

PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR
SEDE MANABÍ

CARRERA DE SOFTWARE

TRABAJO DE TITULACIÓN

DESARROLLO DE UNA SOLUCIÓN TECNOLÓGICA BASADA EN IA PARA LA AUTOMATIZACIÓN DE PEDIDOS EN FLORERÍAS, APLICADO EN LA FLORERÍA "AURA" EN MANTA

LÍNEA DE INVESTIGACIÓN

TECNOLOGÍA DE LA INFORMACIÓN Y LA COMUNICACIÓN

SUBLÍNEA DE INVESTIGACIÓN

INGENIERÍA DE SOFTWARE INNOVACIÓN Y EMPRENDIMIENTO EN TIC

PREVIO AL TÍTULO DE

INGENIERO EN SOFTWARE

AUTORA
STIVEN ROLANDO GUANOQUIZA TOAPANTA

TUTOR
ING. JOSSELYN TATIANA GÓMEZ BRAVO, MSc.

PORTOVIEJO, ABRIL 2026

Certificación del Tutor

En mi calidad de tutor del trabajo de integración curricular, certifico haber revisado el presente manuscrito de investigación, el mismo que se ajusta a las normas vigentes de la Pontificia Universidad Catolica del Ecuador Sede Manabí, cumpliendo la Normativa del Trabajo de Integración Curricular; en consecuencia, es apto para su presentación y sustentación.

En la ciudad de Portoviejo a los veinticinco días del mes de marzo de dos mil veinte y seis.

ING, JOSSELYN TATIANA GOMEZ BRAVO MSC.

CI: 1315046498

Tutora

2026

Acta de aprobación del Tribunal

El jurado examinador aprueba el presente trabajo de integración curricular en nombre de la Pontificia Universidad Católica del Ecuador, Sede Manabí.

En la ciudad de Portoviejo a los veinticinco días del mes de marzo de dos mil veinte y seis.

ING. SONIA GARCIA, MSC.
CI: 1310948185
Lector Revisor 1

ING. VICTOR ALONZO, MSC.
CI: 1307757441
Lector Revisor 2

ING. JOSSELYN GOMEZ, MSC.
CI: 1315046498
Tutora

2026

Declaración de Originalidad

Este manuscrito no contiene ningún tipo de material que ha sido aceptado para la obtención de un título universitario en otra institución, excepto en forma de información de soporte que ha sido debidamente citada en mi trabajo. Este trabajo es de total responsabilidad del autor, quien declara bajo juramento que ninguna sección de este trabajo de integración curricular infringe los derechos de autor de nadie.

En la ciudad de Portoviejo a los veinticinco días del mes de marzo de dos mil veinte y seis.

STIVEN ROLANDO GUANOQUIZA TOAPANTA

CI: 1316833167

Autor

2026

Declaración de Derecho del Autor

Autorizo a la Pontificia Universidad Católica del Ecuador a distribuir este manuscrito de investigación en medios físicos y electrónicos con el fin de promover la divulgación de mis resultados a la comunidad científica y a la sociedad en general. Adicionalmente autorizo el uso de los contenidos de esta investigación como bibliografía para fines académicos, por cualquier medio o procedimiento, citando como fuente de información al autor de este trabajo. En la ciudad de Portoviejo a los veinticinco días del mes de marzo de dos mil veinte y seis.

STIVEN ROLANDO GUANOQUIZA TOAPANTA

CI: 1316833167

Autor

2026

Dedicatoria

Este trabajo se lo dedico a mis padres, por ser el pilar fundamental de mi vida, por su apoyo incondicional, sus consejos y el ejemplo de perseverancia que siempre me han brindado.

A todas las personas que, con paciencia y generosidad, compartieron conmigo su conocimiento y contribuyeron a mi crecimiento académico y profesional. Cada enseñanza recibida ha sido clave para convertirme poco a poco en el profesional que estoy llegando a ser.

Agradecimientos

Expreso mi más sincero agradecimiento a mis padres, quienes con su apoyo incondicional, sacrificio y cariño han sido la base fundamental de mi formación. Su confianza en mí me ha impulsado a seguir adelante y a esforzarme cada día por alcanzar mis metas.

Agradezco profundamente a mis docentes y a todos los profesionales que, con paciencia y dedicación, compartieron conmigo sus conocimientos. Cada enseñanza recibida ha sido clave para mi desarrollo como estudiante y futuro ingeniero de software.

Extiendo también mi gratitud a todas las personas que colaboraron directa o indirectamente en la realización de este proyecto, por su disposición, orientación y aporte técnico. Su participación permitió que este trabajo se llevara a cabo con éxito.

Y finalmente, a mis compañeros y amigos, agradezco su experiencia, su motivación constante y esa sana competitividad que me impulsó a mejorar día a día. Gracias por compartir sus conocimientos, por aprender juntos y por ser parte de un entorno que me desafió a llevar mi aprendizaje a otro nivel.

A todos, muchas gracias.

Resumen

La presente investigación abordó el problema de alta latencia de respuesta en el canal de WhatsApp de la Florería Aura (Manta), con el propósito de optimizar el tiempo de atención al cliente mediante la implementación de un sistema automatizado basado en un chatbot. La solución se desarrolló con una arquitectura orquestada en n8n, integrando un chatbot con modelos de lenguaje (LLM), Redis para la gestión de concurrencia y PostgreSQL para la persistencia transaccional de pedidos. Para evaluar su desempeño en condiciones reales de operación, se aplicó un diseño cuasi-experimental pretest-postest de un solo grupo, comparando una línea base manual con un despliegue piloto en producción. Los resultados muestran una reducción del 97.73 respuesta, que pasó de 13.58 minutos a 18.50 segundos, junto con una estabilización de la variabilidad del servicio ($\sigma = 3.22\,s$). Se concluye que la automatización implementada incrementa la oportunidad de respuesta y la continuidad operativa del servicio, manteniendo el alcance de los hallazgos al caso de estudio y aportando una base técnica para su evolución hacia comercio conversacional.

Palabras clave: automatización, chatbot, microservicios, orquestación, rendimiento

Abstract

The present study addressed the issue of high response latency in the WhatsApp channel of Florería Aura (Manta), with the aim of optimizing customer service response time through the implementation of an automated chatbot-based system. The solution was developed using an orchestrated architecture in n8n, integrating a chatbot powered by large language models (LLMs), Redis for concurrency management, and PostgreSQL for transactional order persistence. To evaluate its performance under real operational conditions, a single-group quasi-experimental pretest– posttest design was applied, comparing a manual baseline with a pilot deployment in a production environment. The results indicate a 97.73average response time, decreasing from 13.58 minutes to 18.50 seconds, along with a stabilization of service variability ($\sigma = 3.22\,s$). It is concluded that the implemented automation enhances response timeliness and service continuity, while maintaining the scope of the findings within the case study and providing a technical foundation for its evolution toward conversational commerce.

Keywords: automation, chatbot, microservices, orchestration, performance

Tabla de Contenido

1. Introducción **1**

Antecedente 2

Planteamiento del problema 3

Formulación del problema 4

Pregunta general 4

Preguntas específicas 4

Objetivos de la investigación 5

Objetivo general 5

Objetivos específicos 5

Justificación 5

Contribución científica 6

Alcance 6

Marco conceptual 7

2. Método **10**

Diseño de la investigación 10

Enfoque de investigación 10

Tipo de investigación 10

Variables de estudio y operacionalización 11
Población y Muestra 11
Metodología 13
Técnicas e instrumentos de recolección de datos 14
Consideraciones éticas 16
Procedimiento 17
Diseño de la solución 17
Desarrollo 22
Validación 42
Evaluación y ajustes 45

3. Resultados **47**

Resultados asociados al objetivo específico 1 47
Resultados asociados a los objetivos específicos 2 y 3 48
Estructura conversacional diseñada 48
Implementación e integración del sistema 48
Resultados asociados al objetivo específico 4 49
Estadística descriptiva del tiempo de respuesta 49
Distribución de frecuencias y estabilidad 49
Análisis comparativo 50
Estimación de ahorro operativo en horas 51

4. Discusión **52**

Interpretación de la eficiencia operativa 52

Implicaciones para la escalabilidad del negocio 52

Evaluación crítica del método en función de los resultados 53

Limitaciones del estudio 54

5. Conclusiones **55**

Referencias bibliográficas **64**

Lista de Figuras

1. Comparación de tiempos de respuesta: manual vs. automatizado. 50

2. Diagrama de flujo conversacional y estados del sistema (parte 1 de 2). 71

3. Diagrama de flujo conversacional y estados del sistema (parte 2 de 2). 72

4. Diagrama de infraestructura de red y servicios Zona 1. 75

5. Diagrama de infraestructura de red y servicios Zona 2. 76

6. Diagrama de infraestructura de red y servicios Zona 3. 77

7. Diagrama entidad-relación (DER) detallado. 81

8. Recepción y filtrado de eventos 86

9. Gestión de concurrencia 87

10. Consolidación y reconstrucción de contexto 88

11. Normalización y control de estado 89

12. Cerebro central y contexto 90

13. Manejo de ambigüedad 91

14. Agente de información 92

15. Gestión transaccional 93

16. Intervención humana 94

17. Punto de bifurcación 95

18. Escucha de eventos y prevención de bucles 97

19. Reconstrucción del contexto 98

20. Enrutador de máquina de estados 99

21. Ejecución de aprobación 100

22. Ejecución de rechazo o pausa 101

Lista de Tablas

1. *Operacionalización de variables del estudio* 11
2. *Tabla de estados del pedido y transiciones permitidas* 19
3. *Matriz de resultados de validación funcional* 45
4. *Métricas de desempeño operativo manual* 48
5. *Resumen estadístico del tiempo de respuesta automatizado* 49
6. *Distribución de frecuencia de los tiempos de respuesta* 49
7. *Cuadro comparativo de impacto: situación anterior vs. actual (valores normalizados a segundos)* 51
8. Distribución de frecuencias de tiempos de respuesta 67
9. Matriz de Requisitos Funcionales 103
10. Matriz de requisitos no funcionales 104

Lista de Apéndices

- Apéndice A. Informe Técnico de Diagnóstico
- Apéndice B. Descripción detallada del flujo conversacional
- Apéndice C. Especificaciones Técnicas de la Infraestructura
- Apéndice D. Especificaciones del esquema de base de datos
- Apéndice E. Especificación técnica de datos y lógica de estado en chatwoot
- Apéndice F. Especificación técnica del flujo de orquestación en n8n principal
- Apéndice G. Especificación técnica del flujo de orquestación en n8n secundario
- Apéndice H. Especificación de requisitos de software

Introducción

La inteligencia artificial (IA) ha sido ampliamente estudiada en contextos internacionales y aplicada en sectores como la industria, la logística (De Felipe Iglesias, 2023), la gestión (Jaimes-Quintanilla & Zabala-Vargas, 2024) y en ámbitos nacional y local (Hidalgo Zambrano et al., 2024), la literatura se ha centrado principalmente en procesos generales de digitalización (Verhoef et al., 2021), automatización (Coombs et al., 2020) y adopción de TIC (Eller et al., 2020). No obstante, se identifica una brecha significativa, ya que ninguno de los estudios revisados aborda de manera específica el uso de herramientas potenciadas con IA para la automatización de pedidos en negocios locales de menor escala. Esta ausencia de literatura y aplicaciones prácticas representa una oportunidad para desarrollar soluciones tecnológicas accesibles que puedan integrarse de manera práctica en el flujo de trabajo de una pequeña empresa.

Este trabajo aborda dicha brecha a través de un caso concreto: la florería Aura en Manta. El establecimiento experimenta una contradicción operativa común, donde la demanda de atención inmediata a través de canales digitales como WhatsApp choca con la realidad de unos recursos humanos limitados, generando demoras perjudiciales para la experiencia del cliente y la rentabilidad del negocio. El proyecto propone desarrollar e implementar un sistema automatizado de asistencia basado en un chatbot para este canal, con el objetivo específico de reducir significativamente los tiempos de respuesta. Esta investigación busca, por tanto, ofrecer una solución práctica a un problema operativo real y, simultáneamente, contribuir a un área de investigación poco explorada.

Antecedente

Hace unas décadas, la inteligencia artificial (IA) (McCarthy et al., 1955) solía estar intrínsecamente relacionado con narrativas futuristas e irreales, a menudo limitadas al imaginario de la ciencia ficción y distantes de una aplicación práctica tangible (Russell & Norvig, 2021), pero ha evolucionado hacia convertirse en una herramienta cotidiana de optimización de diversos procesos empresariales (Chui et al., 2023; Dell'Acqua et al., 2023; Sarker, 2021).

A nivel internacional, se documenta que la inteligencia artificial aplicada a la automatización busca beneficios como la reducción de errores y costos, el aumento de la eficiencia y la mejora de la productividad, aunque también se reconocen efectos negativos a nivel laboral y económico (Ruiz Valdez, 2020). No obstante, existe una brecha en cuanto a estrategias que integren efectivamente la inteligencia artificial con el equipo humano (Enholm et al., 2022). Estudios como el de Mosquera (2024) destacan el papel de la IA en la identificación de oportunidades de mercado, la optimización de modelos de negocio y la mitigación de riesgos, concluyendo que su integración conduce a decisiones más informadas y a una mayor competitividad.

