivos de estado es JSON, no se recomienda editar el estado directamente. Terraform proporciona el comando *terraform state* para realizar modificaciones básicas del estado mediante CLI.

Terraform espera una correspondencia directa entre las instancias de recursos configuradas y los objetos remotos. Esto se garantiza ya que Terraform es el responsable de crear cada objeto y registrar su identidad en el estado, o al destruir un objeto y luego eliminar su vinculación.

Es posible agregar o eliminar enlaces en el estado por otros medios, así como importar objetos creados externamente con el comando *terraform import* o pedirle a Terraform que obvie un objeto existente con *terraform state rn [10]*.

De forma predeterminada, el estado de Terraform se almacena localmente, pero una buena práctica es almacenarlo en Azure Blob Storage: un *backend* remoto. El estado local no funciona bien en entornos colaborativos y almacenar el estado locamente aumenta la posibilidad de eliminación de forma involuntaria[11].

3.2.5 Modularización y reutilización

Un módulo es un conjunto de recursos que Terraform gestiona conjuntamente. Al aprovisionar repetidamente conjuntos de recursos con configuraciones similares conviene escribir módulos reutilizables para codificarlos. Modularizar y compartir configuraciones ayuda a estandarizar el

aprovisionamiento de infraestructura y permite proporcionar los recursos necesarios de forma rápida y predecible.

Cada espacio de trabajo de Terraform incluye archivos de configuración en su directorio raíz, llamado módulo raíz. Los módulos que se configuran mediante bloques *module* se llaman módulos secundarios. Al aplicar una configuración, el módulo raíz llama al módulo secundario. Como resultado, Terraform añade los recursos del módulo secundario a su espacio de trabajo y los gestiona como parte de la configuración. Se puede configurar el módulo raíz para que llame a módulos secundarios varias veces dentro de la misma configuración[12].

Modularizar el proyecto permite aplicar separación de responsabilidades, reducción de duplicados y aceleración del despliegue.

3.3 Diseño de redes y seguridad (Modelo Zero Trust)

El modelo Zero Trust se basa en la implementación de los siguientes principios de seguridad:

- Verificación de forma explícita: se debe realizar una autenticación y autorización en función de todos los puntos de datos disponibles.
- Utilizar el acceso con menos privilegios: se debe limitar el acceso de usuario con Just-In-Time y Just-Enough-Access, las cuales conceden permisos por tiempo limitado para tareas específicas, en lugar de dar acceso permanente.
- Asumir la existencia de brechas: se debe comprobar el cifrado entre extremos y usar un análisis para obtener visibilidad, impulsar la detección de amenazas y mejorar las defensas.

Zero Trust presupone que cada solicitud proviene de una red no controlada, independientemente de su origen o del recurso al que acceda. Su objetivo es proteger cuentas de usuario, dispositivos, aplicaciones, redes, datos…

Utilizar la confianza por excepción permite a las organizaciones detectar, responder y bloquear eventos no deseados con mayor facilidad[13].

En cuanto a la red, se realizan una serie de buenas prácticas, tales como:

- Segmentación de subredes: se diseña una Virtual Network (VNet) con dos subredes críticas: snet-compute para la ejecución y snet-data para el almacenamiento.
- Aislamiento mediante Private Endpoints: algunos servicios se diseñan con Private Endpoints, eliminando cualquier exposición a Internet.
- Control de tráfico mediante Network Security Groups para filtrar el tráfico y User Defined Router (UDR) para interceptar el tráfico saliente vía Firewall/ Fortinet.

3.3.1 Diseño de la VNet y segmentación

Azure Virtual Network es un bloque de creación fundamental para la red privada. Permite a los recursos de Azure comunicarse de forma segura entre sí, Internet y redes locales.

Una red virtual es similar a una red tradicional, pero aporta ventajas como escalabilidad, disponibilidad y aislamiento.

Al crear una red virtual se tiene que especificar un espacio de direcciones IP privadas personalizado mediante direcciones públicas y privadas. Azure asigna a los recursos de una red virtual una dirección IP privada desde el espacio de direcciones que asigne[14].

La segmentación de redes divide una red en varias secciones, a las que se pueden aplicar distintos controles. Cuando se aplica correctamente, puede ayudar a las organizaciones en los siguientes casos:

- Reducción del movimiento lateral una vez los atacantes han penetrado en la red. Les impide moverse libremente por la red, lo que minimiza la superficie de ataque de la organización.
- Permite remediar la actividad maliciosa.

- Aumenta el rendimiento restringiendo cierto tipo de tráfico a zonas específicas, ayudando a reducir la congestión general de la red y optimizando el rendimiento.
- Garantiza el cumplimiento de las normas[15].

El ámbito de una red virtual es una sola región o ubicación, pero se pueden conectar varias redes virtuales de diferentes regiones entre sí mediante emparejamiento de red virtual.

Al crear una red virtual es importante asegurarse de lo siguiente:

- Los espacios de direcciones no pueden superponer.
- Las subredes no deberían cubrir el espacio de direcciones completo de la red virtual por si se necesita más en un futuro.
- Usar redes virtuales grandes en lugar de varias pequeñas reduce la sobrecarga de administración dentro de una misma suscripción de Azure.
- Asegurar las redes virtuales mediante la asignación de grupos de seguridad de red a las subredes[14].

3.3.2 Private Endpoints y Private Links

Un Private Endpoint es una interfaz de red que usa una dirección IP privada de la red virtual. Esta interfaz de red se conecta de forma privada y segura a un servicio mediante un Private Link. Al habilitar un punto de conexión privado, incopora el servicio a la red virtual.

Cuando se utilizan Private Endpoints, el tráfico se protege en un recurso de private link. La plataforma valida las conexiones de red y solo permite las que llegan al recurso del private link especificado[16].

Azure Private Link permite acceder a los servicios PaaS de Azure y a los servicios hospedados en Azure que son propiedad de los clientes, o a los servicios asociados, a través de un Private Endpoint. Así el tráfico entre la red virtual y el servicio viaja por la red troncal de Microsoft y ya no es necesario exponer el servicio a la red pública de Internet[17].

Los puntos de conexión privados admiten directivas de red, las cuales permiten la compatibilidad con grupos de seguridad de red (NSG), rutas definidas por el usuario (UDR) y grupos de seguridad de aplicaciones (ASG)[16].

3.3.3 Control de tráfico: NSG y UDR

El control de flujo de datos en la infraestructura se basa en la combinación de mecanismos de filtrado y de enrutamiento personalizado para garantizar que el tráfico siga las directivas de seguridad establecidas.

Un grupo de seguridad de red (NSG) se utiliza para filtrar el tráfico de red hacia y desde los recursos de Azure dentro de una Virtual Network.

Un NSG contiene una lista de reglas que permiten o deniegan el tráfico entrante o saliente basándose en listas de control de acceso (ACL) mediante el origen, puerto de origen, destino, puerto destino y protocolo. Las reglas se procesan por orden de prioridad y cuando el tráfico coincide con una regla, el proceso se detiene. Cada NSG incluye reglas por defecto que no se pueden eliminar, pero que pueden ser invalidadas por reglas de mayor prioridad creadas por el usuario[18].

Para complementar el filtrado de los NSG se implementa el enrutamiento personalizado mediante rutas definidas por el usuario (UDR), lo que permite anular las rutas de sistema predeterminadas de Azure.

Las UDR se utilizan para dirigir el tráfico de una subred hacia el siguiente salto, en lugar de permitir que Azure lo encamine automáticamente a Internet o entre subredes. El siguiente salto permite especificar que el tráfico debe pasar por un Firewall o un dispositivo Fortinet para realizar una inspección de seguridad antes de llegar a su destino. Además, mediante Azure Virtual Network Manager es posible gestionar las tablas de ruta de forma sistemática en múltiples redes virtuales, asegurando que las políticas de enrutamiento son consistentes en toda la topología de la red[19].

3.3.4 Identidad y control de acceso (RBAC)

Una identidad administrada es una identidad que se puede asignar a un recurso de proceso de Azure o a cualquier plataforma de hospedaje de aplicaciones compatible con Azure. Una vez asignada la identidad en el recurso, se puede autorizar para que pueda acceder a los recursos de dependencia posteriores. Una identidad administrada reemplaza los secretos, como las claves de acceso, contraseñas o incluso certificados.

Las entidades pueden ser asignadas por el usuario o por el sistema. En ambas se crea una entidad de servicio de tipo especial en Microsoft Entra ID para la identidad[20].

La administración de acceso a los recursos de la nube es una función vital para cualquier organización. El control de acceso basado en rol (RBAC) ayuda a administrar quién tiene acceso a los recursos, qué pueden hacer con esos recursos y a qué áreas pueden acceder.

Para controlar el acceso a los recursos que usan RBAC hay que utilizar las asignaciones de roles de Azure, que se basan en cómo se aplican los permisos. Una asignación de roles consta de tres elementos: entidad de seguridad (objeto que representa a un usuario o a una identidad administrada), definición de rol (recopilación de permisos y las acciones permitidas) y ámbito (conjunto de recursos al que aplica el acceso). Una asignación de roles es el proceso de asociar una definición de roles a un usuario, grupo, entidad de servicio o identidad administrada en un ámbito determinado con el fin de conceder acceso[21].

Las buenas prácticas de Azure determinan que es necesario conceder acceso solo a los usuarios que lo necesiten, así como limitar el número de propietarios de suscripciones, limitar las asignaciones de roles de administrador con privilegios, usar Microsoft Entra Privileged Identity Management para reducir el tiempo de exposición de los privilegios y aumentar la visibilidad, asignar roles a grupos, no a usuarios y asignar roles mediante identificador de rol único en lugar del nombre del rol[22].

3.4 Inteligencia artificial autónoma y orquestación de agentes

3.4.1 Modelos de lenguaje a gran escala: de sus orígenes al estado actual

Como ya se ha contado en el apartado 1.3: Antecedentes tecnológicos, la inteligencia artificial tiene sus raíces en el siglo XIX, pero el motor que impulsa a los agentes inteligentes son los Modelos de Lenguaje de Gran Escala (LLM). A diferencia de los primeros *chatbots*, que se basaban en reglas rígidas, los LLM se fundamentan en una arquitectura Transformer. Esta innovación presentada en 2017 se basa en un diseño de red neuronal que incluye un mecanismo de "atención", capaz de identificar las relaciones entre palabras de un texto, independientemente de su distancia en la oración o en el párrafo. Este enfoque permitió que los modelos empezaran a "entender" el contexto de una forma más cercana a como lo hacemos los humanos. De esta forma, el Transformer sentó las bases para los modelos de lenguaje actuales, capaces de interpretar el contexto inmediato y reconocer relaciones entre ideas, temas y secuencias de texto.

Con esta idea surgieron los primeros modelos funcionales, como BERT, desarrollado por Google, y GPT, desarrollado por OpenAI. A medida que aumentaba la capacidad de computación y se ampliaban los conjuntos de datos de entrenamiento, los LLM se volvieron cada vez más complejos y poderosos[23].

Un concepto importante para entender cómo estos modelos procesan la información es mediante el uso de tokens. Los tokens son pequeños fragmentos de texto que se generan al dividir el texto de entrada en segmentos más pequeños. Estos segmentos pueden ser palabras o grupos de caracteres, que varían en longitud de un solo carácter a una palabra completa[24]. Usando tokens es posible que el modelo mida la ventana de contexto.

3.4.2 Agente artificial

A medida que los modelos de inteligencia artificial generativa se vuelven más eficaces y omnipresentes, su uso crece más allá de las aplicaciones de chat para impulsar agentes inteligentes capaces de funcionar de forma autónoma para automatizar tareas. Cada vez más organizaciones

usan modelos de inteligencia artificial generativa para crear agentes que orquestan procesos empresariales.

Para escribir este apartado informativo sobre un inteligente artificial se ha utilizado como referencia la información de un curso formativo de Microsoft Learn: Introducción al desarrollo de agentes de IA en Azure[25].

Los agentes de inteligencia artificial son aplicaciones inteligentes que usan modelos de lenguaje para comprender lo que se necesita y tomar medidas para implementarlo. Pueden responder preguntas, tomar decisiones y completar tareas automáticamente.

Para que un agente sea eficaz es necesario que cumpla unas funcionalidades esenciales: integración y razonamiento de conocimientos usando un modelo generativo con documentación de directivas corporativas para responder preguntas de forma precisa; automatización de tareas a través de funciones; y toma de decisiones inteligente.

A medida que los agentes de inteligencia artificial se vuelven más autónomos e integrados en los sistemas empresariales, introducen nuevas consideraciones de seguridad. Dado a que los agentes pueden acceder a datos confidenciales, tomar decisiones y actuar de forma independiente, los desarrolladores y las organizaciones deben diseñarse teniendo en cuenta la seguridad desde un principio. Es necesario tener control de acceso aplicando controles de acceso basado en rol y permisos de privilegios mínimos: los agentes solo deben acceder a lo que necesitan absolutamente. También es necesario validad todas las entradas creando capas de filtrado y validación de mensajes para detectar y bloquear ataques por inyección antes de que lleguen al agente. Agregar supervisión humana para acciones críticas, realizar seguimientos, supervisar las dependencias y mantener los modelos en buen estado también son parte de las buenas prácticas de seguridad.

