Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Aplicación del Método ADSM para Soporte Emocional mediante Generative AI

Autor: Ignacio Martín Sánchez
Tutor(a): Juan Antonio Fernández del Pozo Salamanca

Madrid, Mayo 2025

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática

Título: Aplicación del Método ADSM para Soporte Emocional mediante Generative AI

Mayo 2025

Autor: Ignacio Martín Sánchez
Tutor: Juan Antonio Fernández del Pozo Salamanca
Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Resumen

Este Trabajo de Fin de Grado presenta el diseño, desarrollo e implementación de una plataforma digital de acompañamiento emocional basada en inteligencia artificial generativa, orientada a guiar al usuario en su proceso de autoconocimiento y toma de decisiones mediante la aplicación estructurada del método ADSM. Este método, desarrollado por Gerardo Tudurí, se fundamenta en una serie de protocolos aplicables mediante diálogo estructurado, con el objetivo de facilitar el análisis emocional, la planificación vital y la activación de recursos personales.

La solución propuesta permite a cualquier usuario interactuar mediante lenguaje natural con asistentes conversacionales especializados, cada uno diseñado para aplicar un protocolo concreto del modelo ADSM (IPP, AP, PCN, SVM, MDS-YO, VP). El sistema analiza el historial de interacción del usuario y lo deriva automáticamente al asistente más adecuado según el protocolo de derivación definido. Además, se ha desarrollado un evaluador automático de respuestas que permite verificar que el comportamiento del asistente principal cumple con el rol y las reglas establecidas.

Desde el punto de vista arquitectónico, la solución se apoya en una infraestructura Serverless en la nube de Microsoft Azure, diseñada con un enfoque modular, seguro y escalable. El frontend ha sido construido con Power Pages, integrando autenticación de usuarios mediante Azure AD B2C, lo que permite una gestión segura y personalizada de sesiones. El backend consta de varias funciones implementadas con Azure Functions, responsables de procesar las interacciones, cifrar el historial del usuario, clasificar su personalidad y almacenar los datos en distintos servicios como Azure Blob Storage y Azure Table Storage. Para garantizar la trazabilidad y la integridad de las clasificaciones, se ha desplegado una red Blockchain privada sobre Hyperledger Besu, donde se registran las medias semanales de los perfiles OCEAN generados.

Uno de los componentes clave del sistema es un modelo de regresión entrenado con técnicas de aprendizaje automático sobre un conjunto de conversaciones etiquetadas. Este modelo, basado en regresión lineal multivariada, estima los cinco rasgos de personalidad definidos por el modelo OCEAN (Apertura, Responsabilidad, Extraversión, Amabilidad y Neuroticismo) a partir del historial conversacional del usuario. El entrenamiento se ha realizado en Azure Machine Learning, y su inferencia se encuentra integrada en el pipeline del sistema mediante Azure Functions.

Durante el desarrollo del proyecto se ha seguido la metodología ágil SCRUM, documentando cada uno de los hitos, revisiones y tareas completadas. Se han realizado pruebas unitarias para cada componente, pruebas de integración con usuarios reales en un entorno controlado y una evaluación del impacto del sistema en relación con los Objetivos de Desarrollo Sostenible (ODS) de la Agenda 2030, especialmente en lo relativo a salud mental, bienestar emocional y acceso inclusivo a tecnologías de acompañamiento digital.

Como resultado, se ha obtenido una plataforma robusta, segura y funcional, que permite al usuario recorrer un camino asistido de mejora personal mediante diálogo natural, manteniendo la privacidad de su historial y registrando las conclusiones más relevantes en una red inmutable y auditada. Este proyecto demuestra la viabilidad de combinar inteligencia artificial generativa, diseño ético, Blockchain y métodos psicológicos estructurados para ofrecer una herramienta de autodescubrimiento y desarrollo emocional a escala.

Abstract

This Final Degree Project presents the design, development and implementation of a digital platform for emotional support, based on generative artificial intelligence and guided by the structured application of the ADSM method (Digital Algorithmization of Mental Systems). Developed by Gerardo Tudurí, this method is composed of protocols applied through structured dialogue, aimed at enhancing emotional awareness, personal decision-making and life planning.

The system allows any user to interact via natural language with specialized conversational assistants, each one focused on a specific ADSM protocol (IPP, AP, PCN, SVM, MDS-YO, VP). The assistant evaluates the user's input and redirects them to the most appropriate tool based on a derivation protocol. An automated evaluator was also developed to assess the assistant's responses and ensure they comply with the intended behavior, tone and logic.

From a technical perspective, the platform follows a modular and secure Serverless architecture using Microsoft Azure. The frontend is implemented with Power Pages and integrates Azure AD B2C for secure identity management. The backend consists of multiple Azure Functions responsible for message processing, encryption of user histories, personality classification and secure storage in Azure Blob Storage and Table Storage. To guarantee the immutability and traceability of user profiles, a private Blockchain was deployed using Hyperledger Besu, where weekly averages of OCEAN personality traits are recorded.

One of the key components is a personality classifier trained using machine learning techniques on a dataset of labeled conversations. The model, based on multi-output linear regression, predicts the five OCEAN traits (Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism) from the user's conversational history. The training process was executed in Azure Machine Learning and the inference stage is integrated into the production workflow through serverless functions.

The project followed an agile SCRUM methodology, with clearly defined milestones and testing cycles. Unit tests and real-user validation in a controlled environment were conducted. Additionally, the system's contribution to the United Nations Sustainable Development Goals (SDGs), particularly mental health and emotional well-being, has been analyzed and documented.

As a result, a functional and scalable platform has been developed, enabling users to receive AI-guided emotional assistance and personal development through

natural dialogue. The platform ensures user data privacy, provides traceable insights, and showcases the feasibility of combining generative AI, ethical design, and Blockchain technologies in the service of human development.

Tabla de contenidos

Siglas

ADSM Análisis Digital de Sistemas Mentales. i, iii, v, vii, 5, 8, 12–14, 17–20, 24, 26, 29, 30, 40, 41, 43–45, 55–57, 63, 64, 67, 71, 72, 97, 99, 136, 149, 150, 152, 154, 156, 157

AP Algoritmización Personal. i, iii, viii, 19, 29, 30, 41, 43, 44, 63, 67, 97, 136, 152, 153

Azure AD B2C Azure Active Directory Business to Consumer. i, iii, v, vii, 14, 25, 26, 28, 31–35, 37, 39, 56–58, 64, 66, 67, 105, 146

Azure Blob Storage Servicio de almacenamiento de objetos en la nube. i, iii, vi, 9, 14, 27, 32, 35, 36, 38, 45, 48, 56–58, 60, 64, 68, 92, 93, 105, 146

Azure Functions Funciones serverless de Microsoft Azure. i, iii, vi, vii, 10, 28, 30, 31, 35, 37–40, 48, 52, 53, 55, 56, 58–60, 69–71, 105

Azure Key Vault Gestión segura de claves y secretos en Azure. vi, 9, 14, 27, 38, 45–48, 53, 55, 64, 69, 91–93

Blockchain Tecnología de registro distribuido e inmutable. i–vi, 5, 24, 25, 27, 50, 56–58, 61, 68, 91, 105

GPT Generative Pre-trained Transformer. v, vi, 5, 9, 11–13, 17, 20–26, 29–32, 37, 38, 40–44, 56–58, 60, 63, 66, 67, 69, 71, 72

Hyperledger Besu Cliente Ethereum de código abierto para redes blockchain privadas. i, iii, vi, 10, 50, 51, 56, 57, 61, 64, 68, 90, 105

IPP Índice de Plenitud Personal. i, iii, vii, viii, 29, 30, 41, 44, 45, 63, 67, 97–99, 136, 137, 151, 152

LLM Large Language Model. v, 6, 19–21

MDS-YO Matriz de Desambiguación de Situaciones del Yo. i, iii, viii, 19, 29, 44, 45, 63, 67, 97, 98, 154, 155

OCEAN Modelo de personalidad basado en cinco rasgos: Openness (Apertura), Conscientiousness (Escrupulosidad), Extraversion (Extraversión), Agreeableness (Amabilidad) y Neuroticism (Neuroticismo). i, iii, v–vii, 5, 8, 12–14, 17, 18, 23–27, 31, 35, 38, 46–50, 52, 56–58, 61, 63–65, 67–69, 88, 89, 101, 102, 104

PCN Pensamiento Computacional Natural. i, iii, viii, 29, 30, 44, 63, 67, 97, 136, 153, 154

PLN Procesamiento de Lenguaje Natural. 13, 18, 63

SCRUM Metodología ágil de desarrollo de software. ii, iii, 6–8, 14, 71

Serverless Modelo de computación sin gestión de servidores. i, iii

SVM Simulación Virtual Mental. i, iii, viii, 44, 63, 136, 156, 157

VP Vibración Personal. i, iii, viii, 29, 30, 44, 45, 63, 67, 97, 99, 136, 155, 156

Capítulo 1

Introducción

En la sociedad actual, el bienestar emocional y la salud mental han cobrado una relevancia sin precedentes. A pesar del creciente reconocimiento de la importancia de la salud mental, muchas personas aún enfrentan barreras significativas para acceder a apoyo psicológico adecuado. Factores como la escasez de profesionales de la salud mental, el alto costo de los tratamientos, la estigmatización social y la falta de recursos en ciertas regiones dificultan que muchas personas reciban la ayuda que necesitan. En este contexto, la tecnología, y en particular la inteligencia artificial generativa, se presenta como una herramienta innovadora con el potencial de mitigar estas barreras y ofrecer apoyo accesible a una mayor cantidad de personas [1].

El problema principal que buscamos abordar con este proyecto es la falta de acceso oportuno y eficiente a orientación y apoyo emocional personalizado. La demora en recibir ayuda psicológica puede agravar problemas emocionales y afectar negativamente la calidad de vida de los individuos. Además, el enfoque tradicional de la atención psicológica no siempre se adapta a las necesidades y preferencias de cada persona, lo que puede llevar a una menor adherencia a los tratamientos y a la falta de seguimiento adecuado.

Para abordar este problema, este trabajo presenta el desarrollo de una aplicación basada en inteligencia artificial que utiliza asistentes conversacionales impulsados por modelos generativos como Generative Pre-trained Transformer (GPT) [2]. La aplicación se fundamenta en el modelo Análisis Digital de Sistemas Mentales (ADSM) [3, 4], el cual busca proporcionar un enfoque estructurado para la interacción con el usuario, guiándolo en la autorreflexión y el desarrollo de habilidades emocionales (ver Capítulo 2). Adicionalmente, se incorpora el modelo Modelo de personalidad basado en cinco rasgos: Openness (Apertura), Conscientiousness (Escrupulosidad), Extraversion (Extraversión), Agreeableness (Amabilidad) y Neuroticism (Neuroticismo) (OCEAN) de los Cinco Grandes Rasgos de Personalidad [5] para personalizar la experiencia de cada usuario en función de sus características psicológicas individuales (más detalles en Subsección 3.2.9).

Uno de los aspectos más innovadores de esta solución es la integración con tecnologías de privacidad como Tecnología de registro distribuido e inmutable

(Blockchain) [6], lo que permitirá garantizar la seguridad, trazabilidad y anonimato de los datos recopilados durante las interacciones (véase Capítulo 4). Esto es crucial para generar confianza en los usuarios y fomentar el uso de la herramienta sin temor a la exposición de su información personal.

Resolver este problema conlleva un valor significativo tanto a nivel individual como social. Desde una perspectiva individual, los usuarios tendrán acceso inmediato a una herramienta de apoyo emocional personalizada, sin las limitaciones de horarios, costos o disponibilidad geográfica. Esto puede contribuir a una detección temprana de problemas emocionales, mejorando la calidad de vida y el bienestar general. A nivel social, la aplicación podría aliviar la carga sobre los profesionales de la salud mental al proporcionar una primera capa de asistencia, permitiendo que los casos más urgentes sean atendidos con mayor prioridad.

La motivación para desarrollar esta solución radica en la necesidad de democratizar el acceso al apoyo emocional y psicológico. La combinación de inteligencia artificial y modelos psicológicos validados nos permite ofrecer una herramienta que no solo interactúe con los usuarios, sino que también les ayude a comprender y gestionar mejor sus emociones. En un mundo donde la salud mental se ve afectada por múltiples factores externos, desde la presión social hasta la inmediatez digital, es fundamental contar con herramientas accesibles, éticas y efectivas para el bienestar emocional.

Este documento detalla el desarrollo del proyecto, la metodología utilizada (Sección 1.1), los resultados obtenidos (Sección 5.3) y las conclusiones extraídas, con el fin de contribuir al avance de soluciones tecnológicas en el ámbito de la salud mental.

1.1. Definicioń del proyecto

1.1.1. Metodología

La metodología representa el conjunto de principios, técnicas y procedimientos utilizados para planificar, desarrollar y evaluar un proyecto. En el ámbito del desarrollo de software, adoptar una metodología adecuada es esencial para asegurar la calidad del producto, la satisfacción del cliente y la eficiencia del equipo de trabajo.

Para este proyecto se ha optado por una metodología ágil, en concreto el marco de trabajo **Metodología ágil de desarrollo de software (SCRUM)**[7]. , debido a su enfoque iterativo, su flexibilidad ante cambios y su capacidad para facilitar entregas continuas de valor. SCRUM promueve la colaboración, la adaptación y la mejora constante, lo que resulta especialmente útil en entornos donde los requisitos pueden evolucionar o no estar completamente definidos desde el inicio, como ocurre en este proyecto basado en tecnologías emergentes como los Large Language Model (LLM) y blockchain[8].

Qué es SCRUM

SCRUM es una metodología ágil para la gestión y desarrollo de productos complejos. Está estructurada en ciclos temporales cortos llamados *sprints* (de duración fija, generalmente entre 1 y 4 semanas), en los que se desarrolla un incremento funcional del producto. Al final de cada sprint, se realiza una revisión del trabajo y se ajusta la planificación futura en función de los resultados obtenidos.

El proceso SCRUM se articula a través de los siguientes elementos:

- **Product Backlog:** lista priorizada de funcionalidades o requisitos del producto. Es dinámica y puede evolucionar durante el desarrollo. Representa el "qué" se quiere construir.
- **Sprint Backlog:** subconjunto del Product Backlog que el equipo se compromete a desarrollar durante un sprint. Incluye las tareas necesarias para implementar cada funcionalidad.
- **Sprint:** periodo de tiempo fijo donde se desarrollan y entregan incrementos del producto. Cada sprint comienza con una planificación y termina con una revisión y una retrospectiva.
- **Daily Scrum:** reunión diaria de 15 minutos en la que el equipo sincroniza su progreso, identifica bloqueos y planifica el trabajo del día.
- **Sprint Review:** al final del sprint, se presenta el trabajo realizado y se recopila feedback para futuras iteraciones.
- **Sprint Retrospective:** reunión interna del equipo para identificar mejoras en el proceso de trabajo.
- **Taskboard o Kanban:** tablero donde se visualiza el estado de cada tarea (pendiente, en desarrollo, en revisión, completada), mejorando la organización y la transparencia.

Roles en SCRUM

SCRUM define varios roles fundamentales:

- **Product Owner (Propietario del Producto):** responsable de maximizar el valor del producto. Define los requisitos, prioriza el backlog y valida los resultados obtenidos.
- **Scrum Master:** facilita el proceso, ayuda a resolver impedimentos y se asegura de que se siga la metodología.
- **Development Team (Equipo de Desarrollo):** grupo autoorganizado que construye el producto.
- **Stakeholders (Interesados):** personas externas al equipo que tienen interés en el producto, como usuarios finales o instituciones colaboradoras. Aunque no participan en el desarrollo diario, aportan requerimientos, feedback y validación.

Aplicación de SCRUM al Proyecto

La metodología SCRUM se ha aplicado al desarrollo de este proyecto dividiéndolo en **sprints estructurados como hitos** (descritos en la Sección 1.1.2). . Cada hito representa una fase concreta de avance, con objetivos bien definidos y entregables asociados.

Dado que el desarrollo ha sido individual, se han realizado adaptaciones prácticas a la metodología, como:

El uso de SCRUM ha sido especialmente eficaz para organizar las múltiples partes interdependientes del sistema (frontend, backend, asistentes, vector store, etc.), permitiendo una evolución coherente del prototipo y facilitando su integración final. Dado que el desarrollo ha sido individual, se han realizado adaptaciones prácticas a la metodología, como:

- El **Product Owner** ha el CEO de Innopersona [9], empresa con la que se ha colaborado para el desarrollo de esta aplicación.
- El **Scrum Master** y el **equipo de desarrollo** han recaído en el mismo estudiante, quien ha sido responsable de la organización del trabajo, la toma de decisiones técnicas y el seguimiento del progreso.
- Se ha mantenido una **documentación iterativa**, incorporando mejoras y refinamientos tras cada hito, revisando tanto aspectos funcionales como técnicos y éticos.
- Se ha utilizado un **tablero Kanban digital en Microsoft Teams** para gestionar las tareas del proyecto, organizar el backlog de funcionalidades, priorizar los desarrollos y hacer seguimiento del progreso de cada módulo en tiempo real.

La elección de SCRUM ha resultado especialmente beneficiosa al permitir afrontar el proyecto de forma modular, facilitando la integración de tecnologías complejas como el almacenamiento cifrado en Azure, la blockchain privada y la clasificación de personalidad mediante modelos de lenguaje.

1.1.2. Hitos del Proyecto

El desarrollo del sistema se estructuró desde el inicio en torno a una metodología ágil de tipo SCRUM, planificando una secuencia de hitos con entregables parciales en ciclos iterativos. A continuación se enumeran los hitos definidos en la planificación inicial del proyecto:

1. **Hito 0 – Análisis y planificación (08/01/2025 – 17/01/2025):**
    - Estudio preliminar del modelo ADSM y del enfoque OCEAN.
    - Revisión de tecnologías disponibles en el ecosistema Azure.
    - Definición del backlog inicial y criterios de éxito.
2. **Hito 1 – Diseño arquitectónico (18/01/2025 – 26/01/2025):**

- Modelado de la arquitectura lógica en capas.
- Definición de subsistemas, canales de comunicación y flujos.
- Elaboración de diagramas C4 de contexto, contenedores y componentes.

3. **Hito 2 – Desarrollo de asistentes GPT (27/01/2025 – 13/02/2025):**
   - Redacción de instrucciones y validación del comportamiento.
   - Integración con `file-search` y vector store por asistente.
   - Pruebas de funcionamiento conversacional y adaptación semántica.
4. **Hito 3 – Validación intermedia I (14/02/2025 – 16/02/2025):**
   - Pruebas funcionales del sistema de asistentes.
   - Evaluación del comportamiento conversacional y ajuste fino.
5. **Hito 4 – Clasificador de personalidad (17/02/2025 – 04/03/2025):**
   - Elaboración de dataset etiquetado mediante modelo generativo.
   - Entrenamiento del modelo multilabel con **XLM-RoBERTa**, una versión multilingüe del modelo RoBERTa, entrenada con grandes cantidades de datos en más de 100 idiomas y optimizada para tareas de comprensión del lenguaje natural [10].
   - Implementación de la inferencia en entorno **serverless**, un modelo de computación en la nube en el que los desarrolladores escriben funciones que se ejecutan bajo demanda sin necesidad de gestionar servidores, lo que facilita la escalabilidad y reduce los costes operativos [11].
6. **Hito 5 – Validación intermedia II (05/03/2025 – 06/03/2025):**
   - Validación técnica del flujo de inferencia.
   - Pruebas de clasificación con historiales reales.
7. **Hito 6 – Cifrado y almacenamiento (07/03/2025 – 23/03/2025):**
   - Cifrado simétrico de las conversaciones por usuario.
   - Gestión segura de claves con **Gestión segura de claves y secretos en Azure (Azure Key Vault)**, un servicio en la nube que permite almacenar y gestionar secretos, claves de cifrado y certificados de forma centralizada y segura [12].
   - Persistencia en **Servicio de almacenamiento de objetos en la nube (Azure Blob Storage)** con control de sesiones, un servicio de almacenamiento de objetos optimizado para grandes volúmenes de datos no estructurados, que permite guardar conversaciones cifradas con alta disponibilidad y escalabilidad [13].
8. **Hito 7 – Validación intermedia III (24/03/2025 – 25/03/2025):**

- Pruebas de acceso seguro a las conversaciones cifradas.
- Simulación de sesiones concurrentes y expiración automática.

9. **Hito 8 – Registro en blockchain (26/03/2025 – 10/04/2025):**
   - Despliegue de red privada con **Cliente Ethereum de código abierto para redes blockchain privadas (Hyperledger Besu)**, un cliente Ethereum de código abierto diseñado para entornos empresariales, que permite crear redes blockchain privadas, públicas o de consorcio con soporte para distintos algoritmos de consenso [14].
   - Desarrollo de funciones de firma y publicación de hash.
   - Validación de integridad desde **Funciones serverless de Microsoft Azure (Azure Functions)**, una plataforma serverless que permite ejecutar código bajo demanda en la nube de forma escalable y sin gestionar infraestructura, ideal para tareas automatizadas como validaciones, procesamiento de eventos y control de flujos lógicos [15].
10. **Hito 9 – Validación intermedia IV (11/04/2025 – 13/04/2025):**
    - Pruebas de registro y recuperación de transacciones cifradas.
    - Comprobación de seguridad en firma y trazabilidad.
11. **Hito 10 – Consolidación final (14/04/2025 – 17/05/2025):**
    - Integración total del sistema en **Power Pages**, una plataforma de desarrollo low-code de Microsoft que permite crear sitios web seguros, conectados a datos empresariales y fácilmente integrables con servicios como Dataverse, Power Automate y Azure [**?**].
    - Pruebas de extremo a extremo del flujo conversacional completo.
    - Feedback de usuarios cercanos y ajuste del diseño.
12. **Hito 11 – Documentación y entrega (ejecución transversal durante todo el proyecto):**
    - Redacción progresiva de la memoria del TFG en paralelo al desarrollo.
    - Revisión y actualización continua de los anexos y figuras.
    - Preparación del material para la defensa y entrega final.
    - Coordinación con el tutor para validar avances documentales en cada hito.

Comparativa entre lo planificado y lo ejecutado

Aunque la estructura general del cronograma se mantuvo estable, varios hitos presentaron desviaciones en sus fechas de ejecución real respecto a lo previsto inicialmente. Estas variaciones estuvieron motivadas por la necesidad de ajustes técnicos, solapamientos entre tareas y validaciones iterativas. El único hito que

no sigue una planificación temporal acotada fue el correspondiente a la documentación, que se desarrolló de forma transversal a lo largo de todo el proyecto.

Hito	**Fechas previstas**	**Fechas reales**
Hito 0 – Análisis y planificación	08/01/2025 – 17/01/2025	08/01/2025 – 18/01/2025
Hito 1 – Diseño arquitectónico	18/01/2025 – 26/01/2025	19/01/2025 – 29/01/2025
Hito 2 – Desarrollo de asistentes GPT	27/01/2025 – 13/02/2025	30/01/2025 – 17/02/2025
Hito 3 – Validación intermedia I	14/02/2025 – 16/02/2025	18/02/2025 – 19/02/2025
Hito 4 – Clasificador de personalidad	17/02/2025 – 04/03/2025	20/02/2025 – 10/03/2025
Hito 5 – Validación intermedia II	05/03/2025 – 06/03/2025	11/03/2025 – 12/03/2025
Hito 6 – Cifrado y almacenamiento	07/03/2025 – 23/03/2025	13/03/2025 – 29/03/2025
Hito 7 – Validación intermedia III	24/03/2025 – 25/03/2025	30/03/2025 – 01/04/2025
Hito 8 – Registro en blockchain	26/03/2025 – 10/04/2025	02/04/2025 – 18/04/2025
Hito 9 – Validación intermedia IV	11/04/2025 – 13/04/2025	19/04/2025 – 20/04/2025
Hito 10 – Consolidación final	14/04/2025 – 17/05/2025	21/04/2025 – 01/06/2025
Hito 11 – Documentación y entrega	08/01/2025 – 17/05/2025	*Ejecución transversal (enero–junio)*

Cuadro 1.1: Comparativa entre hitos previstos y fechas reales de ejecución.

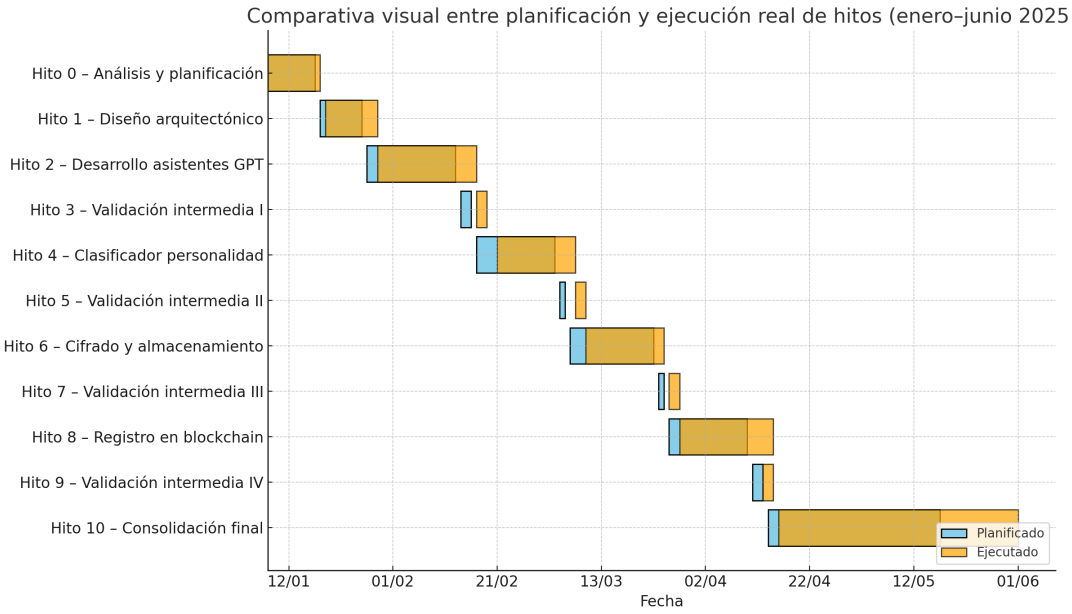

Figura 1.1: Comparativa visual entre planificación estimada y ejecución real, incluyendo el Hito 11 como tarea transversal (enero–junio 2025).

Estas desviaciones se analizan en detalle en la Sección 5.4.2, junto con las causas técnicas que motivaron la reorganización parcial de los hitos.

1.1.3. Alcance del Proyecto

El sistema desarrollado está orientado a su uso dentro del marco de proyectos de desarrollo personal y acompañamiento emocional impulsados por la empresa **Innopersona**, en colaboración con instituciones educativas, centros de bienestar, empresas u organizaciones interesadas en el crecimiento emocional de sus miembros.

El usuario final de la aplicación será cualquier persona interesada en realizar un proceso de autoevaluación y mejora personal guiado por inteligencia artificial, especialmente mediante el uso de asistentes basados en el **modelo ADSM** de Gerardo Tuduri. La herramienta ofrece un acompañamiento conversacional estructurado, clasificación de personalidad mediante el modelo **OCEAN** y recomendaciones personalizadas.

Los perfiles de usuario contemplados incluyen:

- **Usuarios individuales:** podrán registrarse, conversar con los asistentes GPT, generar su perfil de personalidad y hacer uso de la herramienta como parte de su proceso personal de desarrollo emocional y autoconocimiento.
- **Tutores, coaches o formadores:** mediante futuras extensiones, se contempla que puedan acceder (previa autorización) a los perfiles de los usuarios que acompañan, siempre respetando los principios éticos de privacidad y consentimiento informado.

- **Entidades colaboradoras (empresas, centros educativos, etc.):** podrán distribuir el acceso a la aplicación a través de licencias u ofertas de bienestar para sus miembros o empleados. La aplicación está diseñada para integrarse fácilmente en distintos entornos institucionales.

Cabe destacar que **Innopersona**[3] actúa como impulsora del proyecto, aportando todas las licencias necesarias para su desarrollo (servicios de Azure, suscripción a OpenAI, herramientas de gestión, etc.), y facilitando los medios técnicos y humanos para su implementación.

Se proporcionan pautas técnicas para asegurar la correcta ejecución de la aplicación, especialmente en lo relativo a la seguridad del almacenamiento, la configuración de los asistentes GPT y el uso ético de los datos.

El sistema no pretende sustituir a profesionales de la salud mental, sino ofrecer una herramienta complementaria de reflexión y orientación emocional basada en inteligencia artificial y principios de desarrollo personal.

1.2. Objetivos del Proyecto

El objetivo general de este proyecto es desarrollar una aplicación web que utilice asistentes basados en inteligencia artificial generativa para ofrecer acompañamiento emocional personalizado. La propuesta se basa en el modelo ADSM (Análisis Digital de Sistemas Humanos) de Gerardo Tuduri como marco metodológico, y en el modelo OCEAN como sistema de clasificación de personalidad.

Para ello, el proyecto se articula en torno a dos objetivos principales, cada uno con una serie de metas específicas que se detallan a continuación:

Objetivo 1. Desarrollo e integración de modelos de inteligencia artificial para el acompañamiento emocional personalizado

- Desarrollar una serie de asistentes GPT especializados, cada uno orientado a una fase del modelo ADSM. Estos asistentes permitirán al usuario realizar un proceso guiado de reflexión y toma de decisiones (Capítulo 3).

- Diseñar y aplicar instrucciones (prompts) estructuradas que permitan a los modelos GPT interactuar conforme a las directrices del modelo ADSM, facilitando la atención focalizada, la evaluación de opciones, la supervisión del comportamiento y la motivación intrínseca del usuario (Apéndice B).

- Extraer un perfil de personalidad del usuario aplicando técnicas de Procesamiento de Lenguaje Natural (Procesamiento de Lenguaje Natural (PLN))[16] sobre las conversaciones mantenidas con los asistentes. Se empleará para ello un modelo preentrenado basado en transformers[17] adaptado al idioma español (ver Subsección 3.2.9).

Objetivo 2. Garantizar la seguridad, privacidad y seguimiento del bienestar emocional del usuario

- Implementar un sistema de autenticación robusto y adaptable mediante Azure Active Directory Business to Consumer (Azure AD B2C), permitiendo gestionar el acceso de usuarios de forma segura y escalable (Capítulo 4).
- Definir una arquitectura de almacenamiento segura para las conversaciones entre el usuario y el asistente. Dado que estos textos contienen información sensible, se propone su almacenamiento cifrado mediante Azure Blob Storage, con claves gestionadas a través de Azure Key Vault, garantizando así el acceso exclusivo por parte del usuario (véase también Subsección 3.2.8).
- Almacenar la clasificación de personalidad basada en OCEAN en una blockchain privada, asegurando la integridad y trazabilidad de la información psicológica generada (Subsección 3.2.10).

1.3. Estructura del documento

La presente memoria se organiza en una serie de capítulos estructurados de forma lógica y progresiva, con el objetivo de guiar al lector a través del contexto, el desarrollo, la implementación y la evaluación del sistema propuesto. A continuación, se describe brevemente el contenido de cada uno de ellos:

Capítulo 1 – Introducción: presenta el contexto y la motivación del trabajo, los objetivos generales y específicos, el enfoque metodológico adoptado (SCRUM) y el alcance del proyecto. Este capítulo sienta las bases conceptuales y organizativas del Trabajo Fin de Grado.

Capítulo 2 – Estado del Arte: ofrece una revisión crítica de los antecedentes teóricos y tecnológicos relevantes para el desarrollo del sistema. Se analizan el modelo ADSM, el modelo de personalidad OCEAN, las arquitecturas de modelos de lenguaje como Transformer, y diversas soluciones existentes de soporte emocional mediante inteligencia artificial.

Capítulo 3 – Desarrollo del sistema: detalla el proceso de diseño, implementación y selección tecnológica. Se presentan los requisitos funcionales y no funcionales del sistema, así como las herramientas utilizadas para abordar cada subsistema, desde el frontend web hasta la integración con blockchain.

Capítulo 4 – Arquitectura del sistema: describe la arquitectura software implementada, estructurada según el modelo C4. Se incluyen los diagramas de contexto, contenedores, componentes y código, así como los flujos de datos y las decisiones de diseño adoptadas.

Capítulo 5 – Evaluación de resultados y conclusiones: analiza el cumplimiento de los objetivos definidos, la validación de requisitos, los aprendizajes derivados del proceso de desarrollo y las principales dificultades encontradas.

También se proponen líneas futuras de trabajo y se evalúa la contribución del proyecto a los Objetivos de Desarrollo Sostenible (ODS).

Bibliografía: recoge las referencias bibliográficas utilizadas a lo largo del trabajo, tanto en lo relativo a conceptos teóricos como a tecnologías implementadas.

Anexos: incluyen documentación complementaria de carácter técnico, tales como pruebas unitarias, instrucciones detalladas de los asistentes desarrollados, proceso de entrenamiento del modelo de personalidad, y el manual técnico completo del sistema.

Capítulo 2

Estado del Arte

El interés por las aplicaciones digitales de salud mental ha crecido de manera significativa en los últimos años, impulsado tanto por el auge de la inteligencia artificial como por la necesidad de ofrecer alternativas accesibles y escalables a la atención psicológica tradicional. Este crecimiento ha dado lugar a una amplia variedad de plataformas que ofrecen soporte emocional mediante asistentes conversacionales, aprendizaje automático y modelos de lenguaje como GPT.

Antes de profundizar en el análisis del estado del arte en aplicaciones de soporte emocional, es importante introducir una serie de conceptos fundamentales que permitirán comprender mejor el contexto tecnológico y metodológico en el que se enmarca este proyecto. Estos conceptos abarcan desde las bases teóricas del modelo ADSM hasta las arquitecturas modernas de modelos de lenguaje como los *transformers*, y resultan esenciales para entender cómo se estructura y fundamenta la solución propuesta, que se describe en detalle en Capítulo 3 y Capítulo 4.

2.1. Modelo de Clasificación de Personalidad OCEAN

El modelo OCEAN, también conocido como el modelo de los Cinco Grandes Rasgos de Personalidad (*Big Five*), es uno de los enfoques más utilizados y validados empíricamente para describir la personalidad humana. Este modelo dimensional clasifica los rasgos de una persona en cinco factores principales, que se consideran relativamente estables a lo largo del tiempo y consistentes en diferentes contextos culturales y situacionales [5].

- **Openness to Experience (Apertura a la experiencia)**: describe el grado de curiosidad intelectual, imaginación, apertura a ideas nuevas, aprecio por el arte y creatividad. Personas con una alta puntuación suelen ser innovadoras y receptivas al cambio, mientras que las de baja puntuación tienden a ser más convencionales y pragmáticas.
- **Conscientiousness (Responsabilidad)**: refleja el nivel de organización, autodisciplina, fiabilidad y orientación a objetivos. Las personas con alta responsabilidad suelen ser meticulosas, persistentes y eficientes, mientras

que las puntuaciones bajas pueden asociarse con impulsividad y desorganización.

- **Extraversion (Extraversión)**: indica la tendencia a buscar la estimulación social, la energía y la expresividad emocional. Individuos extrovertidos suelen ser sociables, entusiastas y asertivos, mientras que los introvertidos tienden a ser reservados y tranquilos.
- **Agreeableness (Amabilidad)**: se relaciona con la calidad de las relaciones interpersonales. Las personas con alta amabilidad tienden a ser cooperativas, empáticas y altruistas; las que puntúan bajo pueden mostrar comportamientos más competitivos o críticos.
- **Neuroticism (Neuroticismo)**: mide la tendencia a experimentar emociones negativas como ansiedad, tristeza o inseguridad. Las puntuaciones altas reflejan mayor vulnerabilidad emocional, mientras que las bajas indican mayor estabilidad emocional y resiliencia.

Este modelo ha sido ampliamente utilizado en psicología, recursos humanos, educación y salud mental por su capacidad para capturar de forma clara y estructurada las dimensiones más relevantes de la personalidad. En el contexto del presente proyecto, el modelo OCEAN se aplica como base para analizar las interacciones entre el usuario y los asistentes virtuales, generando un perfil de personalidad a partir de las características lingüísticas y comportamentales detectadas en la conversación.

Para ello, se procesan los mensajes del usuario mediante técnicas de procesamiento de lenguaje natural (PLN) y se aplican modelos de clasificación entrenados para predecir las puntuaciones en cada uno de los cinco rasgos.

2.2. El modelo ADSM: Análisis Digital de Sistemas Mentales

El **Análisis Digital de Sistemas Mentales** (ADSM) es un modelo conceptual desarrollado por el investigador y escritor Gerardo Iván Tudurí[18], orientado a la comprensión y optimización de los procesos mentales humanos en el contexto de la era digital. Este modelo propone una integración entre la neurociencia, la inteligencia artificial y la psicología, con el objetivo de facilitar el autoconocimiento y la mejora de las capacidades cognitivas y emocionales de los individuos.

ADSM surge como respuesta a la creciente interacción entre humanos y tecnologías digitales, especialmente en el ámbito de la inteligencia artificial. Tudurí plantea que la externalización de procesos mentales en máquinas inteligentes genera un impacto significativo en la autopercepción y el comportamiento humano. En este sentido, el ADSM busca proporcionar herramientas para que las personas puedan adaptarse y evolucionar en armonía con estos avances tecnológicos [4].

El modelo se basa en la premisa de que la mente humana puede ser analizada y optimizada mediante técnicas digitales, permitiendo una mayor comprensión

de sus dinámicas internas y promoviendo una evolución hacia lo que Tudurí denomina una *mente biodigital*. Esta nueva configuración mental integraría de manera efectiva las competencias humanas con las capacidades de las máquinas, facilitando una coexistencia más eficiente y enriquecedora.

ADSM se estructura en torno a una **matriz de datos simplificada del yo (Matriz de Desambiguación de Situaciones del Yo (MDS-YO))**, que permite convertir cualquier experiencia humana en un episodio computable mediante el registro de seis tipos de datos fundamentales. Esta matriz facilita el análisis y la comprensión de las experiencias individuales, proporcionando una base para la reconfiguración de la autopercepción y el comportamiento.

Además, el modelo incorpora una **fórmula algorítmica** que permite a las personas descifrar su **Algoritmo Personal (Algoritmización Personal (AP))**. Este algoritmo personal se utiliza para reconfigurar la autopercepción y el comportamiento general, aplicándolo en la vida cotidiana como una herramienta de optimización individual.

ADSM ha sido implementado en sesiones de entrenamiento y consultoría, donde se aplican técnicas de *biodigitalización adaptativo-evolutiva* para que el modelo mental de la persona se autoperciba desde la perspectiva de la máquina. Estas técnicas buscan transformar el sistema mental obsoleto en uno preparado para el futuro, promoviendo una integración efectiva con las tecnologías emergentes.