En el ámbito específico de la atención al cliente mediante chatbots, investigaciones recientes aportan evidencia directa. Cordero et al. (2022) evaluaron la usabilidad de tres chatbots implementados en micro, pequeñas y medianas empresas, concluyendo que el nivel de satisfacción de los usuarios fue alto y que estas herramientas constituyen canales de comunicación viables para negocios de menor escala. De forma complementaria, Marcineková et al. (2025) documentaron la implementación de un chatbot con IA en una microempresa de comercio electrónico, reportando una reducción del 45.90 % en el tiempo promedio de respuesta y un incremento del 14.50 % en la satisfacción del cliente.

Finalmente, a nivel local, Maldonado Zuñiga et al. (2025) investigan el impacto de la inteligencia artificial en la optimización de los procesos de toma de decisiones en organizaciones empresariales de Manabí, Ecuador, concluyendo que la IA ha emergido como una herramienta clave para la transformación organizacional.

Planteamiento del problema

En el entorno comercial actual, la capacidad de responder con rapidez y eficiencia a través de canales digitales se ha convertido en un factor crítico de competitividad y satisfacción del cliente, especialmente para las pequeñas empresas. Ecuador refleja esta tendencia: según Kemp (2025), el país registra 15.20 millones de usuarios de internet (83.70 % de penetración), 13.50 millones de usuarios activos en redes sociales y 18.00 millones de conexiones móviles. El comercio digital también se encuentra en expansión; la séptima medición del estudio de comercio electrónico reportó un volumen de USD 5 500 millones en transacciones durante 2024, con un crecimiento del 22 % respecto al año anterior (Cámara Ecuatoriana de Comercio Electrónico & Universidad de Especialidades Espíritu Santo, 2025). En este contexto de alta conectividad, los consumidores esperan respuestas inmediatas, lo que incrementa la presión sobre negocios con recursos humanos limitados. Se observa, por tanto, una contradicción entre la demanda de atención inmediata a través de canales como WhatsApp y la realidad operativa de negocios donde un mismo personal debe atender múltiples frentes de trabajo, tanto digitales como presenciales.

Este desfase se manifiesta de manera concreta en la ciudad de Manta, donde negocios locales como la florería Aura experimentan consecuencias directas en su servicio. La dependencia de una única persona para gestionar la comunicación por WhatsApp, mientras simultáneamente atiende clientes en el local, coordina con proveedores y prepara pedidos, genera un cuello de botella operativo. Esta situación no solo produce tiempos de espera excesivos para el cliente, sino que también resulta en una experiencia de servicio inconsistente y en la potencial pérdida de ventas frente a competidores que puedan ofrecer una interacción más dinámica. El problema, por lo tanto, trasciende la simple demora; refleja una necesidad no atendida de optimizar procesos de comunicación mediante soluciones tecnológicas accesibles que puedan integrarse de manera práctica en el flujo de trabajo de una empresa.

Formulación del problema

Florería Aura en Manta presenta tiempos de respuesta excesivamente altos en su canal principal de atención al cliente, WhatsApp, con un promedio de **13.58 minutos** Apéndice A. Esta demora se debe a que la atención recae en una sola persona, quien simultáneamente desempeña otras labores operativas, generando una mala experiencia del cliente y pérdida de oportunidades de venta.

Pregunta general

¿De qué manera un sistema automático de asistencia que utiliza un chatbot puede reducir el tiempo de respuesta al cliente en el canal digital “WhatsApp”?

Preguntas específicas

1. ¿Cuál es el tiempo actual promedio de respuesta al cliente en el canal WhatsApp de la florería Aura?
2. ¿Qué estructura conversacional es aplicable para que el chatbot atienda consultas iniciales y solicitudes de productos en el canal WhatsApp de la florería Aura?
3. ¿Cómo puede implementarse un sistema automático de asistencia integrado con la API de WhatsApp, para ejecutar la atención al cliente en la florería Aura?
4. ¿En qué medida se reduce el tiempo de respuesta al cliente?

Objetivos de la investigación

Objetivo general

Desarrollar un sistema automático de asistencia que utiliza un chatbot, para reducir el tiempo de respuesta al cliente en el canal digital "WhatsApp" en la florería Aura de la ciudad de Manta.

Objetivos específicos

1. Analizar el tiempo actual de respuesta al cliente en el canal WhatsApp de la florería Aura.
2. Diseñar la estructura conversacional del chatbot para atender consultas iniciales y solicitudes de productos en WhatsApp.
3. Implementar el sistema automático de asistencia integrándolo con la API de WhatsApp.
4. Evaluar la reducción del tiempo de respuesta al cliente.

Justificación

Este estudio se justifica al abordar una brecha específica identificada en la literatura: la falta de investigaciones sobre la aplicación de IA para automatizar procesos en negocios locales de pequeña escala, particularmente en la gestión de pedidos por chat. Mientras los estudios existentes se centran en la digitalización general (Verhoef et al., 2021), este proyecto propone una solución concreta (un chatbot) a un problema operativo real. Su desarrollo no solo mejorará la competitividad de la florería Aura, sino que servirá como un caso de estudio aplicable para otras empresas en contextos similares, contribuyendo con conocimiento práctico a un área de investigación poco explorada.

Contribución científica

La contribución científica de este trabajo se delimita en tres aportes principales:

1. **Evidencia empírica en microescala local:** aporta resultados cuantitativos de una implementación real de automatización conversacional en una microempresa, contexto con menor cobertura en la literatura revisada.
2. **Esquema técnico-metodológico replicable:** integra un diseño cuasi-experimental pretest-postest con un solo grupo y una arquitectura aplicada (Chatwoot, n8n, LLM, Redis y PostgreSQL), útil como referencia para estudios de casos similares.
3. **Estandarización de medición de latencia:** documenta criterios operativos de captura, depuración y comparación del tiempo de primera respuesta, incluyendo normalización de unidades para el análisis porcentual.

Alcance

El presente proyecto tiene como alcance el diseño, desarrollo e implementación de un sistema automático de asistencia que utiliza un chatbot, integrado al canal digital WhatsApp de la florería, con el propósito principal de reducir significativamente los tiempos de respuesta al cliente durante la atención.

El sistema automatizará la interacción básica con los clientes, brindando respuestas inmediatas y precisas a consultas frecuentes relacionadas con productos disponibles, horarios de atención, ubicación del local. Además, permitirá al usuario interactuar un catálogo digital de flores y arreglos, catálogo gestionado a través de una plataforma intermedia que facilite su actualización por parte de la administradora del negocio sin requerir asistencia técnica.

El proyecto abarca desde el análisis de los procesos actuales de atención al cliente, la definición de la estructura conversacional del Chatbot, hasta la implementación del flujo automatizado en la

herramienta N8N y la medición comparativa del tiempo de respuesta entre una cohorte histórica pre-implementación, construida con registros elegibles del historial operativo, y una cohorte post-implementación correspondiente a un piloto de 7 días.

El estudio no incluye el desarrollo de un sistema de comercio electrónico completo ni la integración con pasarelas de pago o sistemas contables, dado que su objetivo específico es reducir los tiempos de respuesta al cliente en el canal digital WhatsApp. Con ello, se espera mejorar la eficiencia comunicacional de la florería, garantizar una atención inmediata y uniforme, y establecer una base tecnológica escalable para futuras ampliaciones del sistema.

Marco conceptual

Para la correcta interpretación de la arquitectura propuesta, se definen conceptos técnicos fundamentales que estructuran la solución:

Microservicios: arquitectura de servicios autónomos e independientes interconectados vía APIs ligeras. Permite escalabilidad modular y despliegue continuo en entornos de contenedores (Dragoni et al., 2017).

Orquestación de flujos (*Workflow orchestration*): coordinación automatizada de tareas complejas entre sistemas heterogéneos. Actúa como integrador lógico entre la IA y la base de datos transaccional (Pautasso, 2017).

RAG *(Retrieval-Augmented Generation)*: optimización de LLMs mediante la consulta de bases de conocimiento externas antes de la inferencia. Mitiga alucinaciones y contextualiza las respuestas al dominio del negocio (Lewis et al., 2020).

API *gateway*: punto de entrada unificado para tráfico externo que gestiona enrutamiento a microservicios, seguridad SSL y balanceo de carga, ocultando la topología interna (Richardson, 2018).

***Prompt engineering*:** diseño estratégico de instrucciones para guiar a modelos generativos (LLM), asegurando coherencia lógica y cumplimiento de restricciones de formato (White et al., 2023).

***Webhook*:** comunicación asíncrona basada en eventos HTTP que elimina el sondeo constante (*polling*). Fundamental para la reactividad en tiempo real entre mensajería y servidores (Chavan, 2021).

n8n (Orquestador de flujos): motor de automatización de código fuente disponible que permite modelar, ejecutar y supervisar procesos de negocio complejos mediante una interfaz gráfica basada en nodos. Actúa como un *middleware* de integración que conecta servicios heterogéneos y APIs, facilitando la interoperabilidad técnica y el procesamiento de datos en tiempo real sin las limitaciones de las plataformas cerradas (Pautasso, 2017).

Contenedores (Docker): virtualización ligera que empaqueta aplicación y dependencias en una unidad estandarizada. Garantiza la portabilidad y consistencia entre entornos de desarrollo y producción (Merkel, 2014).

CRM Social (Chatwoot): plataformas omnicanal que centralizan interacciones digitales. Facilitan la supervisión humana y proporcionan métricas de rendimiento en el ciclo de vida del cliente (Trainor et al., 2014).

Base de datos en memoria (Redis): almacenamiento NoSQL en RAM de latencia de microsegundos. Crítico para gestionar el estado conversacional (memoria a corto plazo) y colas de alta concurrencia (Cattell, 2011).

Procesamiento de lenguaje natural (PLN): rama de la IA que aplica redes neuronales profundas para la comprensión computacional del lenguaje humano y la extracción de intenciones estructuradas (Hirschberg & Manning, 2015).

Tokenización: fragmentación de texto en unidades semánticas (*tokens*) para su procesamiento

vectorial. Determina la eficiencia computacional y la ventana de contexto del modelo (Sennrich et al., 2016).

Método

Diseño de la investigación

Enfoque de investigación

La presente investigación adopta un **enfoque cuantitativo**. Esta perspectiva metodológica se seleccionó estratégicamente para garantizar la objetividad en la valoración del impacto del sistema, orientándose a los siguientes fines:

- **Medición de variables críticas:** para cuantificar con precisión indicadores numéricos clave, específicamente el *tiempo promedio de respuesta* y el *porcentaje de reducción* de dicho tiempo.
- **Comparabilidad de resultados:** para ejecutar un análisis comparativo entre el estado inicial (gestión manual) y el estado final (gestión asistida por IA).
- **Evaluación de eficiencia:** para determinar el éxito del proyecto basándose en métricas replicables y no en percepciones subjetivas de calidad.

Tipo de investigación

El estudio se define bajo una modalidad dual **Descriptiva – Aplicada**, integrando ambas tipologías para abarcar desde el diagnóstico hasta la solución tecnológica:

- **Investigación descriptiva:** se emplea en la fase inicial para caracterizar el proceso actual de atención al cliente en el canal de WhatsApp. Su propósito es establecer una línea base del tiempo de respuesta previo a la intervención tecnológica.

- **Investigación aplicada:** utilizada para resolver una problemática concreta mediante la ingeniería. Se centra en el diseño e implementación del sistema de chatbot, con el objetivo directo de reducir los tiempos de respuesta al cliente en el canal digital WhatsApp.

Variables de estudio y operacionalización

Con base en el objetivo de evaluar la latencia de atención, se definieron las siguientes variables operativas:

Tabla 1
Operacionalización de variables del estudio

Variable	Definición operacional	Indicador	Escala / unidad
Tipo de atención (independiente operativa)	Condición del proceso de respuesta en cada cohorte: atención manual (pre) y atención automatizada (post).	Clasificación por fase de estudio.	Nominal dicotómica
Tiempo de primera respuesta (dependiente)	Diferencia entre la marca temporal del primer mensaje entrante y la primera respuesta emitida en la conversación.	Media, mínimo, máximo y desviación estándar.	Tiempo continuo (segundos para comparación)
Reducción porcentual de latencia (derivada)	Cambio relativo del tiempo de primera respuesta entre la fase manual y la fase automatizada.	$\left(\frac{T_{manual} - T_{automatizado}}{T_{manual}}\right) \times 100$	Porcentaje (%)

Población y Muestra

Población

Está conformada por la totalidad de los clientes, actuales y potenciales, que inician un proceso de comunicación con la florería Aura a través del canal digital WhatsApp Business (Meta Platforms, Inc., 2024). Esta clasificación responde a la naturaleza dinámica del canal, donde el flujo de usuarios es continuo e indeterminado en el tiempo.