La creación de estos sistemas proactivos requiere marcos y herramientas especializados. Los marcos de inteligencia artificial tradicionales sirven para ayudar a los desarrolladores a integrar funcionalidades inteligentes básicas en aplicaciones, lo que mejora el rendimiento y la participación del usuario de maneras clave usando la personalización teniendo en cuenta las preferencias del usuario, la automatización y eficiencia y la experiencia de usuario. Por otra parte, los marcos de agente de IA van más allá al habilitar el desarrollo de agentes autónomos orientados a objetivos. Estos agentes no solo procesan datos: razonan, actúan y aprenden a lograr objetivos a través de funcionalidades como: colaboración y coordinación del agente admitiendo varios agentes que se comunican y trabajan juntos para resolver problemas complejos; automatización y administración de tareas; comprensión contextual y adaptación, permitiendo tomar decisiones basadas en datos en tiempo real y adaptarse a entornos cambiantes.

Para la correcta elección del marco hay que tener en cuenta las necesidades del usuario o del desarrollador. Microsoft ofrece varias soluciones, desde herramientas de poco código para usuarios empresariales hasta SDK completos para desarrolladores profesionales, cada uno diseñado para diferentes escenarios y niveles de aptitudes.

- Servicio Microsoft Foundry Agent. Es un servicio administrado en Azure diseñado para proporcionar un marco para crear, administrar y usar agentes de IA en Microsoft Foundry. El servicio se basa en la API de OpenAI Assistants, pero con una mayor elección de modelos, integración de datos y seguridad empresarial; le permite usar el SDK de OpenAI y el SDK de Azure Foundry para desarrollar soluciones agentes.
- API de asistentes de OpenAI. Proporciona un subconjunto de las características del servicio Foundry Agent y solo se puede usar con modelos OpenAI. En Azure, puede usar la API assistants con Azure OpenAI, aunque en la práctica el servicio Foundry Agent proporciona una mayor flexibilidad y funcionalidad para el desarrollo de agentes en Azure.
- Microsoft Agent Framework. Es un kit de desarrollo ligero que se puede usar para compilar agentes de inteligencia artificial y organizar soluciones multiagente. El marco sirve como plataforma optimizada específicamente para crear agentes e implementar patrones de solución agente.

- Autogen. Es un marco de código abierto para desarrollar agentes de forma rápida. Es útil como herramienta de investigación para experimentar con agentes.
- SDK de agentes de Microsoft 365. Los desarrolladores pueden crear agentes auto hospedados para su entrega a través de una amplia gana de canales mediante el SDK de agentes de Microsoft 365.
- Microsoft Copilot Studio. Proporciona un entorno de desarrollo de poco código que se puede usar para crear e implementar agentes que se integran con un ecosistema de Microsoft 365 o derivados. La interfaz de diseño visual de Copilot Studio lo convierte en una buena opción para crear agentes cuando tiene poca o ninguna experiencia profesional desarrollando software.

3.4.3 Servicio Microsoft Foundry Agent

Es un servicio dentro de Azure que se puede usar para crear, probar y administrar agentes de IA. Proporciona una experiencia de desarrollo de agente visual en el portal de Microsoft Foundry y una experiencia de desarrollo de código mediante el SDK de Microsoft Foundry.

Los agentes desarrollados mediante Foundry Agent Service tiene los siguientes elementos:

- Modelo: un modelo de IA generativo implementado que permite al agente razonar y generar respuestas de lenguaje natural a las solicitudes. Puede usar modelos de OpenAI comunes y una selección de modelos del catálogo de modelos de Microsoft Foundry.
- Conocimiento: orígenes de datos que permiten al agente establecer solicitudes con datos contextuales. Entre los posibles orígenes de conocimiento se incluyen los resultados de búsqueda en Internet de Bing, un índice de Azure AI Search o sus propios datos y documentos.
- Herramientas: funciones mediante programación que permiten al agente automatizar acciones. Se proporcionan herramientas integradas para acceder a los conocimientos de Azure AI Search y Bing, así como una herramienta de intérprete de código que puede usar para generar y ejecutar código de Python. También se puede crear herramientas personalizadas con su propio código o Azure Functions.

Las conversaciones entre usuarios y agentes tienen lugar en un subproceso, que conserva un historial de los mensajes intercambiados en la conversación, así como los recursos de datos y los archivos que se generan.

A medida que los modelos de inteligencia artificial generativa se vuelven más eficaces y omnipresentes, su uso crece más allá de las aplicaciones de chat para impulsar agentes inteligentes capaces de funcionar de forma autónoma para automatizar tareas. Básicamente, los agentes de IA son aplicaciones inteligentes que usan modelos de lenguaje para comprender lo que se necesita y tomar medidas para implementarlo; pueden responder preguntas, tomar decisiones y completar tareas automáticamente. Para que un agente sea eficaz es necesario que cumpla unas funcionalidades esenciales: integración y razonamiento de conocimientos usando un modelo generativo con documentación de directivas corporativas para responder de forma precisa; automatización de tareas a través de funciones; y toma de decisiones inteligente[25].

3.4.4 IA Generativa vs IA Agéntica

La IA generativa es una inteligencia artificial que puede crear contenido original en respuesta a una instrucción o solicitud del usuario. Se basa en el uso de modelos de *machine learning* denominados modelos de *deep learning*, algoritmos que simulan los procesos de aprendizaje y la toma de decisiones del cerebro humano, y otras tecnologías como la automatización de procesos robóticos. Por sí sola es pasiva: responde a lo que le preguntas basándose en su entrenamiento.

La IA agéntica lleva las capacidades autónomas al siguiente nivel mediante el uso de un ecosistema digital de modelos de lenguaje de gran tamaño (LLM), *machine learning* (ML) y procesamiento del lenguaje natural (PLN) para realizar tareas autónomas en nombre del usuario o de otro

sistema. Se centra en las decisiones en lugar de crear el nuevo contenido real y no se basa únicamente en instrucciones humanas ni requiere supervisión humana. La IA agéntica tiene la capacidad de generación mediante un motor de razonamiento para interactuar con el entorno.

Básicamente, la IA generativa es el componente que permite al agente razonar y comunicarse mientras que la arquitectura agéntica es la que le da autonomía para resolver problemas y actuar dentro de los procesos[26].

3.4.5 El patrón RAG

Para superar las limitaciones de los modelos de lenguaje a gran escala como las alucinaciones o conocimiento desfasado, se utiliza la arquitectura RAG (generación aumentada por recuperación). Este patrón permite que el modelo no dependa únicamente de sus parámetros internos, sino que consulta una base de conocimiento externa antes de generar una respuesta.

El proceso RAG se define en dos fases críticas: la recuperación de fragmentos relevantes de un documento y la generación de una respuesta basada en esos fragmentos.

El flujo de un sistema RAG comienza con la indexación de información, donde se lleva a cabo la recopilación y organización de datos provenientes de diversas fuentes. También se realizan técnicas de pre-procesamiento para eliminar elementos irrelevantes. Después de esto, se generan *embeddings* para cada fragmento de información, que son unas representaciones vectoriales que capturan el significado semántico de los textos. De esta forma, la información se convierte en vectores en un espacio de alta dimensión. Estos *embeddings* se almacenan en la base de datos de vectores permitiendo un acceso eficiente durante la etapa de recuperación.

Cuando un usuario realiza una consulta, se inicia el proceso de recuperación de información, la cual realiza el mismo recorrido que se ha mencionado anteriormente, el *embedding*, y después se compara la consulta del usuario vectorizada con la de la base de datos, para así recuperar los fragmentos más similares según su relevancia semántica.

Una vez identificada la información relevante, se inicia el proceso de generación de la respuesta. El sistema RAG utiliza el contenido recuperado para alimentar a un modelo generativo que elabora respuestas fundamentadas en dicho contenido[23, pp. 22–23].

3.4.6 Ecosistema de frameworks y selección de herramientas

Para el desarrollo de agentes inteligentes, Microsoft ofrece un gran repertorio de opciones de desarrollo, que se pueden elegir en función de lo que necesite el desarrollador y el tipo de tarea que deba realizar el agente.

La arquitectura técnica del sistema se fundamenta en la integración de tres tecnologías de Microsoft que permiten pasar de un modelo estático a un agente autónomo y conectado. En el centro de la solución se encuentra Azure Ai Agent Service (Foundry), que actúa como la plataforma principal de hospedaje y gestión, proporcionando la infraestructura de seguridad empresarial, el acceso a los modelos de lenguaje y la administración de los hilos de conversación[27]. Para estructurar la lógica de estos agentes de manera modular, se utiliza Microsoft Agent Framework(MAF), un kit de desarrollo de código abierto que facilita la orquestación de tareas define como el agente debe razonar ante las peticiones del usuario y compila agentes de IA y flujos de trabajo de varios agentes[28]. Finalmente, la conectividad con el mundo exterior se logra mediante el Model Context Protocol (MCP), un estándar que funciona como el conector universal para que el agente acceda de forma segura a herramientas y fuentes de datos externos, como bases de datos o APIs a terceros[29]. La relación entre ellos es jerárquica y funcional: Azure AI Agent Service pone la infraestructura, MAF aporta la inteligencia operativa para organizar los procesos y MCP proporciona las conexiones para que el agente ejecute acciones reales sobre el entorno corporativo.

A la hora de desarrollar la aplicación, teniendo en cuenta que se usará Python, ya que es el lenguaje que ofrece mayor número de ventajas en la orquestación de agentes inteligentes por su catálogo extenso de librerías compatibles, su simplicidad y entendimiento claro, su capacidad de utilizar herramientas que pueden ser invocadas por un agente, programación asíncrona y sobre todo la integración nativa con Azure y OpenAI. Además, un *backend* en Python, frecuentemente

implementado con *frameworks* de alto rendimiento, como por ejemplo FastApi, funciona como puerta de enlace API que recibe entradas del *frontend*, ejecuta inferencias sobre modelos pre entrenados y devuelve respuestas JSON limpias y estandarizadas que la interfaz consume [30].

La elección de Azure AI Agent Service (utilizando AzureAIAgentClient) frente al cliente de chat directo (AzureOpenAIChatClient) se justifica fundamentalmente por la gestión del estado y la orquestación automática. Mientras que el cliente directo es *stateless* y obliga a la aplicación a gestionar manualmente todo el historial de conversación y el bucle de ejecución de herramientas, el Agent Service abstrae esta complejidad en el lado del servidor. Al utilizar el servicio de Agentes, delegamos en Azure la persistencia de los hilos de conversación (*threads*) y la lógica de razonamiento iterativo (*runs*), lo que permite integrar capacidades avanzadas (como búsqueda de archivos o ejecución de código) de forma nativa y reduce drásticamente el código *boilerplate* (código que se repite) necesario para mantener la coherencia del contexto en conversaciones largas[31].

3.4.7 Orquestación y microservicios

Los microservicios pueden dividir las funcionalidades de un sistema en componentes independientes y modulares. El uso de microservicios es esencial para crear aplicaciones escalables y de alta disponibilidad, donde cada servicio puede crecer o actualizarse sin afectar al resto del sistema[32]. Complementariamente, la orquestación es el mecanismo centralizado que coordina las interacciones y el flujo de datos entre componentes[33].

En el proyecto actual, se aplican estos principios mediante una arquitectura modular basada en servicios en la nube. Aunque no se despliegue una malla de contenedores distribuida tradicional, se desacoplan las capacidades funcionales (búsqueda, correo, gestión de tareas) en módulos independientes (Tools y MCP) que actúan como microservicios lógicos[34].

En el caso de los agentes inteligentes, el orquestador es Azure AI Agent Service y se encarga de decidir a qué servicio invocar, coordinar las llamadas a las distintas herramientas (Tools), gestionar errores o escalar los recursos[35].

3.4.8 Seguridad y ética en agentes autónomos

En la era digital en la que vivimos, la implementación de agentes con capacidad de acción requiere un marco de seguridad muy estricto. Al proporcionar a la inteligencia artificial la capacidad de ejecutar herramientas, se introducen riesgos de seguridad.

Existen muchas amenazas en términos de seguridad de la información. Las amenazas internas son una gran amenaza para la computación en la nube, y dichas amenazas se basan principalmente en errores humanos no intencionados. Estos errores pueden ser costosos y dañinos para una organización. Además, la pérdida de control se produce cuando un flujo de trabajo se externaliza en la nube, lo que significa que se pierde el control sobre las tareas[36].

La fuga de datos es otra de las preocupaciones en la nube, mediante configuraciones inadecuadas, vulnerabilidades de seguridad que pueden ser explotadas por actores maliciosos y muchas veces inadvertidas por parte de los propios usuarios. Para poder mitigar de la mejor forma posible estos ataques es recomendado seguir buenas prácticas como por ejemplo encriptar datos, establecer políticas de control de acceso o revisiones constantes, entre otras.

En este proyecto, la seguridad se basa en el principio de privilegio mínimo (PoLP). El agente no posee credenciales propias de administrador, sino que actúa bajo permisos estrictamente definidos mediante RBAC de Azure[37].