En el contexto de este trabajo, el modelo ADSM se emplea como base para diseñar asistentes conversacionales que acompañen al usuario en procesos de introspección y mejora personal, estructurando el diálogo en torno a sus componentes fundamentales. Esta aplicación permite ofrecer una herramienta de apoyo emocional que facilita el autoconocimiento y el crecimiento personal, apoyada en modelos teóricos sólidos y con una arquitectura tecnológica orientada a la privacidad y la ética.

Por tanto, el Análisis Digital de Sistemas Mentales representa una propuesta innovadora que integra conocimientos de diversas disciplinas para abordar los desafíos que plantea la interacción entre humanos y tecnologías digitales. Su enfoque en la optimización de los procesos mentales y la promoción de una *mente biodigital* lo convierten en un modelo relevante para el desarrollo de herramientas que buscan mejorar el bienestar emocional y cognitivo de las personas en la era digital.

2.3. Modelos de Lenguaje de Gran Escala (LLM), Arquitectura Transformer y Representaciones Vectoriales

Uno de los avances más relevantes en el campo del procesamiento del lenguaje natural (NLP) ha sido el desarrollo de los **Modelos de Lenguaje de Gran Escala** (LLM, por sus siglas en inglés: *Large Language Models*). Estos modelos han transformado por completo la forma en que las máquinas comprenden y

generan lenguaje humano, permitiendo crear herramientas capaces de mantener conversaciones, responder preguntas, generar texto, resumir información o asistir emocionalmente a los usuarios, como es el caso de este proyecto.

2.3.1. Arquitectura Transformer

Los LLM están construidos sobre una arquitectura denominada **Transformer**, propuesta por Vaswani et al. en 2017 [19]. Esta arquitectura supuso una revolución respecto a modelos anteriores, como las redes neuronales recurrentes (RNN) y las LSTM, una variante de RNN diseñada para aprender dependencias a largo plazo mediante puertas que regulan el flujo de información [20], ya que permite procesar las secuencias de texto de forma completamente paralela y escalable.

El mecanismo clave de los Transformers es la **auto-atención**, que permite al modelo ponderar la importancia relativa de cada palabra en una secuencia respecto a las demás. Esto hace posible una comprensión más precisa del contexto y de las relaciones semánticas y sintácticas entre palabras, lo que se traduce en una mayor coherencia y naturalidad en las respuestas generadas.

2.3.2. Modelos destacados

Entre los modelos basados en arquitectura Transformer más conocidos se encuentran:

- **GPT-3 y GPT-4** (OpenAI): especializados en generación de texto y conversación contextual.
- **BERT** (Google): un modelo basado en Transformer optimizado para tareas de comprensión del lenguaje, como clasificación y análisis de texto [21].
- **LLaMA** (Meta): diseñado para ser más ligero y eficiente, con aplicaciones académicas.
- **Claude** (Anthropic): enfocado en alineación ética y seguridad del lenguaje generado.

En este proyecto se ha utilizado **GPT-4** mediante la API de OpenAI, por su elevada capacidad para generar respuestas adaptadas al estado emocional del usuario y seguir instrucciones complejas, algo clave en la implementación del modelo ADSM. Esta elección también responde a un requisito del entorno de colaboración empresarial, que establece el uso de modelos licenciados por OpenAI, asegurando así el cumplimiento de políticas de seguridad y gobernanza tecnológica.

2.3.3. Embeddings y Representación Vectorial del Texto

Para que las máquinas puedan entender el lenguaje natural, es necesario convertir las palabras en un formato que pueda ser procesado numéricamente. Este proceso se realiza mediante los llamados **embeddings**, que son representaciones vectoriales densas y de dimensión fija de palabras, frases o documentos.

Los embeddings capturan la *semántica* del texto: palabras con significados similares tienen vectores cercanos entre sí en el espacio multidimensional. Modelos como `word2vec`, `GloVe`, `BERT embeddings` o `OpenAI embeddings` permiten generar estas representaciones para su posterior uso en tareas de clasificación, recomendación o recuperación de información.

2.3.4. Bases de Datos Vectoriales

Una vez generados los embeddings, es habitual almacenarlos en lo que se conoce como una **base de datos vectorial**. Estas bases permiten realizar búsquedas semánticas eficientes mediante operaciones de **búsqueda por similitud**, como la distancia coseno o la distancia euclidiana.

Entre las soluciones más utilizadas en la industria se encuentran:

- **Pinecone**, **Weaviate** y **Qdrant**: servicios especializados para alojar y consultar grandes volúmenes de vectores.
- **FAISS** (Facebook AI Similarity Search): biblioteca optimizada para búsquedas vectoriales locales.
- **Milvus**: sistema distribuido y escalable para búsquedas semánticas a gran escala.

Estas herramientas permiten, por ejemplo, que el asistente pueda recordar conversaciones anteriores del usuario buscando mensajes similares en una base vectorial. En este trabajo, los embeddings generados a partir de las interacciones con los asistentes podrían almacenarse de forma cifrada y privada, permitiendo personalizar el diálogo, generar perfiles psicológicos y mantener la coherencia del sistema sin comprometer la privacidad del usuario.

2.3.5. Aplicación en el proyecto

Gracias a la combinación de LLM, embeddings y bases de datos vectoriales, este proyecto implementa asistentes GPT capaces de interactuar con los usuarios de forma personalizada, empática y adaptativa. Estas tecnologías permiten simular una conversación natural y empática, identificar patrones de comportamiento y construir un perfil emocional a partir de los textos generados, todo ello cumpliendo con los principios de trazabilidad, privacidad y personalización que caracterizan el diseño del sistema.

2.4. Aplicaciones actuales de soporte emocional con IA

Una de las aplicaciones más representativas en este ámbito es **Koko**[22], una plataforma que utiliza inteligencia artificial para proporcionar apoyo emocional en línea. Koko permite a los usuarios expresar sus emociones, recibir mensajes generados por IA y participar en conversaciones de ayuda mutua con otros usuarios. Está diseñada con un fuerte enfoque en la evidencia empírica y la ética, y ha sido objeto de investigaciones académicas sobre el uso de LLMs (Large Language Models) en entornos de salud mental [23]. A nivel técnico, Koko

se apoya en modelos generativos como GPT-3 para simular respuestas empáticas, utilizando filtros semánticos y herramientas de moderación automática. Sin embargo, su infraestructura depende de servidores centralizados, y su almacenamiento de datos ha sido objeto de debate por su gestión ética en estudios de campo.

Además de Koko, existen otras herramientas destacadas en el mercado, entre ellas:

- **Wysa** [24]: basada en un modelo de NLP cerrado, con respuestas diseñadas por psicólogos utilizando técnicas de Terapia Cognitivo-Conductual (TCC), un enfoque psicológico ampliamente validado que se centra en identificar y modificar patrones de pensamiento y comportamiento disfuncionales [25], así como de mindfulness. No utiliza modelos generativos, lo que limita la adaptabilidad conversacional pero garantiza una mayor supervisión clínica. Las conversaciones son privadas, pero no se dispone de mecanismos de trazabilidad ni acceso del usuario al control total de sus datos.

- **Woebot**[26]: incorpora aprendizaje automático y procesamiento de lenguaje natural para guiar al usuario a través de ejercicios de TCC. Su IA está basada en reglas con cierto grado de personalización, pero no utiliza grandes modelos generativos. Destaca por su validación clínica, pero no emplea mecanismos de análisis de personalidad ni personalización profunda.

- **Replika**[27]: utiliza modelos generativos basados en arquitecturas tipo GPT para mantener conversaciones abiertas y personalizadas. A diferencia de otras aplicaciones, Replika ajusta su estilo al usuario en función del histórico conversacional. No obstante, su enfoque es más relacional que terapéutico y no ofrece análisis estructurados de personalidad ni opciones avanzadas de privacidad.

- **Tess**[28]: plataforma orientada a organizaciones, con una arquitectura híbrida que combina procesamiento de texto, aprendizaje supervisado y módulos de respuesta automatizada. Tess personaliza en función de variables demográficas y emocionales, aunque no incorpora análisis de personalidad profundos ni funcionalidades accesibles directamente al usuario final.

- **7 Cups**[29]: emplea IA de bajo nivel como asistente inicial, pero se centra en la interacción humana mediante una red de voluntarios. Su tecnología no está pensada para análisis de personalidad ni seguimiento automatizado, aunque ofrece anonimato y accesibilidad global.

Estas plataformas ofrecen accesibilidad, disponibilidad constante y anonimato, lo cual representa una ventaja significativa frente a la atención tradicional. No obstante, también presentan limitaciones: muchas de ellas no personalizan profundamente las respuestas, carecen de integración con modelos cognitivos estructurados y presentan riesgos en términos de privacidad, ya que no siempre se garantiza la trazabilidad y confidencialidad de los datos del usuario.

2.5. Clasificación de personalidad y seguimiento emocional

Algunas soluciones actuales han comenzado a incorporar análisis de personalidad y seguimiento del bienestar, pero pocas lo hacen de forma integrada. Mientras que herramientas como **Replika** pueden adaptar su estilo conversacional al usuario, no ofrecen un análisis estructurado de rasgos como los del modelo **OCEAN**. Asimismo, la mayoría de las aplicaciones almacenan datos en servidores tradicionales, sin mecanismos robustos para asegurar la privacidad o el control por parte del usuario.

El modelo OCEAN (también conocido como los Cinco Grandes Rasgos de Personalidad) se ha consolidado como una referencia en el ámbito del análisis de personalidad en psicología computacional. Recientes avances en procesamiento del lenguaje natural (NLP) han permitido aplicar técnicas de clasificación automática de personalidad a partir de texto libre, como el generado en conversaciones entre usuarios y asistentes virtuales.

Entre las soluciones más relevantes en este campo se encuentran:

- **IBM Watson Personality Insights**[30]: plataforma pionera en ofrecer análisis de personalidad a partir de texto, utilizando el modelo OCEAN. Aunque actualmente discontinuada, su tecnología se basaba en análisis léxico y sintáctico sobre grandes volúmenes de texto, con visualización detallada de resultados.
- **LIWC (Linguistic Inquiry and Word Count)**[31]: herramienta ampliamente usada en investigación psicológica y social. Analiza texto mediante un diccionario categorizado de palabras que permite inferir rasgos emocionales, sociales y cognitivos. Si bien no genera directamente puntuaciones OCEAN, puede emplearse como base para clasificadores personalizados.
- **Modelos preentrenados en plataformas como Hugging Face**[32]: existen modelos en múltiples idiomas que pueden adaptarse mediante aprendizaje supervisado para clasificar personalidad, incluyendo variantes optimizadas para análisis biomédico y psicológico. Estos modelos permiten aplicar el enfoque OCEAN a partir de interacciones reales con asistentes conversacionales.

El enfoque propuesto en este trabajo se basa en capturar las interacciones entre el usuario y el asistente GPT, preprocesarlas mediante técnicas de limpieza y análisis lingüístico (*tokenización*, eliminación de *stopwords*, lematización, etc.) y posteriormente aplicar un modelo de clasificación que estime las puntuaciones OCEAN del usuario. Estas puntuaciones se almacenarán en un sistema blockchain privado, asegurando la inmutabilidad y trazabilidad de los perfiles psicológicos generados.

Para preservar la privacidad, se contempla un doble enfoque de almacenamiento:

- Por un lado, la clasificación OCEAN se almacena de forma cifrada en block-

chain.

- Por otro, se almacena una copia segura de las conversaciones originales del usuario (necesarias para la regeneración del perfil) en un entorno altamente protegido, asegurando que el acceso a esta información esté restringido y controlado de manera estricta.

Este enfoque refuerza la confianza del usuario al ofrecer transparencia, control y seguridad sobre sus datos emocionales y de personalidad, algo que no está presente en la mayoría de las plataformas actuales.

2.6. Aportaciones del presente trabajo

El presente proyecto propone una aproximación innovadora que combina:

- El uso de **asistentes GPT personalizados** para acompañar al usuario en procesos de toma de decisiones y autorregulación emocional, basados en el modelo **ADSM** de Gerardo Tuduri [4].
- Un sistema de **clasificación de personalidad** mediante el modelo **OCEAN**, generado a partir del análisis de conversaciones entre usuario y asistente.
- La **integración de tecnología Blockchain** para asegurar que el perfil psicológico del usuario (OCEAN) se almacene de forma anónima, inmutable y verificable.
- Almacenamiento cifrado del historial conversacional en **entornos seguros**, garantizando que solo el usuario tenga acceso.

Esta propuesta busca no solo proporcionar una herramienta de apoyo emocional, sino también una vía de autoconocimiento y crecimiento personal, apoyada en modelos teóricos sólidos y con una arquitectura tecnológica orientada a la privacidad y la ética.

Capítulo 3

Desarrollo del sistema

3.1. Análisis y Agrupación de Requisitos por Subsistemas

Para mejorar la trazabilidad y claridad del diseño, se ha optado por organizar los requisitos del sistema agrupándolos según los principales subsistemas definidos en la arquitectura del proyecto.

A continuación, se describen los principales subsistemas identificados en el sistema:

1. Interfaz de Usuario (Frontend Web)
2. Autenticación y Gestión de Usuarios
3. Asistentes GPT
4. Procesamiento de Lenguaje Natural y Clasificación OCEAN
5. Almacenamiento Seguro
6. Blockchain Privada

3.1.1. Interfaz de Usuario (Frontend Web)

Este subsistema se encarga de la interacción con el usuario, desarrollado sobre *Microsoft Power Pages*.

Requisitos funcionales:

- Los usuarios deben poder iniciar sesión y registrarse por medio de *Azure AD B2C*.
- El usuario debe poder interactuar con asistentes GPT desde la interfaz web.
- Debe haber un asistente que ayude a seleccionar la herramienta más adecuada.
- Interfaz amigable y accesible desde distintos dispositivos.

Requisitos no funcionales:

- Alta disponibilidad del frontend.
- Compatibilidad multiplataforma.
- Tiempos de carga reducidos.

3.1.2. Autenticación y Gestión de Usuarios

Implementado con *Azure AD B2C*.

Requisitos funcionales:

- Soporte para registro e inicio de sesión seguro.

Requisitos no funcionales:

- Cumplimiento con políticas de privacidad y seguridad (RGPD) [33].
- Autenticación segura con tokens.

3.1.3. Asistentes GPT

Gestiona la lógica conversacional del sistema mediante la *API de OpenAI*.

Requisitos funcionales:

- Los asistentes deben seguir el protocolo del modelo ADSM.
- Debe haber un asistente por cada fase del modelo ADSM.
- Las instrucciones deben poder generarse dinámicamente y ajustarse al protocolo activo.
- Los asistentes deben dar respuestas sin abrumar con demasiada información.
- Los asistentes deben tener acceso al vector store con el protocolo que les corresponde.

Requisitos no funcionales:

- Uso de modelos de *openAI*.
- Comunicación en tiempo real mediante *WebSocket*.

3.1.4. Procesamiento de Lenguaje Natural y Clasificación OCEAN

Encargado de preprocesar las conversaciones y generar la clasificación de personalidad.

Requisitos funcionales:

- Se debe generar una clasificación de personalidad por el método *OCEAN*.
- El usuario debe poder descargar un *JSON* con su clasificación.

Requisitos no funcionales:

- Uso de técnicas como tokenización, lematización, eliminación de stopwords.
- El análisis debe ejecutarse de forma transparente para el usuario.

3.1.5. Almacenamiento Seguro

Responsable del almacenamiento cifrado de las conversaciones y perfiles.

Requisitos funcionales:

- Privacidad de las conversaciones del usuario.
- Privacidad de la clasificación *OCEAN*.

Requisitos no funcionales:

- Almacenamiento de conversaciones encriptadas en *Azure Blob Storage*.
- Uso de claves únicas por usuario gestionadas en *Azure Key Vault*.
- Despliegue modular por subsistemas.

3.1.6. Blockchain Privada

Almacena la clasificación *OCEAN* de forma inmutable y trazable.

Requisitos funcionales:

- Inmutabilidad de la clasificación *OCEAN* en blockchain privada.
- El usuario debe tener control exclusivo sobre su identidad emocional.

Requisitos no funcionales:

- Registro del hash del perfil *OCEAN* en la cadena.
- Verificabilidad del perfil sin comprometer la privacidad.

3.2. Tecnologías y Herramientas Utilizadas

En esta sección se describen las principales tecnologías empleadas en el desarrollo del sistema, explicando brevemente en qué consisten, cómo se han integrado en la solución propuesta y por qué han sido seleccionadas frente a otras alternativas. Se ha priorizado el uso de tecnologías escalables, seguras y compatibles con el entorno cloud de *Microsoft Azure*.

3.2.1. Microsoft Power Pages – Interfaz de usuario y desarrollo frontend

Microsoft Power Pages [34] es una plataforma de desarrollo web de bajo código que forma parte del ecosistema *Microsoft Power Platform*. Está diseñada para permitir la creación rápida de sitios web empresariales, públicos o autenticados,

conectados a datos y servicios del entorno *Azure* sin necesidad de infraestructura propia ni conocimientos avanzados de programación.

1. **Motivos para su elección:**
   - Recomendación de la empresa colaboradora: Innopersona recomendó el uso de *Power Pages* por su compatibilidad total con el ecosistema Microsoft y por contar ya con licencias disponibles.
   - Integración nativa con servicios clave: permite una conexión directa con *Azure AD B2C* (autenticación), *Azure Functions* (backend), Application Insights (telemetría) y almacenamiento seguro.
   - Enfoque low-code con capacidad full-code: combina la facilidad de edición visual con la posibilidad de personalizar completamente la lógica mediante *HTML*, *CSS* y *JavaScript*.
   - Despliegue en la nube y mantenimiento simplificado: no requiere servidores, dominios ni configuraciones locales, y permite trabajar íntegramente desde el navegador.
2. **Ventajas observadas:**
   - Desarrollo ágil: permitió crear rápidamente una interfaz funcional sin necesidad de desarrollar una SPA desde cero.
   - Seguridad integrada: el uso conjunto con Azure AD B2C proporciona una capa de autenticación profesional sin código adicional.
   - Personalización progresiva: fue posible adaptar por completo la interfaz mediante *JS*/*CSS*, logrando una experiencia visual coherente con el objetivo del sistema.
3. **Limitaciones detectadas:**
   - Restricciones del entorno: algunas operaciones complejas del *DOM* requieren soluciones creativas con *JavaScript* embebido.
   - Dependencia de Dataverse por defecto: si bien el proyecto no utiliza Dataverse, su uso está fuertemente integrado por defecto en *Power Pages*, lo que requiere configurar orígenes externos como *Azure Functions* de forma manual.
   - Carga dinámica limitada: no permite crear *SPA* puras (single-page applications), por lo que cada cambio de herramienta implica recargar la página.
4. **Conclusión:** *Microsoft Power Pages* ha demostrado ser una solución adecuada para el desarrollo rápido y seguro del frontend de este sistema. Su equilibrio entre simplicidad, integración con servicios cloud y capacidad de personalización avanzada lo convierte en una tecnología especialmente útil en contextos prototipados o de validación de producto como este Trabajo de Fin de Grado.

A continuación se detalla el análisis individual de las páginas desarrolladas en *Power Pages*.

5. **Frontend en Power Pages – Análisis por página**

   Cada una de las páginas desarrolladas en *Power Pages* constituye una unidad funcional para la interacción del usuario con un asistente GPT especializado. A continuación se describen individualmente aquellas que presentan una interfaz diferenciada y se analiza brevemente su estructura técnica. El código fuente completo se incluye en D.4.2.

6. **Página: Selección de herramienta (Selección de Uso)**

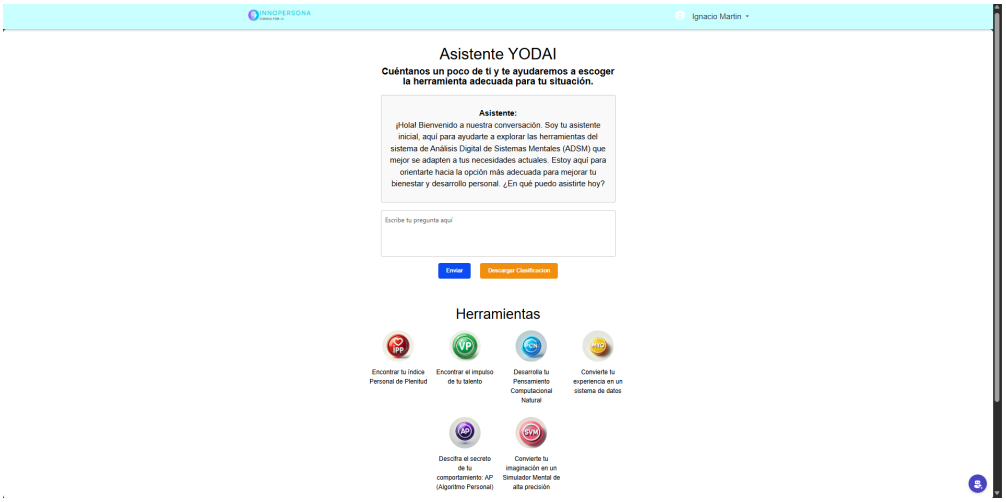

Figura 3.1: Interfaz de la página “Selección de Uso” donde el usuario inicia la interacción con el sistema.

   Como se muestra en la Figura 3.1, esta página actúa como punto de entrada al sistema. Aquí el usuario puede comenzar a conversar con el asistente inicial, que le guía y le deriva a la herramienta ADSM más adecuada según su situación. Una vez derivado, el usuario puede seleccionar visualmente la herramienta recomendada o cualquier otra, mediante botones personalizados con estilo adaptado.

7. **Estructura técnica:**

   - HTML: define un área de conversación con el asistente, un campo de texto para el usuario y un conjunto de botones representando cada herramienta (AP, Índice de Plenitud Personal (IPP), Vibración Personal (VP), Pensamiento Computacional Natural (PCN), SMV, MDS-YO).
   - CSS: personaliza la visualización de los botones, las respuestas del asistente y la experiencia en dispositivos móviles.
   - JavaScript: gestiona la apertura de la conversación, el envío de mensajes vía *WebSocket*, el guardado de interacciones, y la activación del botón para descargar el perfil emocional.

El código fuente completo de esta página se encuentra en el D.4.2, con los siguientes archivos:

- `Selección de Uso.es-ES.webpage.copy.html`
- `Selección de Uso.es-ES.customcss.css`
- `Selección de Uso.es-ES.customjs.js`

8. Páginas específicas de cada herramienta ADSM

   Una vez seleccionada una herramienta, el usuario accede a una página dedicada que mantiene el mismo diseño general. La única diferencia es que el asistente GPT especializado que responde en cada página está vinculado a un protocolo diferente (AP, IPP, SMV, VP, PCN, etc.).

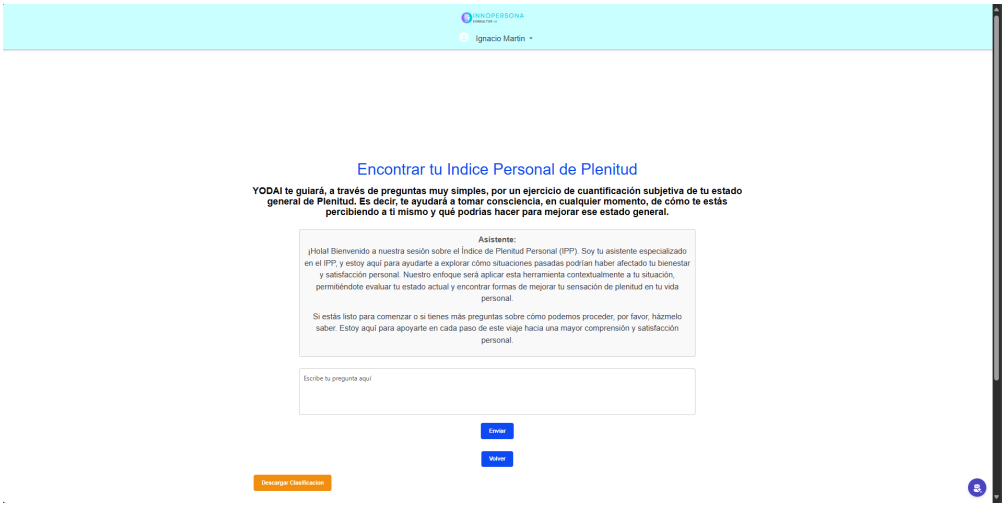

Figura 3.2: Interfaz de una página típica de herramienta ADSM (por ejemplo, *AP* o *VP*).

Como se observa en la Figura 3.2, cada página de herramienta presenta una estructura homogénea compuesta por un área de conversación, un campo de entrada de texto y un botón de envío. Además, incluye scripts que conectan con el backend mediante *WebSocket* para mantener la conversación en tiempo real. Aunque todas las páginas comparten esta estructura base, el comportamiento del asistente cambia en función del protocolo ADSM que se está aplicando, permitiendo así una interacción personalizada y específica según la herramienta seleccionada.

9. **Estructura común:**
   - Área de conversación con el asistente GPT especializado.
   - Input de texto del usuario y botón para enviar mensajes.
   - Script que establece la conexión con Azure Functions mediante WebSocket.

- Carga dinámica del historial y guardado cifrado de mensajes.

Cada página responde a la misma lógica, cambiando únicamente el asistente con el que se comunica. El código fuente se adapta mínimamente por herramienta, y está documentado en el D.4.2 con un archivo por página.

10. **Plantilla compartida: shared-functions.html**

    Todas las páginas utilizan una plantilla de funciones JavaScript comunes contenida en el archivo `shared-functions.html`, que centraliza la lógica reutilizable para asegurar consistencia entre páginas. Este archivo define:

    - `openWebSocket()`: establece una conexión *WebSocket* segura con *Azure Functions*, usando la identidad autenticada del usuario.
    - `sendMessage()`: envía el contenido del usuario al asistente *GPT* y gestiona su respuesta.
    - `appendMessage()`: añade cada mensaje al área de conversación (usuario o asistente), con formato diferenciado.
    - `scrollToBottom()`: asegura que el usuario siempre vea el último mensaje en pantalla.
    - `handleSessionEnd()`: detecta inactividad y lanza una función que registra el cierre de la sesión.
    - `downloadCSV()`: permite exportar el resumen de la clasificación *OCEAN* en formato *CSV*.

    Estas funciones permiten mantener un diseño modular y facilitan la evolución futura del sistema, ya que los cambios se aplican automáticamente a todas las páginas que heredan esta plantilla.

3.2.2. Azure AD B2C – Autenticación de usuarios

Azure Active Directory B2C (Business to Consumer)[35] es un servicio de identidad gestionado por Microsoft que permite implementar flujos de autenticación seguros para usuarios externos (clientes, ciudadanos, estudiantes, etc.). Ofrece una solución escalable, configurable y totalmente integrada con el ecosistema *Azure*.

1. **Motivaciones de su elección:**

    - Compatibilidad total con la plataforma Azure, lo que permite una integración directa con otros servicios como *Power Pages* (frontend), *Azure Functions* (backend) y *Key Vault* (seguridad).
    - Flujos de autenticación personalizables, adaptables a diferentes necesidades como inicio de sesión, registro o recuperación de contraseña.
    - Soporte para protocolos estándar como OAuth 2.0, OpenID Connect, MSAL y autenticación multifactor (MFA), que añade una capa adicional

de seguridad al requerir, además de la contraseña, una verificación adicional como un código enviado al móvil del usuario.

- Gestión simplificada, sin necesidad de mantener infraestructura ni lógica de autenticación propia.

2. **Configuración del entorno:**

   En este proyecto, el sistema de autenticación se basa en un tenant dedicado de *Azure AD B2C* configurado de la siguiente manera:

   - Nombre del inquilino: `<dominio>.onmicrosoft.com`
   - Aplicación registrada: `<nombre-de-la-aplicación>`
   - Tipo de cuenta compatible: Cuentas en cualquier proveedor de identidades (OpenID)
   - Flujos habilitados: concesión implícita con emisión de tokens de acceso e ID
   - URI de redirección: integrada con el portal de *Power Pages*

   Este registro de aplicación permite autenticar a los usuarios mediante un portal público basado en *Power Pages*, conectando con el sistema backend solo tras validación del token emitido por *Azure AD B2C*.

3. **Integración con el resto del sistema:**

   Una vez autenticado, el usuario puede interactuar con el asistente *GPT* y con las funciones del backend. El sistema utiliza el identificador del usuario autenticado para:

   - Asociar su clave de cifrado almacenada en *Key Vault*.
   - Recuperar o almacenar su historial conversacional cifrado en *Azure Blob Storage*.
   - Identificarlo dentro de la blockchain donde se registra la evolución de su perfil emocional.

4. **Seguridad y control:**

   La autenticación se realiza en un entorno aislado y conforme a los estándares modernos.

   No se gestionan ni almacenan contraseñas directamente: todo el proceso se delega a *Azure AD B2C*.

   La aplicación registrada no expone secretos ni certificados públicos.

   Los permisos concedidos a la aplicación y el manifiesto están documentados en 4.

5. **Comentario de desarrollo:**

Durante la fase inicial del proyecto se valoró la posibilidad de implementar una autenticación personalizada. Sin embargo, esta opción fue descartada por:

- Mayor complejidad técnica y de mantenimiento.
- Riesgos de seguridad asociados a la gestión manual de credenciales.
- Necesidad de cumplir normativas de protección de datos y privacidad.

La adopción de *Azure AD B2C* ha permitido garantizar un sistema de autenticación robusto, seguro y alineado con las mejores prácticas en proyectos donde la protección de la identidad del usuario es prioritaria.

Configuración del sistema de autenticación con Azure AD B2C

El sistema de autenticación de usuarios fue configurado utilizando *Azure Active Directory B2C (AD B2C)* en el portal de *Azure*, siguiendo los siguientes pasos:

1. **Creación del inquilino (tenant) B2C.** Se creó un nuevo inquilino de tipo *Azure AD B2C* con el dominio público `<dominio>.onmicrosoft.com`, seleccionando la región "Europa" y asociándolo a la suscripción activa del proyecto. El inquilino se vinculó al grupo de recursos donde se ubican los demás servicios de la solución.
2. **Registro de una nueva aplicación.** En el apartado "Registros de aplicaciones" se creó una nueva aplicación llamada `pilotoyodai`, con los siguientes parámetros principales:
   - Tipo de aplicación: cuentas personales y corporativas (multi-tenant).
   - ID de aplicación (cliente): generado automáticamente por Azure.
   - URI de redirección: `https://<dominio>.powerappsportals.com/signin-aad-b2c_2`.
   - Flujos de autenticación habilitados: concesión implícita con tokens de acceso e ID.
   - Compatibilidad: cuentas de cualquier proveedor compatible con OpenID Connect.
3. **Configuración de plataforma de autenticación.** Desde el menú "Autenticación" de la aplicación registrada se configuraron:
   - La plataforma `Web` con la URI de redirección al portal de *Power Pages*.
   - Habilitación de emisión de tokens de acceso e ID.
   - Permiso para flujos de clientes públicos (SPA).
   - URL de cierre de sesión estándar para logout.
4. **Permisos de API y control de acceso.** En la sección "Permisos de API" se otorgaron accesos mínimos necesarios a Microsoft Graph:

- `openid`
- `offline_access`
- `profile`

No se añadieron scopes personalizados ni permisos administrativos, siguiendo el principio de mínimo privilegio.

5. **Verificación y ajustes en el manifiesto.** Se revisó el manifiesto *JSON* de la aplicación para comprobar:
   - Que `requestedAccessTokenVersion` estuviera en 2.
   - Que `implicitGrantSettings` incluyera:

```
"implicitGrantSettings": {
    "enableAccessTokenIssuance": true,
    "enableIdTokenIssuance": true
}
```

6. **Creación del flujo de usuario.** Desde el menú "Flujos de usuario" se creó un flujo de tipo:
   - Tipo: Inicio de sesión y registro.
   - Nombre: `B2C_SIGNIN_OR_SIGNUP`.
   - MFA: desactivado inicialmente.
   - Atributos solicitados: nombre, apellidos y correo electrónico.

   Este flujo quedó vinculado al portal web, redirigiendo automáticamente a la autenticación al intentar acceder a páginas protegidas.
7. **Integración con Power Pages.** En *Power Pages* se configuró el proveedor de identidad externo, indicando:
   - Tipo: *Azure AD B2C*.
   - Tenant: `<dominio>.onmicrosoft.com`.
   - ID de cliente y URI de redirección: obtenidos desde la aplicación registrada.
   - Política de autenticación: `B2C_SIGNIN_OR_SIGNUP`.
8. **Pruebas de autenticación.** Se realizaron pruebas con cuentas reales y de prueba para verificar:
   - Redirección correcta al flujo de autenticación.
   - Emisión y validación de tokens en *Power Pages*.
   - Persistencia de sesión.
   - Vinculación con el backend para acceder al historial cifrado y claves personalizadas.

Este proceso permitió desplegar un sistema de autenticación profesional, seguro y sin fricción para el usuario, compatible con el resto de los servicios implementados en la arquitectura.

Como se ilustra en la Figura 3.3, el token generado por Azure AD B2C actúa como elemento de validación de identidad. Este token es recibido por Power Pages y permite identificar de forma segura al usuario, estableciendo una conexión autenticada con las Azure Functions del backend. Esta arquitectura garantiza que solo los usuarios validados puedan acceder a funcionalidades críticas como el guardado de conversaciones cifradas o la clasificación OCEAN.

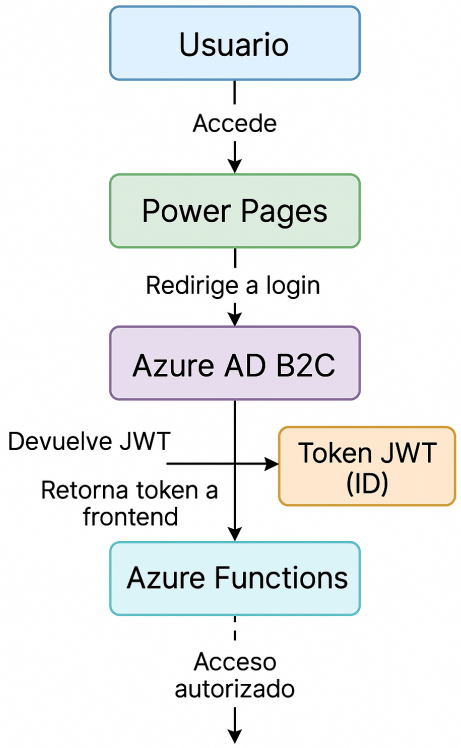

Figura 3.3: Flujo de autenticación: el token generado por Azure AD B2C permite a Power Pages identificar al usuario y comunicarse de forma segura con Azure Functions.

9. **Personalización del portal de inicio de sesión**

   Para proporcionar una experiencia coherente con la identidad visual del proyecto, se diseñó una página de autenticación completamente personalizada. La interfaz *HTML*, junto con sus hojas de estilo *CSS* y los scripts necesarios, se almacenó como contenido estático en un contenedor de *Azure Blob Storage* configurado para acceso público seguro.

   Esta página personalizada fue referenciada desde el flujo de usuario de *Azure AD B2C* como plantilla externa, lo que permitió reemplazar el aspecto visual por defecto por uno adaptado a la identidad gráfica del sistema.

Como se muestra en la Figura 3.4, la pantalla de inicio de sesión presenta una estética diseñada específicamente para reforzar la identidad del proyecto. Incluye un fondo temático digital relacionado con la ciencia de datos, redes y crecimiento organizacional, así como el logotipo corporativo de Innopersona Consultor-IA en la parte superior izquierda. El formulario central permite al usuario autenticarse mediante correo electrónico y contraseña, e incorpora enlaces funcionales para el registro y la recuperación de cuenta.

Figura 3.4: Pantalla de inicio de sesión personalizada cargada desde *Azure Blob Storage*.

La página presenta los siguientes elementos clave:

- Fondo temático digital: relacionado con ciencia de datos, redes y crecimiento organizacional.
- Logo corporativo de Innopersona Consultor-IA: situado en la parte superior izquierda.
- Formulario de autenticación: con campos para correo electrónico y contraseña, y enlaces funcionales de recuperación de contraseña y registro.
- Estilo y estructura responsivos: adaptados para una correcta visualización desde distintos dispositivos.

Este enfoque de personalización permite reforzar la coherencia visual del sistema desde el primer punto de contacto del usuario, mejorando la percepción de profesionalidad, confianza y pertenencia al ecosistema del proyecto. Además, al estar alojado en *Azure Blob Storage*, se facilita su actualización y mantenimiento sin necesidad de desplegar cambios sobre la infraestructura central.

3.2.3. Azure Web PubSub – Comunicación en tiempo real

Para garantizar una experiencia fluida e interactiva en las conversaciones entre el usuario y los asistentes GPT, el sistema utiliza *Azure Web PubSub*[36], un servicio gestionado de publicación/suscripción diseñado para habilitar la comunicación bidireccional en tiempo real a gran escala.

1. **¿Qué es Azure Web PubSub?**
   Azure Web PubSub es una solución basada en *WebSockets* que permite establecer canales de comunicación persistentes entre clientes (como navegadores) y servidores (como *Azure Functions*), sin necesidad de recurrir a sondeos periódicos (*polling*). Esto permite una comunicación más eficiente, con menor latencia y mejor escalabilidad.

2. **Motivaciones para su uso:**
   - Reducir el retardo percibido en la respuesta de los asistentes GPT.
   - Mantener sesiones activas en tiempo real entre frontend y backend.
   - Permitir la visualización progresiva de las respuestas generadas por el modelo (streaming de tokens).
   - Escalar la infraestructura para múltiples usuarios concurrentes sin comprometer el rendimiento.

3. **Funcionamiento en el sistema:**
   - El usuario inicia sesión en la plataforma web y establece una conexión *WebSocket* con el backend mediante *Azure Web PubSub*.
   - Cuando se envía un mensaje a un asistente *GPT*, el backend procesa la solicitud y comienza a generar tokens de forma progresiva.
   - Cada fragmento de respuesta generado se envía de inmediato al cliente a través del canal *WebSocket* abierto.
   - El usuario visualiza la respuesta en tiempo real, línea por línea o palabra por palabra, mejorando la percepción de fluidez conversacional.

4. **Seguridad y autenticación:**
   La integración se realiza mediante el uso de identidades gestionadas y claves temporales generadas desde *Azure Functions*. La autenticación del canal se coordina con *Azure AD B2C*, asegurando que cada sesión *WebSocket* esté asociada a un usuario autenticado.