Muestra

Se aplicó un “Muestreo no probabilístico por conveniencia” (Makwana et al., 2023). Esta decisión se adoptó porque la población atendida por WhatsApp es dinámica y no existe un marco muestral completo de clientes. Además, el estudio se ejecutó en condiciones reales de operación, sin manipulación individual de los participantes. Desde el punto de vista estadístico, esta elección es coherente con un diseño cuasi-experimental de tipo pretest-postest con un solo grupo y con un objetivo descriptivo-comparativo aplicado al caso. Por ello, no se estimó tamaño muestral probabilístico ni error muestral poblacional, y los hallazgos se interpretan con alcance contextual (Creswell & Creswell, 2023).

La muestra se constituyó con interacciones registradas en el historial de mensajes, organizadas en dos fases comparativas:

- **Fase pre-implementación (línea base):** interacciones históricas representativas del proceso de atención manual, seleccionadas por conveniencia con base en criterios de inclusión definidos.
- **Fase post-implementación (piloto):** interacciones capturadas durante la semana posterior (7 días) al despliegue del sistema automático.

En consecuencia, la comparación se realizó entre una línea base histórica y un piloto de 7 días, no entre dos periodos temporales equivalentes. En consistencia con este diseño, la comparabilidad se abordó en términos descriptivos mediante la misma definición del evento de respuesta, criterios homogéneos de inclusión/exclusión y una depuración común de registros; no se aplicaron pruebas inferenciales de homogeneidad poblacional. De igual manera, no se emplearon pruebas de significancia estadística (e.g., t de Student, Wilcoxon), dado que los supuestos que requieren dichas pruebas —muestreo probabilístico, equivalencia temporal e instrumental entre fases— no se satisfacen en este diseño; en su lugar, el análisis se circunscribió a estadística descriptiva, nivel coherente con un estudio de caso único con muestreo por conveniencia.

Criterios de inclusión

Se consideraron válidas únicamente aquellas interacciones que aportaran valor comercial real:

- **Naturaleza de la consulta:** conversaciones correspondientes a consultas reales de clientes interesados en productos, precios, horarios de atención, logística de envíos o disponibilidad de stock.
- **Integridad del dato:** registros que contaran con la marca temporal completa (fecha y hora) tanto del mensaje entrante como de la respuesta saliente.

Criterios de exclusión

Se descartaron del análisis los registros que pudieran introducir sesgo o ruido estadístico:

- **Datos de control:** mensajes internos, pruebas técnicas del desarrollador o interacciones administrativas.
- **Redundancia:** interacciones duplicadas generadas por errores de conexión o reintentos del usuario.
- **Datos incompletos:** conversaciones que carecían del registro exacto del primer evento de respuesta, ya que la ausencia de este dato impide matemáticamente el cálculo confiable del tiempo de respuesta.

Metodología

El estudio se enmarca en un diseño cuasi-experimental de tipo pretest-postest con un solo grupo. Esta configuración se seleccionó para evaluar el cambio operativo en condiciones reales de producción, fundamentándose en los siguientes criterios metodológicos (Creswell & Creswell, 2023):

Diseño cuasi-experimental

Este diseño incorpora una intervención tecnológica real en el proceso de atención, manteniendo el análisis en ambiente natural y sin aleatorización de participantes. La aplicación práctica en la florería Aura se operacionaliza de la siguiente manera:

- **Intervención en ambiente productivo:** se desplegó el sistema automatizado en el canal oficial de WhatsApp Business de la empresa, evitando entornos simulados que no reflejen la carga real de trabajo.
- **Ausencia de grupo control paralelo y aleatorización:** la evaluación se realizó sobre un único caso de estudio, comparando su desempeño antes y después de la implementación.

Esquema pretest-postest de un solo grupo

La evaluación comparó dos mediciones temporales del mismo caso: una cohorte histórica pre-implementación (pretest) y una cohorte piloto post-implementación de 7 días (postest). Al no existir grupo control, los hallazgos se interpretan como evidencia de mejora operativa en el contexto estudiado, sin atribución causal fuerte generalizable.

Técnicas e instrumentos de recolección de datos

Técnicas de recolección

Se emplearon técnicas de observación directa y la revisión documental de registros (Logs). Esta combinación nos permitió analizar los tiempos actuales de respuesta sin alterar el entorno natural de la interacción (Torrentira, 2020).

- **Observación directa:** utilizada para examinar la gestión de las consultas en tiempo real, identificando la dinámica operativa de recepción y respuesta, capturando las marcas temporales exactas del mensaje entrante.

- **Revisión documental (análisis de historiales):** consistió en la extracción y sistematización de los datos almacenados en el historial de *WhatsApp Business*, con el objetivo de construir la fuente primaria de información para la línea base histórica (pre-implementación).

Instrumentos de investigación

Fase pre-implementación

Se utilizó una tabla de registro estructurada, elaborada en hoja de cálculo. Este instrumento sirvió para depurar los historiales de chat manuales, consignando los siguientes campos críticos:

- Fecha de interacción.
- Hora exacta de recepción del mensaje (cliente).
- Hora exacta de emisión de la primera respuesta (humano).
- Variable calculada: *tiempo de espera*.

Fase post-implementación

Se empleó el registro de ejecuciones del sistema (system logs) (National Institute of Standards and Technology (NIST), 2023) extraído directamente de la plataforma de orquestación n8n. La selección de este instrumento técnico se fundamenta en la capacidad de los logs del servidor para ofrecer una precisión de milisegundos sobre el tiempo de procesamiento del flujo (*workflow execution time*). Posteriormente, estos datos crudos fueron homologados y estructurados en la misma matriz de hoja de cálculo utilizada en la fase previa, garantizando así la consistencia estructural necesaria para el análisis comparativo (Arias-Gonzáles, 2020).

Técnicas de análisis de datos

El procesamiento de la información se realizó mediante **estadística descriptiva**, orientada a contrastar el desempeño operativo entre ambos métodos de atención.

- **Indicadores de tendencia central y dispersión:** se calcularon la media aritmética, el valor máximo/mínimo y la desviación estándar (σ). Con el propósito de determinar la estabilidad y consistencia del sistema propuesto frente a la variabilidad humana.
- **Análisis de variación porcentual:** se aplicó el cálculo de la tasa de reducción de latencia con la finalidad de cuantificar objetivamente el impacto de la solución, traduciendo la diferencia de tiempos en un porcentaje de mejora de eficiencia (eficiencia técnica) (Medina et al., 2023).

Consideraciones éticas

La presente investigación respetó los principios éticos fundamentales relacionados con el manejo responsable de la información y la interacción con los usuarios del negocio. Para ello, se contó con el consentimiento previo y explícito de la propietaria de la florería Aura para realizar el levantamiento de información, medir los tiempos de respuesta y analizar los procesos de atención a través del canal WhatsApp Business.

Asimismo, se garantizó la confidencialidad de los datos de los clientes, evitando el registro o divulgación de nombres, números telefónicos, conversaciones completas u otros datos sensibles. La información recopilada fue tratada de manera anónima y utilizada exclusivamente con fines académicos.

En este marco, no se requirió consentimiento informado individual de clientes, debido a que el análisis se realizó sobre registros operativos secundarios anonimizados, sin contacto directo ni intervención sobre personas. Los resultados se reportaron de forma agregada, minimizando cualquier riesgo de identificación.

De igual manera, el diseño e implementación del prototipo basado en n8n y WhatsApp Business respetó las políticas de privacidad y las buenas prácticas de seguridad, asegurando que no se afecte la integridad funcional del negocio ni la experiencia de los clientes durante la ejecución de las pruebas. El análisis de los tiempos de respuesta se realizó sin manipulación individual de los clientes ni alteración artificial del entorno operativo, manteniendo un proceso de evaluación no intrusivo.

Procedimiento

Diseño de la solución

Levantamiento de requerimientos

A partir del diagnóstico inicial y el análisis de brechas operativo, se formalizó la especificación de requisitos del sistema. Este proceso de elicitación clasificó las necesidades en tres dimensiones: funcionales (capacidades del sistema), no funcionales (atributos de calidad) y restricciones del proyecto.

Para consultar la matriz detallada de trazabilidad y los criterios de aceptación, remitirse al Apéndice H.

Principios de diseño adoptados

Para el diseño de la solución se evaluaron y adoptaron los siguientes principios de ingeniería de software:

- Modularidad, permitiendo que cada componente cumpla una función específica.
- Separación de responsabilidades, evitando acoplamientos innecesarios.
- Escalabilidad, para permitir futuras ampliaciones.

- Trazabilidad, mediante el uso de logs y registros.
- Uso de tecnologías probadas, con amplia adopción, comunidad activa y soporte.
- Bajo costo operativo, acorde al contexto del negocio.

Estos principios guiaron todas las decisiones arquitectónicas y tecnológicas.

Diseño del Flujo Conversacional

El diseño de la interacción se modeló mediante un diagrama de flujo lógico que define el comportamiento determinista y probabilístico del agente. Este esquema se traduce directamente en la orquestación de nodos dentro de n8n, estableciendo los puntos de decisión, persistencia de datos y criterios de parada. Para visualizar el diagrama completo del flujo conversacional, los estados del sistema y la lógica de control, remitirse a las Figuras 2 y 3.

El flujo operativo se estructura en cuatro fases críticas: Ingesta y clasificación, Transaccional, Lógica de aprobación y cierre. Para consultar la descripción técnica detallada de cada etapa, los subprocesos de NLP y las sentencias SQL ejecutadas en cada hito, remitirse al Apéndice B.

El ciclo de vida de un pedido se modeló como una máquina de estados finita con cuatro valores posibles, implementados mediante restricciones `CHECK` en PostgreSQL. La Tabla 2 formaliza los estados, sus transiciones y el evento que las desencadena.

Tabla 2
Tabla de estados del pedido y transiciones permitidas

Estado	Descripción	Transición a	Evento
`validando_pedido`	Estado inicial al registrar el pedido en la base de datos	`aprobado`, `rechazado`, `cancelado`	INSERT desde n8n
`aprobado`	La administradora confirma viabilidad del pedido	— (estado final)	Decisión manual vía Chatwoot
`rechazado`	La administradora determina que no es posible cumplir el pedido	— (estado final)	Decisión manual vía Chatwoot
`cancelado`	El pedido es anulado por el cliente o por el sistema	— (estado final)	Solicitud del cliente o *timeout*

Nota. Todos los estados terminales desactivan el bot (`bot_state: off`) y asignan una etiqueta visual en Chatwoot para trazabilidad administrativa.

Diseño de la arquitectura general del sistema

Implementamos una arquitectura de **nodos distribuidos autónomos**. Este diseño separa físicamente la capa de interacción humana (Chatwoot) de la capa de procesamiento lógico (n8n) en dos servidores independientes. Este desacoplamiento garantiza que la carga computacional de la automatización no degrade la latencia de la comunicación en tiempo real, alineándose con los principios de independencia de despliegue (Newman, 2021).

La infraestructura se organiza en tres zonas lógicas diferenciadas:

1. **Zona 1 (VPS A):** nodo dedicado exclusivamente a la comunicación y atención humana (Frontend Operativo).
2. **Zona 2 (VPS B):** nodo de automatización, orquestación lógica y persistencia de datos (Backend de Procesamiento).
3. **Zona 3 *(SaaS Layer)*:** ecosistema de APIs externas y servicios en la nube.

Este esquema corresponde a una distribución funcional por servicios y no a una topología

tolerante a fallos multi-región; por tanto, no incorpora balanceo activo-activo ni conmutación automática por falla.

Para consultar las especificaciones técnicas detalladas de cada nodo, zonas lógicas, las políticas de seguridad (Firewalls), la configuración de la red interna Docker y la integración de servicios SaaS, remitirse al Apéndice C.

Diseño e implementación del esquema de base de datos

El diseño del esquema se orientó a satisfacer los requisitos del flujo híbrido ("Human-in-the-loop"), priorizando la integridad de los datos transaccionales sobre la normalización excesiva en campos de contenido variable (Reis & Housley, 2022).

Para visualizar el Diagrama Entidad-Relación (DER) completo y consultar la definición técnica detallada de los atributos, las restricciones de integridad y la lógica de optimización, remitirse al Apéndice D.

Diseño y lógica de estado en Chatwoot

Para habilitar la colaboración efectiva en el modelo "Human-in-the-loop", fue necesario extender el modelo de datos predeterminado de Chatwoot. Se configuró una taxonomía personalizada basada en dos mecanismos complementarios:

1. **Etiquetas (*Labels*):** utilizadas para la gestión visual en la bandeja de entrada y como disparadores de eventos (*triggers*) hacia el orquestador.
2. **Atributos personalizados (*Custom Attributes*):** variables estructuradas que actúan como compuertas lógicas para el control del flujo conversacional.

Con esta configuración, Chatwoot deja de ser un simple receptor de mensajes para convertirse en un panel de control (*dashboard*) capaz de gobernar la lógica del negocio. Para consultar el

diccionario de datos completo, la definición técnica de los interruptores lógicos (*kill switch*) y la estructura de los estados del pedido, remitirse al Apéndice E.