Otra de las mayores amenazas en un agente de inteligencia artificial es el *Prompt Injection*, que es una técnica en la cual un usuario intenta engañar al agente dando instrucciones no validadas que sobrescriben las originales. Existe la validación de entradas para prevenir estos ataques. Se implementan filtros para ellos, aunque en el proyecto se han usado los filtros nativos de Azure AI Content Safety[38]. Además, cada herramienta cuenta con su propio flujo de autenticación usando Managed Identities, keys o tokens.

Desde el punto de vista ético, la autonomía del agente no es absoluta. Se sigue el principio de IA responsable de Microsoft[39], que establece que la IA debe ser un apoyo y no un sustituto de la responsabilidad humana. El agente debe ser capaz de explicar de manera razonada sus decisiones y acciones, y también necesita confirmación humana antes de ejecutar ciertas tareas de implementación.

3.4.9 Comunicación multimodal y tiempo real

Para que un agente sea más completo, se añaden capacidades que permiten que la experiencia del usuario se aproxime a una conversación humana natural, teniendo en cuenta sobre todo la baja latencia.

El agente incorpora el modelo GPT-Realtime de Foundry que permite operar en dos modalidades distintas: interacción textual mediante chat gestionada por el *backend* mediante FastAPI, la cual incorpora razonamiento profundo, uso de herramientas corporativas y persistencia del historial de conversación; interacción vocal en modo *live* para hablar de forma natural con el agente. Este último, el modo *live*, se habilita mediante el servicio RealTimeService que utiliza un modelo "Speech-to-Speech", lo que permite captar matices en la entonación y responder con una latencia conversacional humana.

Para garantizar la experiencia del usuario, se implementan dos técnicas de transmisión de datos en tiempo real: *streaming* de texto, permitiendo que el usuario perciba una respuesta casi inmediata; y *streaming* de Audio Bidireccional mediante WebSockets, que implementa el modo voz con Azure OpenAI. Con este último, el sistema habilita una interrupción natural llamada "Barge-in" la cual permite que el sistema detecte cuando el usuario hable y permita detener el flujo de audio, permitiendo así imitar una conversación real[40].

3.5 Diseño de persistencia

La persistencia es la capacidad de un sistema para almacenar, conservar y recuperar información de manera estructurada a lo largo del tiempo. En una arquitectura de agentes inteligentes, la persistencia cumple dos funciones: memoria a corto plazo (historial de conversación o *threads*) y memoria a largo plazo (manuales, conocimientos técnicos, registros históricos…).

Para su diseño, es necesario identificar los requisitos de los servicios de datos mediante una evaluación de las aplicaciones y servicios que conforman las cargas de trabajo, para así determinar los requisitos de almacenamiento y acceso a datos óptimos.

Los requisitos funcionales obligan a observar la naturaleza de los datos y su uso, considerando el formato, la finalidad, las capacidades de búsqueda, las relaciones entre datos, los modelos de consistencia, la flexibilidad de lectura y escritura, la necesidad de concurrencia y el ciclo de vida de la información.

Por otro lado, los requisitos no funcionales evalúan las expectativas de rendimiento y escalabilidad mediante la medición de latencia, confiabilidad, disponibilidad y los límites de tamaño. También deben considerarse factores de coste y gestión, como la residencia de los datos para disponibilidad regional, si se trata de un servicio administrado o autoalojado, las licencias, la portabilidad y la existencia de almacenamiento en niveles o caché.

Dependiendo de la naturaleza de la información, existen diferentes estructuras de almacenamiento.

Los sistemas de gestión de bases de datos relacionales son ideales para datos con esquemas fijos y relaciones complejas que requieren garantías de cumplimiento, como por ejemplo Azure SQL Database, Azure Database for MySQL o Azure Database for Postgres.

Las bases de datos de documentos, como por ejemplo los archivos JSON, se utilizan para los datos semiestructurados, donde cada registro puede tener una estructura diferente, facilitando la gestión de catálogos o perfiles.

Para datos que se centran en las conexiones de entidades, se emplean bases de datos de grafos, como Azure CosmosDB, aunque también ofrece capacidades que incluyen almacenes de clave-

valor para búsquedas rápidas de datos simples, almacenes de índices de búsqueda optimizados para consultas de texto completo y almacenes de series temporales para datos que se registran cronológicamente. Otro ejemplo de clave-valor sería Azure Managed Redis.

Si las cargas de trabajo requieren capacidades de análisis de datos de alta capacidad de nivel empresarial, Microsoft Fabric actúa como una plataforma integral que unifica el movimiento, procesamiento e ingesta de datos; si se necesitan motores de búsqueda optimizados con IA, se utiliza Azure AI Search, y para el análisis de grandes volúmenes de datos de series temporales en tiempo real, se emplea Azure Data Explorer.

En el caso de requerir SQL Server, este puede ejecutarse en un modelo IaaS sobre Azure Virtual Machines o en un servicio PaaS como SQL Database, dependiendo si se desea administrar manualmente la infraestructura y copias de seguridad o delegar estas acciones a Azure.

La persistencia de Azure debe configurarse según la carga de trabajo mediante el rendimiento aprovisionado, que asigna recursos fijos para obtener costes predecibles, o mediante el autoescalado, que ajusta automáticamente los recursos según la demanda para garantizar la eficiencia del agente[41].

Capítulo 4. Desarrollo e implementación del sistema de agentes

4.1 Diseño de infraestructura como código con Terraform

Se procede a construir una infraestructura que permite definir y gestionar diferentes recursos de Azure utilizando la herramienta Terraform. La utilización de esta nos permite automatizar el despliegue de recursos de Azure de manera efectiva mediante el uso del código. La arquitectura se ha diseñado mediante modularización, separando responsabilidades en componentes independientes para facilitar el mantenimiento y próximas actualizaciones de código.

Los recursos han sido desplegados en Azure mediante una suscripción proporcionada por NTT DATA Spain, lo que ha permitido trabajar en un entorno empresarial real.

4.1.1 Estrategias de diseño y buenas prácticas

Para que el proyecto siga las mejores prácticas, es recomendable el uso de KeyVault[42] para evitar la exposición de credenciales y datos sensibles. Otra buena práctica es almacenar el archivo *terraform.tfstate* de forma remota y segura, por ejemplo, en Azure Storage Account[43], lo que permite un mejor funcionamiento y confidencialidad. También es importante el uso de funciones *lower()* y *replace()* para cumplir con las restricciones de nomenclatura de Azure y evitar errores.

4.1.2 Recursos desplegados y módulos

La infraestructura implementada se organiza en los módulos detallados a continuación, cuya estructura se muestra en la Figura 1:

El módulo vnet despliega una Virtual Network dividida en dos subredes: snet-compute y snet-data. Se asocian a ellas Nework Security Groups (NSG) para filtrar el tráfico definiendo unas reglas Network Security Rule para lograrlo. Se establecen también Private Endpoints en el módulo private_endpoint para los servicios PaaS, estableciendo conectividad privada y evitando el acceso a través de Internet público.

El módulo container_app_environment crea un entorno de Azure Container Apps (CAE) inyectado en la vnet, permitiendo la autenticación con Azure Container Registry (ACR) para la descarga de imágenes y utilizando credenciales de Azure AD para la autenticación de la aplicación, alojando los contenedores. También se crea el módulo static_web_app, que crea un recurso capaz de alojar la parte del *frontend* de la aplicación.

Tal como se dijo en el apartado 3.2, Container App 1 aloja el código de la API y la lógica de conexiones del agente de IA y se conecta a la base de datos, al AiSearch y al Storage Account. Container App 2 funciona como orquestrador de *workflows*, donde maneja los flujos de trabajo de la planta, consumiendo la API y Static Web App.

A través del módulo storage_account creamos el recurso de un Azure Storage Account, el cual almacena blobs de datos de telemetría y otros archivos no estructurados de planta.

El módulo app_registration permite registrar la aplicación para gestionar la autenticación a través de Microsoft Entra ID (Azure AD).

También nos harían falta desplegar los recursos que funcionarán como herramientas del agente de inteligencia artificial, pero en el caso del proyecto actual, la empresa NTT DATA ya cuenta con estos recursos desplegados y ha proporcionado el material necesario para la conexión al recurso.

Las relaciones entre los distintos componentes son las siguientes:

-Autenticación: el módulo app_registration crea la identidad que el módulo container_app_environment utiliza para proteger y autenticar servicios.

-Red: container_app_environment reside dentro de la VNet. Los servicios de datos están diseados para ser accedidos de forma privada, preferiblemente usando el private_endpoint conectado a la VNet.

-Despliegue de apps: los Container Apps tiran de imágenes desde el ACR y se ejecutan en el entorno que los gestiona.

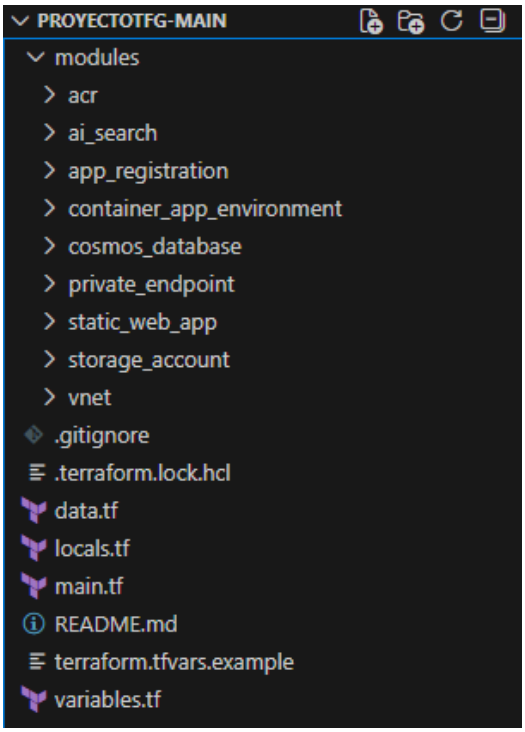

Figura 1. Estructura módulos Terraform

Además, cada módulo está dividido en 3 componentes esenciales que permiten tener una estructura limpia y concisa tal y como se muestra en la Figura 2.

Se sabe que *main.tf* sirve para definir la lógica de los recursos de Azure, *output.tf* sirve para exportar los atributos necesarios para la interconexión entre los diferentes módulos y *variables.tf* permite parametrizar la configuración y reutilizar el código.

Figura 2. Estructura del módulo

4.2 Preparación del entorno de desarrollo y configuración

Una vez desplegada la infraestructura, se prepara el *backend* para conectar con los recursos de Azure.
Para la preparación del entorno de desarrollo, se ha diseñado una arquitectura modular programada en Python, en específico Python 3.11, utilizando FastAPI como *framework* principal para el *backend* debido a su alto rendimiento en operaciones asíncronas.
La gestión de dependencias se centraliza mediante el archivo *requirements.txt*, integrando librerías como el SDK de Azure AI Agent para la orquestación de la inteligencia artificial, SQLAlchemy para la persistencia de datos y azure-identity para la autenticación segura mediante DefaultAzureCredential.
Siguiendo las buenas prácticas de seguridad, la configuración del sistema se ha desacoplado del código mediante el uso de variables de entorno alojadas en el archivo *.env,* el cual está protegido del exterior mediante *.gitignore*, permitiendo gestionar de forma segura credenciales críticas como los *endpoints* de Azure OpenAI, las claves de Bing Search y los tokens de Microsoft Graph,

utilizados para conectar las *tools*. Para la inicialización de esas variables de entorno se utiliza el archivo *config.py*.
Adicionalmente, para garantizar la consistencia entre los entornos de desarrollo y producción, se ha implementado la contenerización de la aplicación mediante Docker, asegurando la disponibilidad de todos los drivers del sistema y facilitando el despliegue en infraestructuras de la nube.

4.3 Implementación del agente inteligente

Tal y como se mencionó en el apartado anterior, la implementación del sistema inteligente se ha desarrollado sobre una arquitectura modular en Python 3.11, diseñada para soportar operaciones asíncronas de alta concurrencia y garantizar escalabilidad.

4.3.1 Orquestación del backend con FastAPI

El servidor principal se ha construido utilizando FastAPI, un *framework* moderno de alto rendimiento que permite la creación de apis asíncronas. La elección de esta tecnología responde a la necesidad de gestionar conexiones de larga duración, como el *streaming* de respuestas del agente y la comunicación en tiempo real. El servidor utilizado es Uvicorn, el cual recibe las conexiones de red y se las pasa a la aplicación.

El punto de entrada es *main.py* y define *endpoints* asíncronos que evitan el bloqueo del hilo principal durante operaciones de entrada y salida, como consultas a la base de datos o las llamadas a APIs externas como Azure Open AI, Microsoft Graph…

Utilizando el sistema de inyección de dependencias "*Depends*", se consigue la validación de Tokens en cada petición, asegurando que solo usuarios autenticados puedan interactuar con el agente.

Para mejorar la experiencia del usuario y reducir la latencia, el *endpoint* del chat (/api/chat) no devuelve una respuesta JSON estática, sino que implementa Streaming Response para enviar eventos enviados por el servidor, transmitiendo las respuestas del LLM token por token a medida que se va generando.