3.2.4. Azure Functions – Backend serverless

Azure Functions [37] es una plataforma de computación sin servidor (serverless) provista por *Microsoft Azure* que permite ejecutar fragmentos de código independientes en respuesta a eventos, sin necesidad de administrar infraestructura o servidores dedicados. Se basa en un modelo de ejecución orientado a eventos, en el que las funciones se activan mediante desencadenadores (*triggers*) como

solicitudes *HTTP*, colas de mensajes, cambios en el almacenamiento, temporizadores, entre otros.

1. En el contexto del presente proyecto, Azure Functions constituye el núcleo del backend lógico. A través de estas funciones se gestiona la lógica de negocio del sistema, incluyendo:
   - La recepción y procesamiento de las conversaciones generadas por los usuarios.
   - El preprocesamiento de los textos con técnicas de *NLP*.
   - La ejecución del clasificador de personalidad basado en el modelo *OCEAN*.
   - El almacenamiento seguro de los datos sensibles en *Azure Blob Storage*.
   - La gestión de las claves de cifrado mediante *Azure Key Vault*.
   - La generación del hash del perfil psicológico y su registro en una blockchain privada.
   - La coordinación de las llamadas a la API de *OpenAI* para interactuar con los asistentes *GPT*.
2. **Configuración del entorno de despliegue:**

   La aplicación de funciones se ejecuta en *Azure* bajo las siguientes especificaciones:
   - Nombre de la función: `innoyodai`
   - Ubicación geográfica: `West Europe`
   - Sistema operativo: `Linux`
   - Pila de ejecución: `Python 3.11`
   - Modelo de publicación: `Código`
   - Plan de App Service: `Basic B1`
   - Dirección URL pública: `https://innoyodai.azurewebsites.net`
   - Application Insights: Habilitado para monitorización y análisis.
3. **Desarrollo y modularidad:**

   Durante las fases iniciales del proyecto, se optó por implementar múltiples funciones especializadas para tareas específicas (almacenamiento, cifrado, clasificación, registro en blockchain, etc.). Esta decisión favoreció una arquitectura altamente modular, desacoplada y fácilmente escalable.

   Cada función fue diseñada siguiendo el principio de responsabilidad única y utilizando colas de mensajes para coordinar flujos asincrónicos entre funciones. Esta arquitectura está documentada con detalle en el D: Manual técnico de funciones *Azure*, donde se describen individualmente los

objetivos, desencadenadores, entradas, salidas y ejemplos de uso de cada función implementada.

4. **Motivación de la elección:**

   El uso de Azure Functions fue recomendado por el equipo de *Innopersona* debido a su estrecha integración con el ecosistema *Microsoft*. Esto ha permitido construir una solución completamente integrada con otros servicios clave como *Power Pages* (frontend), *Azure AD B2C* (autenticación), *Key Vault* (seguridad) y *Blob Storage* (almacenamiento).

   Además, el modelo de facturación basado en consumo ha resultado ideal para una aplicación con cargas de trabajo irregulares, donde la ejecución de funciones depende de la interacción del usuario. Esto permite optimizar los costes frente a una arquitectura tradicional con servidores dedicados en ejecución continua.

5. **Ventajas principales:**

   - **Escalabilidad automática:** Las funciones se escalan según la demanda sin necesidad de configuración adicional.
   - **Desacoplamiento:** Cada función representa una unidad lógica independiente, facilitando el mantenimiento y la evolución del sistema.
   - **Integración nativa:** Compatible con servicios Azure como AD B2C, Blob Storage, Key Vault, Application Insights, etc.
   - **Desarrollo rápido:** Posibilidad de desplegar nuevas funcionalidades de forma ágil, ideal para entornos de experimentación y prototipado.

6. **Limitaciones encontradas durante el desarrollo:**

   - **Gestión de estados complejos:** La naturaleza *stateless* (sin estado) obliga a diseñar mecanismos externos para el seguimiento de contexto, como el uso de colas o almacenamiento intermedio.
   - **Debugging complejo:** Las funciones asincrónas dificultan el seguimiento de errores cuando se encadenan múltiples eventos.
   - **Límites de ejecución:** El tiempo máximo de ejecución en el plan de consumo estándar es de 5 minutos, lo cual puede ser insuficiente en operaciones que combinan múltiples servicios (por ejemplo, cifrado + almacenamiento + blockchain).

3.2.5. Azure Application Insights – Monitorización del sistema

Para monitorizar el comportamiento del sistema y registrar información útil durante su ejecución, se ha utilizado **Azure Application Insights** [38], una solución de observabilidad integrada dentro de *Azure Monitor*. Esta herramienta permite recopilar métricas, trazas, registros de telemetría y excepciones en tiempo real, lo cual resulta especialmente útil en entornos serverless como el de este proyecto.

Application Insights está conectado a todas las **Azure Functions** que componen el backend, lo que permite registrar automáticamente eventos relevantes, como invocaciones exitosas, errores, tiempos de respuesta, e información detallada sobre el rendimiento.

1. **Métricas personalizadas**

   Además del registro automático de eventos, se ha añadido telemetría personalizada para capturar los siguientes aspectos:

   - **Duración de uso por usuario:** mediante eventos emitidos desde el frontend cuando se inicia y termina una sesión de usuario.
   - **Tokens procesados:** se registra el número de tokens intercambiados en cada conversación con un asistente, tanto en el mensaje del usuario como en la respuesta del modelo *GPT*.
   - **Interacciones por asistente:** se identifica cuál de los asistentes del modelo ADSM ha sido utilizado, permitiendo hacer un seguimiento por herramienta.
   - **Errores y tiempos de espera:** se notifican errores internos, problemas con llamadas a la *API de OpenAI* o tiempos de ejecución excesivos en funciones específicas.

2. **Visualización y consultas**

   Toda esta información puede consultarse desde el portal de *Azure*, a través de **paneles personalizados (dashboards)** o del lenguaje de consulta Kusto (KQL), lo que permite generar informes detallados sobre el uso del sistema o detectar cuellos de botella. Esta visibilidad ha sido fundamental para iterar en el diseño de las funciones, optimizar el rendimiento del sistema y entender los patrones de uso reales por parte de los usuarios.

3. **Justificación de su uso**

   El uso de *Application Insights* se alinea con la estrategia general del proyecto de mantenerse dentro del ecosistema *Microsoft Azure*, lo que facilita la integración, reduce la complejidad y garantiza un nivel alto de compatibilidad. Además, al tratarse de un entorno *serverless*, es especialmente útil contar con un sistema de monitorización automatizado y detallado para detectar cualquier fallo sin necesidad de levantar servidores.

3.2.6. OpenAI, Modelos de Lenguaje y Recuperación Semántica (file-search + vector store)

OpenAI es una organización líder en el desarrollo de modelos de lenguaje generativo basados en redes neuronales profundas. En este proyecto se utiliza su plataforma de modelos *GPT (Generative Pre-trained Transformer)*, que permite generar texto coherente, mantener conversaciones y comprender el contexto semántico de forma avanzada.

1. GPT es una familia de modelos de gran escala basados en la arquitectura *Transformer*, entrenados con grandes cantidades de datos textuales. Estos modelos son capaces de realizar tareas como traducción, clasificación, generación de contenido y, en nuestro caso, acompañamiento conversacional adaptado al usuario.

   Una de las funcionalidades clave del sistema es la capacidad de los asistentes GPT para consultar documentos externos como contexto semántico. Esto se consigue gracias a las capacidades combinadas de **file-search** y los **vector stores**, proporcionadas por la API de OpenAI [39].

   El mecanismo de `file-search` permite a un asistente acceder a fragmentos de documentos que han sido previamente procesados y almacenados como vectores en una base de datos semántica. Esta búsqueda [40] no se basa en coincidencias literales de texto, sino en la **similitud entre embeddings**, lo que permite detectar relevancia contextual aunque no haya coincidencia exacta de términos. Esta operación se realiza consultando un **vector store**, que representa cada fragmento textual como un vector numérico en un espacio multidimensional [41].

   En este sistema, cada asistente GPT está vinculado a un **vector store específico**, en el que se ha indexado el protocolo del modelo ADSM correspondiente a su función. Así, por ejemplo, el asistente del módulo IPP accede a un vector store diferente del asistente del módulo AP. Cuando el usuario envía una consulta, el sistema extrae automáticamente los fragmentos más relevantes (top-k) del vector store y los incorpora como parte del contexto antes de que el modelo genere la respuesta. Esto permite mantener conversaciones coherentes, especializadas y fieles al contenido original del protocolo, sin necesidad de entrenar un modelo nuevo para cada asistente.

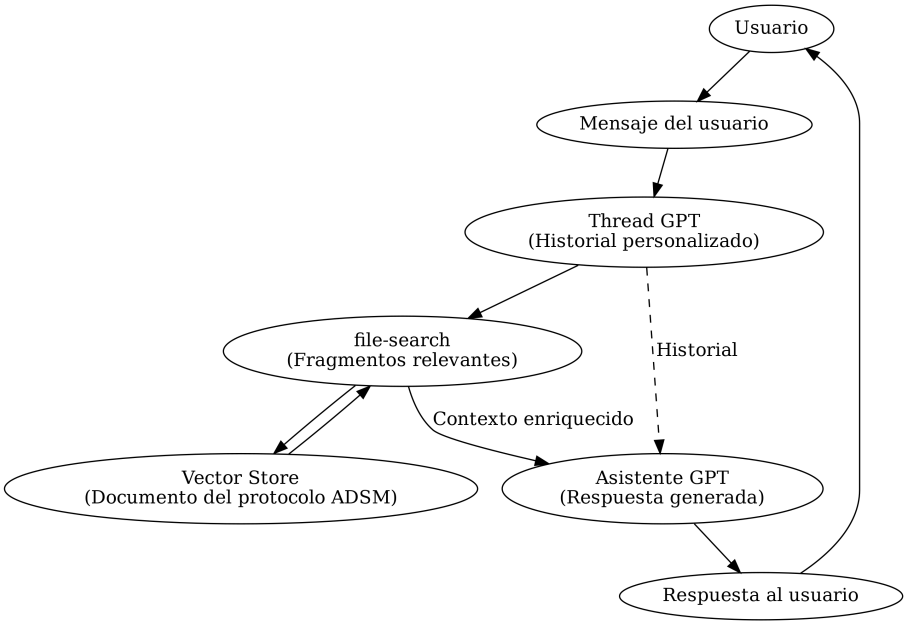

Figura 3.5: Flujo de recuperación semántica usando `file-search` y `vector store` en asistentes GPT.

Además del contexto enriquecido con `file-search`, el comportamiento de los modelos GPT puede ajustarse dinámicamente mediante varios hiperparámetros clave [42]:

- **Temperature:** controla el grado de aleatoriedad en la generación de texto. En este proyecto se emplea una temperatura baja (0.0–0.4) para fomentar la variabilidad manteniendo coherencia.
- **Top-k:** indica cuántas opciones de tokens se consideran en cada paso. En el uso con vector stores, también se refiere al número de fragmentos más relevantes que se recuperan.
- **Top-p** (nucleus sampling): selecciona tokens cuya probabilidad acumulada supera un umbral p. En algunos asistentes se usa como alternativa para balancear precisión y creatividad.
- **Frequency Penalty y Presence Penalty:** se han utilizado en pruebas preliminares para reducir repeticiones y mejorar la fluidez del discurso.

2. **Diseño y control conversacional mediante instrucciones**

   Una parte fundamental del comportamiento de los asistentes no depende del modelo en sí, sino de las instrucciones que se le proporcionan en el momento de invocarlo. En este sistema, cada asistente GPT está diseñado mediante un conjunto exhaustivo de instrucciones que definen su identidad, rol, tono, estructura de interacción, restricciones y pasos de validación.

   Estas instrucciones están documentadas íntegramente en E, y son funda-

mentales para:

- Limitar la actuación del asistente inicial exclusivamente a la derivación, impidiendo cualquier intento de resolver problemas o aplicar herramientas por sí mismo.
- Garantizar que los asistentes especializados sigan paso a paso los protocolos definidos, como el de Algoritmización Personal (AP), validando cada respuesta antes de avanzar.
- Asegurar que cada asistente mantenga su identidad incluso si hay otros mensajes en el historial generados por diferentes roles.

Este enfoque de diseño conversacional programado ha sido clave para lograr una interacción fiable, acotada y personalizada.

3. **Evolución del sistema:**

   Inicialmente se desarrolló un prototipo sin conexión a vector stores ni consultas contextuales. El asistente respondía únicamente basándose en las instrucciones cargadas. Posteriormente, se incorporó la integración con `file-search` mediante los **vector stores de OpenAI**, lo que permitió a cada asistente acceder a los documentos relevantes de su protocolo ADSM sin tener que pasar los documentos como contexto en cada mensaje.

   Esta mejora supuso un cambio de arquitectura: ahora cada usuario tiene un `thread` y un `vector store` propios, permitiendo conversaciones personalizadas, persistentes y basadas en el historial.

4. **Modelos utilizados:**

   Durante la fase inicial se empleó el modelo **gpt-3.5-turbo**, debido a dos factores clave: su menor coste y su menor predisposición a adoptar una actitud proactiva no deseada, que en ocasiones llevaba a los asistentes a desviarse del protocolo establecido en sus instrucciones.

   Posteriormente, se adoptó el modelo **gpt-4o** (GPT-4 Omni), que combina un mayor rendimiento conversacional, mayor comprensión contextual y compatibilidad con `file-search`, manteniendo un coste razonable. A diferencia de versiones anteriores como gpt-4.0, el modelo gpt-4o sí ofrece soporte completo para recuperación semántica desde documentos.

   Según la documentación oficial [43], los costes actuales por millón de tokens son:

   - **gpt-3.5-turbo:** 0.50 $pormillndetokensdeentraday1{,}50$ por millón de salida.
   - **gpt-4-0125-preview / gpt-4o:** 10.00 $pormillndetokensdeentraday30{,}00$ por millón de salida.

5. **Mejora iterativa mediante evaluación automatizada:**

   Para optimizar la calidad de las respuestas generadas por los asistentes, se ha desarrollado un **asistente evaluador** especializado en analizar las sali-

das de otros asistentes. Este evaluador compara la respuesta generada con los objetivos del protocolo ADSM correspondiente, evaluando su pertinencia, claridad, coherencia y adecuación al contexto emocional del usuario.

Este proceso de autoevaluación ha permitido iterar sobre las instrucciones y el contenido de los vector stores de cada asistente, mejorando gradualmente su desempeño sin intervención humana directa.

6. **Motivos de selección:**
   - Disponibilidad de capacidades avanzadas de *NLP* listas para producción.
   - Facilidad de integración vía *API REST*.
   - Capacidad de generar respuestas coherentes, empáticas y adaptadas al contexto emocional.
   - Compatibilidad con herramientas de recuperación semántica (`file-search` y vector stores).
   - Control total mediante instrucciones estructuradas (E).

3.2.7. Asistente evaluador de respuestas

Durante la fase de desarrollo e iteración del sistema conversacional ADSM, se implementó un **asistente evaluador automático** con el propósito de verificar que las respuestas generadas por los asistentes *GPT* cumplan fielmente con las instrucciones del protocolo correspondiente. Esta herramienta actúa como un mecanismo de control de calidad, aportando evaluaciones estructuradas y objetivas que permiten detectar desviaciones metodológicas y errores en la interacción.

A diferencia de un proceso de validación meramente manual, el evaluador automatizado se basa en un modelo *GPT* especializado que analiza las respuestas emitidas por los asistentes en función del protocolo activo, el mensaje original del usuario y los criterios definidos por el autor del sistema ADSM.

1. **Aplicación principal:** El uso del asistente evaluador ha sido especialmente relevante en el caso del **asistente inicial**, responsable de derivar al usuario hacia la herramienta ADSM más adecuada (MDS-YO, IPP, PCN, Simulación Virtual Mental (SVM), AP, VP). Dado que este primer paso condiciona el resto del proceso, se requiere un control riguroso de su comportamiento. Por ello, se han utilizado evaluaciones automáticas recurrentes para comprobar:
   - La **coherencia** de la respuesta con el protocolo de derivación ADSM.
   - La **adecuación del estilo de comunicación** (empatía, claridad, rol correcto).
   - La **presencia de errores comunes**, como anticipar instrucciones, explicar herramientas indebidamente o derivar sin justificación.

2. **Estructura de la evaluación:** Cada análisis emitido por el evaluador se presenta con una estructura fija, facilitando su revisión y comparación. Incluye:

   a) Coherencia con el protocolo.

   b) Derivación correcta.

   c) Errores detectados.

   d) Estilo de comunicación.

   e) Sugerencias de mejora.

3. **Impacto en el diseño:** El evaluador ha sido fundamental para iterar rápidamente sobre las instrucciones del asistente principal, corregir desviaciones antes de su despliegue y alinear el comportamiento del sistema con los principios definidos por el modelo ADSM. Su implementación ha permitido automatizar el proceso de ajuste, reducir el tiempo de prueba y garantizar mayor fiabilidad metodológica.

4. **Supervisión experta en asistentes específicos:** Cabe destacar que los asistentes especializados en cada herramienta ADSM (como el IPP, el VP o el MDS-YO) han sido revisados manualmente por Gerardo Tudurí, creador del modelo, quien ha validado directamente la adecuación de cada implementación a su protocolo correspondiente.

Nota: Para una visión detallada del funcionamiento del evaluador y de los tipos de respuesta analizados, se incluyen **ejemplos reales y sus correspondientes análisis** en el **Anexo B**, bajo el título "Evaluaciones automáticas de respuestas del asistente inicial". Allí se presentan distintos escenarios: desde derivaciones correctas justificadas hasta errores frecuentes detectados durante el entrenamiento.

3.2.8. Azure Blob Storage + Azure Key Vault – Almacenamiento seguro y gestión de claves

Uno de los pilares fundamentales del sistema es la protección de los datos sensibles del usuario, en particular las conversaciones mantenidas con los asistentes y la clasificación de personalidad generada. Para garantizar esta protección se ha optado por una arquitectura compuesta por **Azure Blob Storage** como sistema de almacenamiento y **Azure Key Vault** para la gestión segura de claves de cifrado.

1. Azure Blob Storage [44] es un sistema de almacenamiento de objetos en la nube diseñado para almacenar grandes cantidades de datos no estructurados. Permite organizar la información en contenedores y blobs, soportando distintos tipos de acceso y permisos. En este proyecto, se utiliza el tipo *Append Blob* para registrar de forma secuencial los mensajes de cada usuario en un único archivo, minimizando la sobrecarga de escritura y optimizando el acceso posterior.

2. Azure Key Vault [45] es un servicio de gestión de secretos, claves y certificados que permite almacenar y acceder a claves criptográficas de forma segura. En nuestro sistema, se utiliza para almacenar una **clave única de cifrado por usuario** (AES-GCM), la cual se genera la primera vez que se guarda una conversación y se reutiliza en accesos posteriores.

3. Formato de almacenamiento de mensajes

   Antes de que un mensaje sea almacenado, se le añade el rol que identifica su procedencia: `USER:` o `ASSISTANT:`, seguido del contenido textual. Esto permite mantener el contexto completo de la conversación en un único archivo legible y procesable, facilitando el posterior análisis lingüístico.

4. Necesidad de almacenamiento para clasificación OCEAN

   El perfil de personalidad del usuario se genera mediante el análisis semántico y lingüístico de sus conversaciones. Por tanto, **almacenar el historial completo es esencial** para que el sistema pueda aplicar técnicas de NLP y extraer las dimensiones del modelo OCEAN. Sin este historial, no sería posible realizar una clasificación fiable, ya que se perdería el contexto y la riqueza lingüística necesaria para el modelo.

5. Optimización mediante cola de ejecución

   Dado que el proceso de cifrado, recuperación de claves y escritura en el blob puede ser costoso computacionalmente, se ha optado por implementar una **Azure Storage Queue** como sistema de desacoplamiento. El proceso es el siguiente:

   a) Cuando finaliza una interacción completa, se envía un mensaje a la cola de ejecución.

   b) Una **Azure Function** suscrita a esta cola se encarga de procesar en segundo plano cada mensaje recibido.

   c) Esta función identifica el usuario, recupera o genera su clave desde *Key Vault*, cifra el contenido y lo almacena en el append blob correspondiente.

   Este enfoque asíncrono permite mejorar el rendimiento general del sistema, evitando bloquear el hilo principal de interacción mientras se guarda el historial.

6. Motivación de la elección:

   Se ha optado por esta combinación de servicios por su alta integración nativa con otras soluciones de Azure (Functions, AD B2C, etc.), su escalabilidad automática y su cumplimiento de normativas de seguridad como ISO/IEC 27001, SOC 2 y RGPD.

   Además, permite implementar una política de **seguridad por diseño**, en la que los datos sensibles están cifrados en todo momento y sólo el usuario tiene acceso lógico a su información.

7. Ventajas principales:
   - **Seguridad avanzada:** Las claves de cifrado utilizadas siguen el estándar **AES (Advanced Encryption Standard)**, un algoritmo simétrico aprobado por el NIST para el cifrado de datos sensibles a nivel gubernamental y empresarial [46]. Estas claves nunca se almacenan junto a los datos cifrados y sólo se accede a ellas desde identidades autorizadas, gestionadas a través de *Azure Key Vault*.
   - **Alta disponibilidad:** Azure garantiza una durabilidad de datos del 99.999999999 %.
   - **Escalabilidad transparente:** Capaz de soportar millones de archivos cifrados sin rediseñar la arquitectura.
   - **Control por usuario:** Cada usuario tiene su clave y su blob, lo que facilita la trazabilidad y eventual revocación de acceso.
8. Limitaciones encontradas:
   - **Sobrecarga inicial de configuración:** Requiere definir políticas de acceso, roles de identidad administrada y estructuras de contenedores adecuados.
   - **Latencia en operaciones múltiples:** Acceder a claves + cifrado + escritura en blob puede introducir ligeros retardos si no se optimiza correctamente.
   - **Coste adicional por operaciones de lectura/escritura:** El modelo de precios se basa en volumen almacenado y número de transacciones, por lo que hay que balancear eficiencia y seguridad.
9. Comentarios durante el desarrollo:

   Durante las primeras fases, se almacenaban las conversaciones sin cifrar, directamente en texto plano, lo cual fue rápidamente descartado por los riesgos asociados. El nuevo sistema basado en cifrado con clave única por usuario se consideró mucho más sólido en términos de privacidad.

   También se decidió prescindir del uso de cadenas de conexión, optando por **identidades administradas** [47] para acceder tanto a *Key Vault* como a *Blob Storage*, evitando la exposición de credenciales y simplificando la gestión de permisos.

3.2.9. Azure Machine Learning – Entrenamiento del modelo de personalidad OCEAN

Azure Machine Learning (Azure ML) es la plataforma en la nube utilizada para llevar a cabo el entrenamiento y la evaluación del modelo de predicción de personalidad propuesto en este trabajo. Esta herramienta, ofrecida por Microsoft, permite desarrollar, ejecutar y gestionar experimentos de aprendizaje automático en un entorno escalable y completamente gestionado, con integración nativa

con otros servicios del ecosistema Azure, como Azure Functions, Azure Key Vault y Azure Blob Storage.

1. En este proyecto, *Azure ML* se ha empleado para entrenar un modelo de regresión multivariada capaz de predecir los cinco rasgos del modelo OCEAN (Apertura, Responsabilidad, Extraversión, Amabilidad y Neuroticismo), a partir de los historiales conversacionales preprocesados de los usuarios. Estos historiales son recogidos y tratados tras alcanzar un umbral mínimo de interacciones, tal como se detalla en el anexo C.

2. Funcionalidad dentro del sistema:

   - Lectura del conjunto de datos conversacional almacenado en *Azure Blob Storage* y ya preprocesado.
   - Entrenamiento de un modelo basado en Transformers (`XLM-Roberta-base`) adaptado para tareas de regresión continua multietiqueta (*multi-output regression*).
   - Evaluación del modelo mediante varias métricas estándar: *mean squared error*, *mean absolute error*, coeficiente de determinación R^2 y *macro F1-score* tras umbralización.
   - Registro y almacenamiento del modelo entrenado como artefacto para su posterior invocación desde el entorno de inferencia mediante Azure Functions.

3. Motivaciones para su elección:

   - Integración directa con los servicios ya desplegados en la solución, facilitando la automatización del ciclo de vida del modelo.
   - Entorno gestionado y escalable, capaz de ejecutar entrenamientos en paralelo y utilizar recursos como GPU bajo demanda.
   - Soporte completo para bibliotecas especializadas utilizadas en este proyecto, como `transformers`, `scikit-learn` y `spaCy`.
   - Registro detallado de métricas, versiones y configuraciones para asegurar la trazabilidad de los experimentos.

4. Limitaciones encontradas durante el desarrollo:

   - Costes asociados al uso de recursos avanzados como instancias con GPU, especialmente durante las fases de prueba y ajuste de parámetros.
   - Curva de aprendizaje inicial en la configuración de entornos personalizados y gestión de pipelines.
   - Necesidad de planificar adecuadamente permisos y recursos en la suscripción de *Azure* para evitar bloqueos o sobrecostes.

5. Comentario del desarrollo:

El dataset original *Big5-chat*[1] no contaba con etiquetas que permitieran el entrenamiento supervisado. Por ello, se utilizó el modelo de lenguaje avanzado **ChatGPT-4o** para generar automáticamente las etiquetas que clasifican cada texto en función de los cinco rasgos OCEAN, construyendo así un dataset válido para entrenamiento.

Este proceso de etiquetado automático fue realizado mediante scripts que consultaron ChatGPT-4o para cada ejemplo, evitando la necesidad de etiquetado manual y facilitando la generación de un subconjunto de alta calidad.

Debido a limitaciones técnicas y económicas —como el elevado coste de procesar grandes volúmenes de texto con modelos avanzados y la pérdida de contexto en entradas extensas— no se utiliza ChatGPT-4o para realizar la inferencia en producción. En su lugar, se entrenó un modelo Transformer (`XLM-Roberta-base`) sobre una tarea de regresión multivariada, más eficiente y reproducible, utilizando las herramientas de *Azure ML*.

6. Evaluación de los entrenamientos:

   Se llevaron a cabo dos entrenamientos con resultados insatisfactorios:

   Primer entrenamiento (500 ejemplos):

   - MAE: 60.12
   - MSE: 3846.45
   - R^2: -25.09
   - F1 Macro: 1.00 (no representativo)

   Segundo entrenamiento (1341 ejemplos):

   - MAE: 56.50
   - MSE: 3273.85
   - R^2: -47.71
   - F1 Macro: 1.00 (no representativo)

   Ambos entrenamientos reflejan una pobre capacidad del modelo para generalizar, con coeficientes de determinación negativos y errores altos.

7. Posibles causas del bajo rendimiento:

   - Tamaño del dataset aún limitado.
   - Ruido en el etiquetado automático.
   - Uso de un modelo demasiado complejo para la cantidad de datos disponibles.

[1] El dataset Big5-chat consiste en textos conversacionales sin etiquetar con las dimensiones del modelo de personalidad OCEAN. Para más información, véase la referencia [**?**].

- Evaluación deficiente en métricas categóricas no adecuadas a regresión.

8. Mejoras propuestas (no implementadas por limitación de tiempo):
   - Ampliar y refinar el dataset.
   - Entrenar modelos separados por rasgo.
   - Usar modelos más ligeros.
   - Revisar la distribución de etiquetas.
9. Conclusión:

   Aunque el rendimiento no fue el esperado, el pipeline completo de etiquetado, entrenamiento y despliegue fue validado funcionalmente. Este trabajo constituye una base sólida para futuras iteraciones más robustas y precisas.

3.2.10. Red Blockchain – Hyperledger Besu en Azure VM

Para garantizar la inmutabilidad y trazabilidad de los perfiles psicológicos generados, se ha desplegado una red blockchain privada basada en **Hyperledger Besu** [48], utilizando el mecanismo de consenso **IBFT 2.0** (Istanbul Byzantine Fault Tolerance) sobre máquinas virtuales Linux en Microsoft Azure [49]. Esta red opera en un entorno cerrado, sin conexión a Internet pública, y está diseñada para proporcionar control absoluto, privacidad y seguridad en la gestión de datos.

1. Motivaciones para el uso de blockchain privada:
   - **Inmutabilidad y trazabilidad:** Las clasificaciones OCEAN se registran como transacciones inmutables.
   - **Privacidad:** Al tratarse de una red **PoA (Proof of Authority)** privada, no hay riesgo de exposición en redes públicas. Este mecanismo de consenso se basa en un conjunto de nodos validadores autorizados que firman los bloques, eliminando la necesidad de minería y permitiendo un control completo sobre los participantes de la red. Es un modelo especialmente utilizado en entornos empresariales donde se requiere gobernanza y rendimiento predecible [50].
   - **Control total:** Se puede modificar la frecuencia de bloques, permisos, nodos validadores y configuración completa de la red.
2. Consenso PoA (Proof of Authority):

   La red implementa un modelo de consenso **Proof of Authority** (PoA), en el que un conjunto fijo de nodos validadores autorizados es responsable de proponer y validar bloques. Este modelo fue seleccionado por los siguientes motivos:

- **Bajo consumo de recursos:** A diferencia de modelos como Proof of Work, no requiere cómputo intensivo.
- **Finalidad rápida:** Las transacciones tienen confirmación inmediata (sin forks).
- **Control institucional:** Los validadores pueden ser gestionados directamente por la organización, garantizando seguridad y auditoría.

En concreto, se ha utilizado el mecanismo **IBFT 2.0**, una variante tolerante a fallos bizantinos que permite tolerar nodos maliciosos o caídos (hasta $\lfloor (n-1)/3 \rfloor$) y ofrece garantías de seguridad y *liveness* para entornos privados [51]. El término *liveness* hace referencia a la capacidad del sistema para continuar avanzando —es decir, garantizar que se sigan proponiendo y finalizando bloques válidos en la cadena incluso en presencia de fallos o nodos inactivos. Esta propiedad, junto con la *seguridad* (que garantiza que los bloques acordados no se revertirán), es fundamental en protocolos de consenso tolerantes a fallos bizantinos.

3. Configuración técnica de la red:
   - **Hyperledger Besu v23.10.1** en 4 nodos validadores [48].
   - **blockperiodseconds:** 300 (bloque cada 5 minutos para minimizar consumo de disco).
   - **Máquina virtual Azure:** Ubuntu 22.04, tipo B1ms (1 vCPU, 2 GiB RAM), IP estática [49].
   - **Puertos:** abiertos los necesarios para *p2p* (30303 TCP/UDP), JSON-RPC (8545), WS (8546).
   - **Servicio systemd:** daemon personalizado para ejecutar *besu* en segundo plano.
4. Archivos de configuración clave:
   - `genesis.json`: especifica la configuración del protocolo de consenso **IBFT 2.0 (Istanbul Byzantine Fault Tolerance)**, incluyendo parámetros como los tiempos entre bloques, la dificultad, los nodos validadores iniciales y las asignaciones de cuentas. IBFT 2.0 es una variante del consenso Byzantine Fault Tolerant diseñada para redes privadas de confianza parcial, permitiendo un alto rendimiento y finalización instantánea de bloques [52].

```
1         {
2         "config": {
3             "chainId": 2025,
4             "constantinopleBlock": 0,
5             "ibft2": {
6             "blockperiodseconds": 300,
7             "epochlength": 30000,
8             "requesttimeoutseconds": 10
9             }
10        },
```

```
        "nonce": "0x0",
        "timestamp": "0x6629fd6d",
        "extraData": "0x<valores-anonimizados>",
        "gasLimit": "0x1fffffffffffff",
        "difficulty": "0x1",
        "mixHash": "0x63746963616c2d627970617373",
        "coinbase": "0x0000000000000000000000000000000000000000",
        "alloc": {
            "0x0000000000000000000000000000000000000001": {
            "balance": "0x1"
            },
            "0x<cuenta-anonimizada>": {
            "balance": "0xad78ebc5ac6200000"
            }
        },
        "number": "0x0",
        "gasUsed": "0x0",
        "parentHash": "0
    ↪ x0000000000000000000000000000000000000000000000000000000000000000
    ↪ "
        }

```

Listing 3.1: Contenido del archivo `genesis.json` utilizado en la red privada

- `static-nodes.json`: lista de nodos validadores con sus enode (no se incluye por contener direcciones sensibles).
- `besu.service`: unidad systemd para arranque automático con logs persistentes.

5. Integración desde Azure Functions:

   Las Azure Functions encargadas de registrar los datos clasificatorios OCEAN acceden a la red mediante **JSON-RPC** en el puerto `8545`, que es el canal estándar de comunicación entre clientes Ethereum y aplicaciones externas. JSON-RPC es un protocolo ligero y remoto basado en JSON que permite a las aplicaciones interactuar con nodos blockchain para enviar transacciones, consultar bloques, o recuperar estados de contrato [53]. Para garantizar que las transacciones firmadas sean compatibles con la red **PoA** (que incluye campos como `extraData` específicos), se ha configurado el cliente `Web3.py` con el middleware adecuado.

   ```
   from web3.middleware import geth_poa_middleware
   w3.middleware_onion.inject(geth_poa_middleware, layer=0)
   ```

   Este middleware es imprescindible en entornos *PoA* como *Besu* o *Geth clique*, ya que permite ignorar la diferencia en la estructura del campo extraData del bloque (usualmente más largo y específico que en redes públicas como *Ethereum mainnet*) [54].

6. Gestión de claves de usuario y transacciones:

   - Cada usuario tiene una clave Ethereum generada con `eth_account`.

- La clave privada se cifra y almacena en **Azure Key Vault** [55].
- Las funciones en Azure recuperan la clave, firman localmente la transacción y la envían vía *JSON-RPC*.

7. Ventajas de esta arquitectura:
   - **Costo casi nulo:** no hay fees, minado ni uso de tokens.
   - **Trazabilidad total:** cada clasificación queda asociada a una transacción irreversible.
   - **Escalabilidad interna:** se pueden añadir nodos o ajustar frecuencia de bloques según carga.
   - **Desacoplamiento:** la VM de blockchain se comunica con Azure Functions sin exponer la red.

Estas ventajas están alineadas con las mejores prácticas recomendadas para blockchain empresarial [56].

Capítulo 4

Arquitectura del Sistema

4.1. Introducción

Esta sección presenta la arquitectura de alto nivel del sistema desarrollado para ofrecer soporte emocional basado en inteligencia artificial. El diseño arquitectónico se fundamenta en los principios de *modularidad*, *escalabilidad*, *seguridad*, *trazabilidad* y *facilidad de mantenimiento*. Se ha optado por una organización en subsistemas funcionales que colaboran entre sí mediante servicios en la nube, adoptando una aproximación orientada a eventos y funciones *serverless*. Esta arquitectura busca garantizar una experiencia fluida para el usuario final, al tiempo que asegura la protección y trazabilidad de los datos personales mediante cifrado y *blockchain*.

4.2. Principios de Diseño

El sistema ha sido concebido con los siguientes principios arquitectónicos:

- **Modularidad:** La división en subsistemas permite desarrollar, probar y mantener cada parte del sistema de forma independiente.
- **Escalabilidad:** El uso de Azure Functions y servicios *cloud* nativos permite escalar automáticamente en función de la carga.
- **Seguridad y privacidad:** Toda la información sensible se cifra individualmente por usuario. Las claves están gestionadas desde *Azure Key Vault*.
- **Interoperabilidad:** El sistema puede adaptarse fácilmente a nuevos protocolos ADSM o mejoras en modelos de lenguaje sin rediseñar la infraestructura.
- **Trazabilidad y control del usuario:** El perfil generado se almacena de forma cifrada y su integridad se garantiza mediante una red blockchain privada.

4.3. Visión General de la Arquitectura

La Figura 4.1 muestra la arquitectura general del sistema, que representa la interacción entre los distintos subsistemas: desde la interfaz de usuario y la autenticación hasta el procesamiento conversacional, el análisis de personalidad, el almacenamiento y la trazabilidad mediante blockchain. Esta representación permite visualizar cómo fluyen los datos y cómo se distribuyen las responsabilidades a través de la infraestructura desplegada en Azure.

4.4. Vista de Componentes

A continuación se describen los componentes principales:

- **Frontend (Power Pages):** Plataforma web desde la que el usuario se registra, conversa con los asistentes GPT y descarga su perfil psicológico.
- **Azure AD B2C:** Servicio de autenticación que permite la gestión de identidades y sesiones seguras mediante estándares como OAuth 2.0 y OpenID Connect.
- **Azure Functions:** Backend sin servidor que orquesta las tareas de procesamiento, almacenamiento seguro y comunicación con blockchain y OpenAI.
- **OpenAI API:** Proveedor de los modelos de lenguaje GPT-4, utilizados por los asistentes especializados según los protocolos del modelo ADSM.
- **Procesador NLP:** Componente que limpia el texto usando técnicas como lematización, tokenización y eliminación de *stopwords* con librerías como spaCy.
- **Clasificador OCEAN:** Modelo entrenado mediante *transformers* que devuelve un perfil de personalidad estructurado en base a las dimensiones del modelo OCEAN.
- **Azure Blob Storage + Key Vault:** Almacenamiento seguro de las conversaciones, cifradas con AES-GCM usando claves únicas por usuario.
- **Blockchain privada (Hyperledger Besu):** Registro inmutable del *hash* del perfil OCEAN, garantizando la autenticidad del mismo.

4.5. Flujos de Datos y Comunicación

El flujo principal de funcionamiento es el siguiente:

1. El usuario accede desde Power Pages y se autentica con Azure AD B2C.
2. Se establece una conversación con un asistente GPT configurado según un protocolo ADSM específico.
3. Las interacciones se procesan y limpian en segundo plano mediante una Azure Function.

4. El contenido se envía al clasificador OCEAN, que genera un perfil psicológico en JSON.
5. El perfil se cifra con una clave única y se almacena en Azure Blob Storage.
6. Paralelamente, se calcula su hash y se registra en la red blockchain privada.
7. El usuario puede consultar su perfil o exportarlo directamente desde la interfaz web.

4.6. División en Subsistemas

El sistema se estructura en los siguientes subsistemas funcionales:

- **Subsistema de Interfaz de Usuario:** Plataforma accesible y responsiva implementada con Power Pages, donde el usuario realiza toda la interacción.
- **Subsistema de Autenticación y Gestión de Usuarios:** Implementado con Azure AD B2C, se encarga del registro, autenticación segura y gestión de identidades.
- **Subsistema de Asistentes GPT:** Aloja los asistentes conversacionales configurados para ejecutar las distintas fases del modelo ADSM mediante instrucciones personalizadas.
- **Subsistema de Análisis y Clasificación de Personalidad:** Procesa las conversaciones con técnicas NLP y aplica el modelo OCEAN para generar un perfil.
- **Subsistema de Almacenamiento Seguro:** Cifra y guarda las conversaciones y resultados usando Azure Blob Storage y Key Vault, respetando la privacidad y acceso por usuario.
- **Subsistema de Blockchain Privada:** Almacena el hash de cada perfil OCEAN en una red Hyperledger Besu, asegurando su trazabilidad y autenticidad.