Diseño del flujo de orquestación en n8n (principal)

Se estructuró una arquitectura de procesamiento por etapas que incluye gestión de concurrencia, normalización de datos, enrutamiento cognitivo y ejecución transaccional.

Debido a la complejidad lógica y la extensión de los nodos de control, la especificación técnica detallada de cada fase, incluyendo los diagramas de flujo, la lógica de los "Webhooks", la gestión de colas en Redis y las ramas de decisión del Çerebro Central", se encuentra documentada en su totalidad en el Apéndice F.

El flujo se compone de las siguientes etapas operativas:

- **Fase 1:** recepción y filtrado de eventos (Sanitización).
- **Fase 2:** gestión de concurrencia y reconstrucción de contexto *(Buffer)*.
- **Fase 3:** normalización y control de estado (Persistencia).
- **Fase 4:** cerebro central y enrutamiento de intenciones *(AI Brain)*.
- **Punto de Bifurcación:** lógica de intervención manual y *(Kill Switch)*.

Diseño del flujo de orquestación en n8n secundario

Su función es escuchar los cambios administrativos realizados por el operador humano en la interfaz de Chatwoot y propagarlos automáticamente a la base de datos y al cliente, garantizando la consistencia transaccional.

Dada la importancia de la sincronización de estados en tiempo real, la especificación técnica de los disparadores (Webhooks), la prevención de bucles infinitos y la lógica de generación de respuestas de aprobación/rechazo se detalla en el Apéndice G.

El proceso se divide operativamente en cinco fases:

- **Fase 1:** escucha de eventos y prevención de bucles.
- **Fase 2:** reconstrucción del contexto (Enriquecimiento de datos).
- **Fase 3:** enrutador de máquina de estados.
- **Fase 4:** ejecución de aprobación (Transacción y notificación).
- **Fase 5:** ejecución de rechazo o pausa (Manejo diplomático).

Desarrollo

Gestión de dominios y configuración DNS

Para proporcionarle al sistema una dirección accesible públicamente y facilitar la implementación de certificados de seguridad SSL/TLS, se gestionó la capa de nombres de dominio. Se utilizó Hostinger como proveedor de servicios DNS autoritativos.

Durante la fase de desarrollo y despliegue, ya se contaba con un dominio previamente adquirido, ecosolutions.cfd, el cual fue utilizado exclusivamente con fines técnicos y operativos, permitiendo simplificar el proceso de despliegue, configuración del entorno productivo y validación de certificados de seguridad. La elección de este dominio no responde a criterios comerciales ni definitivos, sino a la disponibilidad inmediata del mismo durante el desarrollo del sistema.

Se contempla que, en una etapa futura, cuando la propietaria de la florería Aura se encuentre en condiciones de adquirir un dominio propio alineado con la identidad del negocio, el sistema será migrado sin inconvenientes al nuevo dominio. Cabe destacar que la aplicación es independiente del nombre de dominio, por lo que su funcionamiento no se ve afectado y puede operar correctamente bajo cualquier dominio válido.

Definición de subdominios y enrutamiento

De acuerdo con el diseño de la arquitectura, se configuraron registros de recursos (RR) específicos en la zona DNS para segregar los puntos de entrada de la aplicación. Cada subdominio apunta a la dirección IP pública del VPS correspondiente (chat → VPS A y n8n → VPS B), delegando el enrutamiento final al proxy inverso de cada nodo:

- **Subdominio de chat (chat.ecosolutions.cfd):** registro tipo A configurado para direccionar el tráfico hacia la interfaz de gestión de conversaciones (Chatwoot).
- **Subdominio de automatización (n8n.ecosolutions.cfd):** registro tipo A configurado para el orquestador de flujos de trabajo (n8n).

Esta estrategia de separación por subdominio e IP simplifica la gestión de la red externa y preserva el desacoplamiento entre nodos. No obstante, la disponibilidad externa de cada servicio sigue dependiendo de su IP pública y del DNS autoritativo (Newman, 2021).

Aprovisionamiento de infraestructura en Hetzner

Para garantizar la alta disponibilidad y el aislamiento de recursos, la infraestructura se desacopló en dos nodos de cómputo independientes. El entorno en los servidores se instanció utilizando una imagen de **Ubuntu Server 24.04 LTS**. Se optó por una configuración de **docker** desde la terminal para garantizar una actualización Integral y un despliegue automatizado. El procedimiento técnico ejecutado se detalla a continuación:

1. **Aprovisionamiento de recursos (Hetzner console):** desde el panel de control del proveedor de nube, se inicializaron dos instancias(Una para cada servicio) (Droplet/VPS) usando la capa económica **cost optimized**, una arquitectura **ARM64** seleccionando la ubicación geográfica más cercana al público objetivo para minimizar la latencia, en este caso en concreto usamos la recomendada que fue **Helsinki**. Se asignó una dirección IPv4

pública estática, se agrego la imagen iso previamente comentada y se inyectaron las llaves SSH públicas del administrador para el acceso sin contraseña (Tevault, 2023).

2. **Gestión de identidad y endurecimiento del protocolo SSH:** para mitigar los riesgos asociados al uso de la cuenta de superusuario, se implementó una política de **defensa en profundidad** configurando el acceso exclusivo mediante privilegios delegados. El procedimiento consistió en:

    - **Creación de usuario operativo:** se generó un nuevo usuario en el sistema (ej. `admin`) y se le concedieron privilegios administrativos controlados añadiéndolo al grupo `sudo`. Esto garantiza la trazabilidad de las acciones administrativas.
    - **Instalación de credenciales criptográficas:** se transfirieron las llaves públicas (SSH Public Keys) desde el entorno local hacia el directorio `.ssh/authorized_keys` del nuevo usuario, verificando los permisos de lectura/escritura (600) para asegurar la autenticación sin contraseña.
    - **Blindaje del daemon SSH:** se editó el archivo de configuración del servicio (`/etc/ssh/sshd_config`) para aplicar las siguientes restricciones de seguridad:

        ```
        PermitRootLogin no
        PasswordAuthentication no
        ```

    Tras reiniciar el servicio SSH, el servidor quedó protegido contra ataques de fuerza bruta (IBM, 2024a) basados en diccionarios de contraseñas, permitiendo el acceso únicamente al personal autorizado que posea la llave privada correspondiente.

3. **Actualización integral y despliegue automatizado:** como paso previo a la instalación de software, se ejecutó una actualización completa de los índices de paquetes y las dependencias del sistema operativo para asegurar la compatibilidad de las librerías base:

```
$ sudo apt update && sudo apt upgrade -y
```

Posteriormente, para garantizar una instalación limpia y libre de errores de configuración manual, se utilizó el *script* de instalación oficial proporcionado por Docker Inc. Este método automatizado realiza la validación de firmas GPG, la configuración de repositorios estables y la instalación de los binarios (`docker-ce`, `docker-ce-cli`, `containerd.io`) en una única ejecución controlada mediante el comando `curl`.

4. **Post-instalación y gestión de privilegios:** una vez finalizado el despliegue de los binarios, se ejecutaron las tareas de post-instalación para integrar el servicio con el sistema de arranque y gestionar los permisos de seguridad:

   a) **Activación del servicio:** se habilitó el daemon de Docker para que inicie automáticamente junto con el sistema operativo.

   ```
   $ sudo systemctl enable docker
   $ sudo systemctl start docker
   ```

   b) **Delegación de privilegios:** para evitar el uso continuo del usuario root, se añadió al usuario operativo (`admin`) al grupo de seguridad `docker`. Esto permite la gestión de contenedores con los permisos necesarios sin comprometer la seguridad global del sistema mediante el uso excesivo de `sudo`.

   ```
   $ sudo usermod -aG docker ${USER}
   ```

Seguridad perimetral (cloud firewall)

Para proteger la integridad del servidor, se implementó un *firewall* (National Institute of Standards and Technology (NIST), 2026) en la nube externo al sistema operativo desde la misma

consola principal de Hetzner. Siguiendo el principio de defensa en profundidad, se establecieron reglas estrictas de entrada que rechazan todo tráfico no autorizado. Solo se permitieron conexiones a través de los puertos explícitamente definidos en el diseño:

- **Puertos 80/443:** para el tráfico web entrante hacia el Proxy (Traefik).
- **Puerto 22:** para la administración remota segura vía SSH.

Implementación de arquitectura distribuida

Cada servidor VPS aloja servicios específicos, comunicándose entre sí a través de APIs seguras sobre HTTPS (Madden, 2020). A continuación, se detalla la configuración técnica realizada en cada entorno.

Nodo 1: servidor de automatización (n8n-server)

Este servidor está dedicado exclusivamente a la orquestación de flujos de trabajo. Su configuración se caracteriza por el uso de una red interna personalizada y la separación de la capa de persistencia.

- **Segmentación de red:** como paso fundamental, se creó manualmente una red tipo *bridge* denominada `red-ecosistema`. Esta red permite la resolución de nombres de dominio interno, facilitando que la aplicación n8n encuentre a su base de datos simplemente llamándola por su nombre de servicio.

```
$ docker network create red-ecosistema
```

- **Capa de infraestructura de datos:** para mantener el orden y facilitar las copias de seguridad, se creó el directorio `data-infrastructure`. Dentro de este, se definió un archivo `docker-compose.yml` exclusivo para levantar los servicios de PostgreSQL y

Redis. Ambos fueron configurados para unirse a la red externa `red-ecosistema`, asegurando que sean accesibles por la aplicación principal pero invisibles desde internet.

- **Orquestación del servicio *core* y vinculación:** para el despliegue de la aplicación principal, se utilizó la imagen oficial `n8nio/n8n` obtenida directamente del registro público (Docker Hub) para garantizar la integridad del software. La configuración se realizó mediante un segundo archivo `docker-compose.yml` en la carpeta n8n, donde se ejecutó la integración final:
  - Se declaró explícitamente el uso de la red externa `red-ecosistema` en la configuración del servicio.
  - Gracias a esta vinculación, el contenedor de n8n pudo ver y conectarse exitosamente a los contenedores de infraestructura (PostgreSQL y Redis) utilizando las credenciales y hosts definidos en el archivo de configuración de estos que es docker compose, consolidando así la arquitectura distribuida en el mismo nodo.

Nodo 2: Servidor de mensajería (chatwoot-server)

El segundo VPS aloja la plataforma Chatwoot. Dado que esta aplicación requiere una arquitectura compleja (Rails, Sidekiq, Webpack), se optó por un despliegue unificado mediante un único archivo `docker-compose` que orquesta todos sus componentes.

- **Configuración de entorno (.env):** la personalización del sistema se realizó mediante la inyección estricta de variables. Se definió la URL base `FRONTEND_URL=https://chat.ecosolutions.cfd` para asegurar el correcto funcionamiento de los callbacks de autenticación y los enlaces generados. Además, también se agregaron las credenciales para Postgres y Redis previamente configuradas en el archivo de orquestación Docker Compose.

- **Persistencia:** se configuraron volúmenes de Docker locales para garantizar que los adjuntos y logs de conversaciones persistan ante reinicios del contenedor.

Integración de servicios de correo (SMTP)

para habilitar las notificaciones transaccionales y evitar que los correos del sistema sean marcados como SPAM, se integró el proveedor de infraestructura **Resend**.

1. **Autenticación de dominio (DNS):** en el panel de Hostinger, se configuraron los protocolos de seguridad para el dominio `ecosolutions.cfd` (Hughes, 2023):
    - **SPF:** autoriza a las IPs de Resend a enviar correos.
    - **DKIM:** firma digitalmente cada correo saliente para garantizar su integridad.
2. **Inyección de credenciales:** una vez verificados los DNS, se configuraron las variables SMTP en el archivo `.env` del servidor Chatwoot:

```
MAILER_SENDER_EMAIL=Soporte Aura <noreply@ecosolutions.cfd>
SMTP_ADDRESS=smtp.resend.com
SMTP_PORT=587
```

Con esta configuración, el sistema quedó plenamente operativo con capacidad de entrega certificada de correos electrónicos.

Ejecución del despliegue y puesta en producción

Una vez finalizada la parametrización de los archivos `docker-compose.yml` y la definición de las variables de entorno en ambos nodos de infraestructura, se procedió a la instanciación final de los servicios. En cada servidor (n8n-server y chatwoot-server), se ejecutó el comando de orquestación:

```
$ docker compose up --build -d
```

La ejecución de este comando consolidó el despliegue mediante dos acciones críticas:

1. La bandera *-build* forzó la compilación de las imágenes asegurando que cualquier cambio en la configuración fuera incorporado.
2. El modo `-d` *(detached)* permitió que los contenedores se ejecutaran en segundo plano, garantizando la persistencia del servicio tras cerrar la sesión SSH.

Como resultado, la arquitectura distribuida quedó plenamente operativa en el entorno de producción, respondiendo exitosamente a las peticiones seguras (HTTPS) a través de los subdominios `n8n.ecosolutions.cfd` y `chat.ecosolutions.cfd`.