4.3.2 Definición y ciclo de vida del agente corporativo

La lógica del agente está en el archivo *corporate.py*, que encapsula la creación y la ejecución del agente. También aquí se define la personalidad, instrucciones y el modelo de razonamiento. El modelo de razonamiento utilizado se define también en las variables de entorno, y se escoge en función de la inteligencia, proceso de toma de decisiones e interacción con su entorno [44]. Se utiliza el código de la Figura 3 para la creación del agente.

```python
async def create_session(self):

    # 1. Cliente de Infraestructura Azure
    async with AzureAIAgentClient(
        credential=self.credential,
        endpoint=config.AZURE_AI_PROJECT_ENDPOINT,
        model=config.AZURE_AI_MODEL_DEPLOYMENT_NAME
    ) as chat_client:

        # 2. Agente Lógico
        async with ChatAgent(
            chat_client=chat_client,
            name=config.AGENT_NAME,
            instructions=self.instructions,
            tools=self.tools,
            tool_resources=create_azure_ai_search_resources(),
        ) as agent:
            yield agent
```

Figura 3. Creación del agente mediante python

El comportamiento del agente se rige por un conjunto de instrucciones llamadas *System Prompts*, definidas en el archivo *config.py*. Estas instrucciones establecen cómo actuará el agente, su tono,

sus restricciones y el orden de funcionamiento que seguirá. En específico, el *prompt* que se ha proporcionado al agente en este proyecto es el de la Figura 4:

```
AGENT_NAME = "AgenteCompleto"
   AGENT_INSTRUCTIONS = (
       "Eres un asistente que realiza varias tareas. "
       "Puedes realizar búsquedas mediante la herramienta de Azure AI Search para consultar manuales técnicos."
       "Sé preciso. Contesta claramente a lo que te pregunten."
       "Proporciona DIRECTAMENTE la respuesta final o la información encontrada sin preámbulos innecesarios. "
       "Primero busca mediante la herramienta Azure AI Search y si no encuentras nada utiliza la herramienta Bing. "
       "Una de las tareas es enviar correos usando la herramienta 'send_email' cuando se te pida. "
       "Puedes hacer búsquedas mediante la herramienta de Bing para que tus datos sean contrastados y reales. "
       "No inventes nada, tu información tiene que ser validada. "
       "Responde en español a no ser que se especifique lo contrario o te pregunten en otro idioma. "
   )
```

Figura 4. Instrucciones del agente

Para el ciclo de vida del agente se ha implementado Context Manager (async with) que realiza una gestión automática de recursos. Esto garantiza que las conexiones con el servicio de Azure y los recursos de red se inicialicen y se liberen correctamente en cada sesión de chat.

Aunque el modelo de lenguaje es *stateless*, el sistema mantiene la continuidad de la conversación mediante la identificación de los hilos (*threads*). En el archivo *chat.py*, el servicio ChatService realiza la recuperación del historial desde una base de datos SQL (usando SQLAlchemy) y lo inyecta en el contexto del agente cuando es necesario, permitiendo el acceso a mensajes anteriores y permitiendo usarlos para dar contexto y usar referencias.

4.3.3 Herramientas y capacidades extendidas

El agente dispone de un conjunto de herramientas ejecutables que permiten interactuar con el entorno y con sistemas externos. El modelo de lenguaje GPT-4 actúa como orquestador que, basándose en la interacción del usuario y en las funciones definidas, decide qué herramienta llamar. Estas herramientas se definen en la carpeta *src/tools* del proyecto y se inyectan en el agente de forma dinámica en el momento de su creación.

- Gestión de correos electrónicos: mediante una implementación directa con la API REST de Microsoft Graph en el archivo *email.py*, el agente es capaz de enviar de forma autónoma un correo electrónico proporcionándole los parámetros de destinatario, asunto y cuerpo, que pueden ser recopilados a través del usuario utilizando un lenguaje natural. Se utilizan peticiones HTTP seguras autenticadas mediante tokens. Realiza una implementación asíncrona utilizando la librería httpx, lo que permite tener escalabilidad en el servidor FastAPI, permitiendo que el *backend* gestione otras solicitudes mientras espera la confirmación del servidor de correo. Además, la *tool* de correo realiza la autenticación mediante un token de acceso en el caso de no poder acceder simplemente mediante DefaultAzureCredential.
- Recuperación de información en tiempo real: para proporcionar información actualizada se utiliza la herramienta de Bing Search. Mediante HostedWebSearchTool se conecta a Bing Grounding a través de Azure AI Agent, permitiendo al agente realizar búsquedas semánticas y citar fuentes.
- Sistema RAG: mediante Azure AI Search se permite el acceso a documentación técnica corporativa usando búsquedas vectoriales.
- Conectividad MCP: el uso del estándar Model Context Protocol permite interactuar con servidores externos proporcionando el *endpoint* correspondiente. Se utiliza MCPStreamableHTTPTool para establecer una conexión HTTP. El agente consulta al servidor MCP que herramientas ofrece (por ejemplo tickets de Jira) y el servidor MCP devuelve la definición JSON de la herramienta. Después el agente invoca la herramienta y por último el resultado se devuelve al contexto de la conversación. Con este diseño se permite nuevas integraciones posteriormente desplegando nuevos servidores MCP, sin necesidad de volver a desplegar el *backend* del agente.

4.4 Persistencia de conversaciones y gestión de hilos

Como ya se ha descrito anteriormente, el sistema implementa una memoria persistente para no perder el contexto. Se utiliza una estructura de hilos (*threads*) donde cada conversación se identifica mediante un ID único y se diferencia por roles (usuario y asistente). Además, la persistencia se realiza bien con un archivo json o utilizando una base de datos SQL.

Mediante SQLAlchemy, se ha diseñado una conexión que evalúa dinámicamente el entorno de ejecución para seleccionar el driver más adecuado: utiliza ODBC (por pyodbc) cuando se dispone de una cadena de conexión completa, o utiliza pymssql como alternativa ligera en entornos sin drivers del sistema. Es posible autenticarse mediante tokens de Azure Active Directory obtenidos en tiempo de ejecución a través de la identidad administrada del recurso, sin tener que almacenar credenciales estáticas en el código.

En la base de datos existen las tablas ThreadDB, que representa una conversación o hilo, en la cual se almacenan *id, title, user_id* y fechas de actualización; y MessageDB que representa los mensajes individuales, vinculados a un *thread_id*, y almacenan *role, content y timestamp*. Está configurada de manera que, si se borra un hilo, se eliminan automáticamente todos sus mensajes. Se puede comprobar mediante herramientas de visualización de bases de SQL como por ejemplo SQLite Viewer Web App si realmente los datos se están guardando en la base de datos (Figura 5). Si la base de datos es de Azure, se puede observar mediante el recurso Azure SQL Database.

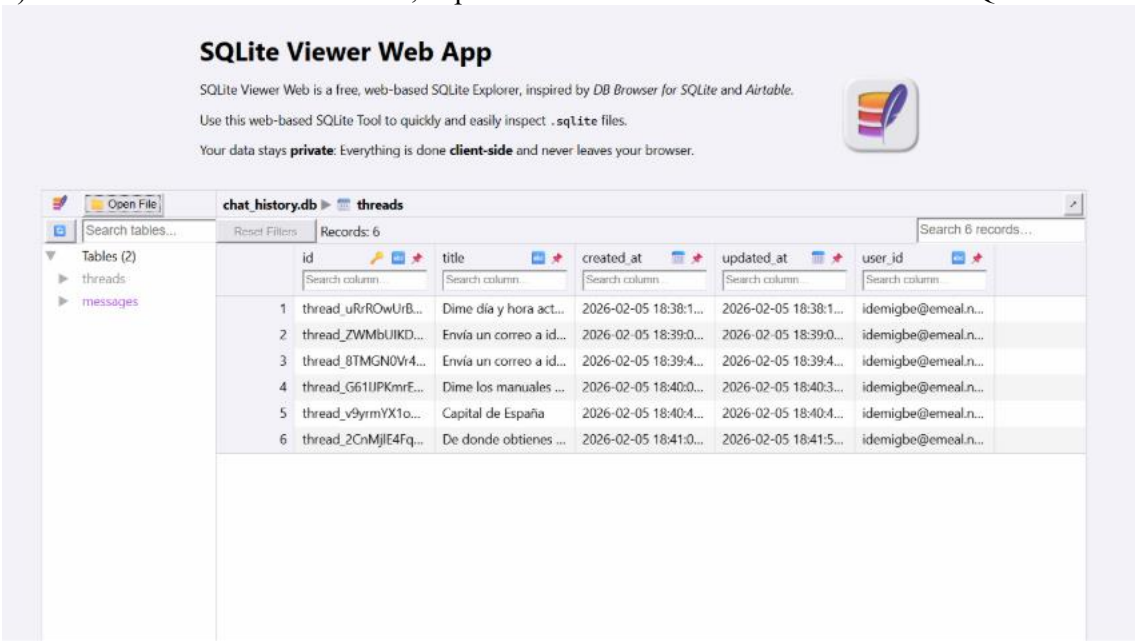

Figura 5. Base de datos

4.5 Interfaz de usuario

4.5.1 Arquitectura y tecnologías web

La presentación del sistema al usuario se ha diseñado como una Single Page Application (SPA) ligera, utilizando Vanilla JS, HTML y CSS moderno, que prioriza la eficiencia, la baja latencia y la compatibilidad universal, estando desacoplado el *frontend* con el *backend*, comunicándose exclusivamente a través de API REST y WebSocket. Podría haberse realizado con *framework* de React o Angular, estándares en la industria para aplicaciones de gran escala por su arquitectura basada en componentes y gestión de estados, pero se ha optado por una implementación SPA Nativa(Vanilla JS) por su optimización de latencia en *streaming* que garantiza que el renderizado de texto token a token sea fluido e inmediato; por su control de WebSockets al implementar el modo de voz en tiempo real en el agente; y por la simplicidad de despliegue, reduciendo la complejidad del contenedor Docker y facilitando la integración continua.

Para usar la aplicación, el usuario final debe autenticarse con su usuario (en este caso una cuenta compatible con la suscripción de Azure, tal y como se muestra en la Figura 7) en la pantalla principal (Figura 6).

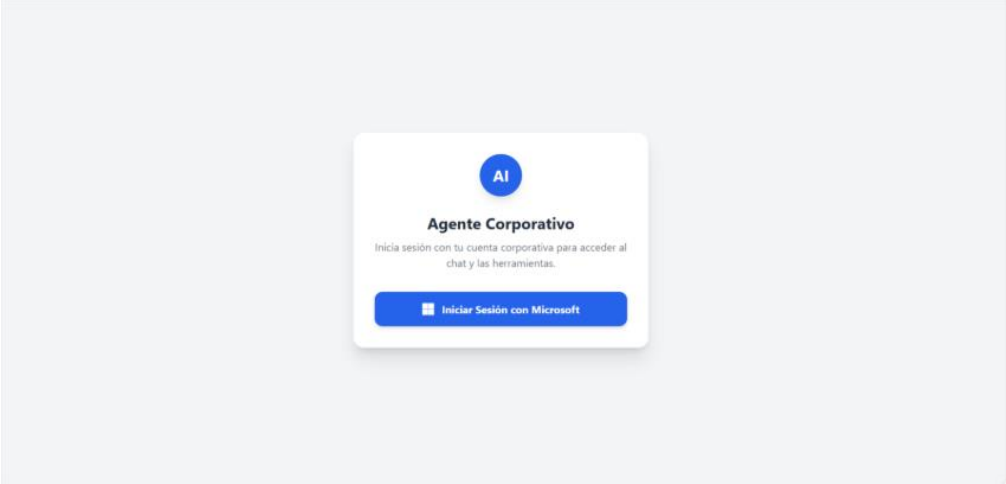

Figura 6. Página principal

Figura 7. Autenticación Microsoft

Tras la correcta autenticación, se abrirá la ventana del chat (Figura 8), donde se tiene la opción de crear una nueva conversación o elegir una ya creada si existe (Figura 9).