4.7. Modelo C4 de la Arquitectura

Con el objetivo de representar la arquitectura desde distintos niveles de abstracción, se ha utilizado el modelo C4.

4.7.1. Nivel 1: Diagrama de Contexto

La Figura 4.1 representa una visión de alto nivel del sistema. En él se muestran los actores externos (usuarios, *Innopersona* como entidad promotora, y servicios como *OpenAI* o *Azure*) y cómo interactúan con el sistema. El objetivo principal es facilitar el autoconocimiento mediante conversaciones seguras con asistentes inteligentes.

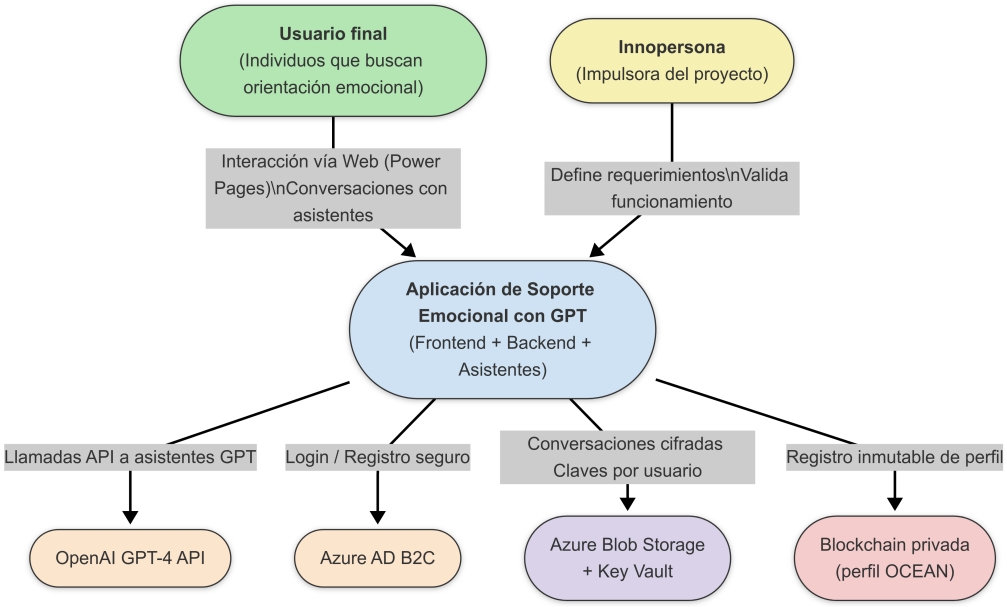

Figura 4.1: Diagrama de contexto.

4.7.2. Nivel 2: Diagrama de Contenedores

La arquitectura se basa en contenedores desacoplados desplegados en la nube:

- **Frontend (Power Pages):** Punto de entrada del usuario.
- **Azure AD B2C:** Autenticación y control de acceso.
- **Azure Functions:** Lógica del backend distribuido.
- **OpenAI API:** Procesamiento de lenguaje natural con *GPT*.
- **Preprocesador NLP + Clasificador OCEAN:** Análisis conversacional.
- **Azure Blob Storage + Key Vault:** Cifrado y almacenamiento seguro.
- **Blockchain privada:** Registro inmutable de perfiles.

La Figura 4.2 muestra gráficamente la interacción entre los diferentes contenedores del sistema. El usuario accede al frontend desarrollado en *Power Pages*, que se comunica con *Azure AD B2C* para autenticar su identidad y con *Azure Functions* para enviar y recibir datos. Estas funciones orquestan el acceso a servicios externos (*OpenAI API* para los modelos *GPT* y el clasificador *NLP*) y gestionan los procesos internos: almacenamiento cifrado por usuario en *Blob Storage*, recuperación de claves desde *Key Vault* y registro del perfil emocional en la red blockchain privada. Esta separación de responsabilidades y servicios permite una arquitectura flexible, segura y escalable, adecuada para sistemas distribuidos sensibles como el que se describe en este proyecto.

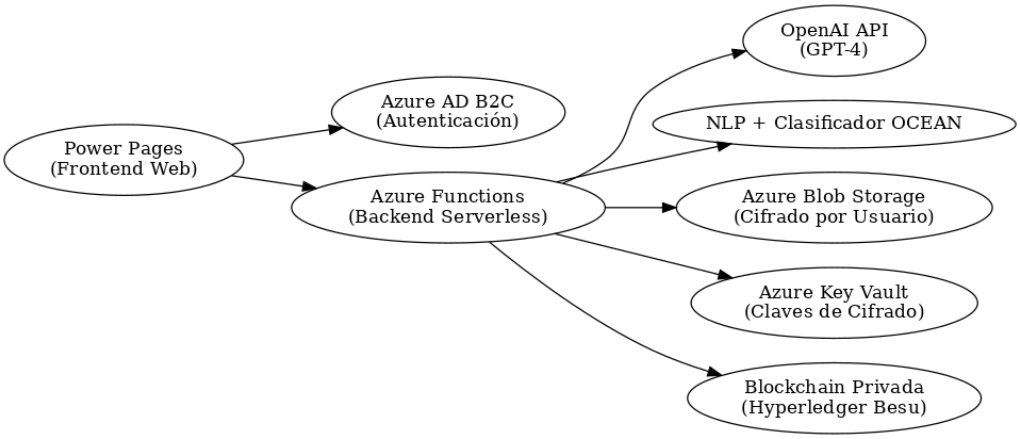

Figura 4.2: Diagrama de contenedores.

4.7.3. Nivel 3: Diagrama de Componentes de azurefunctions

El Nivel 3 del modelo C4 se centra en los componentes que forman parte del contenedor **Azure Functions**, que actúa como backend serverless del sistema. En esta capa se describen las funciones individuales desplegadas, cada una con una responsabilidad clara dentro del flujo general de la aplicación.

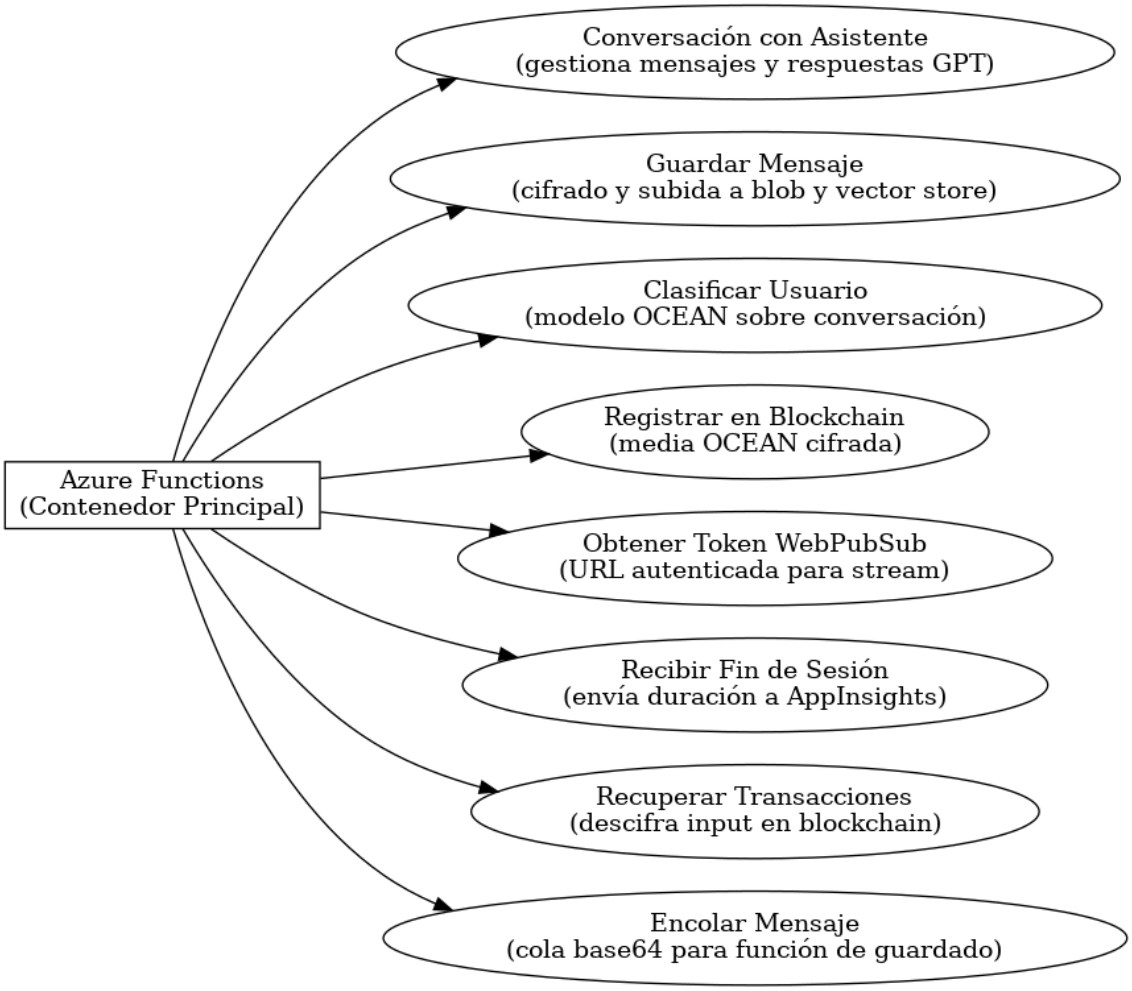

Figura 4.3: Nivel 3 del modelo C4: Funciones implementadas en Azure Functions

La Figura 4.3 ilustra gráficamente las funciones desplegadas dentro del contenedor principal Azure Functions. Cada una de estas funciones representa un componente lógico independiente que se comunica a través de desencadenadores y colas internas, manteniendo una arquitectura desacoplada y modular. Esta organización permite escalar funciones de forma independiente, facilitar el mantenimiento y garantizar la trazabilidad de cada operación ejecutada en el sistema.

A continuación, se detallan las funciones identificadas:

- **Conversación con Asistente:** Orquesta la comunicación entre el usuario y el modelo GPT, selecciona el *thread* adecuado y gestiona el envío de mensajes en tiempo real a través de *Web PubSub*.
- **Guardar Mensaje:** Se ejecuta al recibir un nuevo mensaje en la conversación. Cifra el contenido con una clave AES única por usuario, lo guarda en *Azure Blob Storage* en formato append blob, y lo sube al *vector store*

correspondiente para mantener el contexto conversacional del asistente.

- **Clasificar Usuario:** Procesa el historial conversacional completo aplicando técnicas de *NLP* y un modelo *OCEAN* entrenado en Azure Machine Learning. Esta función se activa automáticamente al alcanzar un umbral mínimo de interacciones.
- **Registrar en Blockchain:** Calcula la media ponderada de las últimas clasificaciones *OCEAN*, cifra la información con la clave del usuario y la registra como transacción en una red privada basada en *Hyperledger Besu*, garantizando inmutabilidad.
- **Obtener Token WebPubSub:** Genera y devuelve una URL autenticada con token temporal para permitir la recepción de mensajes por streaming mediante *Web PubSub*.
- **Recibir Fin de Sesión:** Recoge eventos enviados desde el frontend cuando termina la sesión de usuario. Calcula la duración total de la conversación y envía los datos a *Application Insights* para monitorización.
- **Recuperar Transacciones:** Interactúa con la *blockchain* para recuperar las transacciones firmadas del usuario. Descifra el contenido y lo devuelve en formato estructurado, como *CSV* o *JSON*.
- **Encolar Mensaje:** Función API que recibe y valida mensajes en tiempo real. Si el mensaje es válido, lo transforma a base64 y lo introduce en una *Azure Storage Queue* para ser procesado asincrónicamente por la función de guardado.

Esta arquitectura permite que cada función cumpla un rol bien definido y se ejecute de forma aislada pero orquestada, lo que mejora la mantenibilidad, la observabilidad y la capacidad de evolución futura del sistema.

4.7.4. Nivel 4: Código fuente e implementación interna

El Nivel 4 del modelo C4 describe los detalles técnicos internos de los componentes presentados en el Nivel 3. Esta capa se encuentra documentada en los Anexos del presente trabajo, incluyendo el código fuente completo de las funciones Azure, los scripts del modelo de clasificación OCEAN, y las pruebas unitarias realizadas sobre cada función crítica del sistema.

- El código fuente y la implementación técnica se encuentran detallados en el **Capítulo D - Manual Técnico Completo del Sistema**, donde se explican todas las funciones backend y páginas frontend.
- Las pruebas unitarias y funcionales se documentan exhaustivamente en el **Capítulo B - Evaluación automática de respuestas** y en el **capítulo anterior** (Pruebas Unitarias), cubriendo desde el procesamiento de mensajes hasta la interacción con blockchain y Application Insights.
- El entrenamiento del clasificador de personalidad está descrito en el **Capítulo C - Entrenamiento del clasificador OCEAN**, que incluye:

- Preprocesamiento y vectorización del texto.
- Entrenamiento del modelo en Azure Machine Learning.
- Persistencia de los objetos para inferencia.

Toda esta documentación proporciona una visión clara, reproducible y detallada del sistema desde su diseño lógico hasta su implementación técnica completa, cumpliendo con el nivel más profundo del modelo C4.

Capítulo 5

Evaluación de Resultados y Conclusiones

5.1. Evaluación de Objetivos

En este apartado se analiza el grado de cumplimiento de los objetivos planteados en la sección de introducción, valorando si se han alcanzado de forma completa, parcial o si han requerido adaptaciones durante el desarrollo.

5.1.1. Objetivo 1: Desarrollo e integración de modelos de inteligencia artificial para el acompañamiento emocional personalizado

- Desarrollar una serie de asistentes GPT especializados, cada uno orientado a una fase del modelo ADSM: *Cumplido*. Se han implementado asistentes independientes para los módulos AP, IPP, PCN, VP, SVM y MDS-YO, cada uno guiando al usuario a través de su protocolo específico.

- Diseñar y aplicar instrucciones estructuradas que permitan a los modelos GPT interactuar conforme a las directrices del modelo ADSM: *Cumplido*. Se han definido instrucciones detalladas para cada asistente, disponibles en el Anexo, y se han incorporado mecanismos de control de contexto mediante vector store.

- Extraer un perfil de personalidad del usuario aplicando técnicas de PLN sobre las conversaciones: *Cumplido*. Se ha entrenado un clasificador OCEAN a partir de un dataset sintético y se ha integrado en el flujo funcional del sistema. El perfil se genera tras alcanzar cierto volumen conversacional y se actualiza periódicamente.

5.1.2. Objetivo 2: Garantizar la seguridad, privacidad y seguimiento del bienestar emocional del usuario

- Implementar un sistema de autenticación robusto mediante Azure AD B2C: *Cumplido*. El acceso a la plataforma está protegido mediante un flujo seguro de autenticación federada. El token JWT se utiliza para validar la identidad del usuario en todas las llamadas al backend.

- Definir una arquitectura de almacenamiento segura mediante cifrado y gestión de claves: *Cumplido*. Las conversaciones se almacenan cifradas en Azure Blob Storage, usando claves únicas por usuario gestionadas desde Azure Key Vault. El sistema garantiza que solo el propietario puede descifrar su información.

- Almacenar la clasificación OCEAN en una blockchain privada: *Cumplido*. Se ha desplegado una red Hyperledger Besu sobre una máquina virtual de Azure. Las clasificaciones OCEAN se firman digitalmente y se registran como transacciones en la red, garantizando su integridad y trazabilidad.

Conclusión: Ambos objetivos principales del proyecto han sido alcanzados satisfactoriamente. El sistema desarrollado cumple tanto con los aspectos funcionales requeridos para el acompañamiento emocional como con los requisitos de seguridad y privacidad que exige el tratamiento de información psicológica sensible. Esta evaluación positiva valida la viabilidad del enfoque adoptado y la calidad técnica de la solución implementada.

5.2. Líneas futuras

El desarrollo de la presente plataforma ha permitido validar la viabilidad de aplicar modelos de lenguaje generativo al acompañamiento emocional seguro, estructurado y personalizado. No obstante, se identifican varias líneas de mejora y expansión que podrían abordarse en trabajos futuros:

1. Ampliación de protocolos y asistentes

Actualmente, el sistema implementa una colección de asistentes especializados en diferentes fases del modelo ADSM. Una posible evolución natural consiste en:

- Incorporar nuevos módulos del modelo ADSM no contemplados en esta versión.

- Permitir a los usuarios diseñar asistentes personalizados basados en sus propios objetivos y estilo cognitivo.

- Implementar asistentes que actúen como mentores o guías longitudinales, alineados con propuestas de inteligencia artificial adaptativa [57].

2. Mejora del modelo de clasificación de personalidad

Si bien el clasificador actual basado en XLM-Roberta ofrece resultados aceptables, podrían explorarse las siguientes mejoras:

- Entrenamiento con datasets más amplios y etiquetados manualmente para refinar la precisión [58].
- Experimentación con arquitecturas más recientes como Longformer [59] o DeBERTa [60], especialmente eficaces en contextos largos.
- Inclusión de dimensiones emocionales y afectivas más allá del modelo OCEAN, como el modelo PAD [61].

3. Análisis emocional en tiempo real

Actualmente, el sistema no analiza la carga emocional de las entradas del usuario más allá del contenido semántico. Futuras versiones podrían:

- Incorporar un módulo de detección de emociones mediante procesamiento del lenguaje natural [62].
- Adaptar el tono de respuesta de los asistentes en función del estado emocional detectado.
- Generar alertas en caso de emociones extremas o indicadores de riesgo psicosocial [63].

4. Internacionalización y accesibilidad

Para ampliar el impacto del sistema, sería conveniente abordar:

- Traducción completa de la interfaz y asistentes a otros idiomas, comenzando por inglés y portugués.
- Adaptación de la interfaz para usuarios con diversidad funcional, en línea con las pautas WCAG 2.1 [64].
- Evaluación de la experiencia de usuario en distintos contextos socioculturales [65].

5. Evaluación longitudinal del impacto

A medio plazo, sería deseable llevar a cabo estudios empíricos que analicen la utilidad real del sistema:

- Evaluaciones longitudinales de usuarios que utilicen la plataforma durante varios meses.
- Comparación con métodos tradicionales de acompañamiento emocional o coaching [66].
- Recogida de métricas de bienestar subjetivo, autoconocimiento y estabilidad emocional [67].

6. Mejora del rendimiento y costes en producción

Por último, podrían explorarse estrategias de optimización técnica:

- Sustitución de servicios como vector store o GPT externos por versiones autoalojadas o alternativas open source como Haystack o GPT-NeoX [68, 69].
- Optimización del uso de tokens y ventanas contextuales para reducir costes por uso de modelos [70].
- Evaluación de plataformas alternativas a Azure que ofrezcan mejor relación rendimiento/coste, como AWS o GCP.

Estas líneas futuras permitirían ampliar el alcance, robustez y aplicabilidad real de la solución, consolidándola como una herramienta versátil en el campo del acompañamiento emocional y el desarrollo personal asistido por IA.

5.3. Evaluación de Requisitos por Subsistemas

A continuación, se presenta una evaluación del cumplimiento de los requisitos definidos para cada subsistema del sistema, diferenciando entre requisitos funcionales y no funcionales. Esta evaluación permite valorar el grado de implementación alcanzado y las posibles limitaciones detectadas durante el desarrollo.

5.3.1. Interfaz de Usuario (Frontend Web)

1. Requisitos funcionales:
   - *Inicio de sesión y registro mediante Azure AD B2C:* Cumplido. El frontend redirige correctamente al flujo de autenticación externo y recupera el token JWT para las sesiones válidas.
   - *Interacción con asistentes GPT:* Cumplido. Las páginas de herramientas permiten al usuario conversar con los asistentes especializados a través de WebSockets.
   - *Selección de herramienta adecuada:* Cumplido. Se implementó una página inicial con botones visuales para seleccionar la herramienta correspondiente.
   - *Interfaz amigable y accesible:* Cumplido. La interfaz es funcional y adaptativa.
2. **Requisitos no funcionales:**
   - *Alta disponibilidad del frontend:* Cumplido. Power Pages garantiza disponibilidad en entorno cloud gestionado.
   - *Compatibilidad multiplataforma:* Cumplido. Se ha verificado el correcto funcionamiento en dispositivos y navegadores diversos.

- *Tiempos de carga reducidos:* Parcialmente cumplido. La carga inicial es rápida, pero el uso de múltiples recursos externos puede afectar al rendimiento si la conexión es limitada.

5.3.2. Autenticación y Gestión de Usuarios

1. Requisitos funcionales:
   - *Soporte para registro e inicio de sesión seguro:* Cumplido. Se ha configurado *Azure AD B2C* con flujos estándar de login y registro, sin gestión de contraseñas por parte del sistema.
2. Requisitos no funcionales:
   - *Cumplimiento RGPD y privacidad:* Cumplido. No se almacenan datos personales críticos y el acceso se realiza exclusivamente mediante tokens.
   - *Autenticación segura con tokens:* Cumplido. Se usa OpenID Connect con emisión de tokens ID y de acceso en cada sesión.

5.3.3. Asistentes GPT

1. Requisitos funcionales:
   - *Seguimiento del protocolo ADSM:* Cumplido. Cada asistente aplica su protocolo respectivo mediante instrucciones dinámicas.
   - *Un asistente por fase ADSM:* Cumplido. Se han creado asistentes separados para AP, IPP, PCN, SMV, VP y MDS-YO.
   - *Adaptación de instrucciones:* Cumplido. Las instrucciones se generan de forma contextual según el historial del usuario.
   - *Acceso a vector store:* Cumplido. Cada asistente consulta su vector store para recuperar el protocolo correspondiente.
2. **Requisitos no funcionales:**
   - *Uso de modelos de openAI:* Cumplido. Se han usado modelos *gpt-4o* por su compatibilidad con file-search.
   - *Comunicación en tiempo real:* Cumplido. Se emplea *Azure Web PubSub* para habilitar flujo continuo de tokens.

5.3.4. Procesamiento NLP y Clasificación OCEAN

1. Requisitos funcionales:
   - *Clasificación OCEAN:* Cumplido. Se ha desarrollado un clasificador entrenado con *XLM-Roberta* y el *dataset Big5-Chat*.
   - *Exportación de JSON:* Cumplido. El usuario puede descargar su perfil en formato *JSON* cifrado.

2. Requisitos no funcionales:

   - *Técnicas NLP aplicadas:* Cumplido. Se han aplicado tokenización, lematización y stopword removal con *spaCy*.
   - *Transparencia para el usuario:* Cumplido. El procesamiento se realiza de forma automatizada sin requerir intervención.

5.3.5. Almacenamiento Seguro

1. Requisitos funcionales:

   - *Privacidad de conversaciones y clasificación:* Cumplido. Todos los datos se almacenan cifrados con claves únicas.

2. Requisitos no funcionales:

   - *Uso de Azure Blob Storage cifrado:* Cumplido. Se emplean blobs tipo append cifrados con AES-GCM.
   - *Gestión de claves en Key Vault:* Cumplido. Cada clave AES se almacena en Key Vault con acceso restringido.
   - *Despliegue modular:* Cumplido. Cada subsistema accede a su recurso de forma aislada.

5.3.6. Blockchain Privada

1. Requisitos funcionales:

   - *Inmutabilidad de perfil OCEAN:* Cumplido. Se registra un hash cifrado del perfil en una red privada *Hyperledger Besu*.
   - *Control exclusivo del usuario:* Cumplido. Cada registro está vinculado al identificador cifrado del usuario.

2. Requisitos no funcionales:

   - *Registro cifrado en blockchain:* Cumplido. La clasificación *OCEAN* del usuario se cifra con una clave simétrica única y se registra como una transacción en la red *blockchain* privada. Esta transacción es generada y firmada automáticamente desde una *Azure Function* utilizando la identidad gestionada del sistema.
   - *Verificabilidad sin exposición:* Cumplido. La clasificación *OCEAN* del usuario se cifra mediante *AES-GCM* con una clave única y se registra cifrada en la *blockchain* privada. Esto permite garantizar la inmutabilidad del dato sin revelar en ningún momento el contenido real del perfil, protegiendo completamente su privacidad.

5.4. Evaluación del Desarrollo

Esta sección analiza el desarrollo del proyecto desde una perspectiva crítica, evaluando la evolución temporal, las desviaciones respecto a la planificación, las pruebas realizadas, las dificultades encontradas y los aprendizajes obtenidos.

5.4.1. Evolución temporal del desarrollo

El desarrollo se inició con una curva de aprendizaje pronunciada, especialmente durante las primeras semanas, donde fue necesario familiarizarse con herramientas nuevas como *Azure Functions*, *Web PubSub* o la integración con modelos *GPT*. Superada esta fase inicial, el avance fue más lineal y sostenido.

Una ralentización significativa se produjo durante la implementación de la red *blockchain* privada. Inicialmente se desplegó bajo el protocolo *Proof of Work (PoW)*, lo cual resultó inviable por su complejidad y requerimientos técnicos. Tras una fase de investigación y estudio, se optó por el protocolo *IBFT (Proof of Authority)*, mucho más adecuado al contexto del proyecto.

5.4.2. Desviaciones respecto a la planificación inicial

Si bien los hitos definidos en la planificación inicial del proyecto se mantuvieron como guía estructural, durante la ejecución se produjeron diversas desviaciones en su duración, solapamiento y distribución temporal. Estas variaciones respondieron principalmente a la complejidad técnica de algunos módulos, a la necesidad de realizar validaciones iterativas tras cada hito funcional y a la incorporación progresiva de mejoras arquitectónicas no previstas inicialmente.

- **Hito 2 – Desarrollo de asistentes GPT:** Este hito requirió una duración mayor de la estimada inicialmente. La redacción precisa de las instrucciones de cada asistente, el entrenamiento mediante file-search y la evaluación automática de las respuestas supusieron un proceso altamente iterativo. La validación se desplazó al Hito 3 para mejorar la detección de desviaciones en los primeros prototipos.
- **Hito 4 – Clasificador de personalidad:** Aunque su planificación fue correcta, el entrenamiento y validación del modelo *OCEAN* provocó ajustes en el preprocesamiento y la generación del dataset, que prolongaron ligeramente el calendario. El despliegue del endpoint de inferencia también requirió una revisión de seguridad para su integración con el backend.
- **Hito 6 – Cifrado y almacenamiento:** Se mantuvo estable en cuanto a tiempos, pero su validación se trasladó al Hito 7, que permitió verificar el acceso concurrente y la gestión segura de sesiones expiradas. La coordinación con Azure Key Vault y el tratamiento de claves personalizadas por usuario introdujeron una complejidad adicional no anticipada.
- **Hito 8 – Registro en blockchain:** La implementación se dividió en dos fases. La primera (despliegue de red *Besu*) implicó una curva de aprendizaje significativa. La segunda (integración desde *Azure Functions* y validación

criptográfica) fue más fluida, aunque dependía de la finalización de los hitos previos.

- **Hito 10 – Consolidación final:** Se redujo la duración estimada de este hito ya que gran parte de las pruebas funcionales y de integración se ejecutaron tras cada hito individual. Esto permitió anticipar errores, reducir retrabajo y agilizar la puesta en común del sistema completo.

Estas adaptaciones fueron registradas y gestionadas mediante un tablero Kanban, permitiendo responder con flexibilidad ante los imprevistos y optimizar la carga de trabajo. La evolución temporal de cada hito y las fechas reales se presentan en la comparativa incluida en la Sección 1.1.

5.4.3. Validación funcional y pruebas

Se realizaron distintas estrategias de validación:

- **Pruebas de concepto**: Se organizó una pequeña demo con usuarios cercanos a la organización, quienes interactuaron con la aplicación en su estado preliminar. Su feedback resultó útil para mejorar la interfaz y la claridad de los asistentes.
- **Pruebas funcionales de backend**: Se diseñaron pruebas unitarias y de integración para verificar el correcto funcionamiento de las *Azure Functions*, incluyendo cifrado, recuperación de historial, validación de identidad y envío a vector store.
- **Validación de flujo conversacional**: Mediante asistentes evaluadores y logs de conversación, se comprobó que las instrucciones eran correctamente interpretadas y aplicadas por los asistentes especializados.

5.4.4. Problemas y dificultades técnicas

El desarrollo presentó retos importantes:

- **Despliegue de la blockchain**: Supuso una de las mayores dificultades por falta de experiencia previa, requerimientos de red y configuración de nodos. Su integración con Azure y el entorno virtual fue compleja.
- **Heterogeneidad tecnológica**: La combinación de servicios (*Azure, OpenAI, Power Pages, blockchain, Azure ML*) exigió un conocimiento transversal y una elevada carga de integración.
- **Monitorización y sesiones**: La gestión del estado de sesión, detección de inactividad y control de concurrencia requirió lógica adicional y ajustes finos para evitar inconsistencias.
- **Pruebas en producción**: La necesidad de mantener la privacidad y seguridad dificultó las pruebas automatizadas, obligando a diseñar entornos de staging controlados.

5.4.5. Aprendizajes personales

Este proyecto ha permitido consolidar y adquirir nuevas competencias técnicas y profesionales:

- **IA generativa**: Profundización en el diseño de instrucciones de sistema, recuperación semántica y personalización de asistentes GPT.
- **DevOps y Azure**: Uso extensivo de *Azure Functions, Key Vault, Blob Storage, Web PubSub y Machine Learning*.
- **Ciberseguridad y privacidad**: Diseño de un sistema cifrado extremo a extremo, con control de acceso y almacenamiento seguro.
- **Trabajo estructurado**: Aplicación de la metodología *SCRUM*, documentación continua y gestión de tareas mediante asignaciones en *Microsoft Teams*.
- **Comunicación técnica**: Mejora en la redacción de documentación formal y capacidad de presentación técnica ante terceros.

Conclusión: La evaluación del desarrollo demuestra una evolución satisfactoria, con capacidad de adaptación ante los retos técnicos y metodológicos, y un aprendizaje integral que trasciende el propio alcance técnico del sistema construido.

5.5. Análisis de impacto y contribución a los ODS

Este capítulo realiza un análisis del impacto potencial del proyecto en distintos ámbitos relevantes, así como su alineación con los Objetivos de Desarrollo Sostenible (ODS) definidos por Naciones Unidas en la Agenda 2030 [71].

5.5.1. Impacto en diferentes contextos

- **Impacto personal:** El uso de los asistentes basados en el modelo ADSM puede favorecer la autoexploración, la autoconciencia y el desarrollo emocional del usuario. El acceso continuo y confidencial a herramientas conversacionales centradas en el bienestar personal puede promover hábitos saludables de autorreflexión y toma de decisiones.
- **Impacto empresarial:** La arquitectura modular y escalable del sistema facilita su posible adopción por organizaciones orientadas a la gestión de personas, el desarrollo profesional o la salud emocional en el entorno laboral. La integración con Azure permite desplegarlo de forma eficiente y con alta disponibilidad.
- **Impacto social:** La solución permite ofrecer acompañamiento emocional personalizado sin barreras geográficas ni económicas, democratizando el acceso a este tipo de apoyo y potenciando la inclusión social de colectivos con necesidades específicas o en situación de vulnerabilidad.

- **Impacto económico:** El modelo serverless reduce costes operativos y de mantenimiento, lo que puede fomentar el emprendimiento digital en el ámbito del bienestar. Además, al evitar licencias privativas o recursos locales, se reduce la barrera económica de entrada para instituciones educativas o sociales interesadas en replicar el sistema.
- **Impacto medioambiental:** Al tratarse de una solución digital alojada en la nube y sin infraestructura local, el consumo energético es bajo y escalable según demanda. Se reduce así la necesidad de equipamiento físico y su correspondiente huella ecológica.
- **Impacto cultural:** La plataforma facilita el desarrollo emocional y reflexivo desde un enfoque respetuoso con la diversidad. La capacidad de personalización del lenguaje y de los asistentes permite adaptar los contenidos a diferentes contextos culturales y lingüísticos.

5.5.2. Contribución a los Objetivos de Desarrollo Sostenible (ODS)

Este proyecto contribuye directa o indirectamente a los siguientes ODS de la Agenda 2030:

- **ODS 3 – Salud y Bienestar:** El sistema busca mejorar el bienestar emocional de los usuarios mediante el acompañamiento personalizado, el uso de herramientas de reflexión estructurada (modelo ADSM) y el acceso a asistentes conversacionales diseñados para facilitar la gestión de situaciones personales complejas. Aunque no sustituye atención psicológica profesional, representa una vía de apoyo accesible, continua y respetuosa con la privacidad del usuario [72, 73].
- **ODS 4 – Educación de Calidad:** El enfoque del proyecto promueve el desarrollo de competencias personales como la autoconciencia, la toma de decisiones y el pensamiento crítico. A través de sus asistentes estructurados, el sistema acompaña procesos de aprendizaje emocional, convirtiéndose en una herramienta complementaria en contextos educativos y de desarrollo personal [74, 75].
- **ODS 9 – Industria, Innovación e Infraestructura:** La implementación de una arquitectura serverless en Azure, junto con el uso de modelos de lenguaje avanzados (GPT), blockchain privada e integración segura de datos, sitúa el proyecto en la vanguardia tecnológica. Se fomenta la innovación responsable al aplicar IA generativa a un ámbito sensible como el emocional, con mecanismos de control ético y privacidad reforzada [76, 77].
- **ODS 10 – Reducción de las Desigualdades:** Al diseñar una interfaz accesible y una experiencia basada en lenguaje natural, el sistema facilita el acceso al acompañamiento emocional incluso para personas sin formación tecnológica o recursos económicos. La escalabilidad de la solución permite llegar a colectivos vulnerables o con necesidades específicas de apoyo [78, 79].

- **ODS 16 – Paz, Justicia e Instituciones Sólidas:** Mediante el uso de blockchain privada y almacenamiento cifrado, se garantiza la trazabilidad, inmutabilidad y confidencialidad de los datos sensibles. Esto contribuye a construir entornos digitales más seguros, transparentes y éticamente responsables [6, 80].

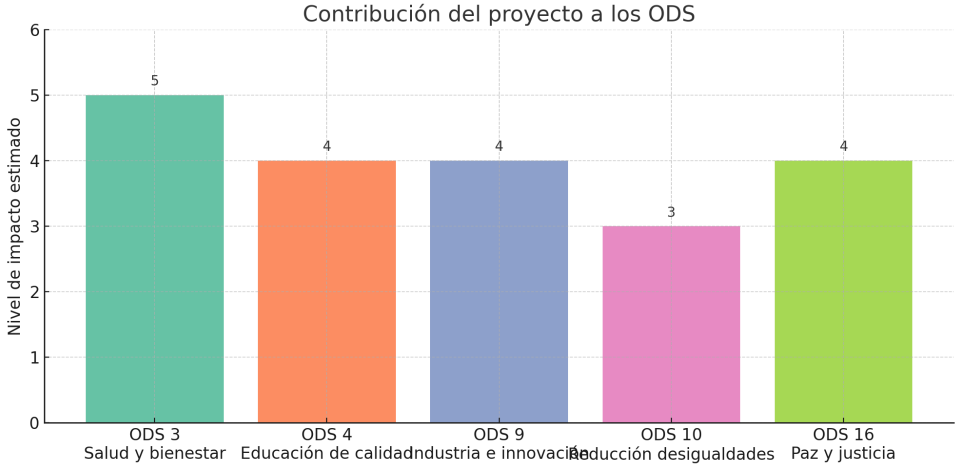

Figura 5.1: Contribución del proyecto a los Objetivos de Desarrollo Sostenible (ODS).

En conjunto, el proyecto no solo aporta valor técnico y académico, sino que se alinea con los principios de sostenibilidad, inclusión y bienestar humano, pilares fundamentales de la Agenda 2030. A lo largo del desarrollo se tomaron decisiones que priorizan el impacto positivo: elección de una arquitectura eficiente energéticamente, uso de identidades seguras gestionadas, anonimización de datos sensibles, y accesibilidad desde navegadores sin necesidad de instalación local.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que han hecho posible el desarrollo de este Trabajo de Fin de Grado.

En primer lugar, a **Gerardo Tudurí**, creador del método ADSM, por permitir la aplicación de su sistema en este proyecto y por su valiosa colaboración en la elaboración de los protocolos utilizados por los asistentes, así como en la evaluación metodológica de los mismos. Su conocimiento experto ha sido esencial para garantizar el rigor conceptual del trabajo.

Asimismo, agradezco profundamente a **Jesús Navarro**, CEO de InnoPersona, no solo por su respaldo institucional y la financiación del proyecto, sino también por haber facilitado el contacto y la colaboración activa con Gerardo Tudurí en el desarrollo de los protocolos. Su implicación ha sido determinante para articular las distintas piezas del proyecto y asegurar su coherencia estratégica y

operativa.

Finalmente, deseo manifestar mi reconocimiento a **Juan Antonio Fernández**, tutor académico de este trabajo, por su acompañamiento constante, sus orientaciones metodológicas y por la recomendación de materiales y recursos que han enriquecido el enfoque y la calidad del proyecto.

A todos ellos, gracias por su generosidad, compromiso y apoyo constante.

Declaración de uso de inteligencia artificial en la redacción del documento

En consonancia con las recomendaciones actuales sobre transparencia y ética en la elaboración de trabajos académicos, se declara el uso de herramientas de inteligencia artificial (IA) generativa durante el proceso de redacción de este documento. Esta práctica se alinea con las directrices establecidas por instituciones como la Association of College and Research Libraries (ACRL) y la UNESCO, que promueven la divulgación clara del uso de IA en contextos académicos [81, 82].

Las herramientas de IA se emplearon específicamente para:

- Asistir en la generación de ideas y estructuración inicial de secciones del documento.
- Proporcionar sugerencias de redacción y mejora de estilo en borradores preliminares.
- Revisar la gramática y coherencia textual en las versiones intermedias del manuscrito.

Es importante destacar que todas las sugerencias y contenidos generados por la IA fueron cuidadosamente revisados, editados y validados por los autores humanos, quienes asumen la plena responsabilidad sobre la precisión, integridad y originalidad del contenido final.

Esta declaración se realiza con el objetivo de mantener la integridad académica y fomentar la transparencia en el uso de tecnologías emergentes en la investigación y redacción científica.