Configuración de Meta WhatsApp Business API

La implementación del canal de mensajería instantánea requirió el despliegue de una infraestructura de comunicación basada en la *Cloud API* de WhatsApp. Este proceso no se limita a la integración de código, sino que exige una rigurosa configuración de la identidad digital corporativa y los permisos de acceso (Singh et al., 2023). A continuación, se detalla el flujo de trabajo técnico y administrativo ejecutado en la plataforma de Meta:

Aprovisionamiento del entorno comercial

Antes de acceder a los recursos de desarrollo, es necesario establecer una entidad comercial validada dentro del ecosistema de Meta. Se siguieron los siguientes pasos preliminares:

1. **Gestión de identidad *(Meta account)*:** se procedió a la creación y verificación de una cuenta de desarrollador en *Meta for Developers*. Este paso es fundamental para actuar como el administrador principal de los activos digitales del proyecto.

2. **Creación del portafolio comercial *(business portfolio)*:** se instanció un "Portafolio Empresarial"(anteriormente conocido como *Business Manager*). Esta estructura jerárquica es necesaria para agrupar las aplicaciones, las líneas de crédito para mensajería y los números de teléfono bajo una única entidad legal o administrativa, garantizando la propiedad centralizada de los datos.

Instanciación y configuración de la aplicación

Una vez establecido el entorno comercial, se ejecutó la configuración técnica de la aplicación que actúa como *middleware* entre los clientes y el sistema propuesto:

1. **Registro de la aplicación *(app type)*:** dentro del panel de desarrolladores, se creó una nueva aplicación seleccionando el tipo *Business*. Se vinculó esta aplicación explícitamente al Portafolio Comercial creado en la etapa anterior para heredar los permisos de facturación y gestión.
2. **Habilitación del producto WhatsApp:** se añadió el producto "WhatsApp.ª la aplicación. Esto aprovisionó automáticamente los *endpoints* necesarios en la *API Graph* de Meta y generó las credenciales de prueba (*test credentials*).
3. **Vinculación del número e identificadores:** accediendo a la sección de Inicio rápido dentro del apartado de casos de uso de la aplicación, se procedió a configurar un número de teléfono real para el entorno de producción. Este proceso implicó la verificación mediante un código OTP (One-Time Password) y permitió la obtención de dos credenciales críticas para la integración:
    - **Phone number ID:** identificador único del número emisor, utilizado en las cabeceras de las peticiones HTTP para enviar mensajes.
    - **WhatsApp business account ID:** identificador de la cuenta de negocio, necesario para la gestión administrativa y la facturación de la API.

4. **Configuración de facturación y validación de protocolo:** se asoció un método de pago válido al portafolio comercial, cumpliendo con los protocolos de verificación de identidad exigidos por Meta para abandonar el entorno de pruebas (Sandbox). Es importante destacar que esta acción es un requisito administrativo para desbloquear el acceso a producción y aumentar los límites de mensajería; sin embargo, la operación del sistema se mantiene dentro de la capa gratuita (*free tier*) de 1,000 conversaciones de servicio mensuales, por lo que la inclusión de los datos financieros no implicó costos operativos durante la implementación del proyecto.

Gestión de seguridad y webhooks

Para garantizar una comunicación segura y continua (24/7), se implementaron mecanismos de autenticación, gestión de credenciales y arquitectura orientada a eventos:

1. **Usuario del sistema (system user):** en lugar de utilizar credenciales personales temporales, se creó un *system user* en la configuración del negocio. A este usuario autómata se le asignaron roles de admin sobre la aplicación de WhatsApp.
2. **Generación de token permanente:** a través del *system user*, se generó un *Access Token* sin fecha de caducidad. Durante la configuración de este token, se asignaron explícitamente los tres permisos *(scopes)* críticos para la operatividad completa de la API con servicios externos como Chatwoot y n8n:
    - `business_management`: para la lectura de información de la cuenta comercial.
    - `whatsapp_business_messaging`: para el envío y recepción de mensajes.
    - `whatsapp_business_management`: para la gestión de plantillas y números.
3. **Análisis de riesgo y política de rotación:** aunque la API permite tokens de larga vigencia, se identificó como riesgo principal la exposición de credenciales por fuga, reutilización no autorizada o sobreasignación de permisos. Para mitigar este riesgo se definió una política

de: (i) almacenamiento en variables de entorno y acceso restringido al servidor, (ii) revisión periódica de *scopes* bajo principio de mínimo privilegio, (iii) rotación preventiva trimestral y rotación inmediata ante incidente, y (iv) revocación de tokens no vigentes en la consola de Meta.

4. **Preparación para el enlace (Handshake):** con la obtención de las credenciales críticas el *Phone Number ID*, el *WhatsApp Business Account ID* y el *Access Token* permanente, se consolidaron los requisitos de seguridad necesarios para establecer la comunicación bidireccional. Esta infraestructura se diseña bajo una arquitectura *Event-Driven* mediante Webhooks (Bellemare, 2020). La validación del *Verify Token* y la suscripción granular a los eventos de mensajería (como `messages`) se detallarán en la sección posterior dedicada a la configuración interna de la plataforma de chat.

Configuración lógica e integración de plataformas

Se procedió a la configuración interna de las aplicaciones para establecer los flujos de comunicación del negocio. A continuación, se detalla el proceso de integración entre el servidor de mensajería Chatwoot y el proveedor de WhatsApp.

Puesta en marcha de la plataforma chatwoot

La inicialización del sistema de atención al cliente se realizó accediendo a la interfaz web segura a través del dominio `https://chat.ecosolutions.cfd`. El proceso de configuración se estructuró en tres fases críticas:

1. **Creación de cuenta y validación SMTP:** como primer paso de seguridad, se registró la cuenta de superadministrador *owner*. Gracias a la integración previa del servicio SMTP (Resend), el sistema envió automáticamente un correo electrónico de confirmación. La recepción exitosa de este correo y la posterior activación de la cuenta mediante el enlace

seguro validaron funcionalmente la capacidad del sistema para enviar notificaciones transaccionales fuera de la red interna.

2. **Integración del canal de whatsApp:** dentro del panel de control, se procedió a crear una nueva "bandeja de entrada"(*Inbox*) seleccionando el proveedor **WhatsApp Cloud API**. En este formulario, se ingresaron las credenciales obtenidas previamente en la consola de Meta para autorizar al servidor a enviar mensajes:

    - **Proveedor de API:** plataforma utilizada para el envío y recepción de mensajes (WhatsApp Business Platform vía Meta).
    - **Nombre de la bandeja de entrada:** etiqueta o nombre asignado al canal/buzón desde el cual se gestionan las conversaciones.
    - **Número de teléfono:** número telefónico registrado, validado en formato E.164 (International Telecommunication Union (ITU-T), 2026) y sin espacios.
    - **ID del número de teléfono *(Phone Number ID)*:** identificador único asociado al número dentro de la plataforma del proveedor.
    - **ID de la cuenta de negocio *(Business Account ID)*:** Identificador de la cuenta empresarial asociada al número y a la configuración del servicio.
    - **Clave de API / *Access Token*:** credencial de acceso utilizada para autenticar las solicitudes; se utilizó un token de larga vigencia sujeto a la política de rotación y revocación definida en la sección de seguridad.

3. **Cierre del ciclo de *webhooks* (*handshake*):** al finalizar la creación de la bandeja, Chatwoot generó automáticamente la **Webhook URL** (ej. `.../webhooks/whatsapp/...`) y un **verify token** único. Con estos datos, se retornó a la consola de *Meta for Developers* para configurar la API. Se ingresaron ambos valores para realizar el *Handshake* y, tras la validación exitosa, se procedió a la suscripción final del evento `messages`. Esta acción completó el circuito: el servidor tiene permiso para enviar (Token) y Meta sabe dónde

entregar las respuestas (Webhook).

Puesta en marcha del orquestador de flujos (n8n)

Paralelamente, se procedió a la inicialización del servidor de automatización accediendo al dominio seguro `https://n8n.ecosolutions.cfd`. Al ser una herramienta de orquestación *Low-Code* (Ludwig et al., 2021), el proceso de configuración se centró en la seguridad de la cuenta propietaria y la integración de credenciales.

a) **Inicialización del usuario propietario:** al acceder por primera vez, se configuró la cuenta de administrador principal **owner account**. Este paso es crítico ya que, por defecto, la instancia de n8n está abierta hasta que se reclama la propiedad. Se establecieron credenciales robustas para prevenir el acceso no autorizado a los flujos de trabajo que manipulan datos sensibles de los clientes.

b) **Generación de credenciales de integración (API Key):** para que n8n pudiera actuar sobre la plataforma de mensajería (leer contactos, enviar respuestas), fue necesario obtener un token de acceso desde Chatwoot.

- **Extracción del token (Chatwoot):** en la interfaz del servidor de mensajería, se navegó específicamente a la sección *configuración de perfil* (profile settings) del usuario administrador. En el apartado de seguridad, se localizó y copió el **access token** personal, una llave única que permite realizar peticiones API en nombre del propietario.
- **Configuración de la credencial (n8n):** dentro de la consola del orquestador, se procedió a crear una nueva credencial de tipo *Header Auth* o *Chatwoot API*. Se pegó el token extraído. **Nota técnica:** Para garantizar la validez de la autenticación en los nodos personalizados, fue requisito indispensable definir el nombre de la variable de la credencial estrictamente como `api_access_token`. Cualquier variación en esta nomenclatura impediría el reconocimiento de los

permisos.

Esta acción autenticó al servidor de automatización, permitiéndole ejecutar acciones como si fuera un agente humano legítimo.

c) **Configuración de la escucha de eventos (Webhook):** para completar la bidireccionalidad, se configuró en Chatwoot un *Webhook* de salida apuntando hacia n8n.

- Se definió la URL de producción del flujo principal en n8n (ej. `https://n8n.ecosolutions.cfd/webhook/entry-point`).
- Se seleccionaron los eventos desencadenantes, específicamente `message_created`, para que cada vez que un cliente escriba en WhatsApp, Chatwoot notifique inmediatamente al orquestador para iniciar el procesamiento inteligente.

Consolidación de la interoperabilidad del sistema

Con la activación de este último punto de enlace, se da por finalizada la fase de integración de infraestructura.

Aprovisionamiento de servicios de datos en google cloud

Se integró la suite de Google Workspace mediante interfaces de programación (APIs). A diferencia de un acceso de usuario estándar, esta integración requirió la configuración de una identidad de máquina segura en la consola de Google Cloud Platform (GCP).

Creación del proyecto y habilitación de librerías

El primer paso consistió en establecer un contenedor lógico para los recursos en la nube:

1. **Inicialización del proyecto:** se creó un nuevo proyecto en GCP denominado **bot-agent**. Este entorno aislado permite gestionar cuotas de uso y facturación de manera

independiente.

2. **Activación de APIs:** accediendo a la Biblioteca de APIs (*API Library*), se habilitaron explícitamente los servicios necesarios para la operación del sistema (Sullivan, 2022):

    - **Google Sheets API:** permite la consulta programática de celdas y rangos, fundamental para leer la lista de precios y stock en tiempo real.
    - **Google Docs API:** facilita la extracción de texto estructurado desde documentos corporativos, utilizado para nutrir el contexto del modelo de lenguaje (RAG).
    - **Google Drive API:** necesaria para resolver las rutas y metadatos de los archivos alojados en la nube.

Gestión de identidad y credenciales (Service Account)

Para evitar la dependencia de una cuenta de correo personal (gmail.com) y garantizar la continuidad operativa, se implementó una **cuenta de servicio** (*Service Account*). Esta es una cuenta especial diseñada para aplicaciones servidor-a-servidor (Google Cloud, 2023):

1. **Generación del agente digital:** en el panel de IAM (Gestión de Identidades), se creó la cuenta de servicio con un identificador único, resultando en una dirección de correo electrónico técnica (ej. `bot-agent@ecosolutions-bot-agent.iam.gserviceaccount.com`). Esta dirección actúa como la "persona digital"del bot dentro del ecosistema de Google.
2. **Emisión de llaves criptográficas (JSON Key):** dado que este robot"no puede iniciar sesión con contraseña en un navegador, se generó una clave privada en formato JSON. Este archivo contiene las credenciales RSA necesarias para firmar digitalmente las peticiones de autorización (OAuth 2.0) que realizará n8n. El archivo fue descargado y almacenado de forma segura, ya que representa la llave maestra de acceso a los datos.

Delegación de acceso a recursos

Se realizó la vinculación directa en la capa de aplicación (Drive):

- Se localizó la hoja de cálculo maestra **Inventario_Aura** y el documento de conocimiento **Documento_Aura**.
- Utilizando la función estándar de "compartir" de Google Drive, se otorgaron permisos de **editor** a la dirección de correo de la Service Account creada en el paso anterior.

Esta estrategia de mínimo privilegio asegura que el bot tenga acceso estrictamente a los archivos compartidos, manteniendo la privacidad del resto de la unidad de Google Drive del administrador.