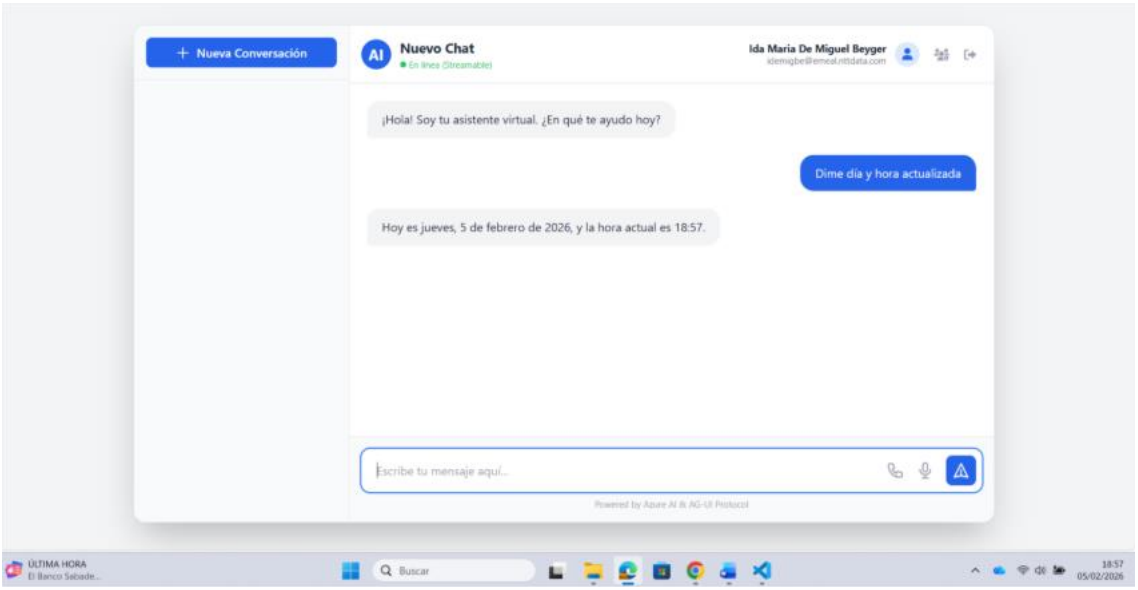

Figura 8. Nuevo chat sin conversaciones guardadas

c

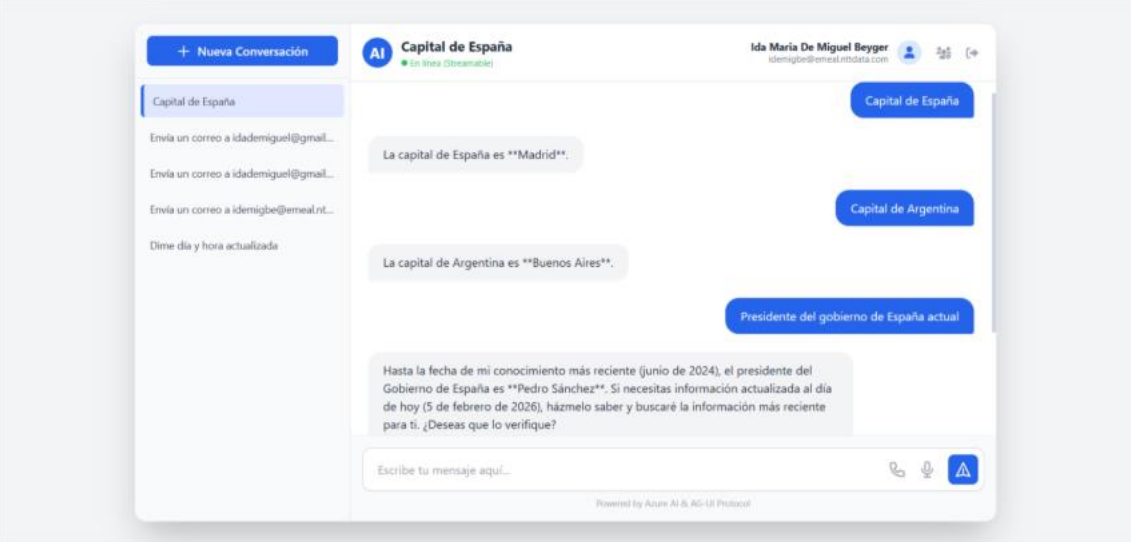

Figura 9. Pantalla de chat con conversaciones anteriores guardadas

Realizando pruebas en el chat, se puede comprobar que las herramientas de envío de correo o búsqueda en internet funcionan correctamente.

Al pedirle que envíe un correo por el chat (Figura 10), se puede comprobar como el agente realiza la acción automáticamente en un breve periodo (5-10 segundos) y el correo llega al destinatario de manera correcta (Figura 11).

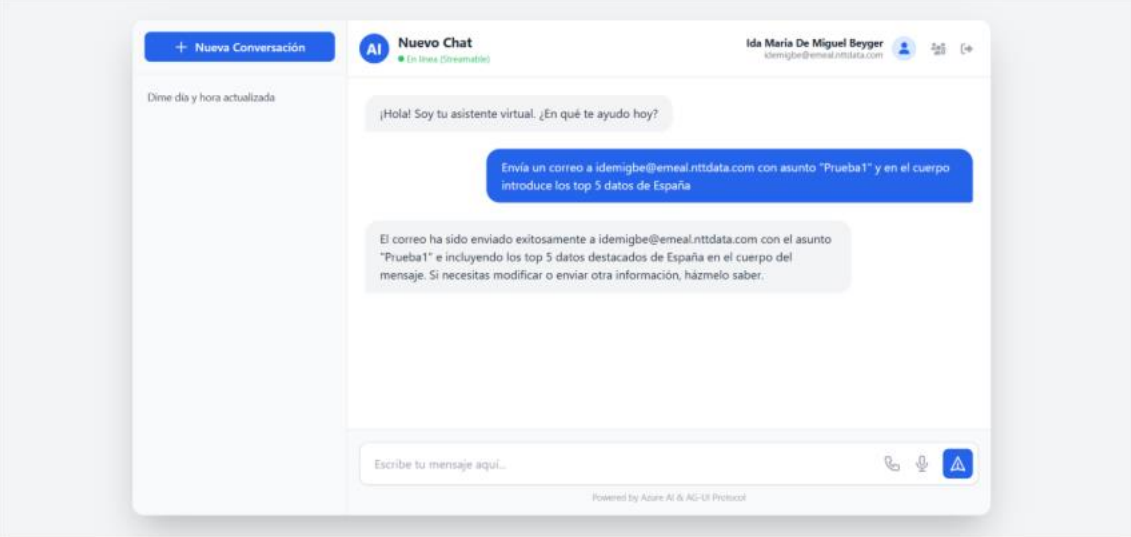

Figura 10. Petición de envío de correo electrónico

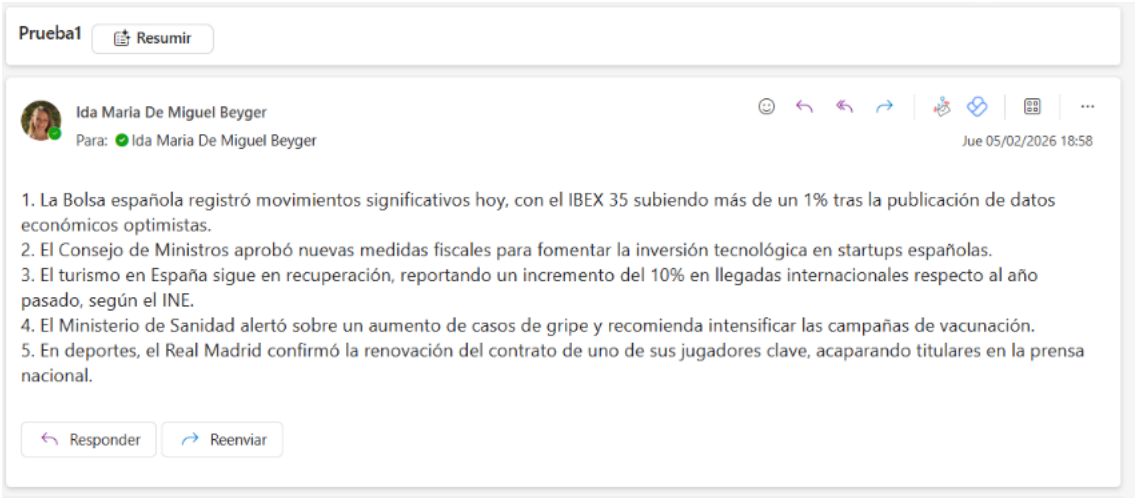

Figura 11. Recepción de email

Para el correcto funcionamiento de la herramienta Azure AI Search, es posible pedir al agente información sobre documentos y manuales técnicos que sabemos que tiene integrados en su base de conocimiento, tal y como se muestra en la Figura 12 y en la Figura 13.

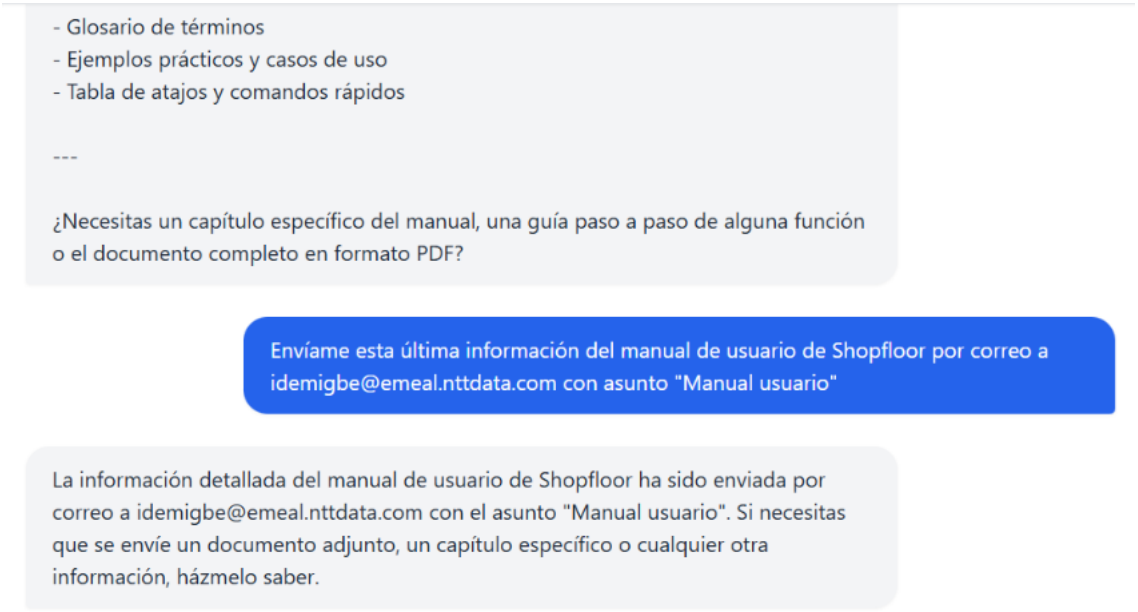

Figura 12. Petición de información de documentos

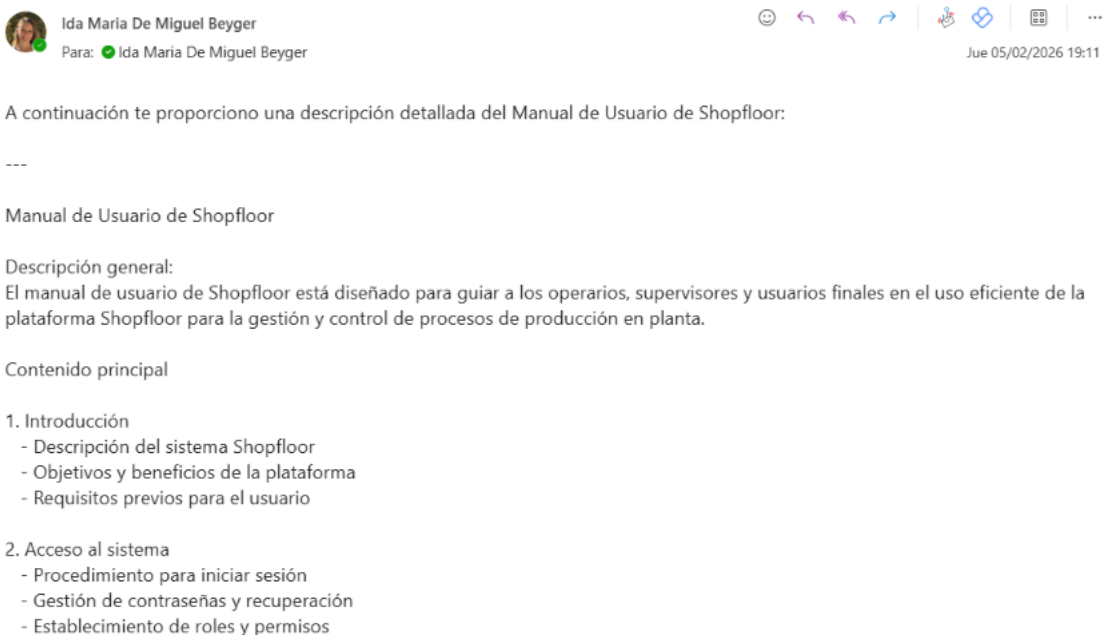

Figura 13. Respuesta del agente con información interna documentada

4.5.2 Gestión de identidad y seguridad

La autenticación en el lado del cliente se realiza mediante la librería de Microsoft Authentication Library (MSAL.js) integrando la aplicación con la seguridad corporativa de Azure Active Directory.

Al cargar, el *frontend* consulta al *endpoint /api/auth-config* para obtener el Client ID y la *Authority* del tenant, evitando insertar a la vista credenciales en el código cliente. La gestión de los tokens se realiza mediante el flujo de autenticación OAuth 2.0, donde el *frontend* se encarga de adquirir, almacenar y renovar los Access Tokens. También se garantiza que ninguna operación se ejecute sin una identificación validada gracias a la cabecera *Authorization*: *Bearer* en cada petición HTTP hacia el *backend*.

4.5.3 Comunicación asíncrona

Para tener una buena latencia de respuesta, el *frontend* utiliza *streaming* de eventos. Se utiliza Fetch API configurada para leer el cuerpo de la respuesta como un flujo de datos, además de utilizar el protocolo Server-Sent Events (SSE) para decodificar en tiempo real.