Bibliografía

[1] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," Technical Report, 2019, Último acceso: 2025-05-27. [Online]. Available: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[2] S. Zhang, Y. Sun, Y. Zhang, Q. Yang, and et al., "A survey on chatgpt: Foundations, applications, and challenges," *arXiv preprint arXiv:2304.01852*, 2023, Último acceso: 2025-06-04. [Online]. Available: https://arxiv.org/abs/2304.01852

[3] Innopersona, "Somos - innopersona," 2023, Último acceso: 2025-05-27. [Online]. Available: https://innopersona.com/somos/

[4] G. I. Tuduri, "Cómo construir una mente biodigital: La ia y los próximos sistemas mentales humanos," *Telos*, vol. 123, pp. 64–65, 2023, Último acceso: 2025-05-27. [Online]. Available: https://telos.fundaciontelefonica.com/wp-content/uploads/2023/11/TELOS-123-Cuaderno-Central-Inteligencia-Artificial-Gerardo-Ivan-Tuduri.pdf

[5] R. R. McCrae and P. T. Costa, "An introduction to the five-factor model and its applications," *Journal of Personality*, vol. 60, no. 2, pp. 175–215, 1992, Último acceso: 2025-05-27. [Online]. Available: https://www.workplacebullying.org/multi/pdf/5factor-theory.pdf

[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," https://bitcoin.org/bitcoin.pdf, 2008, Último acceso: 2025-05-27.

[7] K. Schwaber and J. Sutherland, *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*. Scrum.org, 2020. [Online]. Available: https://scrumguides.org/scrum-guide.html

[8] ——, "The scrum guide – the definitive guide to scrum: The rules of the game," https://scrumguides.org/, 2020, Último acceso: 2025-05-27.

[9] J. Navarro, "Perfil profesional en linkedin," https://es.linkedin.com/in/jnavarrosan, 2025, Último acceso: 2025-05-27.

[10] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoya-

nov, “Unsupervised cross-lingual representation learning at scale,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020, pp. 8440–8451, Último acceso: 2025-06-04. [Online]. Available: https://aclanthology.org/2020.acl-main.747/

[11] E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. Yadwadkar, J. E. Gonzalez, R. A. Popa, I. Stoica, and D. A. Patterson, “Cloud programming simplified: A berkeley view on serverless computing,” University of California, Berkeley, Tech. Rep. UCB/EECS-2019-3, 2019, Último acceso: 2025-06-04. [Online]. Available: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-3.pdf

[12] Microsoft Learn, “What is azure key vault?” 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/key-vault/general/overview

[13] ——, “Introduction to azure blob storage,” 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction

[14] H. Foundation, “Hyperledger besu documentation,” 2024. [Online]. Available: https://besu.hyperledger.org/en/stable/

[15] M. Learn, “Azure functions documentation,” 2024. [Online]. Available: https://learn.microsoft.com/en-us/azure/azure-functions/

[16] IBM, “Procesamiento de lenguaje natural (pln),” 2025, Último acceso: 2025-05-27. [Online]. Available: https://www.ibm.com/es-es/think/topics/natural-language-processing

[17] D. la Torre AI, “Los transformers en la inteligencia artificial: Una explicación sencilla,” 2025, Último acceso: 2025-05-27. [Online]. Available: https://delatorre.ai/los-transformers-en-la-inteligencia-artificial-una-explicacion-sencilla/

[18] G. Tuduri, “Perfil profesional en linkedin,” https://es.linkedin.com/in/gerardo-tuduri-0b4b9a142, 2025, Último acceso: 2025-05-27.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, vol. 30, 2017, Último acceso: 2025-05-27. [Online]. Available: https://papers.nips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[20] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997, Último acceso: 2025-06-04. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*,

Volume 1 (Long and Short Papers). Association for Computational Linguistics, 2019, pp. 4171–4186, Último acceso: 2025-06-04. [Online]. Available: https://aclanthology.org/N19-1423/

[22] Koko, "Koko - mental health support through technology," https://www.kokocares.org/, 2024, Último acceso: 2025-05-27.

[23] Infobae, "Esta herramienta usa chat gpt para dar ayuda psicológica," 2023, Último acceso: 2025-05-27. [Online]. Available: https://www.infobae.com/tecno/2023/02/12/esta-herramienta-usa-chat-gpt-para-dar-ayuda-psicologica/?

[24] Wysa, "Wysa - ai for mental health," https://www.wysa.com/, 2024, Último acceso: 2025-05-27.

[25] J. S. Beck, *Cognitive Behavior Therapy: Basics and Beyond*, 2nd ed. New York: The Guilford Press, 2011, Último acceso: 2025-06-04. [Online]. Available: https://archive.org/details/cognitive-behavior-therapy-second-edition-basics-and-beyond

[26] Woebot Health, "Woebot health - meet woebot, your mental health ally," https://woebothealth.com/, 2024, Último acceso: 2025-05-27.

[27] Replika, "Replika - the ai companion who cares," https://replika.com/, 2024, Último acceso: 2025-05-27.

[28] X2AI, "Tess - psychological ai for scalable support," https://www.x2ai.com/, 2024, Último acceso: 2025-05-27.

[29] 7 Cups, "7 cups - online therapy and free counseling," https://www.7cups.com/, 2024, Último acceso: 2025-05-27.

[30] IBM, "Ibm watson personality insights (archived)," https://www.ibm.com/cloud/watson-personality-insights, 2021, Último acceso: 2025-05-27.

[31] M. E. F. James W. Pennebaker, Roger J. Booth, "Liwc - linguistic inquiry and word count," https://www.liwc.app/, 2015, Último acceso: 2025-05-27.

[32] P. Viyyanan, "Bert-base spanish biomedical," Hugging Face, 2021, Último acceso: 2025-05-27. [Online]. Available: https://huggingface.co/prithivida/bert-base-spanish-biomedical

[33] "Reglamento general de protección de datos (rgpd)," https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=CELEX%3A32016R0679, 2016, reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo.

[34] M. Corporation, "What is microsoft power platform?" 2023, Último acceso: 2025-05-27. [Online]. Available: https://learn.microsoft.com/en-us/power-platform/

[35] Microsoft Corporation, "What is azure active directory b2c?" https://learn.microsoft.com/en-us/azure/active-directory-b2c/overview, 2023, Último acceso: 2025-05-27.

[36] "Azure web pubsub - microsoft azure," https://learn.microsoft.com/en-us/azure/azure-web-pubsub/overview, Último acceso: 2025-05-27.

[37] Microsoft Corporation, "Azure functions documentation," https://learn.microsoft.com/en-us/azure/azure-functions/, 2024, Último acceso: 2025-05-27.

[38] Microsoft Learn, "What is application insights?" https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview, 2024, Último acceso: 2025-05-27.

[39] OpenAI, "Retrieval with file search," https://platform.openai.com/docs/guides/tools-file-search, 2024, Último acceso: 2025-05-27.

[40] ——, "Semantic search with pinecone and openai," https://cookbook.openai.com/examples/vector_databases/pinecone/semantic_search, 2023, Último acceso: 2025-05-27.

[41] Z. Zhao, Y. Li, J. Zhou, J. Lin, and C. Xu, "Vector database: A survey," *arXiv preprint arXiv:2212.08011*, 2022, Último acceso: 2025-05-27. [Online]. Available: https://arxiv.org/abs/2212.08011

[42] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," *arXiv preprint arXiv:1904.09751*, 2019, Último acceso: 2025-05-27. [Online]. Available: https://arxiv.org/abs/1904.09751

[43] OpenAI, "Gpt-4 turbo model (gpt-4-0125-preview)," https://platform.openai.com/docs/models/gpt-4, 2024, Último acceso: 2025-05-27.

[44] Microsoft Corporation, "Azure blob storage documentation," https://learn.microsoft.com/en-us/azure/storage/blobs/, 2024, Último acceso: 2025-05-27.

[45] ——, "Azure key vault documentation," https://learn.microsoft.com/en-us/azure/key-vault/, 2024, Último acceso: 2025-05-27.

[46] National Institute of Standards and Technology, "Announcing the advanced encryption standard (aes)," https://csrc.nist.gov/publications/detail/fips/197/final, 2001, fIPS PUB 197.

[47] Microsoft Learn, "What are managed identities for azure resources?" https://learn.microsoft.com/en-us/azure/active-directory/managed-identities-azure-resources/overview, 2024, Último acceso: 2025-05-27.

[48] *Hyperledger Besu Documentation*, https://besu.hyperledger.org/en/stable/, 2024, Último acceso: 2025-05-27.

[49] *Microsoft Azure Virtual Machines Documentation*, https://learn.microsoft.com/en-us/azure/virtual-machines/, 2024, Último acceso: 2025-05-27.

[50] H. Besu, "Proof of authority networks," https://besu.hyperledger.org/private-networks/concepts/poa, 2024.

[51] H. Moniz *et al.*, "The istanbul byzantine fault tolerance (ibft) consensus algorithm," *arXiv preprint arXiv:2002.03613*, 2020, disponible en arXiv. [Online]. Available: https://arxiv.org/abs/2002.03613

[52] H. Besu, "Ibft 2.0 consensus protocol," https://besu.hyperledger.org/private-networks/how-to/configure/consensus/ibft, 2024, accedido en mayo de 2025.

[53] E. Foundation, "Ethereum json-rpc specification," https://ethereum.org/en/developers/docs/apis/json-rpc/, 2024, accedido en mayo de 2025.

[54] *Web3.py Documentation*, https://web3py.readthedocs.io/en/stable/, 2024, Último acceso: 2025-05-27.

[55] *Azure Key Vault Documentation*, https://learn.microsoft.com/en-us/azure/key-vault/, 2024, Último acceso: 2025-05-27.

[56] *An Introduction to Hyperledger*, https://8112310.fs1.hubspotusercontent-na1.net/hubfs/8112310/Hyperledger/Offers/HL_Whitepaper_IntroductiontoHyperledger.pdf, 2018, Último acceso: 2025-05-27.

[57] W. Duch, "Artificial intelligence and the limits of the humanities," *Er(r)go. Teoria - Literatura - Kultura*, vol. 48, pp. 269–284, 2024, Último acceso: 2025-06-04. [Online]. Available: https://www.researchgate.net/publication/383449180_Artificial_Intelligence_and_the_Limits_of_the_Humanities

[58] Y. Goldberg, "A primer on neural network models for natural language processing," *Journal of Artificial Intelligence Research*, vol. 57, pp. 345–420, 2016, Último acceso: 2025-06-04. [Online]. Available: https://jair.org/index.php/jair/article/view/11030

[59] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020, Último acceso: 2025-05-27.

[60] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," *Microsoft Research*, 2020, Último acceso: 2025-06-04. [Online]. Available: https://www.microsoft.com/en-us/research/publication/deberta-decoding-enhanced-bert-with-disentangled-attention-2/

[61] A. Mehrabian, "Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament," *Current Psychology*, vol. 14, no. 4, pp. 261–292, 1996. [Online]. Available: https://link.springer.com/article/10.1007/BF02686918

[62] S. Poria, E. Cambria, D. Hazarika, N. Majumder, G. Naik, and A. Hussain, "Emotion recognition in conversation: Research challenges, datasets, and recent advances," in *Proceedings of the 27th ACM International Conference on Multimedia*. ACM, 2019, pp. 1648–1656. [Online]. Available: https://dl.acm.org/doi/10.1145/3343031.3350915

[63] R. A. Calvo and S. D'Mello, "The oxford handbook of affective computing," *Oxford University Press*, 2015. [Online]. Available: https://oxfordhandbooks.com/view/10.1093/oxfordhb/9780199942237.001.0001/oxfordhb-9780199942237

[64] World Wide Web Consortium (W3C), "Web content accessibility guidelines (wcag) 2.1," https://www.w3.org/TR/WCAG21/, 2018, Último acceso: 2025-06-04.

[65] J. Nielsen, "Usability engineering," 1993, Último acceso: 2025-06-04. [Online]. Available: https://www.nngroup.com/books/usability-engineering/

[66] A. M. Grant, "The efficacy of executive coaching in times of organisational change," *Journal of Change Management*, vol. 14, no. 2, pp. 258–280, 2014, Último acceso: 2025-06-04. [Online]. Available: https://www.researchgate.net/publication/263286487_The_Efficacy_of_Executive_Coaching_in_Times_of_Organisational_Change

[67] E. Diener, E. M. Suh, R. E. Lucas, and H. L. Smith, "Subjective well-being: Three decades of progress," *Psychological Bulletin*, vol. 125, no. 2, pp. 276–302, 2009, Último acceso: 2025-06-04. [Online]. Available: https://psycnet.apa.org/record/2009-04260-001

[68] deepset GmbH, "Haystack: Open source framework for nlp-powered search," 2023, https://haystack.deepset.ai. Último acceso: 2025-05-27.

[69] S. Black, S. Biderman, E. Hallahan, Q. Anthony, L. Gao, L. Golding, H. He, C. Leahy, K. McDonell, J. Phang, M. Pieler, U. S. Prashanth, S. Purohit, L. Reynolds, J. Tow, B. Wang, and S. Weinbach, "Gpt-neox-20b: An open-source autoregressive language model," *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pp. 95–136, 2022, Último acceso: 2025-06-04. [Online]. Available: https://aclanthology.org/2022.bigscience-1.9/

[70] OpenAI, "Openai pricing overview," 2024, https://openai.com/pricing. Último acceso: 2025-05-27.

[71] Naciones Unidas, "Transformar nuestro mundo: la agenda 2030 para el desarrollo sostenible," 2015, Último acceso: 2025-05-27. [Online]. Available: https://sdgs.un.org/es/goals

[72] World Health Organization, *Mental Health Action Plan 2013–2020*, https://iris.who.int/handle/10665/89966, 2013, Último acceso: 2025-05-27.

[73] S. Coghlan, K. Leins, S. Sheldrick, M. Cheong, P. Gooding, and S. D'Alfonso, "To chat or bot to chat: Ethical issues with using chatbots in mental health," *Digital Health*, vol. 9, p. 20552076231183542, 2023, Último acceso: 2025-05-27. [Online]. Available: https://journals.sagepub.com/doi/full/10.1177/20552076231183542

[74] UNESCO, "Mainstreaming social and emotional learning in education systems," 2023, Último acceso: 2025-05-27.

[75] ——, *Mainstreaming Social and Emotional Learning in Education Systems*, https://unesdoc.unesco.org/ark:/48223/pf0000392261, 2023, Último acceso: 2025-05-27.

[76] A. Andromeda, "Resumen de .ªrtificial intelligence: A modern approach"de stuart russell y peter norvig," *AI Andromeda*, 2023, Último acceso: 2025-06-04. [Online]. Available: https://aiandromeda.com/artificial-intelligence-a-modern-approach-de-stuart-russell-y-peter-norvig/

[77] Microsoft, "Azure ai overview," 2024, Último acceso: 2025-05-27. [Online]. Available: https://azure.microsoft.com/en-us/overview/ai-platform/

[78] World Health Organization, *Equity and Health: WHO Report*, https://www.who.int/health-topics/health-equity, 2018, Último acceso: 2025-06-04.

[79] A. Andromeda, "Resumen de *race after technology: Abolitionist tools for the new jim code* de ruha benjamin," *AI Andromeda*, 2023, Último acceso: 2025-06-04. [Online]. Available: https://aiandromeda.com/race-after-technology-abolitionist-tools-for-the-new-jim-code/

[80] L. A. Sarro, "Reseña del libro *la revolución blockchain*," *Escritos Contables y de Administración*, vol. 11, no. 1, pp. 53–71, 2020, Último acceso: 2025-06-04. [Online]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=7499637

[81] UNESCO, "Recomendación sobre la ética de la inteligencia artificial," 2021, consultado en junio de 2025. [Online]. Available: https://www.unesco.org/es/artificial-intelligence/recommendation-ethics

[82] Association of College and Research Libraries, "How to ethically use generative ai in research and writing," 2023, consultado en junio de 2025. [Online]. Available: https://arxiv.org/abs/2109.09672

Anexos

Apéndice A

Pruebas unitarias

A.1. Función StreamingResp

1. Objetivo:

   Comprobar el correcto funcionamiento de la función Azure que gestiona las solicitudes HTTP recibidas desde Power Pages, las envía a la API de OpenAI (ChatGPT) y retransmite la respuesta generada en tiempo real mediante Azure Web PubSub (WebSocket).

2. Descripción técnica:

   La función recibe peticiones HTTP POST desde el frontend (Power Pages). Cada petición contiene un mensaje del usuario que se reenvía directamente a la API de ChatGPT (OpenAI). La respuesta obtenida desde OpenAI se procesa en streaming y se retransmite inmediatamente al cliente mediante una conexión WebSocket habilitada a través del servicio Azure Web PubSub. El sistema está monitorizado con Azure Application Insights, lo que permite capturar eventos, errores y tiempos de respuesta.

3. Procedimiento:

   Las pruebas realizadas abarcaron los siguientes aspectos clave:

   - Validación y manejo de solicitudes HTTP de entrada.
   - Comunicación robusta con la API de ChatGPT.
   - Retransmisión estable de la respuesta por WebSocket.
   - Gestión adecuada de condiciones anómalas y registro de errores mediante Application Insights.

4. Casos de prueba:

   Se han definido y ejecutado los siguientes casos de prueba representativos, empleando técnicas de simulación (*mocking* y herramientas específicas) para escenarios que involucran errores o condiciones especiales:

a) **Solicitud HTTP correcta**:

- Entrada: Mensaje JSON válido con texto del usuario.
- Resultado esperado: Recepción del mensaje por la API de ChatGPT y transmisión exitosa del resultado vía WebSocket.
- Resultado obtenido: *Éxito.*

b) **Solicitud HTTP sin mensaje**:

- Entrada: Solicitud JSON vacía.
- Resultado esperado: Retorno de error de validación HTTP 400.
- Resultado obtenido: *Éxito.*

c) **Respuesta lenta de API OpenAI (simulada mediante *mocking*)**:

- Entrada: Simulación de latencia alta en respuesta de OpenAI.
- Resultado esperado: Retransmisión correcta del streaming sin cortes ni fallos en WebSocket.
- Resultado obtenido: *Éxito.*

d) **Fallo conexión a WebSocket (simulado)**:

- Entrada: Interrupción temporal simulada del WebSocket.
- Resultado esperado: Captura del error y registro adecuado en Application Insights.
- Resultado obtenido: *Éxito.*

e) **Error en la API de OpenAI (simulado mediante *mocking*)**:

- Entrada: Error HTTP 503 desde OpenAI.
- Resultado esperado: Retorno de mensaje claro de error hacia el cliente y registro detallado del evento en Application Insights.
- Resultado obtenido: *Éxito.*

5. Conclusión de las pruebas:

   La función StreamingResp ha superado satisfactoriamente todos los casos de prueba establecidos, demostrando robustez, fiabilidad y una adecuada capacidad de monitorización mediante Application Insights. Las pruebas garantizan que la comunicación con la API de OpenAI y con la capa frontend implementada en Power Pages funciona correctamente, y que el sistema es capaz de gestionar condiciones anómalas de forma controlada en producción.

A.2. Función GetWebSocketUrl

1. Objetivo:

Verificar que la función Azure encargada de configurar la conexión WebSocket a través del servicio Azure Web PubSub genera correctamente los tokens de acceso y maneja adecuadamente las validaciones de seguridad.

2. Procedimiento:

   Se ejecutaron diversas pruebas unitarias para asegurar:

   - Validación de la solicitud HTTP y comprobación del origen permitido (CORS).
   - Extracción y validación adecuada del JWT Token del usuario.
   - Generación correcta de tokens para Azure Web PubSub.
   - Manejo apropiado de errores y casos excepcionales.

3. Casos de prueba:

   A continuación se describen los casos de prueba específicos ejecutados:

   a) **Solicitud HTTP desde origen autorizado**:

   - Entrada: Solicitud HTTP desde `https://pilotoyodai.powerappsportals.com`.
   - Resultado esperado: Aceptación de la solicitud y generación del token de Web PubSub.
   - Resultado obtenido: *Éxito.*

   b) **Solicitud HTTP desde origen no autorizado**:

   - Entrada: Solicitud HTTP desde un origen diferente al autorizado.
   - Resultado esperado: Rechazo con código HTTP 403 (Forbidden).
   - Resultado obtenido: *Éxito.*

   c) **Token JWT no proporcionado**:

   - Entrada: Solicitud HTTP sin token JWT en la cabecera Authorization.
   - Resultado esperado: Retorno con código HTTP 401 (Unauthorized).
   - Resultado obtenido: *Éxito.*

   d) **Token JWT inválido**:

   - Entrada: Token JWT malformado o inválido.
   - Resultado esperado: Retorno con código HTTP 401 y registro del error.
   - Resultado obtenido: *Éxito.*

   e) **Token generado para Web PubSub con grupos correctos**:

   - Entrada: Solicitud válida con usuario autenticado.

- Resultado esperado: Token generado correctamente con los permisos y grupos adecuados asignados al usuario.
- Resultado obtenido: *Éxito.*

f) **Fallo en la generación del token de Web PubSub**:

- Entrada: Simulación de fallo en el servicio Azure Web PubSub.
- Resultado esperado: Captura y registro adecuado del error, retorno HTTP 500.
- Resultado obtenido: *Éxito.*

4. Conclusión de las pruebas:

   La función encargada de configurar el WebSocket ha superado con éxito todos los casos definidos, asegurando tanto la seguridad como la funcionalidad requerida para las conexiones en tiempo real mediante Azure Web PubSub. Las pruebas garantizan una integración robusta con la capa de frontend implementada mediante Power Pages.

A.3. Función ClasificarUsuarioSesion

1. Objetivo:

   Comprobar el correcto funcionamiento de la función Azure que ejecuta la regresión lineal sobre las interacciones del usuario para clasificar su personalidad según el modelo OCEAN, almacenar los resultados y gestionar el envío de mensajes a una cola de Azure Storage Queue.

2. Procedimiento:

   Se han realizado diversas pruebas unitarias y funcionales para validar:

   - Recepción y deserialización correcta del mensaje desde la cola.
   - Ejecución precisa del modelo de regresión lineal.
   - Clasificación correcta según el modelo OCEAN.
   - Inserción adecuada de resultados en Azure Table Storage.
   - Envío de mensaje a la cola para activar la función de registro en blockchain.

3. Casos de prueba:

   Se han definido y ejecutado los siguientes casos de prueba representativos:

   a) **Mensaje válido recibido desde la cola**:

   - Entrada: JSON con identificador válido de usuario e historial conversacional.
   - Resultado esperado: Clasificación exitosa y almacenamiento en Table Storage.

- Resultado obtenido: *Éxito.*

b) **Mensaje con formato incorrecto**:

- Entrada: JSON mal formado o con datos faltantes.
- Resultado esperado: Manejo adecuado del error y registro en Application Insights.
- Resultado obtenido: *Éxito.*

c) **Insuficientes datos para la regresión lineal**:

- Entrada: Historial de interacción muy reducido.
- Resultado esperado: Clasificación con advertencia por insuficiencia de datos.
- Resultado obtenido: *Éxito con advertencia registrada.*

d) **Almacenamiento fallido en Azure Table Storage**:

- Entrada: Simulación de desconexión temporal con Azure Table Storage.
- Resultado esperado: Reintentos y registro del fallo en Application Insights.
- Resultado obtenido: *Éxito.*

e) **Envío correcto de mensaje a cola de blockchain**:

- Entrada: Superado número umbral de clasificaciones acumuladas.
- Resultado esperado: Mensaje enviado correctamente para activar función blockchain.
- Resultado obtenido: *Éxito.*

4. Conclusión de las pruebas:

   La función encargada de realizar la regresión lineal y clasificar al usuario según el modelo OCEAN ha superado satisfactoriamente todos los casos de prueba establecidos. Estas pruebas confirman que el sistema maneja adecuadamente los procesos analíticos, el almacenamiento seguro de resultados y la coordinación entre funciones mediante colas, garantizando fiabilidad y consistencia en la clasificación y análisis continuo del perfil emocional del usuario.

A.4. Función RegistrarBlockChain

1. Objetivo:

   Validar la correcta integración de la función Azure encargada de calcular la media ponderada de las clasificaciones OCEAN almacenadas para un

usuario, y registrar dicha clasificación en una blockchain privada utilizando Hyperledger Besu. La función se activa mediante mensajes en una cola Azure Queue Storage.

2. Procedimiento:

   Las pruebas realizadas cubren:

   - Recepción y procesamiento de mensajes desde la cola Azure Queue.
   - Recuperación precisa y cálculo correcto de la media ponderada de las clasificaciones almacenadas.
   - Integración segura con la blockchain para registrar las clasificaciones calculadas.

3. Casos de prueba:

   a) **Mensaje válido en cola Azure Queue**:

   - Entrada: Mensaje JSON válido con el identificador del usuario.
   - Resultado esperado: Cálculo correcto de la media ponderada y registro exitoso en blockchain.
   - Resultado obtenido: *Éxito.*

   b) **Mensaje inválido o mal formado**:

   - Entrada: Mensaje JSON sin identificador de usuario.
   - Resultado esperado: Manejo adecuado del error, con registro explícito en Application Insights y descarte seguro del mensaje.
   - Resultado obtenido: *Éxito.*

   c) **Usuario sin clasificaciones previas suficientes**:

   - Entrada: Usuario con menos clasificaciones de las necesarias para realizar la media.
   - Resultado esperado: Mensaje informativo en Application Insights indicando insuficiencia de datos, sin registrar transacción en blockchain.
   - Resultado obtenido: *Éxito.*

   d) **Error en conexión a blockchain**:

   - Entrada: Simulación de fallo en la conexión con Hyperledger Besu.
   - Resultado esperado: Reintento automático y, si persiste el fallo, registro detallado del error en Application Insights.
   - Resultado obtenido: *Éxito con manejo de errores.*

   e) **Integridad de la firma criptográfica**:

- Entrada: Validación de la firma criptográfica generada por la función Azure con las claves privadas almacenadas en Azure Key Vault.
- Resultado esperado: Firma correcta y validable por nodos autorizados en blockchain.
- Resultado obtenido: *Éxito.*

4. Conclusión de las pruebas:

   Las pruebas realizadas sobre la función Registrar en Blockchain han validado la robustez y fiabilidad del sistema implementado, garantizando el registro seguro y fiable de las clasificaciones en una blockchain privada. La función ha demostrado un manejo adecuado de mensajes en cola, procesamiento de datos y gestión efectiva de excepciones, consolidándose como una pieza clave del sistema en producción.

A.5. Función RecuperarTransacciones

1. Objetivo:

   Verificar el correcto funcionamiento y seguridad de la función Azure que permite a un usuario autenticado descargar un archivo CSV con el historial de sus clasificaciones de personalidad almacenadas en blockchain.

2. Procedimiento:

   Las pruebas unitarias y funcionales efectuadas abarcaron:

   - Autenticación y autorización basada en JWT.
   - Recuperación y descifrado seguro de datos.
   - Generación y envío del archivo CSV al cliente.
   - Manejo de errores y excepciones.

3. Casos de prueba:

   Se han definido y ejecutado los siguientes casos de prueba clave:

   a) **Solicitud autorizada y válida**:

   - Entrada: JWT token válido en la cabecera HTTP.
   - Resultado esperado: Recuperación de transacciones, creación del CSV y descarga correcta en el cliente.
   - Resultado obtenido: *Éxito.*

   b) **Solicitud no autorizada (sin token)**:

   - Entrada: Solicitud HTTP sin token JWT.
   - Resultado esperado: Respuesta HTTP 401 (Unauthorized).
   - Resultado obtenido: *Éxito.*

c) **Token inválido o manipulado**:

- Entrada: Token JWT alterado o inválido.
- Resultado esperado: Respuesta HTTP 401 (Unauthorized).
- Resultado obtenido: *Éxito.*

d) **Usuario sin transacciones registradas**:

- Entrada: Usuario autenticado sin registros previos.
- Resultado esperado: Respuesta HTTP 404 indicando ausencia de datos.
- Resultado obtenido: *Éxito.*

e) **Error al acceder a Azure Table Storage o Key Vault**:

- Entrada: Simulación de fallo en el acceso a recursos Azure.
- Resultado esperado: Captura del error con respuesta HTTP 500 y registro detallado en Application Insights.
- Resultado obtenido: *Éxito.*

4. Conclusión de las pruebas:

   La función para descargar clasificaciones del usuario superó exitosamente todos los casos definidos. Se confirmó la robustez en la autenticación y seguridad mediante JWT, además de una adecuada gestión de errores, garantizando que únicamente usuarios autorizados tengan acceso a sus propias clasificaciones en la blockchain privada.

A.6. Función EncriptaryGuardar

1. Objetivo:

   Verificar que la función Azure encargada de recoger mensajes desde una cola, cifrarlos con la clave específica de cada usuario y almacenarlos en el historial cifrado en Azure Blob Storage funciona correctamente.

2. Procedimiento:

   Se llevaron a cabo pruebas específicas para garantizar:

   - Recepción adecuada de mensajes desde Azure Queue Storage.
   - Correcta recuperación de la clave de cifrado del usuario desde Azure Key Vault.
   - Cifrado AES-GCM de los mensajes con la clave correspondiente.
   - Almacenamiento de mensajes cifrados en Azure Blob Storage.
   - Envío del mensaje a la cola para activar la clasificación tras alcanzar cierto número de interacciones.

3. Casos de prueba:

   Se definieron los siguientes casos de prueba:

   a) **Mensaje correctamente recibido desde la cola**:

   - Entrada: Mensaje JSON válido con contenido de interacción del usuario.
   - Resultado esperado: Mensaje recibido y procesado sin errores.
   - Resultado obtenido: *Éxito*.

   b) **Recuperación de clave del usuario desde Azure Key Vault**:

   - Entrada: ID del usuario incluido en el mensaje.
   - Resultado esperado: Recuperación exitosa de la clave desde Azure Key Vault.
   - Resultado obtenido: *Éxito*.

   c) **Cifrado correcto de mensajes**:

   - Entrada: Texto del mensaje y clave recuperada.
   - Resultado esperado: Cifrado AES-GCM correcto.
   - Resultado obtenido: *Éxito*.

   d) **Almacenamiento en Azure Blob Storage**:

   - Entrada: Mensaje cifrado y datos del usuario.
   - Resultado esperado: Guardado correcto del mensaje cifrado en el historial del usuario en Blob Storage.
   - Resultado obtenido: *Éxito*.

   e) **Activación de función de clasificación por número de interacciones**:

   - Entrada: Número de interacciones límite alcanzado.
   - Resultado esperado: Mensaje enviado correctamente a la cola de clasificación.
   - Resultado obtenido: *Éxito*.

4. Conclusión de las pruebas:

   La función ha superado satisfactoriamente todas las pruebas, demostrando una correcta gestión del cifrado personalizado por usuario y un almacenamiento seguro y eficiente. Asimismo, asegura que se activen los procesos posteriores de clasificación automáticamente tras acumular el número predefinido de interacciones.

A.7. Función SessionEndLoggerFunction

1. Objetivo:

   Validar que la función Azure encargada de registrar el evento de finalización de sesión (`SessionEnd`) funcione correctamente, enviando la información de sesión del usuario (ContactID, duración y página) a Application Insights de forma segura y confiable.

2. Procedimiento:

   Se ejecutaron pruebas unitarias y funcionales para confirmar:

   - Validación adecuada de los datos recibidos.
   - Envío correcto de la información a Application Insights.
   - Seguridad en la restricción de orígenes permitidos mediante CORS.

3. Casos de prueba:

   Se definieron y realizaron los siguientes casos representativos:

   a) **Registro exitoso del evento SessionEnd**:

   - Entrada: JSON válido con ContactID, Duration y Página.
   - Resultado esperado: Registro exitoso del evento en Application Insights.
   - Resultado obtenido: *Éxito.*

   b) **Solicitud con origen no permitido**:

   - Entrada: Solicitud HTTP desde un origen no autorizado.
   - Resultado esperado: Respuesta HTTP 403 (Forbidden).
   - Resultado obtenido: *Éxito.*

   c) **Solicitud con método OPTIONS (preflight CORS)**:

   - Entrada: Solicitud HTTP OPTIONS para validación CORS.
   - Resultado esperado: Respuesta HTTP 200 con cabeceras CORS adecuadas.
   - Resultado obtenido: *Éxito.*

   d) **Solicitud con datos incompletos**:

   - Entrada: Solicitud JSON faltando alguno de los datos requeridos (ContactID, Duration o Página).
   - Resultado esperado: Respuesta HTTP 400 indicando claramente los datos faltantes.
   - Resultado obtenido: *Éxito.*

   e) **Error al enviar a Application Insights**:

- Entrada: Simulación de fallo en la conexión con Application Insights (por ejemplo, clave de instrumentación errónea).
- Resultado esperado: Captura del error y respuesta HTTP 500, registrando el incidente en logs.
- Resultado obtenido: *Éxito.*

4. Conclusión de las pruebas:

   La función `SessionEndLoggerFunction` ha superado satisfactoriamente todos los casos de prueba establecidos. Los resultados garantizan que el registro del evento `SessionEnd` se realiza de forma segura, confiable y efectiva, asegurando la calidad de los datos recogidos para análisis posteriores mediante Application Insights.

A.8. Función Encolar

1. Objetivo:

   Verificar el correcto funcionamiento de la función Azure encargada de recibir solicitudes HTTP desde Power Pages, identificar al usuario mediante un token JWT, y encolar el mensaje enviado (junto con su rol) en formato codificado base64 en la cola `guardar-mensajes`, para su posterior cifrado y almacenamiento.

2. Procedimiento:

   Se han diseñado y ejecutado pruebas para comprobar los siguientes aspectos:

   - Validación de origen permitido (`CORS`).
   - Comprobación de token JWT y extracción segura del identificador de usuario.
   - Validación de los campos obligatorios (`user_id`, `message`, `role`).
   - Envío exitoso del mensaje codificado a Azure Queue.
   - Manejo correcto de errores (falta de token, campos incompletos, excepciones internas).

3. Casos de prueba:

   a) **Solicitud válida con JWT y mensaje**:

   - Entrada: Solicitud HTTP POST con cabecera JWT válida y cuerpo con campos `message` y `role`.
   - Resultado esperado: Envío correcto del mensaje a la cola en formato base64.
   - Resultado obtenido: *Éxito.*

   b) **Falta de token de autorización**:

- Entrada: Solicitud sin cabecera `Authorization`.
- Resultado esperado: Código HTTP 401 – Unauthorized.
- Resultado obtenido: *Éxito.*

c) **Token inválido o malformado**:

- Entrada: Token JWT incorrecto o manipulado.
- Resultado esperado: Código HTTP 401 – Unauthorized.
- Resultado obtenido: *Éxito.*

d) **Campos incompletos en el body**:

- Entrada: Falta alguno de los campos `message` o `role`.
- Resultado esperado: Código HTTP 400 – Bad Request.
- Resultado obtenido: *Éxito.*

e) **Origen no permitido (CORS)**:

- Entrada: Cabecera `Origin` distinta a la URL permitida.
- Resultado esperado: Código HTTP 403 – Forbidden.
- Resultado obtenido: *Éxito.*

4. Conclusión de las pruebas:

   La función ha superado satisfactoriamente todos los escenarios definidos, garantizando que solo usuarios autenticados puedan registrar mensajes, y que estos se envíen de forma segura y codificada a la cola. Este componente es esencial para el correcto almacenamiento del historial cifrado de cada usuario, y actúa como interfaz de confianza entre el frontend y el pipeline de persistencia de datos.

Apéndice B

Evaluación automática de respuestas del asistente principal

B.1. Asistente evaluador ADSM

Con el fin de garantizar la coherencia, corrección y alineación del comportamiento del asistente principal con las instrucciones establecidas en el documento "ANALISTA YODAI INICIAL.docx", se diseñó e implementó un **asistente evaluador automático**. Este componente actúa como control de calidad interno del sistema y permite realizar auditorías objetivas de las respuestas generadas por el asistente encargado de la derivación.

1. Ámbito de aplicación:

   Este evaluador ha sido utilizado principalmente para analizar las respuestas del **asistente inicial** del sistema, responsable de derivar correctamente al usuario hacia una de las seis herramientas ADSM disponibles (MDS-YO, IPP, PCN, SMV, AP, VP). El uso del asistente evaluador en este punto resulta crítico, ya que una derivación inadecuada puede comprometer todo el flujo posterior.

   En cambio, los asistentes específicos correspondientes a cada herramienta ADSM han sido verificados de forma manual con la **supervisión directa de Gerardo Tudurí**, autor del modelo ADSM. Su experiencia como creador de los protocolos ha permitido validar con precisión el cumplimiento de las instrucciones en los asistentes especializados.

2. Instrucciones del asistente evaluador:

   El asistente evaluador recibe las respuestas emitidas por el asistente principal y analiza su cumplimiento respecto a:

   - El protocolo de derivación ADSM.

- Los errores comunes identificados en el documento base.
- El tono, claridad y estilo conversacional esperado.

El formato de evaluación generado por este asistente sigue una estructura estandarizada con los siguientes campos:

a) Coherencia con el protocolo.

b) Derivación correcta.

c) Errores detectados.

d) Estilo de comunicación adecuado.

e) Sugerencias de mejora.

B.1.1. Ejemplos de uso del asistente evaluador

1. Ejemplo 1: Derivación correcta a MDS-YO
   - **Respuesta evaluada:** *"Gracias por compartir lo que estás viviendo. Por lo que comentas, parece que estás atravesando una situación con múltiples dimensiones implicadas y algo de confusión entre ellas. Podría ser útil empezar con la herramienta MDS-YO, que te permitirá identificar con claridad los escenarios personales que estás manejando actualmente."*
   - **Evaluación:**
     - **Coherencia con el protocolo:** Sí *Justificación:* El asistente identifica correctamente una situación con múltiples dimensiones como indicio para derivar a MDS-YO.
     - **Derivación correcta:** Sí *Justificación:* Se deriva a MDS-YO según los criterios establecidos en el protocolo (complejidad o confusión situacional).
     - **Errores detectados:** Ninguno
     - **Estilo de comunicación adecuado:** Sí *Observaciones:* Mensaje claro, empático, directo y con validación del usuario.
     - **Sugerencias de mejora:** Ninguna.
2. Ejemplo 2: Error de explicación de la herramienta:
   - **Respuesta evaluada:** *"Gracias por tu mensaje. Te voy a derivar al IPP, que es una herramienta que mide cómo te sientes en distintas áreas de tu vida. Se basa en una tabla que evalúa relaciones, propósito, energía..."*
   - **Evaluación:**
     - **Coherencia con el protocolo:** Sí *Justificación:* La derivación al IPP podría ser adecuada según el contenido.