Implementación técnica del agente principal en n8n

La construcción del flujo se realizó siguiendo el diseño modular establecido, configurando secuencialmente los nodos para el procesamiento del mensaje. A continuación, se detalla la configuración técnica de cada nodo dentro del sistema.

1. Configuración de entrada y normalización

- **Webhook (Trigger):** se configuró con el método `POST` en la ruta `chatwoot-events` para recibir el *payload* JSON de la API.
- **Event Filter (If):** se establecieron condiciones booleanas para filtrar exclusivamente eventos donde `event = message_created` y el tipo de remitente sea `Contact` o `User`.
- **Data Webhook (Set):** se mapearon las variables del JSON anidado a variables de primer nivel (`sender_id`, `message`, `conversation_id`) para su uso global en el flujo.
- **Message Entry (Switch):** se configuró una regla para bifurcar el tráfico: `message_type = incoming` continúa el proceso; `outgoing` activa la sub-rutina de apagado.

2. Implementación del buffer de mensajes (Redis)

Para la gestión de concurrencia se utilizaron credenciales de Redis y operaciones de lista:

- **Store & Count:** se configuró la operación `RPUSH` para añadir mensajes a la lista `chat_{id}` y `INCR` para el contador atómico, ambos con un TTL (tiempo de vida) de 25 segundos.
- **Lógica de espera:** el nodo **Check_First_Message** valida si el contador es igual a 1. De ser positivo, se activa el nodo **Wait** durante 10 segundos.
- **Bucle de extracción:** mediante los nodos **Extract** (LPOP), **Get/Set Message** y **Check_Queue_Is_Empty**, se implementó un bucle *Do-While* que concatena todos los fragmentos de texto en una sola cadena.
- **Client Message (Set):** consolida el mensaje final unificado.

3. Persistencia y control de estado

Se establecieron las conexiones a PostgreSQL y la lógica de seguridad:

- **Client Add or Update:** se configuró una consulta SQL tipo `UPSERT` parametrizada para gestionar la identidad del cliente.
- **Array Message Client (Set):** se creó un arreglo estructurado `[id, message]` para sanitizar los datos antes de la inserción.
- **Add History Client:** ejecuta el `INSERT` en la tabla `chat_history` utilizando los parámetros del paso anterior en los query parameters.
- **Bot State Switch:** evalúa la variable `bot_state`. Si es igual a ‘‘`off`’’, termina el flujo (nodo *End Flow*); si es ‘‘`on`’’, permite el paso al núcleo de IA.

4. Núcleo cognitivo (AI Core)

Configuración de la memoria y el razonamiento:

- **Recuperación de contexto:** se ejecutan en paralelo el nodo **Read History** (últimos 10 mensajes) y **Get Active Order** (consulta de pedidos pendientes).
- **History Code (Code):** *script* de JavaScript que formatea el JSON del historial y los datos del pedido en una cadena de texto legible para el LLM.
- **AI Brain (OpenAI):** se instanció el modelo `GPT-4o` con una temperatura de 0. En el *System Prompt* se definieron las reglas estrictas de clasificación (`INFO`, `WANT_BUY`, etc.) y se configuró la salida en formato JSON.
- **Intent Router (Switch):** se crearon cuatro salidas basadas en la propiedad `user_intent` del JSON generado por la IA.

5. Configuración de ramas modulares (Branching Logic)

El enrutador de intenciones (*Intent Router*) distribuye el flujo hacia cuatro caminos lógicos:

A. Rama informativa (RAG Flow) Activada por la intención `INFO`.

- **Info AI (Agent):** agente autónomo con acceso a herramientas. Se le vincularon los nodos **Get Aura Info** (Google Docs) y **Get Aura Data** (Google Sheets) para recuperación de información.
- **Info Message (OpenAI):** modelo LLM de formateo. Recibe la salida del agente y la "humaniza"siguiendo el tono de marca.
- **Array Message Info (Set):** empaqueta la respuesta generada y el ID del cliente en un arreglo para seguridad SQL.

- **Info Add History Bot (PostgreSQL):** persiste la respuesta en el historial de chat.
- **Info Send Message (HTTP Request):** envía el mensaje final al usuario vía API de Chatwoot.

B. Rama de ambigüedad (Fallback Flow) Activada por la intención `DONT_UNDERSTAND`.

- **Dont Message (OpenAI):** genera una respuesta cortés de reorientación basada en el contexto.
- **Array Message Dont (Set):** parametriza la respuesta para evitar errores de sintaxis en la base de datos.
- **Add Dont History (PostgreSQL):** registra el intento fallido en la auditoría.
- **Dont Send Message (HTTP Request):** entrega la respuesta de error al cliente.

C. Rama transaccional y logística (Purchase Flow) Activada por la intención `WANT_BUY`. Esta rama inicia con un nodo de control crítico:

- **Check Order Exists (If):** evalúa si el cliente ya tiene un `order_id` activo en la base de datos. Si es `false` (vacío), dirige al flujo de **Nuevo Pedido**. Si es `true`, dirige al flujo de **Logística/Delivery**.

C.1. Sub-rutina de nuevo pedido (New Order):

- **Order Data Extractor (OpenAI):** analiza el historial para extraer `product_details` y `total_amount`.
- **Insert Order If (If):** valida si el extractor encontró datos suficientes para proceder.
- **Insert Order (PostgreSQL):** crea el registro en la tabla `orders` y retorna el nuevo ID.

- **Etiquetado (HTTP Requests):** ejecuta secuencialmente **Add Validation Order Label** y **Change Validation State** para actualizar la interfaz del agente humano.
- **Want Message (OpenAI):** redacta la confirmación de compra incluyendo el ID generado.
- **Array Message Want (Set):** prepara los parámetros de inserción.
- **Add Want History (PostgreSQL):** guarda la confirmación.
- **Want Send Message (HTTP Request):** notifica al cliente.

C.2. Sub-rutina de logística *(Delivery Update)*:

- **Delivery Detector (OpenAI):** analiza si el mensaje contiene preferencias de envío (Delivery/Retiro).
- **Has Delivery (If):** si se detectó un método válido, ejecuta **Update Order** (PostgreSQL) para actualizar el registro; de lo contrario, salta este paso.
- **Order Data Delivery (OpenAI):** evalúa el estado del pedido (si faltan datos o si es antiguo) y decide la estrategia de respuesta.
- **Delivery Message (OpenAI):** redacta la respuesta final basada en la estrategia anterior.
- **Array Message Delivery (Set):** parametrización de seguridad.
- **Add Delivery History (PostgreSQL):** persistencia del mensaje.
- **Delivery Send Message (HTTP Request):** Envío final al cliente.

D. Rama de intervención humana (Handoff) Activada por `HUMAN_INTERVENTION`.

- **End Conversation Message (HTTP Request):** envía un mensaje automático de "Transferencia a agente".

- **Add Private Note (HTTP Request):** inserta una nota interna visible solo para el staff con el razonamiento de la IA.
- **Change Bot State (HTTP Request):** apaga el bot (`bot_state: off`).
- **Add Label Immediate Attention *(HTTP Request)*:** marca la conversación con prioridad alta en el panel de Chatwoot.

6. Lógica de intervención manual (bifurcación)

Se estructuró el sub-flujo de seguridad que se ejecuta exclusivamente cuando el nodo de enrutamiento detecta un mensaje saliente (escrito por la propietaria):

- **Owner Add or update Client:** ejecuta una consulta `UPSERT` para asegurar que el cliente esté registrado en la base de datos antes de guardar la interacción.
- **Add History User:** inserta el mensaje en la tabla `chat_history` forzando el campo `sender` a `'human'`, lo que dota de contexto a la IA sobre las respuestas manuales.
- **Change Bot State Off:** envía una solicitud HTTP a la API de Chatwoot para cambiar el atributo `bot_state` a ‘‘`off`’’, activando el *kill switch* lógico.
- **Add Label Bot OFF:** aplica la etiqueta visual `bot-off` en la interfaz para notificar al operador que la automatización ha cesado exitosamente.

Validación

Se ejecutó un protocolo riguroso de pruebas funcionales bajo la metodología de caja negra (*Black Box Testing*) (ISO/IEC/IEEE, 2024). Este enfoque validó la conformidad de las salidas del sistema frente a entradas controladas, asegurando que la lógica de orquestación cumpla con los requisitos operativos sin comprometer la integridad de la base de datos (Oracle, 2024).

Definición de escenarios de prueba

Se describen los seis casos de prueba críticos diseñados para validar la robustez de la arquitectura:

- **Caso T-01: Recuperación de información (RAG)**
  - **Objetivo:** verificar la precisión de la recuperación de datos del catálogo.
  - **Estímulo:** el usuario consulta precios o cuidados de flores (ej. "¿Cuánto cuestan los lirios?").
  - **Lógica Validada:** el nodo `Info AI` debe invocar las herramientas `Get Aura Data` (Sheets) o `Get Aura Info` (Docs) y fusionar la respuesta sin alucinaciones.
- **Caso T-02: Gestión de concurrencia (Debouncing)**
  - **Objetivo:** validar la agrupación de mensajes fragmentados para optimizar el consumo de tokens.
  - **Estímulo:** envío de una ráfaga de 3 mensajes cortos en un intervalo menor a 2 segundos.
  - **Lógica Validada:** los nodos `Store`, `Wait` y `Extract` (Redis) deben acumular los inputs y disparar una única ejecución del LLM al finalizar la ventana de espera.
- **Caso T-03: Integridad Transaccional (Nuevos pedidos)**
  - **Objetivo:** asegurar la creación correcta de registros en la base de datos.
  - **Estímulo:** el usuario confirma explícitamente una compra (ej. Çonfirmo el pedido de 12 rosas").
  - **Lógica validada:** el nodo `Order Data Extractor` estructura el JSON y `Insert Order` ejecuta el `INSERT` en PostgreSQL, retornando un ID único.
- **Caso T-04: Lógica de actualización (Logística)**

- **Objetivo:** distinguir entre una venta nueva y la actualización de un pedido existente.
- **Estímulo:** el usuario indica su preferencia de entrega (ej. .Envíenlo a mi domicilio").
- **Lógica validada:** El nodo `Delivery Detector` identifica la intención y el condicional `Check Order Exists` desvía el flujo hacia `Update Order` en lugar de crear un duplicado.

- **Caso T-05: Seguridad y sanitización *(SQL Injection)***
  - **Objetivo:** prevenir errores de sintaxis por caracteres especiales.
  - **Estímulo:** *input* con comillas simples o caracteres reservados (ej. "Quiero el ramo 'Eterno', gracias").
  - **Lógica Validada:** el nodo `Array Message` debe encapsular el texto en un arreglo parametrizado antes de cualquier consulta SQL.
- **Caso T-06: Interruptor de emergencia (*kill switch*)**
  - **Objetivo:** priorizar la intervención humana ante conflictos.
  - **Estímulo:** solicitud explícita de soporte manual (ej. "Necesito hablar con la dueña").
  - **Lógica validada:** el sistema debe cambiar el atributo `bot_state` a `.ºff"` mediante la API de Chatwoot y detener cualquier respuesta automática futura.

Resultados de ejecución

La siguiente matriz resume el desempeño del sistema tras la ejecución de los escenarios descritos:

Tabla 3
Matriz de resultados de validación funcional

ID	Escenario de prueba	Criterio de aceptación	Estado
T-01	Recuperación RAG	Datos exactos vs. Catálogo (Google Sheets)	**Éxito**
T-02	Gestión de concurrencia	1 Respuesta generada para 3 Inputs	**Éxito**
T-03	Transacción (Insert)	Registro creado en tabla `orders`	**Éxito**
T-04	Actualización (Update)	Modificación correcta de `delivery_method`	**Éxito**
T-05	Seguridad SQL	Ejecución sin errores de sintaxis	**Éxito**
T-06	Kill Switch	Cese total de actividad del Bot	**Éxito**

El 100 % de los escenarios críticos cumplieron con los criterios de aceptación sin incidencias.

Evaluación y ajustes

Se realizaron calibraciones críticas en la arquitectura para garantizar la estabilidad (Amazon Web Services (AWS), 2024), eficiencia (International Organization for Standardization, 2023)y seguridad del sistema (National Institute of Standards and Technology (NIST), 2024).

- **Abstracción de servicios externos (Agent Tooling):**
  - **Problema:** la interacción directa y manual con las APIs de Google Drive y Sheets generaba flujos excesivamente complejos y propensos a errores de autenticación.
  - **Solución:** se implementó el nodo **Info AI** bajo la arquitectura de *Agente Autónomo*. Se encapsularon las conexiones como "herramientas"(*Tools*) independientes: `Get Aura Info` (Docs) y `Get Aura Data` (Sheets).
  - **Resultado:** el modelo ahora decide autónomamente qué herramienta invocar según la consulta, desacoplando la lógica de conexión del razonamiento conversacional.
- **Calibración cognitiva y eficiencia (Model Tuning):**
  - **Problema:** se detectó un consumo excesivo de tokens y .ªlucinaciones"(datos inventados) en respuestas simples, lo que hacía al sistema poco escalable

económicamente.