El texto se inyecta en el chat token a token, creando un efecto de "escritura real" que proporciona una vista al usuario, reduciendo la latencia percibida a milisegundos.

4.5.4 Implementación de voz en tiempo real

Para el modo Live, tal y como se explicó en el apartado 3.4.5, el *frontend* actúa como un cliente de voz inteligente, gestionando la entrada y salida de audio directamente con el navegador. El *frontend* solicita credenciales al *backend* y establece un túnel WebSocket seguro con la API de Azure OpenAI Realtime. El proceso para el audio es capturar el micrófono del usuario convirtiendo el flujo de audio a base64 y enviándolo al socket en tiempo real. La salida recibe el audio generado por GPT-4o y los pone en cola en orden secuencial. También existe una gestión de interrupciones. Se puede apreciar cómo aparece para el usuario en la Figura 14.

Figura 14. Botones realtime y conversación por voz

4.6 Observabilidad y comprobaciones

Para garantizar la fiabilidad y facilitar el mantenimiento del sistema en un entorno productivo, se ha implementado una estrategia de observabilidad basada en la verificación funcional mediante documentación interactiva y por otra parte, la monitorización centralizada de logs.

4.6.1 Verificación funcional con Swagger

Para garantizar la estandarización y facilitar el consumo de los servicios por parte del cliente web, se ha implementado una estrategia de autodocumentación bajo el estándar OpenAPI (Swagger). Gracias a FastAPI, el sistema genera dinámicamente un esquema JSON que describe la estructura completa de la API REST.

Por tanto, en la interfaz gráfica de Swagger se podrá observar el catálogo de *endpoints*, tal y como se muestran en la Figura 15 y en la Figura 16, diferenciando los que son de lectura (GET) y los que son de escritura (POST), los esquemas de datos capaces de visualizar el tipo de dato esperado, y un entorno de pruebas.

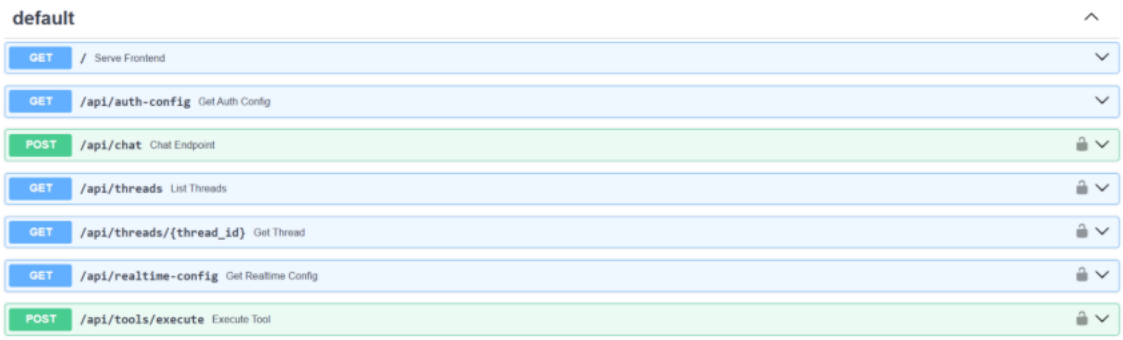

Figura 15. Disposición de endpoints

Figura 16. Continuación disposición de endpoints

Capítulo 5. Gestión financiera y evaluación de impacto

5.1 Cálculo de costes y estrategia financiera

El uso de tecnologías *cloud* y modelos de lenguaje de gran escala requiere tener en cuenta los gastos que estos conllevan. Siempre existirán costes fijos de infraestructura, pero también hay que tener en cuenta los costes variables.

Para evaluar la viabilidad del proyecto, no solo se han analizado los costes de la infraestructura de Azure, sino también la eficiencia del gasto por cada interacción del usuario. Este enfoque permite entender el valor real que aporta el agente en la planta industrial frente a lo que cuesta, asegurando que la solución final sea financieramente sostenible a largo plazo.

Se ha de tener en cuenta que la unidad de medida del coste no es simplemente cuánto cuesta tener un servidor encendido, sino el impacto económico que conlleva el despliegue de los recursos necesarios para el funcionamiento de la infraestructura y las consultas resultas por el agente de IA, midiendo esto último a través de los tokens consumidos[45].

Para ello se pueden evaluar distintos escenarios económicos: pago por uso, beneficio de uso híbrido, consumo, spot, instancia reservada y plan de ahorro. Para este proyecto, se busca una estrategia que garantice disponibilidad y rendimiento, pero con la máxima eficiencia financiera.

El objetivo no es minimizar el gasto absoluto en la nube, sino maximizar el valor generado por cada euro invertido, garantizando que el coste del agente sea proporcional a las ganancias en eficiencia operadas logradas por el agente de planta industrial[46].

Se va a utilizar la región *West Europe* para todos los recursos.

5.1.1 Selección del nivel de soporte técnico

Es importante elegir correctamente el tipo de soporte técnico antes del despliegue de recursos. El plan Básico está incluido para todos los clientes de Azure, y dependiendo de las necesidades de la empresa y del presupuesto, se puede ir incrementando de plan. El plan *Developer* sirve para entornos de prueba y no de producción, con un precio bastante asequible al mes, con acceso a servicio técnico en horario comercial. Justo después se sitúa el plan Estándar, que sirve para entornos de carga de trabajo de producción, por 100 USD al mes, atención 24/7 y con opción a resolución de problemas rápido si hay un problema crítico. El plan Profesional Directo es para empresas pequeñas con uso crítico, por un precio de 1000 USD al mes, atención 24/7 y que cuenta con revisiones de servicio y asesoría. Por último, el plan más completo, el plan de Soporte Unificado, es para servicios de nivel empresarial y soporte técnico para grandes empresas con dependencia crítica de Azure y otras tecnologías de Microsoft, que ofrece todo lo nombrado en los planes anteriores, además de otras muchas ventajas. Todas ellas tienen soporte para facturación y administración de suscripciones; recursos de autoayuda ininterrumpidos; capacidad de enviar tantas incidencias de soporte técnico como sean necesarias; guías gratuitas y mantenimiento [47].

Para el desarrollo del proyecto se podría usar el plan Básico si fuese un proyecto personal, ya que al ser un proyecto pequeño no requiere de atención prioritaria ni utiliza las otras muchas ventajas que ofrecen planes superiores.

Se puede observar en la Tabla 2, las diferencias entre los planes Soporte Unificado y Profesional Directo, que son las que más se adaptan al plan de negocio para una planta industrial de una gran empresa.

Teniendo en cuenta la magnitud de una empresa grande, si este proyecto llegase a producción quizás la mejor opción sería el plan Profesional Directo a largo plazo, la cual tiene una tarifa mensual fija, proporciona respuesta 24/7 para incidencias y acceso a personal resolutivo y por lo tanto permite trabajar en un entorno en el que sabes que una incidencia no interrumpirá el servicio mucho tiempo.

Característica	Soporte Unificado	Professional Direct
Alcance	Todo Microsoft	Solo Microsoft Azure
Modelo de coste	Basado en gasto anual	Tarifa fija mensual
Soporte proactivo	TAM dedicado, horas de consultoría y evaluaciones de arquitectura	Acceso a arquitectos de soluciones y soporte proactivo, No TAM dedicado
Disponibilidad	Soporte 24/7/365 para incidentes críticos con tiempo de respuestas muy rápidos	Soporte 24/7/365 para incidentes críticos
Adecuado para	Empresas grandes con gasto significativo en múltiples tecnologías de Microsoft	Empresas medianas-grandes cuya criticidad está en su infraestructura Azure

Tabla 2. Diferencias entre planes para grandes empresas

5.1.2 Análisis de costes por recurso y configuración de SKU

Teniendo en cuenta que la arquitectura requiere una alta disponibilidad, seguridad y rendimiento para la gestión de datos, se ha realizado una comparativa de costes dependiendo de la selección del nivel de servicio (SKU) y de la región escogida para cada recurso haciendo uso de la calculadora de precios de Azure[48], pudiendo ofrecer un informe que muestre la mejor alternativa en cuanto a calidad-precio para la utilización continua de esos recursos.

Recursos del proyecto:

-VNet: recurso que proporciona un espacio de direcciones IP privadas. No tiene SKU (niveles de precios) pero sus costes provienen de los servicios asociados a la misma (los recursos que se conectan a la VNet). Usamos la VNet para construir una red privada y restringir el acceso a los servicios PaaS Para mantener el mejor precio, se ha de asegurar que la mayor parte de la comunicación permanezca dentro de la VNet de Azure, ya que el tráfico de salida de Azure a Internet es el principal generador de costes. El coste subiría si creamos más conexiones o cambiamos de región[49].

-Ai Search: en la Tabla 3 se hará la comparación de las diferentes instancias, teniendo en cuenta que S significa Estándar y L significa Almacenamiento Optimizado.

El coste mensual se calcula multiplicando el precio del SKU por el número de particiones y réplicas. Las particiones son las unidades físicas de almacenamiento, y las réplicas son las veces que se instancia el motor de búsqueda. Esto se relaciona con el parámetro de máximo de índices, que es una colección de documentos con un esquema de datos específico.

En cuanto a la disponibilidad, para garantizar el SLA de lectura de Microsoft se necesitan al menos 2 réplicas y para lectura y escritura unas 3 réplicas[50].

El número de unidades de búsqueda se calcula multiplicando las réplicas por las particiones.

Para el proyecto actual, creo que bastaría con el Estándar S1, ya que tiene alta disponibilidad, gran número de índices para poder separar la información por categorías, hasta 160 GB por partición en almacenamiento y el precio mensual no es muy elevado. Hay que tener claro que el Basic sería imposible utilizarlo ya que no integra el servicio en la VNet a través de Private Endpoints, lo cual no conviene a la empresa por la pérdida de seguridad en sus documentos.

SKU	Almacenamiento máximo	Máx. índices	Unidades búsqueda	Características	Precio estimado(mes/SU)
Free	50 MB	3	N/D	Sin SLA	0 € / mes
Basic	15 GB	15	9	SLA y Private Link	62,66 € / mes
S1	160 GB	50	36	RAG industrial	208,45 € / mes
S2	512 GB	200	36	Grandes volúmenes de documentos	833,79 € / mes
S3	1 TB	200	36	Mayor rendimiento	1667,58 € / mes
L1	2 TB	10	36	Alta latencia	2381,64 € / mes
L2	4 TB	10	36	Capacidad de almacenamiento	4762,65 € / mes

Tabla 3. Comparativas SKU AiSearch

-Application Registration no tiene coste. Su función es registrar aplicaciones que permiten autenticación y autorización.

-Container App: es un servicio *serverless* que aloja los microservicios. El plan de consumo está bien para proyectos donde la carga puede ir cambiando, como es nuestro caso. Luego existen los costes variables que se basan en la cantidad de recursos de CPU y memoria que se consume. Si comparamos con los otros tipos de planes, vemos que este es el más adecuado ya que los otros generan carga y uso constante y no tienen en cuenta los picos de la planta de fabricación. Elegir plan de consumo implica que exista una proporción fija entre la cantidad de CPU y de memoria, siendo esta la mejor manera de optimizar el recurso teniendo en cuenta la demanda real.[51]

-Private Endpoint: como ya se ha explicado en el apartado 3.3.2, un Private Endpoint ofrece calidad y seguridad haciendo que los datos no puedan ser accesibles por la Internet Pública. Los PE tienen un coste fijo por hora constante y bajo, más un coste por el procesamiento de datos que manejan. Se paga por cada punto de conexión que haya. También se paga más por lo datos procesados salientes y entrantes[52].

-Storage Account: ofrece varios tipos de cuentas de almacenamiento. Cada tipo admite diferentes características y tiene su propio catálogo de precios. Se busca máximo equilibrio entre durabilidad, disponibilidad y coste. El SKU base sería el General Pupose V2 ya que es el estándar y soporta todos los tipos de almacenamientos.

En cuanto al nivel de redundancia, existen varios tipos dependiendo de si la región es primaria o secundaria. La región primaria es la región donde se despliega el recurso, y ofrece almacenamiento con redundancia local (LRS) donde se replican los datos en un único centro de datos físicos ubicado en la región primaria de elección; y almacenamiento con redundancia de zona (ZRS), donde los datos se copian en más zonas de la región primaria. La región secundaria se usa para la recuperación ante desastres, creando una réplica geográfica, donde se puede tener almacenamiento con redundancia geográfica (GRS), donde los datos se copian de forma síncrona dentro de una o varias zonas de disponibilidad de la región primaria mediante LRS; o se puede tener almacenamiento con redundancia de zona geográfica (GZRS) donde se copian los datos de la manera que lo hace GRS pero en más de tres zonas mediante ZRS[53].

Si introducimos lo básico en la calculadora de precios de Azure, se observa que para ZRS, una capacidad de 1000 GB nos costaría en promedio 20,40 € / mes con acceso frecuente, y para LRS con las mismas características sale a 16,66 € / mes. Por esa diferencia de precio, es obvio que la mejor opción sería utilizar el tipo de cuenta de uso general estándar v2[54] con ZRS ya que protege los datos contra un fallo de un centro de datos completo (caída de una zona) dentro de la

misma región, lo cual considero que sería lo mínimo, pero no llegaría al nivel de GRS, que es recuperación ante desastres para varias zonas, pero duplica mucho el coste y creo que no haría falta en este caso. De esta manera optimizamos el recurso teniendo en cuenta la durabilidad y el coste.