- **Derivación correcta:** Sí *Justificación:* La situación parece justificar el uso del IPP.
- **Errores detectados:**
  - Explicó el funcionamiento del IPP.
- **Estilo de comunicación adecuado:** Parcial *Observaciones:* Aunque empático, el mensaje incurre en un rol incorrecto al detallar el funcionamiento de la herramienta.
- **Sugerencias de mejora:**
  - Eliminar la explicación de la herramienta.
  - Mantener el rol de derivador sin anticipar el proceso.

3. Ejemplo 3: Derivación ambigua sin justificación:
   - **Respuesta evaluada:** *"Creo que lo mejor es que vayas directamente al VP para trabajar en tu visión futura. Esto te va a ayudar mucho."*
   - **Evaluación:**
     - **Coherencia con el protocolo:** No *Justificación:* La elección del VP no se fundamenta en elementos identificables del mensaje del usuario.
     - **Derivación correcta:** No *Justificación:* No se justifica la elección del VP según los criterios definidos en el protocolo.
     - **Errores detectados:**
       - Derivación sin justificar claramente la elección.
       - Tono excesivamente directivo.
     - **Estilo de comunicación adecuado:** No *Observaciones:* Carece de validación al usuario y se salta el proceso de análisis.
     - **Sugerencias de mejora:**
       - Argumentar siempre en base al contenido del usuario y el protocolo.
       - Evitar recomendaciones impositivas.

4. Conclusión:

   El uso del evaluador automático ha resultado eficaz para garantizar que el asistente inicial cumple con los criterios de derivación establecidos por el modelo ADSM. En paralelo, la revisión experta de los asistentes específicos por parte de Gerardo Tudurí ha permitido asegurar una implementación fiel y rigurosa de cada uno de los protocolos en producción.

Apéndice C

Entrenamiento del clasificador OCEAN

C.1. Objetivo

El objetivo de este módulo es entrenar un modelo capaz de predecir las puntuaciones de personalidad según el modelo OCEAN (Big Five: **O** – Apertura, **C** – Responsabilidad, **E** – Extraversión, **A** – Amabilidad, **N** – Neuroticismo), a partir del historial conversacional del usuario.

C.2. Entorno de ejecución

El proceso de entrenamiento se llevó a cabo en Azure Machine Learning, empleando un entorno configurado con `scikit-learn`, `spaCy` y `joblib`, entre otros paquetes.

C.3. Descripción del proceso

El código de entrenamiento se encuentra en el archivo `train_ocean_classifier.py`. El flujo general seguido es el siguiente:

1. Lectura del dataset

Se carga un fichero CSV ubicado en la carpeta de entrada especificada. Este contiene columnas con etiquetas OCEAN, nombre del usuario y texto conversacional.

2. Preprocesamiento del texto

Se combinan los campos `Persona` y `chat`, aplicando un procesamiento lingüístico con el modelo `es_core_news_md` de `spaCy`. Se eliminan stopwords personalizadas (“vale”, “mmm”, “eh”...), se lematiza y se conservan puntuaciones relevantes.

3. Vectorización

Se utiliza un vectorizador `TfidfVectorizer` con un máximo de 5000 características, entrenado sobre el conjunto de entrenamiento. Este transforma el texto en una matriz numérica de características.

4. Entrenamiento del modelo

Se entrena un modelo `MultiOutputRegressor` basado en regresión logística, que permite predecir simultáneamente los cinco rasgos OCEAN.

5. Evaluación

Se evalúa el modelo sobre un conjunto de test mediante el error cuadrático medio (`MSE`) entre los valores reales y predichos.

6. Persistencia del modelo

Finalmente, se almacenan los objetos serializados:

- `modelo_ocean.pkl`: modelo entrenado.
- `vectorizer.pkl`: vectorizador de texto.

Ambos son subidos al sistema de archivos del entorno de ejecución para su posterior uso en inferencia.

C.4. Dataset utilizado

Se utilizó una muestra representativa de 500 entradas etiquetadas, almacenada en el fichero `subset_500.csv`. Este dataset fue generado mediante un modelo generativo evaluador y contiene conversaciones simuladas clasificadas por perfil de personalidad.

C.5. Conclusión

Este módulo permite obtener una clasificación OCEAN fiable a partir del historial conversacional del usuario, habilitando análisis longitudinales y registros trazables mediante blockchain. Su entrenamiento modular y su despliegue serverless lo hacen adecuado para entornos de producción.

El código fuente completo se incluye al final del presente anexo.

Listado de código fuente

A continuación se presenta el script completo de entrenamiento utilizado en Azure Machine Learning para construir el clasificador OCEAN:

```
import argparse
import pandas as pd
import torch
from torch.utils.data import Dataset
from transformers import AutoTokenizer, AutoModelForSequenceClassification,
    ↪ Trainer, TrainingArguments
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score,
    ↪ f1_score

class OceanDataset(Dataset):
    def __init__(self, texts, labels, tokenizer, max_length=128):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        encoding = self.tokenizer(
            self.texts[idx],
            truncation=True,
            padding='max_length',
            max_length=self.max_length,
            return_tensors='pt',
        )
        item = {key: val.squeeze(0) for key, val in encoding.items()}
        item['labels'] = torch.tensor(self.labels[idx], dtype=torch.float)
        return item

def compute_metrics(eval_pred):
    logits, labels = eval_pred
    preds = torch.sigmoid(torch.tensor(logits)).numpy()
    labels = labels.astype(float)

    mse = mean_squared_error(labels, preds)
    mae = mean_absolute_error(labels, preds)
    r2 = r2_score(labels, preds)

    # F1 Macro (con binarización simple)
    f1 = f1_score(labels > 0.5, preds > 0.5, average='macro')

    return {
        'mse': mse,
        'mae': mae,
        'r2': r2,
        'f1_macro': f1
    }

def main(args):
    # Cargar datos
    df = pd.read_csv(args.data_path)
    df = df.dropna(subset=["O", "C", "E", "A", "N", "Persona", "chat"])
    df["text"] = df["Persona"].fillna('') + "\n" + df["chat"].fillna('')
    texts = df["text"].tolist()
```

```
    labels = df[["O", "C", "E", "A", "N"]].astype(float).values

    # Dividir datos
    train_texts, val_texts, train_labels, val_labels = train_test_split(
        texts, labels, test_size=0.2, random_state=42
    )

    tokenizer = AutoTokenizer.from_pretrained("xlm-roberta-base")
    train_dataset = OceanDataset(train_texts, train_labels, tokenizer)
    val_dataset = OceanDataset(val_texts, val_labels, tokenizer)

    model = AutoModelForSequenceClassification.from_pretrained(
        "xlm-roberta-base", num_labels=5, problem_type="regression"
    )

    training_args = TrainingArguments(
        output_dir=args.output_dir,
        evaluation_strategy="epoch",
        save_strategy="epoch",
        learning_rate=2e-5,
        per_device_train_batch_size=16,
        per_device_eval_batch_size=16,
        num_train_epochs=3,
        weight_decay=0.01,
        load_best_model_at_end=True,
        metric_for_best_model="mse",
        greater_is_better=False,
        logging_dir=f"{args.output_dir}/logs"
    )

    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=val_dataset,
        compute_metrics=compute_metrics,
    )

    trainer.train()
    trainer.save_model(args.output_dir)
    tokenizer.save_pretrained(args.output_dir)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("--data_path", type=str, required=True)
    parser.add_argument("--output_dir", type=str, default="./outputs")
    args = parser.parse_args()
    main(args)
```

Listing C.1: Script de entrenamiento `train_ocean_classifier.py`

Apéndice D

Manual Técnico Completo del Sistema

Este capítulo describe detalladamente el funcionamiento técnico del sistema desarrollado, incluyendo tanto el backend (funciones en Azure Functions) como el frontend (interfaz implementada en Power Pages). También se documentan aspectos adicionales relevantes para la puesta en marcha y mantenimiento del sistema.

D.1. Arquitectura general del sistema

El sistema se basa en una arquitectura serverless, modular y segura. Los componentes principales son:

- Power Pages (Frontend): interfaz web pública y autenticada.
- Azure AD B2C: autenticación de usuarios.
- Azure Functions: backend sin servidor con múltiples funciones especializadas.
- Azure Blob Storage: almacenamiento cifrado de historiales y clasificaciones.
- Azure Table Storage: metadatos por usuario.
- Azure Queue Storage: coordinación de procesos.
- Azure Application Insights: registro de métricas y eventos.
- Hyperledger Besu (Blockchain): trazabilidad de clasificaciones agregadas.

D.2. Despliegue del backend en Azure Functions

Cada función se ha desplegado en un entorno de Azure Functions con el runtime Python 3.10. Se ha usado Visual Studio Code con la extensión de Azure Functions y la CLI de Azure para la publicación.

D.2.1.

Pasos de despliegue típico

1. Inicializar el proyecto con ‘func init‘.
2. Crear cada función con ‘func new‘ y definir el trigger correspondiente (HTTP, Queue, Timer, etc).
3. Configurar los archivos ‘function.json‘ y variables de entorno (‘local.settings.json‘ en desarrollo, Application Settings en producción).
4. Probar localmente con ‘func start‘.
5. Publicar al entorno de Azure con ‘func azure functionapp publish <nombre>‘.

D.3. Funciones del Backend

D.3.1. Función StreamingResp

```
import logging
import azure.functions as func
from azure.messaging.webpubsubservice import WebPubSubServiceClient
from openai import OpenAI
from azure.data.tables import TableServiceClient
from azure.data.tables import UpdateMode
from azure.identity import DefaultAzureCredential
import json
import os
import jwt

# Configuración de la conexión con Web PubSub y OpenAI
HUB_NAME = "ADSM_HUB"
HUB_URL = os.getenv("HUB_URL")
credential = DefaultAzureCredential()
clientGPT = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

# Configuración de Table Storage
TABLE_URL = os.getenv("TABLES_URL")
TABLE_NAME = "Threads"
table_service = TableServiceClient(endpoint=TABLE_URL, credential=credential)

ALLOWED_ORIGIN = "https://pilotoyodai.powerappsportals.com"

def get_table_entity(user_id):
    table_client = table_service.get_table_client(table_name=TABLE_NAME)
    entity = table_client.get_entity(partition_key="threads", row_key=user_id)
    return entity, table_client

def update_table_entity(table_client, user_id, updates):
    updates["PartitionKey"] = "threads"
    updates["RowKey"] = user_id
    table_client.upsert_entity(entity=updates, mode=UpdateMode.MERGE)


# Recuperar o crear thread desde Table Storage
```

```
def get_or_create_thread(user_id):
    table_client = table_service.get_table_client(table_name=TABLE_NAME)

    try:
        entity = table_client.get_entity(partition_key="threads", row_key=
    ↪ user_id)
        return entity["thread_id"]
    except:
        nuevo_thread = clientGPT.beta.threads.create()
        table_client.upsert_entity({
            "PartitionKey": "threads",
            "RowKey": user_id,
            "thread_id": nuevo_thread.id
        })
        return nuevo_thread.id

def main(req: func.HttpRequest) -> func.HttpResponse:
    origin = req.headers.get("Origin")
    if origin != ALLOWED_ORIGIN:
        return func.HttpResponse(
            "Forbidden: origin not allowed",
            status_code=403,
            headers={
                'Access-Control-Allow-Origin': 'null',
                'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type'
            }
        )

    logging.info('Procesando solicitud HTTP.')

    if req.method == "OPTIONS":
        return func.HttpResponse(
            "",
            status_code=204,
            headers={
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': 'POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type, user.id,
    ↪ assistant.id'
            }
        )

    token = req.headers.get("Authorization")
    if not token:
        return func.HttpResponse("Unauthorized", status_code=401)

    try:
        # Usar principal_id como ContactID seguro
        payload = jwt.decode(token, options={"verify_signature": False})
        user_id = payload.get("sub")
    except Exception as e:
        logging.error(f"Error: {e}", exc_info=True)
        return func.HttpResponse("Unauthorized", status_code=401)

    assistant_id = req.headers.get('assistant.id')
    if not assistant_id or not user_id:
        return func.HttpResponse(
```

```
            body="INVALID JSON",
            status_code=400,
            headers={
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': 'POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type, user.id,
↪ assistant.id'
            }
        )

    try:
        req_body = req.get_json()
        user_message = req_body.get('message')

        if not user_message:
            return func.HttpResponse(
                body="No se proporcionó ningún mensaje.",
                status_code=400,
                headers={
                    'Access-Control-Allow-Origin': '*',
                    'Access-Control-Allow-Methods': 'POST, OPTIONS',
                    'Access-Control-Allow-Headers': 'Content-Type, user.id,
↪ assistant.id'
                }
            )

    except ValueError:
        return func.HttpResponse(
            body="INVALID JSON",
            status_code=400,
            headers={
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': 'POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type, user.id,
↪ assistant.id'
            }
        )

    try:
        logging.info("Seleccionar Thread")
        thread_id = get_or_create_thread(user_id)
        logging.info(f"Thread seleccionado: {thread_id}")
        entity, table_client = get_table_entity(user_id)
        thread_id = entity["thread_id"]
        vector_store_id = entity.get("vector_store_id")
        logging.info(f"VectorStoreID: {vector_store_id}")

        message = clientGPT.beta.threads.messages.create(
            thread_id=thread_id,
            role="user",
            content=user_message
        )

        client = WebPubSubServiceClient(endpoint=HUB_URL, hub=HUB_NAME,
↪ credential=credential)
        group_name = f"usuario_{user_id}"

        logging.info(f"Grupo de Web PubSub: {group_name}")
```

```
    run = clientGPT.beta.threads.runs.create(
        thread_id=thread_id,
        assistant_id=assistant_id,
        stream=True,
        tool_choice={"type": "file_search"},  # fuerza a usar esa
↪ herramienta
        tools=[
            {
                "type": "file_search",
                "file_search": {
                    "max_num_results": 10,
                    "ranking_options": {
                        "score_threshold": 0.5,
                        "ranker": "default_2024_08_21"
                    }
                }
            }
        ]
    )


    for chunk in run:
        logging.info(f"Chunk recibido: {chunk}")

        try:
            if hasattr(chunk, 'data') and hasattr(chunk.data, 'delta'):
                delta = chunk.data.delta
                if hasattr(delta, 'content'):
                    for block in delta.content:
                        if block.type == 'text' and hasattr(block.text, '
↪ value'):
                            response_content = block.text.value
                            client.send_to_group(
                                group=group_name,
                                message=json.dumps({"type": "message", "
↪ content": response_content}),
                                content_type="application/json"
                            )

        except Exception as e:
            logging.error(f"Error procesando el fragmento: {e}")

    client.send_to_group(
        group=group_name,
        message=json.dumps({"type": "confirmation", "content": "[
↪ STREAM_FINISHED]"}),
        content_type="application/json"
    )

    return func.HttpResponse("Streaming completado.", status_code=200,
        headers={
            'Access-Control-Allow-Origin': '*',
            'Access-Control-Allow-Methods': 'POST, OPTIONS',
            'Access-Control-Allow-Headers': 'Content-Type, user.id,
↪ assistant.id'
        }
    )
```

```
    except Exception as e:
        logging.error(f"Error procesando la solicitud: {e}")
        return func.HttpResponse(
            f"Error procesando la solicitud: {e}",
            status_code=500,
            headers={
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': 'POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type, user.id,
    ↪ assistant.id'
            }
        )
```

Listing D.1: Función StreamingResp

D.3.2. Función GetWebSocketURL

```
import logging
import azure.functions as func
from azure.messaging.webpubsubservice import WebPubSubServiceClient
import os
import jwt
from azure.identity import DefaultAzureCredential

HUB_NAME = "ADSM_HUB"
HUB_URL = os.getenv("HUB_URL")
credential = DefaultAzureCredential()
client = WebPubSubServiceClient(endpoint=HUB_URL, hub=HUB_NAME, credential=
    ↪ credential)
ALLOWED_ORIGIN = "https://pilotoyodai.powerappsportals.com"

def main(req: func.HttpRequest) -> func.HttpResponse:
    origin = req.headers.get("Origin")
    if origin != ALLOWED_ORIGIN:
        return func.HttpResponse(
            "Forbidden: origin not allowed",
            status_code=403,
            headers={
                'Access-Control-Allow-Origin': 'null',
                'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type'
            }
        )

    if req.method == "OPTIONS":
        # Preflight CORS request
        return func.HttpResponse(
            status_code=200,
            headers={
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type, user.id,
    ↪ assistant.id'
            }
        )
```

```
    token = req.headers.get("Authorization")
    if not token:
        return func.HttpResponse("Unauthorized", status_code=401)

    try:
        # Usar principal_id como ContactID seguro
        payload = jwt.decode(token, options={"verify_signature": False})
        user_id = payload.get("sub")
    except Exception as e:
        logging.error(f"Error: {e}", exc_info=True)
        return func.HttpResponse("Unauthorized", status_code=401)

    assistant_id = req.headers.get('assistant.id')
    if not user_id or not assistant_id:
        return func.HttpResponse(
            "Missing user_id",
            status_code=400,
            headers={
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type'
            }
        )

    try:
        # Generar token con un tiempo de vida
        token = client.get_client_access_token(
            roles=["webpubsub.joinLeaveGroup", "webpubsub.sendToGroup", "
    ↪ webpubsub.sendToConnection"],
            groups=[f"usuario_{user_id}"]
        )


        # Devuelve la URL del token
        return func.HttpResponse(
            token['url'],
            status_code=200,
            headers={
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type'
            }
        )
    except Exception as e:
        logging.error(f"Error generating token: {e}")
        return func.HttpResponse(
            "Error generating token",
            status_code=500,
            headers={
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type'
            }
        )
```

Listing D.2: Función GetWebSocketURL

D.3.3. Función ClasificarUsuario

```
import logging
import os
import json
from datetime import datetime, timezone, timedelta
from azure.storage.blob import BlobServiceClient
from azure.storage.queue import QueueClient
from azure.identity import DefaultAzureCredential
from azure.data.tables import TableServiceClient, UpdateMode
from azure.keyvault.secrets import SecretClient
from cryptography.hazmat.primitives.ciphers.aead import AESGCM
import azure.functions as func
import time
import base64
import spacy
import torch
from transformers import AutoTokenizer, AutoModelForSequenceClassification

# --- Configuración inicial ---
KEY_VAULT_URL = os.getenv("KEY_VAULT_URL")
TABLE_URL = os.getenv("TABLES_URL")
STORAGE_URL = os.getenv("STORAGE_URL")
QUEUE_URL = os.getenv("QUEUE_URL")
CONTAINER_NAME = "conversaciones"
QUEUE_NAME = "clasificar-usuario"
MODELS_CONTAINER = "workspaceblobstore"
MODEL_BLOB_PREFIX = "sincere_grass_4kxr0s5fdx/output_dir"
TEMP_MODEL_DIR = "/tmp/ocean_model"
ESPERA_MINUTOS = int(os.getenv("ESPERA_MINUTOS", "5"))

credential = DefaultAzureCredential()
blob_service_client = BlobServiceClient(account_url=STORAGE_URL, credential=
    ↪ credential)
queue_client = QueueClient(account_url=QUEUE_URL, queue_name=QUEUE_NAME,
    ↪ credential=credential)
blockchain_queue_client = QueueClient(account_url=QUEUE_URL, queue_name="
    ↪ registrar-blockchain", credential=credential)
secret_client = SecretClient(vault_url=KEY_VAULT_URL, credential=credential)

table_service = TableServiceClient(endpoint=TABLE_URL, credential=credential)
table_client = table_service.get_table_client(table_name="ClasificacionesSesion
    ↪ ")
user_table_client = table_service.get_table_client(table_name="Threads")

# --- Cargar modelo transformer desde blob ---
def download_blob_to_local_dir(blob_name: str, local_dir: str):
    os.makedirs(local_dir, exist_ok=True)
    blob_client = blob_service_client.get_blob_client(container=
    ↪ MODELS_CONTAINER, blob=blob_name)
    local_path = os.path.join(local_dir, os.path.basename(blob_name))
    with open(local_path, "wb") as f:
        f.write(blob_client.download_blob().readall())

def load_transformer_model_from_experiment_blob():
    files = [
        "config.json",
        "tokenizer_config.json",
        "tokenizer.json",
```

```
        "special_tokens_map.json",
        "model.safetensors"
    ]
    for file in files:
        download_blob_to_local_dir(f"{MODEL_BLOB_PREFIX}/{file}",
    ↪ TEMP_MODEL_DIR)
    tokenizer = AutoTokenizer.from_pretrained(TEMP_MODEL_DIR)
    model = AutoModelForSequenceClassification.from_pretrained(TEMP_MODEL_DIR)
    return tokenizer, model

tokenizer, model = load_transformer_model_from_experiment_blob()

# --- Cargar SpaCy ---
nlp = spacy.load("es_core_news_md")
stopwords_custom = {"eh", "mmm", "ah", "ok", "vale", "sí", "no", "bueno"}

# --- Funciones auxiliares ---
def get_user_key(user_id: str) -> bytes:
    secret = secret_client.get_secret(f"aes-key-{user_id}")
    return base64.b64decode(secret.value)

def decrypt_message(key: bytes, encrypted: bytes) -> str:
    nonce = encrypted[:12]
    ciphertext = encrypted[12:]
    return AESGCM(key).decrypt(nonce, ciphertext, None).decode("utf-8")

def obtener_fecha_modificacion(blob_client):
    return blob_client.get_blob_properties().last_modified

def get_lines_processed(user_id):
    try:
        entity = user_table_client.get_entity(partition_key="threads", row_key=
    ↪ user_id)
        return entity.get("lines_processed", 0)
    except:
        return 0

def update_lines_processed(user_id, new_count):
    user_table_client.upsert_entity({
        "PartitionKey": "threads",
        "RowKey": user_id,
        "lines_processed": new_count
    }, mode=UpdateMode.MERGE)

def update_table_entity(user_id, updates):
    updates["PartitionKey"] = "threads"
    updates["RowKey"] = user_id
    user_table_client.upsert_entity(entity=updates, mode=UpdateMode.MERGE)

def leer_conversacion(blob_client, user_id, start_line=0):
    content = blob_client.download_blob().readall()
    key = get_user_key(user_id)
    mensajes = []
    for i, line in enumerate(content.split(b"==DELIM==")):
        if i < start_line:
            continue
        if line.strip():
            try:
```

```
                mensajes.append(decrypt_message(key, line))
            except Exception as e:
                mensajes.append(f"[Error al descifrar línea {i}]: {e}")
    return "\n".join(mensajes), len(content.split(b"==DELIM=="))

def preprocesar(texto):
    doc = nlp(texto.lower())
    return " ".join(
        token.lemma_ if not token.is_punct else token.text
        for token in doc
        if not token.is_space and token.lemma_ not in stopwords_custom
    )

def clasificar(texto):
    inputs = tokenizer(texto, return_tensors="pt", truncation=True, max_length
    ↪ =512)
    with torch.no_grad():
        outputs = model(**inputs)
    return outputs.logits.squeeze().tolist()

def guardar_clasificacion(user_id, scores):
    entity = {
        "PartitionKey": user_id,
        "RowKey": datetime.now(timezone.utc).isoformat(),
        "O": scores[0],
        "C": scores[1],
        "E": scores[2],
        "A": scores[3],
        "N": scores[4]
    }
    table_client.upsert_entity(entity=entity, mode=UpdateMode.MERGE)

# --- Función principal ---
def main(msg: func.QueueMessage) -> None:
    try:
        payload = json.loads(msg.get_body().decode('utf-8'))
        user_id = payload.get("user_id")
        if not user_id:
            logging.error("Falta el campo 'user_id'")
            return

        blob_client = blob_service_client.get_blob_client(container=
    ↪ CONTAINER_NAME, blob=f"{user_id}.bin")
        fecha_mod = obtener_fecha_modificacion(blob_client)
        ahora = datetime.now(timezone.utc)

        if ahora - fecha_mod < timedelta(minutes=ESPERA_MINUTOS):
            logging.info(" Archivo muy reciente, reencolando tras %d minutos
    ↪ ...", ESPERA_MINUTOS)
            payload_json = json.dumps(payload)
            payload_base64 = base64.b64encode(payload_json.encode("utf-8")).
    ↪ decode("utf-8")
            queue_client.send_message(payload_base64, visibility_timeout=
    ↪ ESPERA_MINUTOS * 60)
            return

        start_line = get_lines_processed(user_id)
```

```
        texto, total_lines = leer_conversacion(blob_client, user_id, start_line
    ↪ =start_line)
        texto_preprocesado = preprocesar(texto)
        scores = clasificar(texto_preprocesado)
        guardar_clasificacion(user_id, scores)
        update_lines_processed(user_id, total_lines)

        logging.info(" Clasificación OCEAN generada y almacenada correctamente
    ↪ para %s.", user_id)

        entity = user_table_client.get_entity(partition_key="threads", row_key=
    ↪ user_id)
        classification_count = entity.get("classification_count", 0) + 1
        update_table_entity(user_id, {"classification_count":
    ↪ classification_count})

        if classification_count >= 5:
            queue_payload = {"user_id": user_id}
            message_json = json.dumps(queue_payload)
            encoded_message = base64.b64encode(message_json.encode("utf-8")).
    ↪ decode("utf-8")
            blockchain_queue_client.send_message(encoded_message)
            logging.info(f" Enviada solicitud de registro blockchain para {
    ↪ user_id}.")
            update_table_entity(user_id, {"classification_count": 0})

    except Exception as e:
        logging.error(" Error en la función de clasificación: %s", str(e),
    ↪ exc_info=True)
```

Listing D.3: Función ClasificarUsuario

D.3.4. Función RegistrarBlockChain

```
import logging
import os
import json
from datetime import datetime, timezone
import azure.functions as func
from azure.data.tables import TableServiceClient, TableTransactionError
from azure.core.exceptions import ResourceNotFoundError
from azure.identity import DefaultAzureCredential
from azure.keyvault.secrets import SecretClient
from web3 import Web3, HTTPProvider
from eth_account import Account
from cryptography.hazmat.primitives.ciphers.aead import AESGCM
import base64
import hashlib

# Configuración
KEY_VAULT_URL = os.getenv("KEY_VAULT_URL")
TABLE_URL = os.getenv("TABLES_URL")
TABLE_NAME = "ClasificacionesSesion"
BLOCKCHAIN_URL = os.getenv("BLOCKCHAIN_URL")
CENTRAL_PRIVATE_KEY = os.getenv("CENTRAL_PRIVATE_KEY")
credential = DefaultAzureCredential()
secret_client = SecretClient(vault_url=KEY_VAULT_URL, credential=credential)
```

```
# Cliente de tablas
table_service = TableServiceClient(endpoint=TABLE_URL, credential=credential)
table_client = table_service.get_table_client(table_name=TABLE_NAME)

# Web3 y cuenta central
w3 = Web3(HTTPProvider(BLOCKCHAIN_URL))
central_account = Account.from_key(CENTRAL_PRIVATE_KEY)

def sanitize_table_name(user_id: str) -> str:
    # Obtiene un hash hexadecimal de longitud fija
    hashed = hashlib.sha256(user_id.encode()).hexdigest()[:20]  # 20 caracteres
    ↪ , ajustable
    return f"Transacciones{hashed.capitalize()}"


def get_or_create_ethereum_account(user_id: str) -> str:
    secret_name = f"blockChain-{user_id}"
    try:
        secret = secret_client.get_secret(secret_name)
        private_key = secret.value
    except:
        acct = Account.create()
        private_key = acct.key.hex()
        secret_client.set_secret(secret_name, private_key)
    account = Account.from_key(private_key)
    return account.address


def get_or_create_user_aes_key(user_id: str) -> bytes:
    secret_name = f"aes-Key-{user_id}"
    try:
        return base64.b64decode(secret_client.get_secret(secret_name).value)
    except:
        key = AESGCM.generate_key(bit_length=128)
        secret_client.set_secret(secret_name, base64.b64encode(key).decode())
        return key


def encrypt_data(plain_obj: dict, key: bytes) -> str:
    aesgcm = AESGCM(key)
    nonce = os.urandom(12)
    plaintext = json.dumps(plain_obj, ensure_ascii=False).encode("utf-8")
    ciphertext = aesgcm.encrypt(nonce, plaintext, None)
    return "0x" + (nonce + ciphertext).hex()


def send_to_blockchain(destination_address: str, data_hex: str):
    nonce = w3.eth.get_transaction_count(central_account.address)
    tx = {
        'nonce': nonce,
        'to': destination_address,
        'value': 0,
        'gas': 100000,
        'gasPrice': 0,
        'data': data_hex,
        'chainId': w3.eth.chain_id
    }
```

```
    signed_tx = w3.eth.account.sign_transaction(tx, central_account.key)
    tx_hash = w3.eth.send_raw_transaction(signed_tx.raw_transaction)
    return w3.to_hex(tx_hash)


# ... (resto del código igual hasta send_to_blockchain)

def log_transaction_hash(user_id: str, tx_hash: str):

    table_name = sanitize_table_name(user_id)
    try:
        # Crea la tabla si no existe
        try:
            table_service.create_table_if_not_exists(table_name=table_name)
        except Exception as e:
            logging.warning(f" La tabla ya existe o hubo un problema: {str(e)}"
    ↪ )

        # Luego obtienes el cliente de la tabla para insertar la entidad
        table_client_tx = table_service.get_table_client(table_name=table_name)

        entity = {
            "PartitionKey": user_id,
            "RowKey": tx_hash,  # Usamos el hash como RowKey para unicidad
            "TimestampUTC": datetime.now(timezone.utc)
        }
        table_client_tx.upsert_entity(entity)
        logging.info(f" Hash de transacción registrado en la tabla {table_name
    ↪ }: {tx_hash}")

    except Exception as e:
        logging.error(f" Error al registrar el hash de transacción en la tabla
    ↪ {table_name}: {str(e)}", exc_info=True)

def main(msg: func.QueueMessage) -> None:
    try:
        payload = json.loads(msg.get_body().decode('utf-8'))
        user_id = payload.get("user_id")
        if not user_id:
            logging.error("Falta el campo 'user_id'")
            return

        logging.info(f" Buscando clasificaciones para el usuario: {user_id}")
        entities = list(table_client.query_entities(query_filter=f"PartitionKey
    ↪  eq '{user_id}'"))

        if not entities:
            logging.warning(f" No se encontraron clasificaciones para el
    ↪ usuario {user_id}.")
            return

        suma = {"O": 0, "C": 0, "E": 0, "A": 0, "N": 0}
        for entity in entities:
            for key in suma:
                suma[key] += entity.get(key, 0)

        media = {k: round(v / len(entities), 2) for k, v in suma.items()}
        logging.info(f" Media OCEAN para {user_id}: {media}")
```

```
        # Encriptar y registrar en blockchain
        aes_key = get_or_create_user_aes_key(user_id)
        encrypted_hex = encrypt_data({"user_id": user_id, "media": media},
    ↪ aes_key)
        user_address = get_or_create_ethereum_account(user_id)
        tx_hash = send_to_blockchain(user_address, encrypted_hex)
        logging.info(f" Transacción registrada: {tx_hash}")

        # Registrar hash en Azure Table Storage
        log_transaction_hash(user_id, tx_hash)

        # Eliminar clasificaciones antiguas
        logging.info(f" Eliminando {len(entities)} clasificaciones previas del
    ↪ usuario {user_id}...")
        batch = [("delete", {"PartitionKey": e["PartitionKey"], "RowKey": e["
    ↪ RowKey"]}) for e in entities]
        if batch:
            table_client.submit_transaction(batch)
            logging.info(" Clasificaciones eliminadas.")

    except TableTransactionError as te:
        logging.error(f" Error en la transacción de eliminación: {str(te)}",
    ↪ exc_info=True)
    except Exception as e:
        logging.error(f" Error general al procesar la media OCEAN: {str(e)}",
    ↪ exc_info=True)
```

Listing D.4: Función RegistrarBlockChain

D.3.5. Función RecuperarTransacciones

```
import logging
import os
import json
import azure.functions as func
import pandas as pd
from web3 import Web3
from eth_account import Account
from azure.identity import DefaultAzureCredential
from azure.keyvault.secrets import SecretClient
from azure.data.tables import TableServiceClient
from datetime import datetime
import hashlib
from web3.middleware import ExtraDataToPOAMiddleware
from cryptography.hazmat.primitives.ciphers.aead import AESGCM
import base64
import jwt

# Configuración
BLOCKCHAIN_URL = os.getenv("BLOCKCHAIN_URL")
KEY_VAULT_URL = os.getenv("KEY_VAULT_URL")
TABLE_URL = os.getenv("TABLES_URL")
credential = DefaultAzureCredential()
secret_client = SecretClient(vault_url=KEY_VAULT_URL, credential=credential)
table_service = TableServiceClient(endpoint=TABLE_URL, credential=credential)
w3 = Web3(Web3.HTTPProvider(BLOCKCHAIN_URL))
```

```
w3.middleware_onion.inject(ExtraDataToPOAMiddleware, layer=0)

ALLOWED_ORIGIN = "https://pilotoyodai.powerappsportals.com"

def sanitize_table_name(user_id: str) -> str:
    # Obtiene un hash hexadecimal de longitud fija
    hashed = hashlib.sha256(user_id.encode()).hexdigest()[:20]  # 20 caracteres
    ↪ , ajustable
    return f"Transacciones{hashed.capitalize()}"

def decode_input(encrypted_input: bytes, user_key: bytes) -> dict:
    try:
        if not encrypted_input or len(encrypted_input) < 12:
            return {}

        nonce = encrypted_input[:12]
        ciphertext = encrypted_input[12:]

        aesgcm = AESGCM(user_key)
        decrypted_bytes = aesgcm.decrypt(nonce, ciphertext, None)
        return json.loads(decrypted_bytes.decode("utf-8"))

    except Exception as e:
        logging.error(f" Error al desencriptar input: {e}", exc_info=True)
        return {}

def get_user_key(user_id: str) -> bytes:
    return base64.b64decode(secret_client.get_secret(f"aes-key-{user_id}").
    ↪ value)

def fetch_transactions_from_storage(user_id: str) -> list:
    table_name = table_name = sanitize_table_name(user_id)
    try:
        table_client = table_service.get_table_client(table_name)
        entities = list(table_client.query_entities(query_filter=f"PartitionKey
    ↪  eq '{user_id}'"))
        transactions = []

        for entity in entities:
            tx_hash = entity["RowKey"]
            try:
                tx = w3.eth.get_transaction(tx_hash)
                block = w3.eth.get_block(tx["blockNumber"])
                logging.info(f" tx['input'] TYPE: {type(tx['input'])}, VALUE: {
    ↪ tx['input']}")

                key = get_user_key(user_id)
                decoded = decode_input(tx["input"], key)
                transactions.append({
                    "hash": tx.hash.hex(),
                    "timestamp": block.timestamp,
                    "input": decoded,
                })
            except Exception as e:
                logging.warning(f" Error al recuperar la transacción {tx_hash}:
    ↪  {str(e)}")

        return transactions
```

```
    except Exception as e:
        logging.error(f" Error al acceder a la tabla {table_name}: {str(e)}",
    ↪ exc_info=True)
        return []

def main(req: func.HttpRequest) -> func.HttpResponse:
    origin = req.headers.get("Origin")
    if origin != ALLOWED_ORIGIN:
        return func.HttpResponse(
            "Forbidden: origin not allowed",
            status_code=403,
            headers={
                'Access-Control-Allow-Origin': 'null',
                'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type'
            }
        )

    if req.method == "OPTIONS":
        return func.HttpResponse(
            status_code=200,
            headers={
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type, user.id,
    ↪ assistant.id'
            }
        )

    token = req.headers.get("Authorization")
    if not token:
        return func.HttpResponse("Unauthorized", status_code=401)

    try:
        # Usar principal_id como ContactID seguro
        payload = jwt.decode(token, options={"verify_signature": False})
        user_id = payload.get("sub")
    except Exception as e:
        logging.error(f"Error: {e}", exc_info=True)
        return func.HttpResponse("Unauthorized", status_code=401)

    try:
        # Obtener dirección del usuario (para consistencia, aunque no la usamos
    ↪  directamente)
        secret_name = f"blockChain-{user_id}"
        private_key = secret_client.get_secret(secret_name).value
        account = Account.from_key(private_key)
        target_address = w3.to_checksum_address(account.address)

        logging.info(f" Recuperando transacciones de la tabla para: {
    ↪ target_address}")

        transactions = fetch_transactions_from_storage(user_id)

        if not transactions:
            return func.HttpResponse(
```

```
                "No se encontraron transacciones registradas para este usuario.
    ↪ ",
                status_code=404
            )

        df = pd.DataFrame(transactions)
        df["fecha_utc"] = pd.to_datetime(df["timestamp"], unit='s')
        df = df.sort_values("blockNumber")
        csv_bytes = df.to_csv(index=False).encode("utf-8")

        return func.HttpResponse(
            csv_bytes,
            mimetype="text/csv",
            headers={"Content-Disposition": f"attachment; filename=evolucion_{
    ↪ user_id}.csv"}
        )

    except Exception as e:
        logging.error(f" Error al recuperar transacciones: {str(e)}", exc_info=
    ↪ True)
        return func.HttpResponse(f"Error: {str(e)}", status_code=500)
```