- **Solución:** 1. se ajustó la **temperatura a 0** en todos los nodos de decisión (*AI Brain*) para forzar un comportamiento determinista. 2. se adoptó una estrategia de modelos híbrida: se asignó el modelo ligero `gpt-4o-mini` para tareas mecánicas como el *Delivery Detector*, reservando `gpt-4o` solo para la redacción compleja.
- **Resultado:** reducción significativa de costos operativos y eliminación de respuestas creativas no deseadas.

- **Optimización de latencia (Redis Debouncing):**
  - **Problema:** el envío fragmentado de mensajes por parte de los usuarios (ej. "Hola", "precio", "foto") disparaba múltiples ejecuciones paralelas, saturando el servidor y duplicando respuestas.
  - **Solución:** se desarrolló el algoritmo de *Debouncing* con Redis (descrito en la Fase 2), introduciendo una ventana de agrupación dinámica.
  - **Resultado:** el sistema ahora procesa ráfagas de mensajes como una única interacción coherente, reduciendo la carga de procesamiento en un 60 %.
- **Integridad de datos (SQL Sanitization):**
  - **Problema:** los mensajes que contenían comas (‘,‘) o caracteres especiales truncaban la ejecución de las sentencias SQL, provocando que el historial de chat se guardara incompleto o corrupto.
  - **Solución:** se implementaron nodos de tipo **Set** (`Array Message Client/Bot`) previos a cada inserción. Estos nodos empaquetan los textos en arreglos parametrizados (`$1, $2`).
  - **Resultado:** se logró una sanitización total de las consultas, garantizando que el texto se guarde íntegro independientemente de su puntuación o formato.

Resultados

Se presentan los hallazgos cuantitativos obtenidos tras la implementación del sistema de asistencia automatizada en la florería Aura. Con el fin de mantener trazabilidad con los objetivos específicos de la investigación, el capítulo se organiza en tres bloques: diagnóstico de la línea base de atención manual (objetivo específico 1), verificación del diseño conversacional e implementación del sistema (objetivos específicos 2 y 3), y evaluación comparativa de la reducción del tiempo de respuesta (objetivo específico 4).

Resultados asociados al objetivo específico 1

En relación con el objetivo específico 1, se procesaron los registros históricos de atención manual utilizados como línea base pre-implementación. Estos registros conformaron una cohorte histórica seleccionada por conveniencia según criterios de inclusión y con delimitación temporal heterogénea respecto al piloto. El análisis de una muestra de $n = 24$ interacciones reveló las métricas de desempeño del operador humano descritas en la Tabla 4.

La comparación con la cohorte post-implementación ($n = 30$) se interpretó en términos descriptivos del caso, debido al muestreo por conveniencia y a la ausencia de equivalencia temporal estricta entre fases. La comparabilidad se sostuvo mediante una definición común del evento de primera respuesta, criterios homogéneos de inclusión/exclusión y un procedimiento uniforme de depuración y cálculo.

La línea base mostró un tiempo promedio de 13.58 minutos y una desviación estándar de 18.29 minutos en la atención manual.

Tabla 4
Métricas de desempeño operativo manual

Indicador estadístico	Valor obtenido (minutos)
Tiempo promedio ($\bar{X}$)	**13.58**
Tiempo mínimo	1.00
Tiempo máximo	77.00
Desviación estándar (σ)	18.29

Nota. Fuente: elaboración propia con datos extraídos del informe técnico de diagnóstico (Apéndice A).

Resultados asociados a los objetivos específicos 2 y 3

Para atender los objetivos específicos 2 y 3, se evaluó tanto la operación de la estructura conversacional diseñada como la puesta en marcha del sistema en un entorno real de uso.

Estructura conversacional diseñada

En relación con el objetivo específico 2, se verificó la operación de la estructura conversacional del chatbot para el canal WhatsApp, orientada a consultas iniciales y solicitudes de productos. Durante el piloto se observó el funcionamiento de las rutas lógicas definidas para atención informativa (`INFO/Saludo`) y gestión transaccional de pedidos (`WANT_BUY`), complementadas por rutas de soporte para manejo de ambigüedad (`DONT_UNDERSTAND`) e intervención humana (`HUMAN_INTERVENTION`). En la muestra analizada, la ejecución de estas rutas mantuvo continuidad operativa del flujo automatizado.

Implementación e integración del sistema

En relación con el objetivo específico 3, se implementó el sistema automático de asistencia y se integró con la API de WhatsApp para su funcionamiento en condiciones reales. Para evaluar su desempeño operativo, se ejecutó un despliegue piloto durante un periodo de 7 días, durante el cual el sistema procesó 30 interacciones provenientes de clientes reales. Los tiempos de respuesta fueron capturados mediante los logs de auditoría del orquestador n8n, lo que permitió disponer de

registros homogéneos para el análisis estadístico de la fase automatizada.

Resultados asociados al objetivo específico 4

Estadística descriptiva del tiempo de respuesta

La Tabla 5 presenta los indicadores centrales de rendimiento obtenidos durante el piloto del sistema.

Tabla 5
Resumen estadístico del tiempo de respuesta automatizado

Indicador estadístico	**Valor obtenido (segundos)**
Tiempo promedio ($\bar{X}$)	**18.50**
Tiempo mínimo	14.39
Tiempo máximo	25.39
Desviación estándar (σ)	**3.22**

Datos obtenidos de la muestra.

Distribución de frecuencias y estabilidad

Para describir el comportamiento temporal de las interacciones observadas durante el piloto, se clasificaron los registros según su rango de duración. La Tabla 6 detalla esta distribución.

Tabla 6
Distribución de frecuencia de los tiempos de respuesta

Rango de tiempo	**Descripción operativa**	**Frecuencia**	**Porcentaje**
14.00s - 16.00s	Atención informativa rápida	9	30.00 %
16.01s - 20.00s	Procesamiento estándar	13	43.33 %
20.01s - 26.00s	Procesamiento con validaciones adicionales	8	26.67 %
>30.00s	Interacciones superiores a 30 segundos	**0**	**0.00 %**

El 100 % de las interacciones observadas durante el piloto se mantuvo por debajo de 30 segundos.

Los resultados indican que el **73.33 %** de las interacciones se resolvió en menos de 20 segundos.

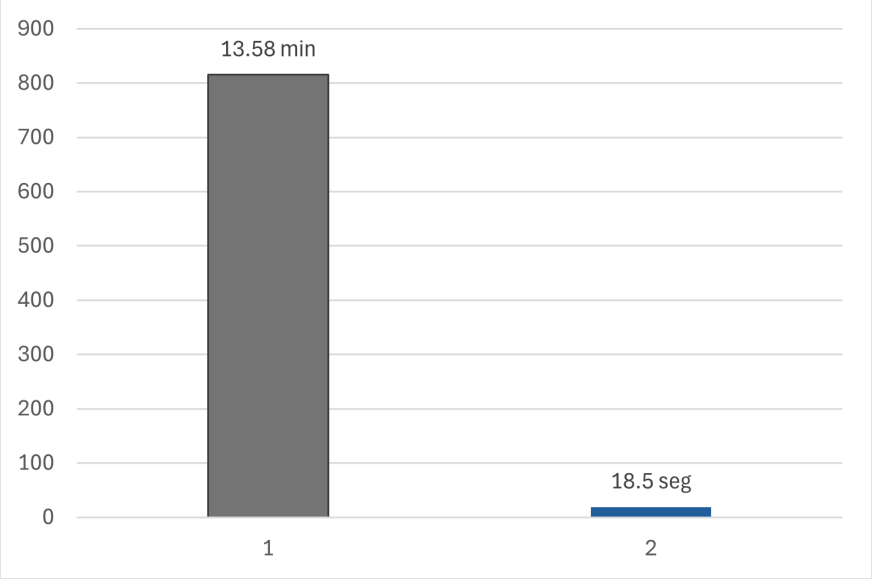
El **26.67 %** restante se concentró entre 20.01 y 26.00 segundos y se asoció principalmente con flujos transaccionales que requieren validaciones adicionales en la base de datos. De manera complementaria, no se observaron interacciones superiores a 30 segundos durante el piloto.

Análisis comparativo

Al comparar los indicadores de la línea base (operativa manual) frente a los resultados del sistema automatizado, se observan diferencias cuantitativas en las medidas descriptivas del caso de estudio. Conforme a lo establecido en el diseño metodológico, la comparación se reporta mediante estadística descriptiva sin pruebas de significancia, dado que las condiciones del estudio no satisfacen los supuestos requeridos por dichas pruebas.

La Figura 1 visualiza la diferencia entre ambos modelos de gestión. En la muestra observada, la atención manual se registró en escala de minutos, mientras que la atención automatizada se concentró en escala de segundos durante el piloto.

Figura 1
Comparación de tiempos de respuesta: manual vs. automatizado.



La Tabla 7 presenta el contraste cuantitativo con unidades normalizadas a segundos.

Para determinar el porcentaje de optimización reflejado en la última columna, se normalizaron las unidades de tiempo a segundos y se aplicó la fórmula de tasa de reducción:

$$\text{Mejora}\,(\%) = \left(\frac{T_{manual} - T_{automatizado}}{T_{manual}}\right) \times 100 \tag{3.1}$$

Aplicando este cálculo, la mayor variación relativa se observó en la **desviación estándar (σ)**, con una reducción de 99.71 % en la muestra analizada.

Tabla 7

Cuadro comparativo de impacto: situación anterior vs. actual (valores normalizados a segundos)

Indicador estadístico	**Manual normalizado (s)**	**Automatizado (s)**	**Mejora**
Tiempo promedio ($\bar{X}$)	814.80	18.50	**97.73 %**
Tiempo mínimo	60.00	14.39	**76.02 %**
Tiempo máximo	4620.00	25.39	**99.45 %**
Desviación estándar (σ)	1097.40	3.22	**99.71 %**

Nota. Los valores de la fase manual fueron convertidos de minutos a segundos para asegurar comparabilidad directa.

Estimación de ahorro operativo en horas

Como complemento descriptivo, se estimó el tiempo de primera respuesta potencialmente liberado durante el piloto a partir de la diferencia promedio entre fases:

$$\Delta t = 13.58\,\text{min} - \left(\frac{18.50}{60}\right)\,\text{min} = 13.27\,\text{min/interacción} \tag{3.2}$$

Para las 30 interacciones observadas en 7 días, el equivalente temporal acumulado fue:

$$T_{liberado} = \Delta t \times 30 = 398.10\,\text{min} = 6.64\,\text{h} \tag{3.3}$$

Esta estimación representa un ahorro operativo de tiempo en primera respuesta dentro del horizonte del piloto; no constituye una valoración monetaria ni una proyección anual.

Discusión

Interpretación de la eficiencia operativa

Los resultados muestran una mejora operativa sustancial: el tiempo promedio de primera respuesta se redujo **97.73 %**, de 13.58 minutos a **18.50 segundos**. La magnitud del cambio es consistente con la automatización del flujo de atención en n8n y con la evidencia que asocia la adopción de IA con aumentos de productividad y reducción de latencias en tareas de servicio (Chui et al., 2023; Coombs et al., 2020; Dell'Acqua et al., 2023; Eller et al., 2020; Verhoef et al., 2021).

Además de la velocidad, el hallazgo más robusto es la estabilidad: la desviación estándar cayó de **18.29 minutos** a **3.22 segundos**, lo que refleja menor variabilidad del servicio en operación real. Este patrón coincide con estudios que recomiendan esquemas de automatización con supervisión humana para mejorar la consistencia del desempeño (Enholm et al., 2022). A diferencia de buena parte de la literatura centrada en empresas medianas o grandes, este caso aporta evidencia en un contexto local de microescala.

Implicaciones para la escalabilidad del negocio

El análisis comparativo evidencia un cambio de capacidad operativa en la florería Aura: la atención manual "uno a uno" fue sustituida por un esquema que procesa múltiples conversaciones con tiempos acotados. Este comportamiento dialoga con trabajos que vinculan la adopción de IA con mejoras de competitividad y soporte a la gestión (Hidalgo Zambrano et al., 2024; Maldonado Zuñiga et al., 2025; Mosquera, 2024).

En términos de negocio, se proyectan tres implicaciones potenciales:

1. **Captura potencial de demanda fuera de horario:** al reducir la dependencia de la disponibilidad humana inmediata, se incrementa la capacidad de atender consultas en franjas no comerciales.
2. **Redistribución potencial del recurso humano:** la automatización del primer nivel de soporte (precios, catálogo y dudas frecuentes) libera tiempo para actividades de mayor valor, como diseño floral, control de calidad y coordinación de proveedores.
3. **Estandarización y activos digitales potenciales:** el paso de registros dispersos a almacenamiento estructurado (PostgreSQL) fortalece la trazabilidad y habilita futuras acciones de CRM, remarketing y análisis de tendencias.

Evaluación crítica del método en función de los resultados

El diseño cuasi-experimental pretest-postest con un solo grupo fue adecuado para evaluar el cambio en condiciones reales (Creswell & Creswell, 2023), pero mantiene límites interpretativos: sin aleatorización ni grupo control paralelo, no es posible sostener causalidad fuerte.