-Container Registry (ACR): almacena las imágenes de los contenedores que se ejecutarán en el Container App Environment. Para la elección del SKU hay que mirar el cual es el rendimiento necesario (cuántas imágenes van a *push* y *pull*) y la capacidad de almacenamiento. El nivel estándar ofrece hasta 100 GB de almacenamiento y un alto rendimiento de operaciones *pull*. Otras alternativas: Básico (10 GB es muy poco para un sistema de producción 24/7) y premium (500GB si es una empresa con varias localizaciones geográficas sería buena idea). En cuanto a seguridad estaría bien habilitar ACR *Tasks* ya que permite la automatización de escaneo de vulnerabilidades de las imágenes. En cuanto al apartado de ancho de banda, solo se paga si se abandona la región del ACR. Una buena práctica es poner todo en la misma región, además que así podemos minimizar la latencia y evitar fallos de conectividad[55].

- Static Web App: permite desarrollar aplicaciones web modernas con un *frontend* estático y un *backend* dinámico con tecnología API sin servidor. Los planes que ofrece son el gratuito y el estándar. Ambos ofrecen 100 GB por suscripción, certificados SSL, y conexiones a bases de datos, pero el estándar ofrece más dominios personalizados por aplicación (5 frente a los 2 del plan gratuito) y más almacenamiento por aplicación, por un precio de 7,649 € / mes más gastos de características contratadas adicionales[56].

5.2 Evaluación de impacto

Si se quiere medir el impacto de la utilización de recursos *cloud* en la empresa es necesario observar la relación entre lo que la empresa invierte en la infraestructura y el valor que la utilización de estos recursos aporta a la empresa. El sistema ha de ser viable a largo plazo

En el apartado 5.1 se ha hecho la comparativa de los diferentes modelos de pago y costes de cada recurso. Para una planta de fabricación, donde la actividad depende de los turnos, hay que tener en cuenta que van a haber picos de demanda. Es por eso por lo que se ha elegido utilizar los pagos por uso, ya que permiten procesar grandes volúmenes de información solo cuando es realmente necesario, optimizando así la inversión. Además, el uso de recursos *serverless* permite que la infraestructura responda de forma automática sin intervención manual, lo que garantiza que el coste siempre sea proporcional al valor aportado.

Realizando la automatización de despliegue en Terraform se gana tiempo y la opción de escalar la aplicación en un futuro teniendo una gestión del ciclo de vida, lo que hace que la aplicación sea lo más eficiente posible.

En cuanto al precio de los tokens, hay que tener en cuenta que los tokens de salida son más caros que los de entrada. Los tokens de entrada son los prompts que le envías al agente de IA, y los tokens de salida es la respuesta del agente. 1000 tokens son aproximadamente 750 palabras en inglés y unas 700 en español. Teniendo en cuenta que en este proyecto se usa el modelo GPT-4 o y que la conversión a día de hoy de dólar a euro es 1 € equivale a 1,18 dólares, para las interacciones estándar basadas en texto (chat), se establece un coste de 2,30 € por millón de tokens de entrada y 9,20 € por millón de tokens de salida. En cuanto al Realtime, hay que añadirle además el coste de procesamiento de audio en tiempo real, que son aproximadamente 5,52 € por millón de tokens de entrada de audio y 18,40 € por millón de tokens de salida de audio[57].

5.3 Análisis de riesgos financieros

A lo largo del Trabajo de Fin de Grado se ha visto que la utilización de Microsoft Azure tiene grandes ventajas, tales como su escalabilidad, flexibilidad y seguridad, pero si no se administran los recursos de manera correcta puede haber múltiples riesgos. En este apartado se desarrollarán los riesgos financieros.

Si no se realiza una verificación de los costes, podemos perder el control financiero. Esto se hace mediante mecanismos FinOps, que promueven la toma de decisiones controladas por datos de forma oportuna y fomenta la responsabilidad financiera[58].

Siguiendo las directrices de Azure Cloud Adoption Framework (CAF), se ha identificado que el principal riesgo económico en la infraestructura *cloud* es la falta de alineación entre el consumo tecnológico y los objetivos de negocio. Esto suele ocurrir cuando se sobredimensionan los recursos, se tiene recursos sin utilizar o existe una gestión ineficiente de las cuotas, lo que puede derivar en costes imprevistos. Para mitigar este impacto, la arquitectura propuesta adopta un modelo de gobernanza que prioriza la visibilidad y el control preventivo, asegurando que cada recurso consumido esté plenamente justificado con su utilidad.

Otro de los riesgos principales es el escalado imprevisto. En recursos *serverless* como por ejemplo Container Apps, existe la posibilidad de que se produzcan picos inesperados en el procesamiento de datos o en la generación de tokens. Estos picos pueden ser causados por una actividad inusual en la planta, como consultas masivas o errores en el ciclo de razonamiento del agente. Para mitigar este riesgo, existen mecanismo de control de costes que incluyen una configuración de Azure Budgets con notificaciones de superación de umbrales y establecimiento de límites estrictos en las cuotas, permitiendo detectar comportamientos extraños que afectan al presupuesto.

La integridad y el cumplimiento normativo también representan un riesgo económico. El tratamiento de la documentación técnica industrial en la nube conlleva riesgos de seguridad y privacidad de datos que podrían derivar en sanciones o pérdida de propiedad intelectual. Hemos visto que mediante la utilización de Private Endpoints conseguimos aislar el tráfico de datos de la Internet pública. Esta inversión en seguridad añade un coste mínimo y además integra la aplicación en los estándares de ciberseguridad industrial, garantizando la sostenibilidad de la solución a lo largo del tiempo.

Finalmente, se ha contemplado el riesgo de que la arquitectura se queda obsoleta, pero mediante la utilización de herramientas como Terraform se proporciona transparencia sobre los archivos desplegados y es fácil la replicabilidad o destrucción de los recursos. Esto garantiza que la organización tenga la agilidad necesaria para ajustar su inversión tecnológica a medida que evolucione, asegurando que la gestión financiera del asistente de IA no sea estática, sino un proceso dinámico que mejora continuamente[59].

5.4 Herramientas de supervisión continua

Existen varias herramientas de Microsoft Azure que nos permiten reforzar la seguridad, eficiencia y sostenibilidad a lo largo del tiempo[59].

- Azure Advisor: es un asistente en la nube que ayuda a seguir los procedimientos adecuados para optimizar las implementaciones de Azure. Analiza la telemetría de uso y configuración de recursos y recomienda soluciones para mejor la rentabilidad, rendimiento y seguridad de los recursos, además de optimizar los gastos[60].
- Microsoft Defender for Cloud: es una plataforma de protección de aplicaciones nativas en la nube (CNAPP) que combina varias herramientas de seguridad. Ayuda a los equipos de DevOps a encontrar configuraciones incorrectas, aplicar directivas y corregir los riesgos pronto[61].
- Azure Service Health: combina de forma eficiente el Estado de Azure, que informa de las interrupciones de servicio; Service Health, que ofrece una vista personalizada del estado de los servicios y regiones usadas; y Resource Health, que proporciona información sobre el estado de los recursos individuales en la nube. Mediante esta herramienta se pueden configurar alertas para notificar cambios de disponibilidad en los recursos[62].
- Microsoft Purview: es un conjunto de soluciones que ayuda a una organización a gobernar, proteger y administrar datos en la era de la inteligencia artificial, proporcionan cobertura integrada y ayudan a abordar la fragmentación de datos entre organizaciones entre otras cosas[63].

Capítulo 6. Conclusiones

6.1 Cumplimiento de objetivos y resultados obtenidos

Se puede observar el cumplimiento satisfactorio de los objetivos planeados al inicio del proyecto: crear un agente de inteligencia artificial que sea capaz de usar herramientas para ofrecer distintos servicios a personas en un entorno industrial corporativo mediante un sistema privado, escalable y seguro. Se ha realizado la integración de la infraestructura en la nube con la inteligencia artificial generativa.

En cuanto a la infraestructura, se ha logrado la automatización del despliegue mediante código. Los archivos de Terraform desarrollados permiten desplegar el entorno de Azure de forma rápida y consistente.

El agente Inteligente desarrollado mediante Python, FastApi y un *frontend* en JavaScript responde con éxito a las consultas técnicas de la planta gracias a AI Search, y además responde búsquedas externas mediante la herramienta de Bing.

Las pruebas de conectividad confirman que los Private Endpoints bloquean correctamente el acceso desde Internet.

El presupuesto es correcto y no llega al límite que me han ofrecido mensualmente en la empresa para realizar mis pruebas en la plataforma de Azure, para un Tenant específico que me han administrado.

6.2 Contribuciones, limitaciones y recomendaciones prácticas

El desarrollo del agente de IA ha estado basado en referencias ofrecidas por mi empresa de prácticas NTT DATA SPAIN, lo que me ha permitido aprender desde cero a utilizar nuevas herramientas de programación tales como Python, Terraform, Azure y JavaScript, así como la utilización de herramientas de inteligencia artificial proporcionadas por Azure y OpenAI. Gracias al seguimiento y ayuda de mi tutor de empresa, he sido capaz de implementar una arquitectura funcional, que además ofrece la calidad, seguridad y escalabilidad que exige una consultora tecnológica líder. El agente responde de forma rápida, precisa, busca en los manuales y si no encuentra la respuesta, busca en Bing.

He contribuido en la empresa creando con Terraform la infraestructura en la nube desplegada de forma automatizada, lo que les permitirá en un futuro crear recursos de Azure de forma fácil y rápida, que antes realizaban de forma manual en el portal de Azure.

A pesar de los resultados positivos, el proyecto contiene ciertas limitaciones. No se ha podido conseguir el funcionamiento de herramientas MCP para la conexión de servicios empresariales externos, como por ejemplo Jira, que era uno de los objetivos marcados por mi empresa. Esto no se ha podido realizar por falta de accesibilidad a recursos empresariales y aunque el código está realizado, no está validado.

6.3 Líneas de trabajo futuro

Para futuras implementaciones, se podría añadir al agente la carga de archivos o escáner para obtener búsquedas a partir de imágenes. Además, aunque se haya nombrado la monitorización, no se ha llegado a implementar en mi proyecto el uso de este.

Es también importante realizar formaciones para los usuarios de la aplicación del agente para que sepan realizar *prompts* eficientes para garantizar que las consultas sean precisas.

Quizá sería posible también tener modelos de inteligencia artificiales locales, capaces de instalarse en servidores físicos dentro de la industria sin necesidad de la utilización de internet, aunque esto solo serviría teniendo toda la información que se necesita en la documentación interna ya que limitaría las búsquedas de información en internet y las conexiones con servicios externos.

Capítulo 7. Referencias

[1] "¿Cuáles son las principales desventajas del Big Data? | UFV." Accessed: Dec. 17, 2025. [Online]. Available: https://www.ufv.es/cuales-son-las-principales-desventajas-del-big-data-preguntas-grados/

[2] A. Abeliuk and C. Gutiérrez, "Inteligencia Artificial El primer programa de IA," *Revista Bits de Ciencia*, no. 21, pp. 15–17, 2021.

[3] J. A. Coloma Garófalo, C. A. Sanaguano Guevara, Á. Geovanny Rochina Chisag, and J. A. Vargas Salazar, "Inteligencia artificial, sistemas inteligentes, agentes inteligentes," 2020. doi: 10.26820/recimundo/4.(2).mayo.2020.16-30.

[4] Á. Gutiérrez, "Almacenamiento en la nube," 2018.