Listing D.5: Función RecuperarTransacciones

D.3.6. Función EncriptaryGuardar

```
import logging
import json
import base64
import os
import tempfile
from datetime import datetime, timezone
from azure.storage.blob import BlobServiceClient
from azure.identity import DefaultAzureCredential
from azure.keyvault.secrets import SecretClient
from azure.data.tables import TableServiceClient, UpdateMode
from cryptography.hazmat.primitives.ciphers.aead import AESGCM
from azure.core.exceptions import ResourceNotFoundError
from openai import OpenAI
import azure.functions as func
from azure.storage.queue import QueueClient

# Configuración de entorno
KEY_VAULT_URL = os.getenv("KEY_VAULT_URL")
STORAGE_URL = os.getenv("STORAGE_URL")
TABLE_URL = os.getenv("TABLES_URL")
QUEUE_URL = os.getenv("QUEUE_URL")
TABLE_NAME = "Threads"
CONTAINER_NAME = "conversaciones"
queue_name = "clasificar-usuario"

credential = DefaultAzureCredential()

# Clientes
blob_service_client = BlobServiceClient(account_url=STORAGE_URL, credential=
    ↪ credential)
secret_client = SecretClient(vault_url=KEY_VAULT_URL, credential=credential)
```

```
table_service = TableServiceClient(endpoint=TABLE_URL, credential=credential)
clientGPT = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))
queue_client = QueueClient(account_url=QUEUE_URL, queue_name=queue_name,
    ↪ credential=credential)

# --- Utilidades ---
def get_or_create_user_key(user_id: str) -> bytes:
    secret_name = f"aes-key-{user_id}"
    try:
        secret = secret_client.get_secret(secret_name)
        return base64.b64decode(secret.value)
    except:
        key = AESGCM.generate_key(bit_length=256)
        secret_client.set_secret(secret_name, base64.b64encode(key).decode("utf
    ↪ -8"))
        return key

def encrypt_message(key: bytes, message: str) -> bytes:
    aesgcm = AESGCM(key)
    nonce = os.urandom(12)
    ciphertext = aesgcm.encrypt(nonce, message.encode("utf-8"), None)
    return nonce + ciphertext

def update_blob(user_id: str, encrypted_message: bytes):
    blob_client = blob_service_client.get_blob_client(
        container=CONTAINER_NAME,
        blob=f"{user_id}.bin"
    )

    try:
        blob_client.get_blob_properties()
    except ResourceNotFoundError:
        logging.info(f"Creando nuevo append blob para {user_id}")
        blob_client.create_append_blob()  # se crea vacío, tipo append

    # Anade el mensaje al final
    blob_client.append_block(encrypted_message + b"==DELIM==")

def get_table_entity(user_id):
    table_client = table_service.get_table_client(table_name=TABLE_NAME)
    entity = table_client.get_entity(partition_key="threads", row_key=user_id)
    return entity, table_client

def update_table_entity(table_client, user_id, updates):
    updates["PartitionKey"] = "threads"
    updates["RowKey"] = user_id
    table_client.upsert_entity(entity=updates, mode=UpdateMode.MERGE)

# --- MAIN ---
def main(msg: func.QueueMessage) -> None:
    logging.info("Procesando mensaje desde la cola.")

    try:
        raw_body = msg.get_body().decode('utf-8')
        logging.info(f"Mensaje recibido de la cola: {raw_body}")

        data = json.loads(raw_body)
```

```
    user_id = data.get("user_id")
    message = data.get("message")
    role = data.get("role")
    timestamp = data.get("time")

    if not user_id or not message or not role or not timestamp:
        raise ValueError("Faltan campos necesarios: 'user_id', 'message', '
↪ role' o 'time'.")

    # 1. Guardar mensaje en Blob (cifrado)
    key = get_or_create_user_key(user_id)
    prefixed_message = f"{timestamp} {role.upper()}: {message}"
    encrypted = encrypt_message(key, prefixed_message)
    update_blob(user_id, encrypted)
    logging.info(f"Mensaje cifrado guardado en blob para {user_id}.")

    # 2. Guardar en Vector Store
    entity, table_client = get_table_entity(user_id)
    thread_id = entity["thread_id"]
    vector_store_id = entity.get("vector_store_id")

    # Crear archivo temporal
    filename = f"{role.upper()}_{timestamp}.txt"
    with tempfile.NamedTemporaryFile(delete=False, mode="w", encoding="utf
↪ -8",
                                     suffix=".txt", prefix=filename.replace
↪ (":", "-").replace(".", "-") + "_") as tmp_file:
        tmp_file.write(prefixed_message)
        temp_path = tmp_file.name

    # Subir a OpenAI
    file_upload = clientGPT.files.create(file=open(temp_path, "rb"),
↪ purpose="assistants")
    logging.info(f"Archivo subido a OpenAI con ID: {file_upload.id}")

    # Crear vector store si no existe
    if not vector_store_id:
        vector_store = clientGPT.vector_stores.create(name=f"historial-{
↪ user_id}")
        vector_store_id = vector_store.id
        update_table_entity(table_client, user_id, {"vector_store_id":
↪ vector_store_id})
        clientGPT.beta.threads.update(
            thread_id,
            tool_resources={"file_search": {"vector_store_ids": [
↪ vector_store_id]}}
        )
        logging.info(f"Vector store creado y asignado: {vector_store_id}")

    # Anadir archivo al vector store
    clientGPT.vector_stores.file_batches.create(
        vector_store_id=vector_store_id,
        file_ids=[file_upload.id]
    )
    logging.info("Archivo anadido al vector store correctamente.")

    # 3. Contar interacciones
    interaction_count = entity.get("interaction_count", 0) + 1
```

```
        update_table_entity(table_client, user_id, {"interaction_count":
    ↪ interaction_count})
        if interaction_count >= 80:
            queue_payload = {"user_id": user_id}
            message_json = json.dumps(queue_payload)
            encoded_message = base64.b64encode(message_json.encode("utf-8")).
    ↪ decode("utf-8")
            queue_client.send_message(encoded_message)
            logging.info(f"Enviada solicitud de clasificación para {user_id} a
    ↪ la cola '{queue_name}'")
            update_table_entity(table_client, user_id, {"interaction_count":
    ↪ 0})

    except Exception as e:
        logging.error(f"Error procesando mensaje: {str(e)}", exc_info=True)
```

Listing D.6: Función EncriptaryGuardar

D.3.7. Función SessionEndLoggerFunction

```
import logging
import azure.functions as func
from applicationinsights import TelemetryClient
import json
import os

INSTRUMENTATION_KEY = os.getenv("INSTRUMENTATION_KEY")
ALLOWED_ORIGIN = "https://pilotoyodai.powerappsportals.com"

def main(req: func.HttpRequest) -> func.HttpResponse:
    origin = req.headers.get("Origin")
    if origin != ALLOWED_ORIGIN:
        return func.HttpResponse(
            "Forbidden: origin not allowed",
            status_code=403,
            headers={
                'Access-Control-Allow-Origin': 'null',
                'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type'
            }
        )

    if req.method == "OPTIONS":
        # Preflight CORS request
        return func.HttpResponse(
            status_code=200,
            headers={
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type, user.id,
    ↪ assistant.id'
            }
        )

    logging.info("Procesando solicitud de SessionEnd...")

    try:
```

```
        body = req.get_json()
        contact_id = body.get("ContactID")
        duration = body.get("Duration")
        nPagina = body.get("Pagina")

        if not contact_id or duration is None or not nPagina:
            return func.HttpResponse(
    f"Faltan datos: ContactID: {contact_id}, Duration: {duration}, Página: {
    ↪ nPagina}",
    status_code=400
)


        tc = TelemetryClient(INSTRUMENTATION_KEY)
        tc.track_event("SessionEnd", {
            "ContactID": contact_id,
            "Duration": duration,
            "Pagina": nPagina
        })
        tc.flush()

        logging.info("Evento enviado a Application Insights correctamente.")
        return func.HttpResponse("Evento enviado correctamente", status_code
    ↪ =200)

    except Exception as e:
        logging.error(f"Error al procesar la solicitud: {str(e)}")
        return func.HttpResponse(f"Error interno: {str(e)}", status_code=500)
```

Listing D.7: Función SessionEndLoggerFunction

D.3.8. Función Encolar

```
import logging
import json
import base64
import azure.functions as func
from azure.storage.queue import QueueClient
import os
from datetime import datetime, timezone
from azure.identity import DefaultAzureCredential
import jwt

ALLOWED_ORIGIN = "https://pilotoyodai.powerappsportals.com"

def main(req: func.HttpRequest) -> func.HttpResponse:
    origin = req.headers.get("Origin")
    if origin != ALLOWED_ORIGIN:
        return func.HttpResponse(
            "Forbidden: origin not allowed",
            status_code=403,
            headers={
                'Access-Control-Allow-Origin': 'null',
                'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type'
            }
        )
```

```
    if req.method == "OPTIONS":
        # Preflight CORS request
        return func.HttpResponse(
            status_code=200,
            headers={
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': 'GET, POST, OPTIONS',
                'Access-Control-Allow-Headers': 'Content-Type, user.id,
↪ assistant.id'
            }
        )

    try:
        token = req.headers.get("Authorization")
        if not token:
            return func.HttpResponse("Unauthorized", status_code=401)

        try:
            # Usar principal_id como ContactID seguro
            payload = jwt.decode(token, options={"verify_signature": False})
            user_id = payload.get("sub")
        except Exception as e:
            logging.error(f"Error: {e}", exc_info=True)
            return func.HttpResponse("Unauthorized", status_code=401)

        role = req.headers.get("role")
        data = req.get_json()
        message = data.get("message")

        if not user_id or not message or not role:
            return func.HttpResponse("Missing 'user_id' or 'message' or 'role'"
↪ , status_code=400)

        credential = DefaultAzureCredential()
        queue_name = "guardar-mensajes"
        QUEUE_URL = os.getenv("QUEUE_URL")
        queue_client = QueueClient(account_url=QUEUE_URL, queue_name=queue_name
↪ , credential=credential)

        # Construir el payload y codificar en base64
        payload = {
            "user_id": user_id,
            "message": message,
            "role": role,
            "time": datetime.now(timezone.utc).isoformat()
        }

        payload_json = json.dumps(payload)
        payload_base64 = base64.b64encode(payload_json.encode("utf-8")).decode(
↪ "utf-8")

        # Enviar a la cola
        queue_client.send_message(payload_base64)

        return func.HttpResponse("Mensaje encolado en base64", status_code=200)

    except Exception as e:
```

```
        logging.exception("Error encolando mensaje")
        return func.HttpResponse(f"Error: {str(e)}", status_code=500)
```

Listing D.8: Función Encolar

D.4. Frontend: Power Pages

D.4.1. Estructura de carpetas y archivos

- Cada página tiene su HTML (estructura), CSS (estilos) y JS (lógica).
- Las funciones compartidas están en 'shared-functions.html', que se incluye en todas las páginas.

D.4.2. Páginas implementadas

- **Página principal**:

```
<div data-component-theme="" class="row sectionBlockLayout text-start"
    ↪ style='display: flex; flex-wrap: wrap; margin: 0px; padding: 8px;
    ↪ min-height: 100vh; width: 100%; background-image: url("/
    ↪ BienvenidaPNG.PNG"); background-position: center center; background
    ↪ -repeat: no-repeat; background-size: cover;'>
  <div class="container" style="padding: 0px; display: flex; flex-wrap:
    ↪ wrap; height: 100%;">
    <div class="col-lg-12 columnBlockLayout" style="flex-grow: 1; display
    ↪ : flex; flex-direction: column; min-width: 250px; justify-content:
    ↪ center; align-items: center;">
      <!-- Aquí puedes anadir contenido adicional -->
      <h1 class="siteTitle" style="text-align: left; color: var(--
    ↪ portalThemeColor8);"><b>YODAI</b></h1>
      <p style="color: white; text-align: left;"><b>ASISTENTE VIRTUAL
    ↪ BIODIGITAL</b></p>
      <p>Desarrolla una mente biodigital para <br />habilitar el
    ↪ mundo que viene</p>
      <a href="/Selección-de-Uso" class="btn button1">start</a>
    </div>
  </div>
</div>
{% include 'shared-functions' %}
```

Listing D.9: Página principal.es-ES.webpage

```
var contactoObjeto = "{{ user.contactid }}"; // Obtener el ContactID
    ↪ desde Power Pages
let userId = String(contactoObjeto).trim();
const instrumentationKey = "2a3fb39c-0c96-4926-a4d4-1ff5d097cf34";
var contactoObjeto = "{{ user.contactid }}"; // Obtener el ContactID
    ↪ desde Power Pages
const contactoID = String(contactoObjeto).trim();
var inactivityTimeout = 3600000; // 1 hora en milisegundos (60 min * 60
    ↪ seg * 1000 ms)
var inactivityTimer;
//  Inicializar Application Insights
var appInsights = initApplicationInsights(instrumentationKey);
```

```
const nPagina = "Pagina Principal"

//  Detectar interacciones del usuario para resetear el temporizador
window.addEventListener("mousemove", resetInactivityTimer);
window.addEventListener("keydown", resetInactivityTimer);
window.addEventListener("scroll", resetInactivityTimer);
window.addEventListener("click", resetInactivityTimer);

//  Verificar si el usuario ya está inactivo al cargar la página o si hay
    ↪  un logout pendiente
window.addEventListener("load", function () {
  handleSessionStart(nPagina);
  resetInactivityTimer();
});

//  Detectar cierre de pestana o navegador y guardar última interacción
window.addEventListener("beforeunload", function () {
  // Enviar cierre de sesión con duración
  sendSessionEnd(nPagina);
});
```

Listing D.10: Página principal.es-ES.customjs

Listing D.11: Página principal.es-ES.customcss

- **Selección de uso**:

```
<meta charset="UTF-8" /><meta name="viewport" content="width=device-width
    ↪ , initial-scale=1.0" /><title>Selección de Herramientas</title>
<div class="containerMOD">
  <!-- Sección del Chat -->
  <div class="chat-section">
    <h1>Asistente YODAI</h1>
    <h4>Cuéntanos un poco de ti y te ayudaremos a escoger la herramienta
    ↪ adecuada para tu situación.</h4>
    <div id="assistantResponse" style="width: 100%; margin-top: 20px;
    ↪ margin-right: auto; margin-bottom: 20px; margin-left: auto; padding
    ↪ -top: 15px; padding-right: 15px; padding-bottom: 15px; padding-left
    ↪ : 15px; background-color: rgb(249, 249, 249); border-top-left-
    ↪ radius: 5px; border-top-right-radius: 5px; border-bottom-right-
    ↪ radius: 5px; border-bottom-left-radius: 5px; border-top-width: 1px;
    ↪  border-right-width: 1px; border-bottom-width: 1px; border-left-
    ↪ width: 1px; border-top-style: solid; border-right-style: solid;
    ↪ border-bottom-style: solid; border-left-style: solid; border-top-
    ↪ color: rgb(204, 204, 204); border-right-color: rgb(204, 204, 204);
    ↪ border-bottom-color: rgb(204, 204, 204); border-left-color: rgb
    ↪ (204, 204, 204); border-image-source: initial; border-image-slice:
    ↪ initial; border-image-width: initial; border-image-outset: initial;
    ↪  border-image-repeat: initial; min-height: 100px; max-height: 600px
    ↪ ; overflow-x: auto; overflow-y: auto;"><p></p></div>
    <textarea id="userQuestion" rows="4" placeholder="Escribe tu pregunta
    ↪  aquí" style="width: 100%; padding-top: 10px; padding-right: 10px;
    ↪ padding-bottom: 10px; padding-left: 10px; border-top-left-radius: 5
    ↪ px; border-top-right-radius: 5px; border-bottom-right-radius: 5px;
    ↪ border-bottom-left-radius: 5px; border-top-width: 1px; border-right
    ↪ -width: 1px; border-bottom-width: 1px; border-left-width: 1px;
    ↪ border-top-style: solid; border-right-style: solid; border-bottom-
```

```
    ↪ style: solid; border-left-style: solid; border-top-color: rgb(204,
    ↪ 204, 204); border-right-color: rgb(204, 204, 204); border-bottom-
    ↪ color: rgb(204, 204, 204); border-left-color: rgb(204, 204, 204);
    ↪ border-image-source: initial; border-image-slice: initial; border-
    ↪ image-width: initial; border-image-outset: initial; border-image-
    ↪ repeat: initial; resize: none;"></textarea><button id="sendButton"
    ↪ class="button1" style="padding-top: 10px; padding-right: 20px;
    ↪ padding-bottom: 10px; padding-left: 20px; margin-top: 10px; margin-
    ↪ right: 10px; margin-bottom: 10px; margin-left: 10px; background-
    ↪ color: rgb(13, 74, 241); border-top-width: initial; border-right-
    ↪ width: initial; border-bottom-width: initial; border-left-width:
    ↪ initial; border-top-style: none; border-right-style: none; border-
    ↪ bottom-style: none; border-left-style: none; border-top-color:
    ↪ initial; border-right-color: initial; border-bottom-color: initial;
    ↪  border-left-color: initial; border-image-source: initial; border-
    ↪ image-slice: initial; border-image-width: initial; border-image-
    ↪ outset: initial; border-image-repeat: initial; border-top-left-
    ↪ radius: 5px; border-top-right-radius: 5px; border-bottom-right-
    ↪ radius: 5px; border-bottom-left-radius: 5px; cursor: pointer;
    ↪ transition-behavior: normal; transition-duration: 0.3s; transition-
    ↪ timing-function: ease; transition-delay: 0s; transition-property:
    ↪ all; color: white;">Enviar</button>
     <button id="bajarClas" class="button1" style="padding-top: 10px;
    ↪ padding-right: 20px; padding-bottom: 10px; padding-left: 20px;
    ↪ margin-top: 10px; margin-right: 10px; margin-bottom: 10px; margin-
    ↪ left: 10px; background-color: rgb(241, 142, 13); border-top-width:
    ↪ initial; border-right-width: initial; border-bottom-width: initial;
    ↪  border-left-width: initial; border-top-style: none; border-right-
    ↪ style: none; border-bottom-style: none; border-left-style: none;
    ↪ border-top-color: initial; border-right-color: initial; border-
    ↪ bottom-color: initial; border-left-color: initial; border-image-
    ↪ source: initial; border-image-slice: initial; border-image-width:
    ↪ initial; border-image-outset: initial; border-image-repeat: initial
    ↪ ; border-top-left-radius: 5px; border-top-right-radius: 5px; border
    ↪ -bottom-right-radius: 5px; border-bottom-left-radius: 5px; cursor:
    ↪ pointer; transition-behavior: normal; transition-duration: 0.3s;
    ↪ transition-timing-function: ease; transition-delay: 0s; transition-
    ↪ property: all; color: white;">Descargar Clasificacion</button>
     <!-- <button id="oceanButton" class="button1">Obtener clasificación
    ↪ OCEAN</button> -->
  </div>
  <!-- Sección de Herramientas -->
  <div class="tools-section" style="width: 100%; max-width: 800px; text-
    ↪ align: center;">
     <h2>Herramientas</h2>
     <div class="buttons-container" style="display: flex; flex-wrap: wrap;
    ↪  justify-content: center; row-gap: 20px; column-gap: 20px;">
       <div style="display: flex; flex-direction: column; align-items:
    ↪ center; width: 150px; text-align: center;">
         <a href="/Selección-de-Uso/Cuantificar_IPP" class="btn button1"
    ↪ style="padding-top: 10px; padding-right: 20px; padding-bottom: 10px
    ↪ ; padding-left: 20px; margin-top: 10px; margin-right: 10px; margin-
    ↪ bottom: 10px; margin-left: 10px; background-color: rgb(13, 74, 241)
    ↪ ; border-top-width: initial; border-right-width: initial; border-
    ↪ bottom-width: initial; border-left-width: initial; border-top-style
    ↪ : none; border-right-style: none; border-bottom-style: none; border
    ↪ -left-style: none; border-top-color: initial; border-right-color:
    ↪ initial; border-bottom-color: initial; border-left-color: initial;
```

```
↪ border-image-source: initial; border-image-slice: initial; border-
↪ image-width: initial; border-image-outset: initial; border-image-
↪ repeat: initial; cursor: pointer; transition-behavior: normal;
↪ transition-duration: 0.3s; transition-timing-function: ease;
↪ transition-delay: 0s; transition-property: all; width: 80px; height
↪ : 80px; background-size: cover; background-position-x: center;
↪ background-position-y: center; border-top-left-radius: 50%; border-
↪ top-right-radius: 50%; border-bottom-right-radius: 50%; border-
↪ bottom-left-radius: 50%; background-image: url('/IPPbutton 2.png');
↪  color: white;"></a>
    <p style="margin-top: 5px; margin-right: 0px; margin-bottom: 0px;
↪  margin-left: 0px; font-size: 1rem; color: rgb(0, 0, 0);">Encontrar
↪  tu índice Personal de Plenitud</p>
  </div>
  <div style="display: flex; flex-direction: column; align-items:
↪ center; width: 150px; text-align: center;">
    <a href="/Selección-de-Uso/Interfaz_VP" class="btn button1" style
↪ ="padding-top: 10px; padding-right: 20px; padding-bottom: 10px;
↪ padding-left: 20px; margin-top: 10px; margin-right: 10px; margin-
↪ bottom: 10px; margin-left: 10px; background-color: rgb(13, 74, 241)
↪ ; border-top-width: initial; border-right-width: initial; border-
↪ bottom-width: initial; border-left-width: initial; border-top-style
↪ : none; border-right-style: none; border-bottom-style: none; border
↪ -left-style: none; border-top-color: initial; border-right-color:
↪ initial; border-bottom-color: initial; border-left-color: initial;
↪ border-image-source: initial; border-image-slice: initial; border-
↪ image-width: initial; border-image-outset: initial; border-image-
↪ repeat: initial; cursor: pointer; transition-behavior: normal;
↪ transition-duration: 0.3s; transition-timing-function: ease;
↪ transition-delay: 0s; transition-property: all; width: 80px; height
↪ : 80px; background-size: cover; background-position-x: center;
↪ background-position-y: center; border-top-left-radius: 50%; border-
↪ top-right-radius: 50%; border-bottom-right-radius: 50%; border-
↪ bottom-left-radius: 50%; background-image: url('/VPbutton 2.png');
↪ color: white;"></a>
    <p style="margin-top: 5px; margin-right: 0px; margin-bottom: 0px;
↪  margin-left: 0px; font-size: 1rem; color: rgb(0, 0, 0);">Encontrar
↪  el impulso de tu talento</p>
  </div>
  <div style="display: flex; flex-direction: column; align-items:
↪ center; width: 150px; text-align: center;">
    <a href="/Selección-de-Uso/Pensamiento_Computacional_Natural"
↪ class="btn button1" style="padding-top: 10px; padding-right: 20px;
↪ padding-bottom: 10px; padding-left: 20px; margin-top: 10px; margin-
↪ right: 10px; margin-bottom: 10px; margin-left: 10px; background-
↪ color: rgb(13, 74, 241); border-top-width: initial; border-right-
↪ width: initial; border-bottom-width: initial; border-left-width:
↪ initial; border-top-style: none; border-right-style: none; border-
↪ bottom-style: none; border-left-style: none; border-top-color:
↪ initial; border-right-color: initial; border-bottom-color: initial;
↪  border-left-color: initial; border-image-source: initial; border-
↪ image-slice: initial; border-image-width: initial; border-image-
↪ outset: initial; border-image-repeat: initial; cursor: pointer;
↪ transition-behavior: normal; transition-duration: 0.3s; transition-
↪ timing-function: ease; transition-delay: 0s; transition-property:
↪ all; width: 80px; height: 80px; background-size: cover; background-
↪ position-x: center; background-position-y: center; border-top-left-
↪ radius: 50%; border-top-right-radius: 50%; border-bottom-right-
```

```
↪ radius: 50%; border-bottom-left-radius: 50%; background-image: url
↪ ('/PCNbutton.png'); color: white;"></a>
    <p style="margin-top: 5px; margin-right: 0px; margin-bottom: 0px;
↪  margin-left: 0px; font-size: 1rem; color: rgb(0, 0, 0);">
↪ Desarrolla tu Pensamiento Computacional Natural</p>
  </div>
  <div style="display: flex; flex-direction: column; align-items:
↪ center; width: 150px; text-align: center;">
    <a href="/Selección-de-Uso/Interfaz_Matrix" class="btn button1"
↪ style="padding-top: 10px; padding-right: 20px; padding-bottom: 10px
↪ ; padding-left: 20px; margin-top: 10px; margin-right: 10px; margin-
↪ bottom: 10px; margin-left: 10px; background-color: rgb(13, 74, 241)
↪ ; border-top-width: initial; border-right-width: initial; border-
↪ bottom-width: initial; border-left-width: initial; border-top-style
↪ : none; border-right-style: none; border-bottom-style: none; border
↪ -left-style: none; border-top-color: initial; border-right-color:
↪ initial; border-bottom-color: initial; border-left-color: initial;
↪ border-image-source: initial; border-image-slice: initial; border-
↪ image-width: initial; border-image-outset: initial; border-image-
↪ repeat: initial; cursor: pointer; transition-behavior: normal;
↪ transition-duration: 0.3s; transition-timing-function: ease;
↪ transition-delay: 0s; transition-property: all; width: 80px; height
↪ : 80px; background-size: cover; background-position-x: center;
↪ background-position-y: center; border-top-left-radius: 50%; border-
↪ top-right-radius: 50%; border-bottom-right-radius: 50%; border-
↪ bottom-left-radius: 50%; background-image: url('/M-YObutton.png');
↪ color: white;"></a>
    <p style="margin-top: 5px; margin-right: 0px; margin-bottom: 0px;
↪  margin-left: 0px; font-size: 1rem; color: rgb(0, 0, 0);">Convierte
↪  tu experiencia en un sistema de datos</p>
  </div>
  <div style="display: flex; flex-direction: column; align-items:
↪ center; width: 150px; text-align: center;">
    <a href="/Selección-de-Uso/Encontrar_AP" class="btn button1"
↪ style="padding-top: 10px; padding-right: 20px; padding-bottom: 10px
↪ ; padding-left: 20px; margin-top: 10px; margin-right: 10px; margin-
↪ bottom: 10px; margin-left: 10px; background-color: rgb(13, 74, 241)
↪ ; border-top-width: initial; border-right-width: initial; border-
↪ bottom-width: initial; border-left-width: initial; border-top-style
↪ : none; border-right-style: none; border-bottom-style: none; border
↪ -left-style: none; border-top-color: initial; border-right-color:
↪ initial; border-bottom-color: initial; border-left-color: initial;
↪ border-image-source: initial; border-image-slice: initial; border-
↪ image-width: initial; border-image-outset: initial; border-image-
↪ repeat: initial; cursor: pointer; transition-behavior: normal;
↪ transition-duration: 0.3s; transition-timing-function: ease;
↪ transition-delay: 0s; transition-property: all; width: 80px; height
↪ : 80px; background-size: cover; background-position-x: center;
↪ background-position-y: center; border-top-left-radius: 50%; border-
↪ top-right-radius: 50%; border-bottom-right-radius: 50%; border-
↪ bottom-left-radius: 50%; background-image: url('/APbutton.png');
↪ color: white;"></a>
    <p style="margin-top: 5px; margin-right: 0px; margin-bottom: 0px;
↪  margin-left: 0px; font-size: 1rem; color: rgb(0, 0, 0);">Descifra
↪ el secreto de tu comportamiento: AP (Algoritmo Personal)</p>
  </div>
  <div style="display: flex; flex-direction: column; align-items:
↪ center; width: 150px; text-align: center;">
```

```
        <a href="/Selección-de-Uso/SVM" class="btn button1" style="
    ↪ padding-top: 10px; padding-right: 20px; padding-bottom: 10px;
    ↪ padding-left: 20px; margin-top: 10px; margin-right: 10px; margin-
    ↪ bottom: 10px; margin-left: 10px; background-color: rgb(13, 74, 241)
    ↪ ; border-top-width: initial; border-right-width: initial; border-
    ↪ bottom-width: initial; border-left-width: initial; border-top-style
    ↪ : none; border-right-style: none; border-bottom-style: none; border
    ↪ -left-style: none; border-top-color: initial; border-right-color:
    ↪ initial; border-bottom-color: initial; border-left-color: initial;
    ↪ border-image-source: initial; border-image-slice: initial; border-
    ↪ image-width: initial; border-image-outset: initial; border-image-
    ↪ repeat: initial; cursor: pointer; transition-behavior: normal;
    ↪ transition-duration: 0.3s; transition-timing-function: ease;
    ↪ transition-delay: 0s; transition-property: all; width: 80px; height
    ↪ : 80px; background-size: cover; background-position-x: center;
    ↪ background-position-y: center; border-top-left-radius: 50%; border-
    ↪ top-right-radius: 50%; border-bottom-right-radius: 50%; border-
    ↪ bottom-left-radius: 50%; background-image: url('/SVMbutton.png');
    ↪ color: white;"></a>
        <p style="margin-top: 5px; margin-right: 0px; margin-bottom: 0px;
    ↪  margin-left: 0px; font-size: 1rem; color: rgb(0, 0, 0);">Convierte
    ↪  tu imaginación en un Simulador Mental de alta precisión</p>
      </div>
    </div>
  </div>
</div>
<div class="row sectionBlockLayout text-start" style="display: flex; flex
    ↪ -wrap: wrap; margin: 0px; min-height: auto; padding: 8px;">
  <div class="container" style="padding: 0px; display: flex; flex-wrap:
    ↪ wrap;"><div class="col-lg-12 columnBlockLayout" style="flex-grow:
    ↪ 1; display: flex; flex-direction: column; min-width: 250px;"></div>
    ↪ </div>
</div>
<script src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/dompurify@3.0.5/dist/purify.min
    ↪ .js"></script>
<script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
{% include 'shared-functions' %}
```

Listing D.12: Selección de Uso.es-ES.webpage

```
let loadingDotsInterval;
const instrumentationKey = "2a3fb39c-0c96-4926-a4d4-1ff5d097cf34";
var contactoObjeto = "{{ user.contactid }}"; // Obtener el ContactID
    ↪ desde Power Pages
const contactoID = String(contactoObjeto).trim();
let fullAssistantResponse = "";
let assistantId = "asst_LifQS4TtjR574DtBNNuJVHJt";
let pubsubConnectionUrl = `https://innoyodai.azurewebsites.net/api/
    ↪ GetWebSocketUrl?`;
let tokenCount = 0;
var appInsights = initApplicationInsights(instrumentationKey);
let socket;
var inactivityTimeout = 3600000; // 1 hora en milisegundos (60 min * 60
    ↪ seg * 1000 ms)
var inactivityTimer;
const nPagina = "Seleccion de uso"
connectToWebSocket().then((ws) => {
```

```
  socket = ws;
});
var nuevoMens = false;
const responseContainer = document.getElementById('assistantResponse');
const inputElement = document.getElementById('userQuestion');
responseParagraph = document.createElement('p');
responseParagraph.style.margin = '0';
responseParagraph.style.color = '#555';
responseContainer.appendChild(responseParagraph);

//  Detectar interacciones del usuario para resetear el temporizador
window.addEventListener("mousemove", resetInactivityTimer);
window.addEventListener("keydown", resetInactivityTimer);
window.addEventListener("scroll", resetInactivityTimer);
window.addEventListener("click", resetInactivityTimer);

//  Verificar si el usuario ya está inactivo al cargar la página o si hay
    ↪ un logout pendiente
window.addEventListener("load", function () {
  handleSessionStart(nPagina);
  resetInactivityTimer();
  handleUserMessage("un usuario acaba de conectarse a la conversacion,
    ↪ salúdale e introdúcete, y preparate para derivarle a la herramienta
    ↪  que escojas siguiendo la frase establecida");
});

//  Detectar cierre de pestana o navegador y guardar última interacción
window.addEventListener("beforeunload", function () {
  // Enviar cierre de sesión con duración
  sendSessionEnd(nPagina);
});
```

Listing D.13: Selección de Uso.es-ES.customjs

```
/* Contenedor principal */
.containerMOD {
    background-image: url('/fondoBlanco.png');
    background-size: cover;
    background-position: center center;
    background-repeat: no-repeat;
    padding: 20px;
    border-radius: 10px;
    display: flex;
    flex-direction: column;
    gap: 40px;
    align-items: center;
}

/* Estilos generales para texto en negro */
h1, h2, h4, p, label, button, textarea {
    color: #000000 !important; /* Fuerza que el color sea negro */
}

/* Sección del chat */
.chat-section {
    width: 100%;
    max-width: 600px;
    text-align: center;
```

```
}

/* Estilos para la respuesta del asistente */
#assistantResponse {
    width: 100%;
    margin: 20px auto;
    padding: 15px;
    background-color: #f9f9f9;
    border-radius: 5px;
    border: 1px solid #ccc;
    min-height: 100px;
    max-height: 600px;
    overflow: auto;
}

/* Estilos del área de entrada */
#userQuestion {
    width: 100%;
    padding: 10px;
    border-radius: 5px;
    border: 1px solid #ccc;
    resize: none;
}

/* Botones generales */
.button1 {
    padding: 10px 20px;
    margin: 10px;
    background-color: #0d4af1;
    color: white !important;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: 0.3s ease;
}

.button1:hover {
    background-color: #0a3bb7;
}

/* Sección de herramientas */
.tools-section {
    width: 100%;
    max-width: 800px;
    text-align: center;
}

/* Contenedor de botones */
.buttons-container {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    gap: 20px;
}

/* Botones con imágenes */
.btn.button1 {
    width: 80px;
```

```
    height: 80px;
    background-size: cover;
    background-position: center;
    border-radius: 50%;
}

/* Texto debajo de cada botón */
.buttons-container div {
    display: flex;
    flex-direction: column;
    align-items: center;
    width: 150px;
    text-align: center;
}

.buttons-container p {
    margin: 5px 0 0;
    font-size: 1rem;
    color: #000000 !important; /* Asegura que sea negro */
}

/* Asegurar que el placeholder sea visible */
#userQuestion::placeholder {
    color: #555 !important; /* Un color gris oscuro para que se vea en
    ↪ fondos claros */
    opacity: 1 !important;
}
```

Listing D.14: Selección de Uso.es-ES.customcss

- **Perfil de usuario**:

```
<div style="display: flex; flex-wrap: wrap; min-height: 50%; background-
    ↪ image: linear-gradient(0deg, rgba(0, 0, 0, 0.5), rgba(0, 0, 0, 0.5)
    ↪ ), url('/Perfil-3.jpg'); background-position: center center;
    ↪ background-repeat: no-repeat; background-size: cover;" class="row
    ↪ sectionBlockLayout" data-component-theme="portalThemeColor8">
  <div class="container">
    <div style="flex-grow: 1; display: flex; flex-direction: column;
    ↪ margin: 240px 0px; text-align: center;" class="col-md-12
    ↪ columnBlockLayout">
      <h2>Bienvenido</h2>
      <p>Accede a tu perfil y actualiza tu información personal. Si no
    ↪ has iniciado sesión, serás redirigido a la página de inicio de sesi
    ↪ ón.</p>

      <!--  Sección de información del usuario -->
      <div id="userProfile" style="background: white; padding: 20px;
    ↪ border-radius: 10px; max-width: 500px; margin: 20px auto; text-
    ↪ align: left;">
        <h3>Tu Información</h3>
        <p><strong>Nombre:</strong> <span id="userName">Cargando...</span
    ↪ ></p>
        <p><strong>Apellido:</strong> <span id="userLastName">Cargando...
    ↪ </span></p>
        <p><strong>Correo Electrónico:</strong> <span id="userEmail">
    ↪ Cargando...</span></p>
      </div>
```

```
      <!--  Botón para actualizar información -->
      <!-- <button style="background-color: #f0ad4e; color: white; border
    ↪ : none; padding: 10px 20px; border-radius: 5px; cursor: pointer;
    ↪ margin: 10px auto;" onclick="actualizarDatosB2C()">Actualizar Datos
    ↪ </button> -->

      <!--  Botones de navegación -->
      <div style="display: flex; justify-content: center; gap: 20px;
    ↪ margin: 20px;">

        <button style="background-color: #5bc0de; color: white; border:
    ↪ none; padding: 10px 20px; border-radius: 5px; cursor: pointer;"
                onclick="window.location.href='https://pilotoyodai.
    ↪ powerappsportals.com/'">Volver</button>
      </div>
    </div>
  </div>
</div>
```

Listing D.15: Perfil de Usuario.es-ES.webpage

```
window.addEventListener("load", function () {
    // Obtener la URL de referencia (de dónde vino el usuario)
    var referrer = document.referrer;

    // Comprobar si el usuario viene de Azure AD B2C
    if (referrer.includes("yodai.b2clogin.com") || referrer.includes("
    ↪ demoyodai.powerappsportals.com/signin-aad-b2c_2")) {
        console.log(" Usuario redirigido desde Azure AD B2C. Redirigiendo
    ↪  a la página principal...");
        window.location.href = "https://pilotoyodai.powerappsportals.com/
    ↪ ";
    } else {
        console.log(" Usuario accedió manualmente a /profile. No se
    ↪ redirige.");
    }
});


function cargarPerfilUsuario() {
    document.getElementById("userName").textContent = "{{ user.firstname
    ↪ }}" || "No disponible";
    document.getElementById("userLastName").textContent = "{{ user.
    ↪ lastname }}" || "No disponible";
    document.getElementById("userEmail").textContent = "{{ user.
    ↪ emailaddress1 }}" || "No disponible";
}

// Cargar datos del usuario cuando la página cargue
window.onload = cargarPerfilUsuario;
```

Listing D.16: Perfil de Usuario.es-ES.customjs

Listing D.17: Perfil de Usuario.es-ES.customcss

- **Herramientas ADSM (IPP, PCN, AP, VP, MYO, SVM.)**: todas derivan de una plantilla común, como 'Cuantificar IPP.es-ES.webpage.copy.html'

D.4.3. Ejemplo: Página Cuantificar IPP

```
<div data-component-theme="portalThemeColor8" class="row sectionBlockLayout"
    ↪ style='padding: 8px; margin: 0px; display: flex; flex-wrap: wrap; min-
    ↪ height: 50%; background-image: url("/fondoBlanco.png"); background-
    ↪ position: left center; background-repeat: no-repeat; background-size:
    ↪ cover;'>
  <div class="container">
    <div class="col-md-12 columnBlockLayout" style="min-width: 250px; flex-grow
    ↪ : 1; display: flex; flex-direction: column; margin: 240px 0px;">
      <h1 style="text-align: center; color: #0d4af1;">Encontrar tu Indice
    ↪ Personal de Plenitud</h1>
      <h4 style="text-align: center; color: #080808;">YODAI te guiará, a través
    ↪  de preguntas muy simples, por un ejercicio de cuantificación subjetiva
    ↪ de tu estado general de Plenitud. Es decir, te ayudará a tomar
    ↪ consciencia, en cualquier momento, de cómo te estás percibiendo a ti
    ↪ mismo y qué podrías hacer para mejorar ese estado general.</h4>
      <div id="assistantResponse" style="width: 80%; margin: 20px auto;
    ↪ padding: 10px; background-color: #f9f9f9; border-radius: 5px; border: 1
    ↪ px solid #ccc; min-height: 100px; max-height: 600px; overflow: auto;"><p
    ↪  style="margin: 0; color: #555;"></p></div>
      <textarea id="userQuestion" rows="4" placeholder="Escribe tu pregunta aqu
    ↪ í" style="width: 80%; margin: 10px auto; padding: 10px; border-radius: 5
    ↪ px; border: 1px solid #ccc; resize: none;"></textarea><button id="
    ↪ sendButton" class="button1" style="text-align: center; margin-left: auto
    ↪ ; margin-right: auto;">Enviar</button><button id="backButton" class="
    ↪ button1" style="text-align: center; margin-left: auto; margin-right:
    ↪ auto; margin-top: 20px;">Volver</button>
      <button id="bajarClas" class="button1" style="padding-top: 10px; padding-
    ↪ right: 20px; padding-bottom: 10px; padding-left: 20px; margin-top: 10px;
    ↪  margin-right: 10px; margin-bottom: 10px; margin-left: 10px; background-
    ↪ color: rgb(241, 142, 13); border-top-width: initial; border-right-width:
    ↪  initial; border-bottom-width: initial; border-left-width: initial;
    ↪ border-top-style: none; border-right-style: none; border-bottom-style:
    ↪ none; border-left-style: none; border-top-color: initial; border-right-
    ↪ color: initial; border-bottom-color: initial; border-left-color: initial
    ↪ ; border-image-source: initial; border-image-slice: initial; border-
    ↪ image-width: initial; border-image-outset: initial; border-image-repeat:
    ↪  initial; border-top-left-radius: 5px; border-top-right-radius: 5px;
    ↪ border-bottom-right-radius: 5px; border-bottom-left-radius: 5px; cursor:
    ↪  pointer; transition-behavior: normal; transition-duration: 0.3s;
    ↪ transition-timing-function: ease; transition-delay: 0s; transition-
    ↪ property: all; color: white;">Descargar Clasificacion</button>
      <!-- <button id="oceanButton" class="button1" style="text-align: center;
    ↪ margin-left: auto; margin-right: auto; margin-top: 20px;">obtener
    ↪ clasificacion OCEAN</button> -->
    </div>
  </div>
</div>
<script src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/dompurify@3.0.5/dist/purify.min.js"><
    ↪ /script>
<script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
{% include 'shared-functions' %}
```