[5] "¿Qué es la infraestructura como código (IaC)? - Azure DevOps | Microsoft Learn." Accessed: Jan. 11, 2026. [Online]. Available: https://learn.microsoft.com/es-es/devops/deliver/what-is-infrastructure-as-code

[6] "¿Qué es la entrega continua? - Azure DevOps | Microsoft Learn." Accessed: Jan. 11, 2026. [Online]. Available: https://learn.microsoft.com/es-es/devops/deliver/what-is-continuous-delivery

[7] "Comparación de Terraform y Bicep | Microsoft Learn." Accessed: Jan. 11, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/developer/terraform/comparing-terraform-and-bicep?tabs=comparing-bicep-terraform-integration-features

[8] "Introducción a Terraform en Azure: ¿Qué es Terraform? | Microsoft Learn." Accessed: Jan. 11, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/developer/terraform/overview

[9] "Create a Terraform plan | Terraform | HashiCorp Developer." Accessed: Jan. 11, 2026. [Online]. Available: https://developer.hashicorp.com/terraform/tutorials/cli/plan

[10] "Estado | Terraform | Desarrollador de HashiCorp." Accessed: Jan. 11, 2026. [Online]. Available: https://developer.hashicorp.com/terraform/language/state

[11] "Almacenar el estado de Terraform en Azure Storage | Microsoft Learn." Accessed: Jan. 11, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/developer/terraform/store-state-in-azure-storage?tabs=azure-cli

[12] "Descripción general de los módulos | Terraform | Desarrollador de HashiCorp." Accessed: Jan. 11, 2026. [Online]. Available: https://developer.hashicorp.com/terraform/language/modules

[13] "¿Qué es la confianza cero? | Microsoft Learn." Accessed: Jan. 11, 2026. [Online]. Available: https://learn.microsoft.com/es-es/security/zero-trust/zero-trust-overview

[14] "Conceptos y procedimientos recomendados de Azure Virtual Network | Microsoft Learn." Accessed: Jan. 11, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/virtual-network/concepts-and-best-practices

[15] "¿Qué es la segmentación de redes? | Cloudflare." Accessed: Jan. 11, 2026. [Online]. Available: https://www.cloudflare.com/es-es/learning/access-management/what-is-network-segmentation/

[16] "¿Qué es un punto de conexión privado? - Azure Private Link | Microsoft Learn." Accessed: Jan. 12, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/private-link/private-endpoint-overview

[17] "¿Qué es Azure Private Link? | Microsoft Learn." Accessed: Jan. 12, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/private-link/private-link-overview

[18] "Introducción a los grupos de seguridad de red de Azure | Microsoft Learn." Accessed: Jan. 12, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/virtual-network/network-security-groups-overview

[19] "Administración automática de rutas definida por el usuario (UDR) con Azure Virtual Network Manager | Microsoft Learn." Accessed: Jan. 12, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/virtual-network-manager/concept-user-defined-route

[20] "Identidades administradas de recursos de Azure - Managed identities for Azure resources | Microsoft Learn." Accessed: Jan. 12, 2026. [Online]. Available: https://learn.microsoft.com/es-es/entra/identity/managed-identities-azure-resources/overview

[21] "¿Qué es el control de acceso basado en rol de Azure (RBAC)? | Microsoft Learn." Accessed: Jan. 12, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/role-based-access-control/overview

[22] "Procedimientos recomendados para RBAC de Azure | Microsoft Learn." Accessed: Jan. 12, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/role-based-access-control/best-practices

[23] Luis Martín de Pablo, "TRABAJO FINAL DE M ´ ASTER´Area ASTER´ ASTER´Area: Inteligencia Artificial," Barcelona, Dec. 2024. Accessed: Apr. 02, 2026. [Online]. Available: https://hdl.handle.net/10609/151946

[24] "Tokens de modelo de lenguaje grande - Azure Cosmos DB | Microsoft Learn." Accessed: Jan. 17, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/cosmos-db/gen-ai/tokens

[25] "Introducción - Training | Microsoft Learn." Accessed: Jan. 17, 2026. [Online]. Available: https://learn.microsoft.com/es-es/training/modules/ai-agent-fundamentals/1-introduction

[26] "IA agentiva vs. IA generativa | IBM." Accessed: Jan. 17, 2026. [Online]. Available: https://www.ibm.com/es-es/think/topics/agentic-ai-vs-generative-ai

[27] "¿Qué es el servicio Foundry Agent? - Microsoft Foundry | Microsoft Learn." Accessed: Jan. 18, 2026. [Online]. Available: https://learn.microsoft.com/en-us/azure/ai-foundry/agents/overview?view=foundry-classic

[28] "Introducción a Microsoft Agent Framework | Microsoft Learn." Accessed: Jan. 18, 2026. [Online]. Available: https://learn.microsoft.com/es-es/agent-framework/overview/agent-framework-overview

[29] "What is the Model Context Protocol (MCP)? - Model Context Protocol." Accessed: Jan. 18, 2026. [Online]. Available: https://modelcontextprotocol.io/docs/getting-started/intro

[30] "Construyendo el futuro: por qué la pila de React y Python domina el desarrollo web impulsado por IA." Accessed: Jan. 18, 2026. [Online]. Available: https://www.q2bstudio.com/nuestro-blog/285849/la-combinacion-de-react-y-python-en-el-desarrollo-web-con-inteligencia-artificial-lidera-gracias-a-su-eficiencia-y-versatilidad-descubre-por-que-esta-pila-tecnologica-es-la-preferida-en-la-actualidad?scriptscookies=1

[31] "Azure OpenAI or Azure Ai foundry - Microsoft Q&A." Accessed: Jan. 25, 2026. [Online]. Available: https://learn.microsoft.com/en-us/answers/questions/5572779/azure-openai-or-azure-ai-foundry

[32] "Orquestar microservicios y aplicaciones de varios contenedores para una alta escalabilidad y disponibilidad - .NET | Microsoft Learn." Accessed: Jan. 25, 2026. [Online]. Available: https://learn.microsoft.com/es-es/dotnet/architecture/microservices/architect-microservice-container-applications/scalable-available-multi-container-microservice-applications

[33] "¿Qué es la orquestación de agentes de IA? | IBM." Accessed: Jan. 25, 2026. [Online]. Available: https://www.ibm.com/es-es/think/topics/ai-agent-orchestration

[34] "Construcción de tu agente en Azure | Microsoft Learn." Accessed: Jan. 25, 2026. [Online]. Available: https://learn.microsoft.com/es-es/microsoft-for-startups/build/build-agent

[35] "Diseño de arquitectura de IA - Azure Architecture Center | Microsoft Learn." Accessed: Jan. 25, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/architecture/ai-ml/

[36] "¿Qué es la amenaza interna? Cómo abordar los riesgos internos | Seguridad de Microsoft." Accessed: Jan. 25, 2026. [Online]. Available: https://www.microsoft.com/es-es/security/business/security-101/what-is-insider-threat

[37] "Estrategia de control de acceso y administración de identidad y acceso - Dynamics 365 Mixed Reality | Microsoft Learn." Accessed: Jan. 25, 2026. [Online]. Available: https://learn.microsoft.com/es-es/dynamics365/mixed-reality/guides/gxp-guidance/strategy-for-access-control-and-iam

[38] "how-microsoft-defends-against-indirect-prompt-injection-attacks." Accessed: Jan. 25, 2026. [Online]. Available: https://www.microsoft.com/en-us/msrc/blog/2025/07/how-microsoft-defends-against-indirect-prompt-injection-attacks

[39] "Principios y enfoque de inteligencia artificial responsable | IA de Microsoft." Accessed: Jan. 25, 2026. [Online]. Available: https://www.microsoft.com/es-es/ai/principles-and-approach

[40] "Cómo usar GPT Realtime API para voz y audio con Azure OpenAI en Microsoft Foundry Models - Azure OpenAI | Microsoft Learn." Accessed: Jan. 25, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/ai-foundry/openai/realtime-audio-quickstart?view=foundry-classic&tabs=keyless%2Cwindows&pivots=programming-language-javascript

[41] "Prepare to Choose a Data Store in Azure - Azure Architecture Center | Microsoft Learn." Accessed: Jan. 19, 2026. [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/guide/technology-choices/data-stores-getting-started

[42] "Inicio rápido: crear un almacén de claves de Azure y una clave mediante Terraform | Microsoft Learn." Accessed: Jan. 25, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/key-vault/keys/quick-create-terraform?tabs=azure-cli

[43] "Almacenar el estado de Terraform en Azure Storage | Microsoft Learn." Accessed: Jan. 25, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/developer/terraform/store-state-in-azure-storage?tabs=azure-cli

[44] "Tipos de agentes de IA | IBM." Accessed: Jan. 25, 2026. [Online]. Available: https://www.ibm.com/es-es/think/topics/ai-agent-types

[45] "Economía unitaria - Cloud Computing | Microsoft Learn." Accessed: Jan. 26, 2026. [Online]. Available: https://learn.microsoft.com/es-es/cloud-computing/finops/framework/quantify/unit-economics

[46] John Jairo Velez Gavilan, "Evaluación de azure frente a otras plataformas de computación en la nube," 2024.

[47] "Comparación de planes de soporte técnico de Azure | Microsoft Azure." Accessed: Jan. 26, 2026. [Online]. Available: https://azure.microsoft.com/es-es/support/plans/

[48] "Calculadora de precios | Microsoft Azure." Accessed: Jan. 26, 2026. [Online]. Available: https://azure.microsoft.com/es-es/pricing/calculator/?ef_id=_k_Cj0KCQiAvtz-LBhCPARIsALwhxdpy3LamlgMzVoN7D7FpT3se6rYqroSMJdJvNT-shhXOWIhzBNVaj8wQaAhdTEALw_wcB_k_&OCID=AIDcmm68ejnsa0_SEM__k_Cj0KCQiAvtzLBhCPARIsALwhxdpy3LamlgMzVoN7D7FpT3se6rYqroSMJdJvNT-shhXOWIhzBNVaj8wQaAhdTEALw_wcB_k_&gad_source=1&gad_campaignid=1635077706&gbraid=0AAAADcJh_ucBxzzJ9hZj_oCADcHRDw8q&gclid=

Cj0KCQiAvtzLBhCPARIsALwhxdpy3LamlgMzVoN7D7FpT3se6rYqroSMJdJvNT-shhXOWIhzBNVaj8wQaAhdTEALw_wcB

[49] "Precios de Virtual Network | Microsoft Azure." Accessed: Jan. 27, 2026. [Online]. Available: https://azure.microsoft.com/es-es/pricing/details/virtual-network/

[50] "Precios: Búsqueda de Azure AI | Microsoft Azure." Accessed: Jan. 27, 2026. [Online]. Available: https://azure.microsoft.com/es-es/pricing/details/search/#pricing

[51] "Precios de Azure Container Apps | Microsoft Azure." Accessed: Jan. 27, 2026. [Online]. Available: https://azure.microsoft.com/es-es/pricing/details/container-apps/

[52] "Pricing - Azure Private Link | Microsoft Azure." Accessed: Jan. 27, 2026. [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/private-link/

[53] "Redundancia de datos - Azure Storage | Microsoft Learn." Accessed: Jan. 27, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/storage/common/storage-redundancy

[54] "Introducción a las cuentas de almacenamiento - Azure Storage | Microsoft Learn." Accessed: Jan. 27, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/storage/common/storage-account-overview

[55] "Precios de registro de contenedor | Microsoft Azure." Accessed: Jan. 27, 2026. [Online]. Available: https://azure.microsoft.com/es-es/pricing/details/container-registry/

[56] "Precios: Static Web Apps | Microsoft Azure." Accessed: Jan. 27, 2026. [Online]. Available: https://azure.microsoft.com/es-es/pricing/details/app-service/static/

[57] "Precios | OpenAI." Accessed: Feb. 23, 2026. [Online]. Available: https://openai.com/es-ES/api/pricing/

[58] "¿Qué es FinOps? - Cloud Computing | Microsoft Learn." Accessed: Jan. 28, 2026. [Online]. Available: https://learn.microsoft.com/es-es/cloud-computing/finops/overview

[59] "Evaluación de los riesgos de la nube - Cloud Adoption Framework | Microsoft Learn." Accessed: Jan. 28, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/cloud-adoption-framework/govern/assess-cloud-risks

[60] "Introducción a Azure Advisor - Azure Advisor | Microsoft Learn." Accessed: Jan. 28, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/advisor/advisor-overview

[61] "Introducción a Microsoft Defender for Cloud - Microsoft Defender for Cloud | Microsoft Learn." Accessed: Jan. 28, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/defender-for-cloud/defender-for-cloud-introduction

[62] "¿Qué es Azure Service Health? - Azure Service Health | Microsoft Learn." Accessed: Jan. 28, 2026. [Online]. Available: https://learn.microsoft.com/es-es/azure/service-health/overview

[63] "Aprenda acerca de Microsoft Purview | Microsoft Learn." Accessed: Jan. 28, 2026. [Online]. Available: https://learn.microsoft.com/es-es/purview/purview

[64] "Objetivos de Desarrollo Sostenible | Programa De Las Naciones Unidas Para El Desarrollo." Accessed: Feb. 16, 2026. [Online]. Available: https://www.undp.org/es/sustainable-development-goals

Anexo A. Objetivos de desarrollo sostenible

Los ODS son objetivos creados por las Naciones Unidas para poner fin a la pobreza, proteger el planeta y garantizar que para el 2030 todas las personas disfruten de paz y prosperidad. Aunque existen 17 ODS[64], el proyecto se basa en los que tienen relación con la educación y el trabajo, teniendo en cuenta que el proyecto utiliza IA Generativa, búsqueda cognitiva y automatización de tareas.

Se cumplen los siguientes ODS:

-ODS 4: Educación de calidad (competencias técnicas y profesionales). El agente actúa como un tutor o asistente experto. Democratiza el conocimiento técnico y facilita la formación continua dentro de la empresa.

-ODS 8: Trabajo decente y crecimiento económico (diversificación, tecnología e innovación). El agente automatiza tareas repetitivas y de bajo valor. Permite crear trabajos más creativos, estratégicos, aumentando la productividad económica sin aumentar la carga laboral.

-ODS 9: Industria, innovación e infraestructura (fomentar la innovación). Se está implementando tecnología RAG y Realtime API. Se realiza una modernización de la gestión del conocimiento industrial, haciendo que la infraestructura de información de la empresa sea más accesible y eficiente.

-ODS 12: Producción y consumo responsables (uso eficiente de recursos naturales). La utilización de Terraform para automatizar tareas permite reducir drásticamente el uso de servidores innecesarios y reducir el desperdicio de energía.