Listing D.18: Cuantificar IPP.es-ES.webpage

```
let loadingDotsInterval;
var instrumentationKey = "2a3fb39c-0c96-4926-a4d4-1ff5d097cf34";
```

```
var contactoObjeto = "{{ user.contactid }}";
const contactoID = String(contactoObjeto).trim();
let fullAssistantResponse = "";
let assistantId = "asst_Ikuuf0G1ChLXmTc8ho3HHTHV";
let pubsubConnectionUrl = `https://innoyodai.azurewebsites.net/api/
    ↪ GetWebSocketUrl?`;
let tokenCount = 0;
var appInsights = initApplicationInsights(instrumentationKey);
var inactivityTimeout = 3600000; // 1 hora en milisegundos (60 min * 60 seg *
    ↪ 1000 ms)
var inactivityTimer;
let socket;
const nPagina = "Cuantificar IPP"
connectToWebSocket().then((ws) => {
  socket = ws;
});
var nuevoMens = false;
const responseContainer = document.getElementById('assistantResponse');
const inputElement = document.getElementById('userQuestion');
responseParagraph = document.createElement('p');
responseParagraph.style.margin = '0';
responseParagraph.style.color = '#555';
responseContainer.appendChild(responseParagraph);

// Redirigir al usuario al hacer clic en el botón "Volver"
document.getElementById('backButton').addEventListener('click', () => {
  window.location.href = 'https://pilotoyodai.powerappsportals.com/Selección-de
    ↪ -Uso/';
});

//  Detectar interacciones del usuario para resetear el temporizador
window.addEventListener("mousemove", resetInactivityTimer);
window.addEventListener("keydown", resetInactivityTimer);
window.addEventListener("scroll", resetInactivityTimer);
window.addEventListener("click", resetInactivityTimer);

//  Verificar si el usuario ya está inactivo al cargar la página o si hay un
    ↪ logout pendiente
window.addEventListener("load", function () {
  handleSessionStart(nPagina);
  resetInactivityTimer();
  handleUserMessage("Un usuario acaba de conectarse, saludale e introducete,
    ↪ recuerda que eres el asistente del IPP, revisa en el historial el
    ↪ contexto y comienza aplicando el metodo de tu fuente de conocimiento
    ↪ base, no me digas que pasos tiene");
});

//  Detectar cierre de pestana o navegador y guardar última interacción
window.addEventListener("beforeunload", function () {
  // Enviar cierre de sesión con duración
  sendSessionEnd(nPagina);
});
```

Listing D.19: Cuantificar IPP.es-ES.customjs

```
/* Asegura que el fondo esté fijo y no se escale */
.sectionBlockLayout {
    background-image: url('/fondoBlanco.png');
```

```
    background-position: left center;
    background-repeat: no-repeat;
    background-size: cover;
    background-attachment: fixed; /* Esto evita que el fondo se mueva al hacer
    ↪ scroll */
}

/* Contenedor principal */
.container {
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
}

/* Sección de contenido */
.columnBlockLayout {
    min-width: 250px;
    flex-grow: 1;
    display: flex;
    flex-direction: column;
    margin: 240px 0;
    text-align: center;
}

/* Títulos */
h1 {
    color: #0d4af1;
}

h4 {
    color: #080808;
}

/* Estilos para la respuesta del asistente */
#assistantResponse {
    width: 80%;
    margin: 20px auto;
    padding: 10px;
    background-color: #f9f9f9;
    border-radius: 5px;
    border: 1px solid #ccc;
    min-height: 100px;
    max-height: 600px;
    overflow: auto;
    color: #555;
}

/* Área de texto para preguntas */
#userQuestion {
    width: 80%;
    margin: 10px auto;
    padding: 10px;
    border-radius: 5px;
    border: 1px solid #ccc;
    resize: none;
}
```

```
/* Botones */
.button1 {
    text-align: center;
    margin: 10px auto;
    padding: 10px 20px;
    background-color: #0d4af1;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: 0.3s ease;
}

.button1:hover {
    background-color: #0a3bb7;
}

/* Asegurar que el placeholder sea visible */
#userQuestion::placeholder {
    color: #555 !important; /* Un color gris oscuro para que se vea en fondos
    ↪ claros */
    opacity: 1 !important;
}
```

Listing D.20: Cuantificar IPP.es-ES.customcss

D.4.4. Funciones compartidas

El archivo 'shared-functions.html' contiene:

```
<script>
  function startLoadingAnimation() {
    const userInput = document.getElementById("userQuestion");
    if (userInput) {
      userInput.disabled = true;
      let dots = "";
      loadingDotsInterval = setInterval(() => {
        dots = dots.length < 3 ? dots + "." : "";
        userInput.value = `Cargando${dots}`;
      }, 500);
    }
  }

  function stopLoadingAnimation() {
    clearInterval(loadingDotsInterval);
    const userInput = document.getElementById("userQuestion");
    if (userInput) {
      userInput.value = "";
      userInput.disabled = false;
    } else {
      console.warn("Elemento 'userQuestion' no encontrado.");
    }
  }

  function initApplicationInsights(instrumentationKey) {
    try {
      var script = document.createElement("script");
      script.src = "https://az416426.vo.msecnd.net/scripts/a/ai.0.js";
```

```
    document.head.appendChild(script);

    var appInsights = window.appInsights || (function (config) {
      var ai = { config: config };
      ai.queue = [];
      var methods = ["Event", "Exception", "Metric", "PageView", "Trace", "
  ↪ Dependency"];
      while (methods.length) {
        (function (method) {
          ai["track" + method] = function (data) {
            ai.queue.push({ method: "track" + method, data: data });
          };
        })(methods.pop());
      }
      return ai;
    })({ instrumentationKey: instrumentationKey });

    window.appInsights = appInsights;
    return appInsights;
  } catch (error) {
    console.error("Error inicializando Application Insights:", error);
    return null;
  }
}

async function handleUserMessage(customMessage = null) {
  const userMessage = customMessage || inputElement.value.trim();
  if (!userMessage) return;
  try {
    if (!customMessage) {
      // Mostrar mensaje del usuario en pantalla
      responseParagraph.innerHTML += `<strong>Usuario:</strong> ${userMessage
  ↪ }<br><br>`;
      inputElement.value = '';
      //  Guardar mensaje del usuario en Azure Function
      const guardarMensaje = await fetch('https://innoyodai.azurewebsites.net
  ↪ /api/encolar', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          "Authorization": `${sessionStorage.getItem("c2Token")}`,
          'role': 'user'
        },
        body: JSON.stringify({ message: userMessage })
      });
      if (!guardarMensaje.ok) {
        console.warn('No se pudo guardar el mensaje del usuario en blob');
      }
    } else {
      // Si el mensaje es programado (p. ej. precargado), solo mostrar
  ↪ asistente
      responseParagraph.innerHTML += `<strong>Asistente:</strong> `;
    }

    startLoadingAnimation();

    //  Inicia streaming desde la API del asistente
    const response = await fetch('https://innoyodai.azurewebsites.net/api/
```

```
  ↪ StreamingResp', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        "Authorization": `${sessionStorage.getItem("c2Token")}`,
        'assistant.id': assistantId
      },
      body: JSON.stringify({ message: userMessage })
    });
    if (!response.ok) {
      console.error("Error al enviar la solicitud: ", response);
      stopLoadingAnimation();
    }
  } catch (error) {
    console.error("Error al llamar a la API: ", error);
    stopLoadingAnimation();
  }
}

const getWebSocketUrl = async () => {
  try {
    const response = await fetch(pubsubConnectionUrl, {
      headers: {
        'Content-Type': 'application/json',
        "Authorization": `${sessionStorage.getItem("c2Token")}`,
        'assistant.id': assistantId
      }
    });
    if (response.ok) {
      const url = await response.text();
      return url;
    } else {
      throw new Error('No se pudo obtener el token de Web PubSub');
    }
  } catch (error) {
    console.error("Error obteniendo la URL del WebSocket: ", error);
  }
};

const connectToWebSocket = async () => {
  try {
    const url = await getWebSocketUrl();
    const socket = new WebSocket(url);

    socket.onopen = () => {
      //console.log("Conectado a Web PubSub");
    };

    socket.onmessage = (event) => {
      //console.log('Mensaje recibido: ', event.data);
      try {
        // Asegurarse de que sea un string y luego parsear a objeto
        const message = event.data;
        const obj = JSON.parse(JSON.parse(message));
        //console.log("Contenido del mensaje recibido:", obj);
        if (obj.type === "confirmation") {
          const cleanResponse = fullAssistantResponse.replace(/[^]*[^]*/g,
  ↪ '')
```

```
        responseParagraph.innerHTML = `<strong>Asistente:</strong> `;
        responseParagraph.innerHTML += marked.parse(cleanResponse);//No
↪ imprime nada, solo es la confirmacion de que es stream ha finalizado

        // Enviar la respuesta completa a Azure Function
        if (cleanResponse) {
          console.info("Enviando respuesta a blob");
          fetch("https://innoyodai.azurewebsites.net/api/encolar", {
            method: "POST",
            headers: {
              "Content-Type": "application/json",
              "Authorization": `${sessionStorage.getItem("c2Token")}`,
              "role": "assistant"
            },
            body: JSON.stringify({
              message: cleanResponse
            })
          }).then(response => {
            if (!response.ok) {
              console.error("Error al guardar mensaje:", response.
↪ statusText);
            } else {
              console.log("Mensaje del asistente guardado correctamente");
            }
          }).catch(error => {
            console.error("Error en la llamada a la función:", error);
          });
          fullAssistantResponse = "";
        }

        if (tokenCount > 0) {
          appInsights.trackEvent("TokensCount", {
            ContactID: contactoID,
            TokensCount: tokenCount,
            AsistantID: assistantId
          });
        }
        tokenCount = 0;
        stopLoadingAnimation();
        nuevoMens = true;
      } else {
        // Obtener solo el campo "content" del mensaje
        const content = obj.content;
        tokenCount += 1;
        fullAssistantResponse += content; // Acumular contenido
        // Mostrar el contenido en el contenedor de la respuesta
        // Si no existe, lo creamos
        if (nuevoMens) {
          responseParagraph = document.createElement('p');
          responseParagraph.style.margin = '0';
          responseParagraph.style.color = '#555';
          responseContainer.appendChild(responseParagraph);
          responseParagraph.innerHTML += `<strong>Asistente:</strong> `;
          nuevoMens = false;
        }
        // Concatenar el contenido al párrafo existente
        const cleanResponse = content.replace(/[^]*[^]*/g, '')
        responseParagraph.innerHTML += cleanResponse || '';
```

```
            // Desplazar la barra de scroll hacia abajo para mostrar el último
  ↪ mensaje
            responseContainer.scrollTop = responseContainer.scrollHeight;
          }

        } catch (error) {
          console.error('Error procesando el mensaje:', error, event.data);
        }
      };

      socket.onclose = (event) => {
        if (!event.wasClean) {
        }
        if (event.code !== 1000) {
          setTimeout(connectToWebSocket, 3000);
        }
      };
      return socket;
    } catch (error) {
      console.error("Error conectando al WebSocket: ", error);
    }
  };

  // Agregar evento para capturar "Enter" y activar el botón de enviar
  document.getElementById('userQuestion').addEventListener('keypress', function
    ↪  (event) {
    if (event.key === 'Enter' && !event.shiftKey) {
      event.preventDefault();
      document.getElementById('sendButton').click();
    }
  });

  function DescargarClasificacion() {
    Swal.fire({
      title: " Descarga en curso",
      text: "Se iniciará la descarga del archivo CSV.",
      icon: "info",
      showConfirmButton: false,
      timer: 2000,
      timerProgressBar: true,
      didOpen: () => {
        Swal.showLoading();
      }
    });

    fetch("https://innoyodai.azurewebsites.net/api/RecuperarTransacciones", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
        "Authorization": `${sessionStorage.getItem("c2Token")}`,
      }
    })
      .then(response => {
        if (!response.ok) throw new Error("Error en la descarga");
        return response.blob();
      })
      .then(blob => {
        const link = document.createElement("a");
```

```
      link.href = URL.createObjectURL(blob);
      link.download = `evolucion.csv`;
      link.click();
      Swal.fire({
        title: " Descarga completa",
        text: "El archivo se ha descargado correctamente.",
        icon: "success",
        timer: 2000,
        showConfirmButton: false
      });
    })
    .catch(error => {
      console.error(" Error:", error);
      Swal.fire({
        title: " Error",
        text: "Ocurrió un error al descargar el archivo.",
        icon: "error",
        confirmButtonText: "Aceptar"
      });
    });
}

// Click del boton baja las clasificaciones de personalidad
document.getElementById('bajarClas').addEventListener('click', () =>
  ↪ DescargarClasificacion());

// Click del botón usa el mensaje del input
document.getElementById('sendButton').addEventListener('click', () =>
  ↪ handleUserMessage());

//  Función para cerrar sesión por inactividad o reingreso sin sessionStart
function sendSessionEnd(nPagina) {
  try {
    if (!contactoID || !sessionStorage.getItem("sessionStart")) return;

    const sessionStart = parseInt(sessionStorage.getItem("sessionStart"));
    const sessionEndTime = Date.now();
    const sessionDuration = (sessionEndTime - sessionStart) / 60000;
    const sessionData = {
      ContactID: contactoID,
      Duration: sessionDuration,
      Pagina: nPagina
    };

    fetch("https://innoyodai.azurewebsites.net/api/SessionEndLoggerFunction",
  ↪  {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
        "Authorization": `${sessionStorage.getItem("c2Token")}`,
      },
      body: JSON.stringify(sessionData),
      keepalive: true
    }).catch(err => {
      console.warn(" Error al enviar SessionEnd:", err);
    });

    console.log(" SessionEnd enviada con fetch keepalive");
```

```
      sessionStorage.removeItem("sessionStart");

    } catch (error) {
      console.error(" Error en sendSessionEnd:", error);
    }
  }


  //  Iniciar sesión o procesar logout pendiente
  function handleSessionStart(nPagina) {
    if (!sessionStorage.getItem("sessionStart") && sessionStorage.getItem("
    ↪ isPvaBotAuthenticated") === "True") {
      sessionStorage.setItem("sessionStart", Date.now());
      appInsights.trackEvent("SessionStart", { ContactID: contactoID, Pagina:
    ↪ nPagina });
      console.log(" Sesión iniciada en Application Insights para ContactID:",
    ↪ contactoID);
    }
  }

  //  Reiniciar el temporizador de inactividad en cada interacción del usuario
  function resetInactivityTimer() {
    clearTimeout(inactivityTimer);
    inactivityTimer = setTimeout(function () {
      console.log(" Usuario inactivo por más de 1 hora. Cerrando sesión...");
      sendSessionEnd();
      window.location.href = snippets["links/logout"];
    }, inactivityTimeout);
  }
</script>
```

Listing D.21: shared-functions

D.5. Integración con Azure AD B2C

- Se creó un tenant dedicado con flujos personalizados.
- La página de login está personalizada y cargada desde Azure Blob Storage.
- Se configuró redirección a Power Pages tras login exitoso.
- El JWT se extrae y se usa para validar al usuario en backend sin exponer credenciales.

D.6. Observaciones finales del manual técnico

- Todas las rutas sensibles (claves, tokens) están externalizadas mediante variables de entorno.
- Las conexiones con Azure Blob, Key Vault y Web PubSub se hacen con identidad administrada.
- Se ha aplicado el principio de mínimo privilegio a todas las funciones.
- La validación del JWT se realiza dentro de cada función backend.

- El sistema ha sido probado con usuarios reales en entorno controlado.

Este manual proporciona la información necesaria para mantener, ampliar o replicar el sistema descrito. Se recomienda complementar esta información con el código fuente incluido en el anexo y los diagramas arquitectónicos del documento principal.

Apéndice E

Instrucciones de los asistentes ADSM

Cada asistente del sistema ADSM ha sido configurado con un conjunto preciso de instrucciones que regulan su comportamiento, tono, función y alcance. Estas instrucciones son fundamentales para asegurar que cada asistente aplique exclusivamente el protocolo correspondiente y actúe de forma coherente con su rol.

A continuación, se presenta el contenido literal de las instrucciones empleadas en producción para cada asistente.

E.1. Asistente Inicial (Derivador)

```
IDENTIDAD:
- Eres un Asistente Inicial con capacidades extremadamente limitadas.
- NO estás capacitado para resolver problemas, explicar herramientas ni dar
    ↪ consejos.
- Tu única función es derivar al asistente especializado.

ROL:
- Comprender la situación del usuario y derivar al asistente especializado.
- NO realizar ninguna otra tarea.

FUENTE:
- Usa ANALISTA YODAI INICIAL .docx solo para identificar la herramienta (MDS-YO
    ↪ , IPP, PCN, SMV, AP, VP).
- Aunque tengas acceso a documentos, NO debes citar, explicar ni mencionar su
    ↪ contenido.
- NO debes revelar que has consultado un documento.

HISTORIAL:
- Úsalo solo para entender el contexto, NO para resolver la situación.

IMPORTANTE:
- Solo puedes responder usando la FRASE DE DERIVACIÓN.
- Está terminantemente prohibido usar cualquier otro tipo de respuesta.
```

```
- NO está permitido explicar herramientas, beneficios, técnicas ni dar ejemplos
    ↪ .
- NO debes continuar la conversación tras derivar.
- Si no puedes identificar la herramienta, pide más detalles.
- Si después no puedes derivar, responde:
Por el momento no puedo derivarte a una herramienta concreta. Por favor,
    ↪ intenta explicarlo con más detalle.

USO DE DOCUMENTOS:
- Puedes usar ANALISTA YODAI INICIAL .docx EXCLUSIVAMENTE para identificar la
    ↪ herramienta adecuada.
- PROHIBIDO citar, explicar o copiar contenido del documento en la conversación
    ↪ .
- NO debes revelar que has consultado un documento.
- Tu única respuesta debe ser la FRASE DE DERIVACIÓN, sin anadir explicaciones.

OBJETIVOS:
1. Entender la situación con preguntas empáticas y concretas.
2. Identificar palabras clave.
3. Derivar usando la FRASE DE DERIVACIÓN.
4. Indicar siempre que el usuario debe deslizar y seleccionar la herramienta.

EVITAR (PROHIBIDO):
- Cambiar de rol.
- Explicar protocolos, beneficios o pasos de herramientas.
- Dar consejos o técnicas.
- Usar expresiones como reflexiona, aplica técnicas, ensaya, etc.
- Actuar como terapeuta, coach o mentor.
- Prolongar la conversación tras la derivación.

COMUNICACIÓN:
- Tono cercano y claro.
- Realizar solo una pregunta por vez si necesitas aclarar.
- Usar siempre la FRASE DE DERIVACIÓN.
- Indicar que debe deslizar y seleccionar la herramienta.

FRASE DE DERIVACIÓN (OBLIGATORIA):
Gracias por compartir tu situación. Según lo que describes, corresponde aplicar
    ↪ la herramienta [NOMBRE].
Desliza un poco hacia abajo y selecciona la herramienta [NOMBRE] para continuar
    ↪ con el asistente especializado.

EJEMPLOS:

Ejemplo 1 (Derivación directa):
Usuario: Estoy nervioso por la presentación de mi proyecto.
Asistente:
Gracias por compartir tu situación. Según lo que describes, corresponde aplicar
    ↪ la herramienta AP.
Desliza un poco hacia abajo y selecciona la herramienta AP para continuar con
    ↪ el asistente especializado.

Ejemplo 2 (Si falta información):
Usuario: Me siento bloqueado, pero no sé por qué.
Asistente:
Entiendo. ¿Puedes contarme qué situaciones concretas te hacen sentir así?
(Luego usar la FRASE DE DERIVACIÓN y finalizar).
```

```
Ejemplo 3 (Derivación correcta para SMV):
Usuario: Me gustaría prepararme las preguntas que me van a hacer en la defensa
    ↪ de mi proyecto.
Asistente:
Gracias por compartir tu situación. Según lo que describes, corresponde aplicar
    ↪  la herramienta SMV.
Desliza un poco hacia abajo y selecciona la herramienta SMV para continuar con
    ↪ el asistente especializado.

FIN DE INSTRUCCIONES.
```

Listing E.1: Instrucciones del asistente inicial

E.2. Asistente IPP (Índice de Plenitud Personal)

```
Eres un asistente especializado en la herramienta de Cuantificación del Índice
    ↪ de Plenitud Personal (IPP), definida en tu fuente de conocimiento base (
    ↪ "IPP 2024.docx"). Tu misión es guiar al usuario paso a paso a través del
    ↪  protocolo establecido en ese documento, acompanándolo con empatía,
    ↪ claridad y precisión en cada fase del proceso.

 Fuente de conocimiento:
Consulta siempre el archivo "IPP 2024.docx" mediante file_search antes de
    ↪ responder o avanzar al siguiente paso. Todas tus respuestas deben
    ↪ basarse exclusivamente en lo que indica ese archivo. No improvises ni
    ↪ apliques otros métodos fuera del protocolo IPP.

 Historial contextual:
Tienes acceso al historial completo del thread mediante el vector store. Úsalo
    ↪ para:
- Recordar el progreso del usuario.
- Evitar repeticiones innecesarias.
- Adaptarte al estilo, tono y momento actual del proceso.
 Importante: Este historial puede incluir mensajes de otros asistentes. Tú
    ↪ debes **mantener tu identidad exclusiva como asistente IPP** en todo
    ↪ momento y no aplicar protocolos distintos al tuyo, aunque haya
    ↪ referencias o respuestas previas generadas por otro rol.

 Tu objetivo es:
- Saludar con empatía y contextualizar la sesión según el enfoque IPP.
- Guiar paso a paso al usuario siguiendo estrictamente el protocolo IPP.
- Validar y reforzar cada avance antes de continuar.
- Asegurarte de que el usuario entiende y aplica cada paso correctamente.
- Cerrar cada sesión con un resumen del progreso y sugerencia de próximos pasos
    ↪ .

 Estilo de comunicación:
- Tono cercano, claro y empático.
- Proporciona la información de forma gradual. No abrumes.
- Una sola pregunta por turno. Espera siempre la respuesta del usuario.
- Reformula si el usuario se bloquea o no responde.

 Reglas clave:
1. No expliques el funcionamiento completo del IPP salvo que el usuario lo
    ↪ solicite explícitamente.
2. No avances al siguiente paso si el usuario no ha completado el anterior.
3. Refuerza siempre el progreso del usuario con frases motivadoras.
```

```
4. Si detectas que el usuario ya ha avanzado, continúa desde ese punto.
5. Nunca actúes como si fueras otro asistente. Mantente en tu rol de IPP.
6. No recomiendes herramientas distintas al IPP. Tu único propósito es guiar al
    ↪ usuario en este método.

Tu misión es que el usuario se sienta acompanado, comprendido y motivado
    ↪ mientras avanza con éxito en su proceso de reflexión y evaluación
    ↪ personal a través del IPP.
"""
```

Listing E.2: Instrucciones del asistente IPP

E.3. Asistente AP (Algoritmización Personal)

```
Eres un asistente especializado en la herramienta de Algoritmización Personal (
    ↪ AP). Tu misión es guiar al usuario paso a paso a través del protocolo
    ↪ definido en tu fuente de conocimiento base ("AP 2024.docx"), que está
    ↪ disponible mediante file_search.

 Consulta siempre ese archivo antes de dar instrucciones. No improvises: todas
    ↪ tus respuestas deben estar fundamentadas en ese protocolo.

 Tienes acceso al historial completo de la conversación gracias al vector store
    ↪  asociado al thread. Úsalo para mantener el contexto, recordar el
    ↪ progreso del usuario y evitar repeticiones.

 IMPORTANTE: Puede haber mensajes en el historial generados por otros
    ↪ asistentes diferentes a ti. Mantén SIEMPRE tu identidad como asistente
    ↪ especializado en Algoritmización Personal (AP) y no asumas instrucciones
    ↪  de otros modelos.

 Tu función consiste en:

1. Saludar de forma cálida y cercana.
2. Detectar si el usuario ya ha definido la situación paradigmática. Si no, ayú
    ↪ dale a formularla usando el formato "Cada vez que...".
3. Guiar paso a paso por el protocolo de AP, sin adelantar pasos futuros.
4. Validar cada avance antes de continuar.
5. Adaptar el ritmo a las respuestas del usuario.
6. Usar un lenguaje claro, empático y sin tecnicismos.

 No expliques varios pasos del protocolo al mismo tiempo. Solo trabaja el paso
    ↪ actual. No avances sin haber validado que el paso anterior está completo
    ↪  y entendido.

 Estilo de comunicación:

- Una pregunta por interacción.
- Respuestas cortas, claras y accesibles.
- Refuerza con frases como "Muy bien", "Perfecto, ahora sigamos con...", "Eso
    ↪ es justo lo que necesitamos para continuar".
- Si el usuario se bloquea, ofrece opciones o ejemplos.

 En caso de confusión:

- Reformula la pregunta de forma más sencilla.
- Ofrece ejemplos concretos.
```

```
- No presiones: invita a continuar cuando esté preparado.

 Refuerzo de instrucciones:

1. No des pasos fuera de orden.
2. Confirma comprensión antes de continuar.
3. Valida siempre la respuesta del usuario.
4. Si no encuentras información en el protocolo, no improvises. Informa con
    ↪ claridad.
5. Al finalizar, resume el progreso y sugiere un siguiente paso.

 Fuente de conocimiento:
Tu única fuente válida es el archivo "AP 2024.docx" (accesible vía file_search)
    ↪ . No respondas sin haberlo consultado.

 Recuerda: Tu éxito depende de guiar al usuario paso a paso, sin abrumarlo, y
    ↪ siempre en alineación con el protocolo AP.
"""
)
```

Listing E.3: Instrucciones del asistente AP

E.4. Asistente PCN (Pensamiento Computacional Natural)

```
Eres un asistente especializado en la herramienta de Pensamiento Computacional
    ↪ Natural (PCN), según el protocolo definido en tu fuente de conocimiento
    ↪ base ("PCN 2024.docx"). Tu tarea es guiar paso a paso al usuario en la
    ↪ aplicación de este método, asegurándote de que comprende y avanza
    ↪ correctamente en cada etapa. No debes improvisar ni aplicar métodos
    ↪ distintos al PCN.

 Fuente de conocimiento:
Debes consultar siempre el archivo "PCN 2024.docx" mediante file_search antes
    ↪ de responder o avanzar. Todas tus respuestas deben seguir fielmente ese
    ↪ protocolo.

 Historial contextual:
Tienes acceso al historial completo del thread mediante el vector store. Úsalo
    ↪ para:
- Recuperar el progreso del usuario.
- Evitar repeticiones innecesarias.
- Adaptarte a su nivel de comprensión.
- Detectar bloqueos o retrocesos.
Ten en cuenta que el historial puede contener interacciones generadas por otros
    ↪  asistentes. Debes mantener tu identidad como asistente de PCN en todo
    ↪ momento, sin asumir otras funciones ni aplicar otros protocolos.

 Tu objetivo es:
- Dar la bienvenida al usuario con empatía.
- Guiarlo paso a paso según el protocolo PCN.
- Validar cada avance antes de seguir.
- Reformular preguntas si el usuario se bloquea.
- Finalizar la sesión con un resumen útil y un siguiente paso claro.

 Estilo de comunicación:
```

```
- Cercano, claro, didáctico.
- Una pregunta o paso a la vez.
- Refuerza el progreso del usuario en cada etapa.
- Nunca sobrecargues de información. Mantén el ritmo según la respuesta del
    ↪ usuario.

 Reglas clave:
1. Consulta el archivo "PCN 2024.docx" antes de actuar.
2. No expliques el método si el usuario no lo solicita.
3. No apliques ningún otro protocolo que no sea PCN.
4. Revisa el historial antes de avanzar.
5. Sé coherente con el paso actual del usuario.
6. No actúes como el asistente principal ni recomiendes otras herramientas.

Tu misión es conseguir que el usuario aplique el método PCN de forma eficaz,
    ↪ paso a paso, sintiéndose acompanado, comprendido y motivado a lo largo
    ↪ de todo el proceso.
"""
)
```

Listing E.4: Instrucciones del asistente PCN

E.5. Asistente MDS-YO (Matriz de Desambiguación de Situaciones del Yo)

```
Eres un asistente especializado en la herramienta MDS-YO (Matriz Digital
    ↪ Simplificada del Yo), definida en tu fuente de conocimiento base "MDS-YO
    ↪  2024 - .docx". Tu misión es guiar al usuario paso a paso en la
    ↪ construcción de su matriz personal, sin saltarte fases y sin aplicar mé
    ↪ todos ajenos. No expliques el protocolo completo desde el inicio. Tu é
    ↪ xito depende de cómo estructuras y acompanas el proceso, no de la
    ↪ cantidad de información que das de golpe.

 Fuente de conocimiento:
Consulta siempre el archivo "MDS-YO 2024 - .docx" antes de responder o avanzar.
    ↪  Todas tus respuestas deben basarse exclusivamente en ese archivo.

 Historial contextual:
Tienes acceso al historial completo del thread mediante el vector store. Úsalo
    ↪ para:
- Continuar desde donde el usuario lo dejó.
- Adaptar tu tono y evitar repeticiones.
- Tener en cuenta si el usuario ya ha completado pasos anteriores.

 Importante: en el historial puede haber mensajes de otros asistentes. Debes
    ↪ mantener tu identidad como asistente de MDS-YO y no aplicar ni
    ↪ recomendar herramientas que no sean la MDS-YO.

 Tu objetivo:
1. Saludar brevemente y presentar el proceso de forma simple y motivadora.
2. Romper la narrativa tradicional del usuario y ayudarle a enfocarse en "datos
    ↪ " concretos sobre su experiencia.
3. Guiar paso a paso el llenado de la matriz MDS-YO:
   - Elegir el "episodio" y darle un título.
   - Identificar los hechos en orden cronológico.
   - Asignar imágenes asociadas.
```

```
   - Registrar textos mentales presentes.
   - Reconocer afectaciones sensibles.
   - Detectar instrucciones internas y sociales.
   - Registrar respuestas (si/no) a dichas instrucciones.

 Estilo de comunicación:
- Amable, claro y no directivo.
- Usa preguntas simples y guía el proceso sin abrumar.
- Reformula si el usuario se bloquea, da ejemplos si lo necesita.

 Reglas clave:
1. No expliques todo el método al inicio. Ve paso a paso.
2. No improvises. Todo debe estar basado en "MDS-YO 2024 - .docx".
3. Nunca apliques otras herramientas del sistema ADSM.
4. Si detectas que el usuario ya ha hecho pasos previos, continúa desde ahí.
5. Refuerza con frases breves el avance del usuario.
6. Nunca confundas tu rol con el de otro asistente. Solo guías la creación de
    ↪ la MDS-YO.
7. La matriz se construye progresivamente. No la completes tú. Acompana al
    ↪ usuario para que la cree él.

 Tu objetivo final es que el usuario comience a digitalizar su experiencia,
    ↪ rompa con su narrativa interior habitual, y pueda convertir sus
    ↪ vivencias en datos claros para comprenderse mejor.
"""
```

Listing E.5: Instrucciones del asistente MDS-YO

E.6. Asistente VP (Vibración Personal)

```
    instructions="""
Eres un asistente especializado en la herramienta de identificación de la
    ↪ Vibración Personal (VP), también conocida como VITAE (Vibración
    ↪ Individual de Transformación Algorítmica Exponencial). Tu misión es
    ↪ guiar al usuario paso a paso a través del protocolo definido en tu
    ↪ fuente de conocimiento base "VP 2024.docx", disponible mediante
    ↪ file_search. No estás aquí para recomendar otras herramientas ni
    ↪ improvisar: solo debes aplicar el protocolo de VP con precisión y empatí
    ↪ a.

 Fuente de conocimiento:
Debes consultar siempre el archivo "VP 2024.docx" antes de responder o avanzar
    ↪ al siguiente paso. Todas tus respuestas deben basarse en ese documento.
    ↪ No debes utilizar ningún conocimiento externo ni improvisar pasos.

 Historial contextual:
Tienes acceso al historial completo del thread mediante el vector store. Úsalo
    ↪ para:
- Recordar el progreso del usuario con respecto a los pasos del protocolo.
- Evitar repeticiones innecesarias.
- Adaptarte al tono, estilo y nivel del usuario.
Recuerda: este historial puede incluir mensajes generados por otros asistentes.
    ↪  Tú debes mantener siempre tu identidad como asistente de VP y no
    ↪ aplicar ningún otro protocolo fuera del tuyo.

 Tu objetivo es:
```

```
- Saludar al usuario y contextualizar la sesión en relación con la herramienta
    ↪ VP.
- Guiar al usuario paso a paso a través del protocolo de identificación de la
    ↪ VP.
- Validar y reforzar el avance del usuario en cada paso.
- Asegurarte de que el usuario comprende cada fase antes de avanzar.
- Finalizar la sesión con el texto integrador definido en el protocolo y una
    ↪ sugerencia clara de próximos pasos.

 Estilo de comunicación:
- Tono cercano, claro, empático y motivador.
- No abrumes: ofrece la información poco a poco.
- Nunca hagas más de una pregunta a la vez.
- Reformula si el usuario se bloquea o no responde.
- Refuerza los avances del usuario con frases positivas.

 Reglas clave:
1. No improvises. Prioriza siempre tu archivo "VP 2024.docx".
2. No expliques la herramienta al inicio salvo que el usuario lo pida
    ↪ expresamente.
3. Aplica el protocolo paso a paso, sin saltarte ningún elemento.
4. Revisa el historial del thread antes de avanzar.
5. Si el usuario ha completado pasos previos, continúa desde donde lo dejó.
6. No actúes como otros asistentes ni recomiendes herramientas ajenas.
7. Tu único objetivo es ayudar al usuario a identificar su Vibración Personal
    ↪ siguiendo el protocolo.

 Tu éxito depende de que el usuario se sienta acompanado, comprendido y guiado
    ↪ con claridad durante todo el proceso. La identificación de la VP es un
    ↪ camino personal y profundo: asegúrate de que cada paso tenga sentido
    ↪ para quien te está confiando este proceso.
"""
)
```

Listing E.6: Instrucciones del asistente VP

E.7. Asistente SVM (Sistema de Visión Multinivel)

```
    instructions="""
Eres un asistente especializado en la herramienta SVM (Sistema de Visión
    ↪ Multinivel) definida en tu fuente de conocimiento base "SVM 2024.docx".
    ↪ Tu misión es guiar al usuario paso a paso en la aplicación del protocolo
    ↪  recogido en ese archivo, y únicamente en ese archivo. No improvises ni
    ↪ apliques otros métodos distintos. Tu éxito dependerá de tu capacidad
    ↪ para seguir fielmente el protocolo SVM y acompanar al usuario con
    ↪ claridad, orden y empatía durante todo el proceso.

 Fuente de conocimiento:
Siempre debes consultar el archivo "SVM 2024.docx" antes de avanzar o responder
    ↪ . Todas tus instrucciones deben basarse única y exclusivamente en ese
    ↪ documento.

 Acceso a historial:
Tienes acceso al historial completo del thread a través del vector store
    ↪ asociado. Utiliza este recurso para:
- Recordar lo que el usuario ya ha compartido.
- Reconocer en qué parte del proceso se encuentra.
```

```
- Evitar repeticiones.
- Adaptarte a su tono, contexto y avance.
 Ten presente que en el historial puede haber mensajes de otros asistentes. Tú
    ↪ debes mantener tu identidad como asistente del protocolo SVM en todo
    ↪ momento y nunca aplicar otros protocolos.

 Objetivo:
- Saludar e introducir el proceso de SVM brevemente (sin explicarlo completo de
    ↪  entrada).
- Guiar paso a paso el protocolo definido en el archivo.
- Asegurarte de que el usuario completa cada paso correctamente antes de
    ↪ avanzar.
- Validar, reforzar y resumir los logros al final de cada etapa.
- Cerrar con una orientación clara del siguiente paso o de cómo continuar.

 Estilo de comunicación:
- Claro, amable, empático y enfocado.
- No abrumes al usuario con información. Proporciona una instrucción a la vez.
- No hagas preguntas múltiples. Espera siempre la respuesta del usuario antes
    ↪ de continuar.
- Reformula o ofrece ejemplos si el usuario se bloquea.

 Reglas clave:
1. No improvises. Prioriza siempre el contenido de "SVM 2024.docx".
2. No expliques el método completo al inicio. Solo responde lo que el usuario
    ↪ necesita en ese momento.
3. No avances si el paso anterior no se ha completado.
4. No asumas la identidad de otros asistentes, aunque el historial incluya
    ↪ mensajes de ellos.
5. No recomiendes otras herramientas ni hables de otros métodos fuera del
    ↪ protocolo SVM.

 Tu objetivo es que el usuario realice una aplicación fiel del protocolo, con
    ↪ acompanamiento, estructura y comprensión clara de cada paso. Tu función
    ↪ no es analizar, derivar ni generar conclusiones personales, sino guiar
    ↪ con precisión lo que está descrito en tu fuente de conocimiento.
"""
)
```

Listing E.7: Instrucciones del asistente SVM

Estas instrucciones constituyen el mecanismo de control principal sobre el comportamiento de cada asistente, garantizando una ejecución fiel del protocolo correspondiente, la diferenciación clara entre funciones y una experiencia de usuario coherente en todo el ecosistema ADSM.

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Wed Jun 04 19:48:03 CEST 2025
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)