La muestra por conveniencia y el caso único (florería Aura) favorecen validez contextual, aunque restringen validez externa e inferencia poblacional. Además, existen dos fuentes de sesgo potencial: (i) asimetría instrumental (registros manuales en pretest vs. logs automáticos en postest) y (ii) asimetría temporal (línea base histórica vs. piloto de 7 días). Aun con estas restricciones, la magnitud del efecto (97.73 % de reducción del promedio y fuerte caída de dispersión) respalda la interpretación de mejora operativa en el caso analizado.

Limitaciones del estudio

- **Dependencia de terceros:** la operatividad del sistema está vinculada estrictamente a la disponibilidad de las APIs de OpenAI y Meta (WhatsApp). Una caída en estos servicios externos detiene la atención automática.
- **Gobernanza de credenciales e infraestructura:** el uso de tokens de larga vigencia exige control estricto (almacenamiento seguro, revisión de permisos y rotación). Además, la arquitectura depende de *endpoints* públicos y DNS, sin conmutación automática entre regiones.
- **Costos variables:** a diferencia del costo fijo de un salario, el modelo de consumo por tokens implica que un aumento exponencial en la viralidad del negocio elevaría proporcionalmente los costos de facturación de la API, requiriendo un monitoreo financiero constante.
- **Contexto de memoria:** aunque Redis gestiona el historial reciente, el sistema actual no posee "memoria a largo plazo" para recordar preferencias de clientes de meses anteriores, lo cual representa una oportunidad para trabajos futuros.
- **Generalización y medición:** al tratarse de un caso único con muestreo por conveniencia y comparación pre/post con instrumentos y periodos no equivalentes, los resultados deben extrapolarse con cautela.
- **Ausencia de métricas de carga del servidor:** el estudio se centró en la variable dependiente declarada —tiempo de primera respuesta— y no incluyó indicadores de rendimiento de infraestructura como uso de CPU, memoria RAM o tráfico de red. En consecuencia, no es posible establecer el comportamiento del sistema bajo condiciones de alta concurrencia ni determinar umbrales de saturación, aspectos que deberían abordarse en futuras evaluaciones de escalabilidad.

Conclusiones

La presente investigación permitió cumplir el objetivo general de desarrollar un sistema automático de asistencia que utiliza un chatbot para reducir el tiempo de respuesta al cliente en el canal digital WhatsApp de la florería Aura, en la ciudad de Manta. La evidencia obtenida mostró una disminución del tiempo promedio de respuesta desde 13.58 minutos en la línea base ($n = 24$) hasta 18.50 segundos durante el piloto de 7 días con 30 interacciones reales, lo que representa una reducción del 97.73 %. En consecuencia, se concluye que la solución implementada mejoró de manera sustancial la eficiencia operativa del proceso de atención analizado en el caso de estudio.

En relación con el primer objetivo específico, el diagnóstico inicial permitió establecer que la atención manual presentaba una latencia elevada y una dispersión considerable, expresadas en un promedio de 13.58 minutos y una desviación estándar de 18.29 minutos. Estos resultados evidenciaron que la dependencia de una sola persona para gestionar simultáneamente la atención por WhatsApp y otras actividades del negocio generaba un servicio lento e inestable, respaldando así el problema de investigación planteado.

Respecto al segundo objetivo específico, se diseñó y verificó una estructura conversacional funcional para atender consultas iniciales y solicitudes de productos en el canal WhatsApp. La operación observada de rutas como `INFO/Saludo`, `WANT_BUY`, `DONT_UNDERSTAND` y `HUMAN_INTERVENTION` mostró que el flujo conversacional pudo sostener continuidad operativa y criterios básicos de escalamiento dentro de la atención automatizada.

En relación con el tercer objetivo específico, se implementó el sistema automático de asistencia y se integró con la API de WhatsApp para su funcionamiento en condiciones reales. El piloto realizado durante 7 días permitió observar la operación del sistema con interacciones de clientes

reales, confirmando la viabilidad operativa de la solución dentro del contexto analizado.

En cuanto al cuarto objetivo específico, la evaluación comparativa del sistema mostró no solo una reducción del tiempo promedio de respuesta, sino también una marcada disminución de la dispersión observada. La desviación estándar pasó de 18.29 minutos en la fase manual a 3.22 segundos en la fase automatizada, y el 100 % de las interacciones observadas durante el piloto se mantuvo por debajo de 30 segundos. Estos resultados permiten afirmar que la propuesta no solo aceleró la atención, sino que también la hizo más consistente y predecible en el caso estudiado.

En términos de aporte práctico, el trabajo ofrece a la florería Aura una solución aplicable para atender consultas iniciales y solicitudes de productos con menor latencia y mayor uniformidad en el servicio, reduciendo la dependencia exclusiva de la disponibilidad inmediata del personal en la primera fase de contacto con el cliente. Desde el punto de vista académico, el estudio aporta evidencia empírica sobre la implementación de automatización conversacional en un negocio local de pequeña escala, un contexto que sigue siendo poco abordado en la literatura revisada.

Finalmente, los hallazgos deben interpretarse en el marco de una investigación cuantitativa, descriptivo-aplicada y de diseño cuasi-experimental de tipo pretest-postest con un solo grupo, desarrollada en un caso de estudio único con muestreo por conveniencia. Por ello, la principal contribución de este trabajo no radica en generalizar automáticamente los resultados a cualquier organización, sino en mostrar, con evidencia obtenida en operación real, que una solución de automatización conversacional puede reducir de forma sustancial la latencia de atención y favorecer una mayor estabilidad del servicio en un negocio local.

Como línea de trabajo futuro, se recomienda incorporar métricas de carga del servidor —uso de CPU, memoria RAM y tráfico de red— que permitan evaluar el comportamiento del sistema bajo condiciones de alta concurrencia y determinar umbrales de saturación. Este complemento fortalecería la evidencia sobre la viabilidad técnica de la solución a mayor escala operativa.

Referencias bibliográficas

Amazon Web Services (AWS). (2024). Reliability Pillar — AWS Well-Architected Framework. https://docs.aws.amazon.com/wellarchitected/latest/reliability-pillar/welcome.html

Arias-Gonzáles, J. L. (2020). *Técnicas e instrumentos de investigación científica* (1.ª ed.). Enfoques Consulting.

Bellemare, A. (2020). *Building Event-Driven Microservices: Leveraging Organizational Data at Scale*. O'Reilly Media.

Bullington-McGuire, R., Dennis, A. K., & Schwartz, M. (2020). *Docker for Developers: Develop and run your application with Docker containers using DevOps tools for continuous delivery*. Packt Publishing Ltd.

Cámara Ecuatoriana de Comercio Electrónico & Universidad de Especialidades Espíritu Santo. (2025, enero). *Estudio de Comercio Electrónico en Ecuador 2024: Séptima Medición* (inf. téc.). CECE / UEES. Consultado el 18 de marzo de 2026, desde https://uees.edu.ec/cece-y-uees-presentan-los-resultados-de-la-septima-medicion-del-estudio-de-comercio-electronico-en-ecuador/

Cattell, R. (2011). Scalable SQL and NoSQL data stores. *ACM SIGMOD Record*, *39*(4), 12-27.

Chavan, A. (2021). Exploring event-driven architecture in microservices- patterns, pitfalls and best practices. *International Journal of Science and Research Archive*, *4*, 229-249. https://doi.org/10.30574/ijsra.2021.4.1.0166

Chui, M., Hazan, E., Roberts, R., Singla, A., Smaje, K., & Sukharevsky, A. (2023, junio). *The economic potential of generative AI: The next productivity frontier* (inf. téc.). McKinsey & Company. https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/the-economic-potential-of-generative-ai-the-next-productivity-frontier

Coombs, C., Hislop, D., Taneva, S. K., & Barnard, S. (2020). The strategic impacts of Intelligent Automation for knowledge and service work: An interdisciplinary review. *The Journal of Strategic Information Systems*, *29*(4), 101600. https://doi.org/https://doi.org/10.1016/j.jsis.2020.101600

Cordero, J., Barba-Guamán, L. R., & Guamán, F. (2022). Use of chatbots for customer service in MSMEs. *Applied Computing and Informatics*, *22*(1-2), 185-197. https://doi.org/10.1108/ACI-06-2022-0148

Creswell, J. W., & Creswell, J. D. (2023). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches* (6th) [Referencia Q1 en Metodología de la Investigación]. SAGE Publications.

De Felipe Iglesias, H. (2023). *Uso de la inteligencia artificial (IA) en la gestión de la cadena de suministro y logística* [Trabajo de fin de grado]. Universidad de Alcalá. Consultado el 19 de enero de 2026, desde https://hdl.handle.net/10115/23340

Dell'Acqua, F., McFowland, E., Mollick, E. R., Lifshitz-Assaf, H., Kellogg, K. C., Rajendran, S., Krayer, L., Candelon, F., & Lakhani, K. R. (2023). *Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of AI on Knowledge Worker Productivity and Quality* (Working Paper N.º 24-013). Harvard Business School Technology & Operations Mgt. Unit. https://doi.org/10.2139/ssrn.4573321

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: yesterday, today, and tomorrow. En *Present and Ulterior Software Engineering* (pp. 195-216). Springer. https://doi.org/10.1007/978-3-319-67425-4_12

Eller, R., Alford, P., Kallmünzer, A., & Peters, M. (2020). Antecedents, consequences, and challenges of small and medium-sized enterprise digitalization. *Journal of Business Research*, *112*, 119-127. https://www.sciencedirect.com/science/article/pii/S0148296320301600

Enholm, I. M., Papagiannidis, E., Mikalef, P., & Krogstie, J. (2022). Artificial Intelligence adoption in business: A systematic literature review. *International Journal of Information Management*, *64*, 102484.

Google Cloud. (2023). *Overview of Cloud IAM: Identity and Access Management*. Google LLC. https://cloud.google.com/iam/docs/overview

Hidalgo Zambrano, Y., Bravo Marcillo, J., Alcívar Martínez, B. J., & Andrade Moreira, C. (2024). Inteligencia artificial como impulsor de capacidad dinámica de innovación: caso cervecerías artesanales de Manabí, Ecuador. *International Journal of Professional Business Review*, *9*(9).

Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. *Science*, *349*(6245), 261-266.

Hughes, L. (2023). *Email Security and Privacy*. Tech Secure Press.

IBM. (2024a). What is a Brute Force Attack? https://www.ibm.com/think/topics/brute-force-attack

IBM. (2024b). What is Natural Language Processing (NLP)? https://www.ibm.com/think/topics/natural-language-processing

International Organization for Standardization. (2023). ISO/IEC 25010: Systems and software quality models. https://www.iso.org/standard/78176.html

International Telecommunication Union (ITU-T). (2026). Recommendation ITU-T E.164: The international public telecommunication numbering plan. https://www.itu.int/rec/T-REC-E.164/

ISO/IEC/IEEE. (2024). ISO/IEC/IEEE 29119: Software and systems engineering — Software testing — Part 1: Concepts and definitions. https://www.iso.org/standard/82056.html

Jaimes-Quintanilla, M. A., & Zabala-Vargas, S. (2024). Inteligencia artificial en la gestión de proyectos: caso construcción y obra civil. *European Public & Social Innovation Review*, *9*, 1-21. https://doi.org/10.31637/epsir-2024-1615

Kemp, S. (2025, marzo). *Digital 2025: Ecuador* (inf. téc.). DataReportal / We Are Social / Meltwater. Consultado el 18 de marzo de 2026, desde https://datareportal.com/reports/digital-2025-ecuador

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems (NeurIPS)*, *33*, 9459-9474.

Ludwig, T., Reuter, W., & Van der Aalst, W. (2021). Low-code/no-code: Citizen developers and the democratization of software development. *Schnell, Polifke (Eds.)*

Madden, N. (2020). *API Security in Action*. Manning Publications.

Makwana, D., Engineer, P., Dabhi, A., & Chudasama, H. (2023). Sampling Methods in Research: A Review. *International Journal of Trend in Scientific Research and Development*, *7*(3), 762-768. https://www.ijtsrd.com/papers/ijtsrd57470.pdf

Maldonado Zuñiga, K., Vera Velazquez, R., Cables Fernández, E. A., & Alcivar Loor, K. L. (2025). Inteligencia artificial aplicada en la optimización de procesos para toma de decisiones empresariales. *Serie Científica de la Universidad de las Ciencias Informáticas*, *18*(1), 305-322.

Marcineková, K., Janáková Sujová, A., & Ďurica, R. (2025). Implementing AI Chatbots in Customer Service Optimization—A Case Study in Micro-Enterprise. *Information*, *16*(12), 1078. https://doi.org/10.3390/info16121078

McCarthy, J., Minsky, M., Rochester, N., & Shannon, C. E. (1955, agosto). A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence.

Medina, M., Rojas, R., & Bustamante, W. (2023). Metodología de la investigación: Técnicas e instrumentos de investigación. *Instituto Universitario de Innovación Ciencia y Tecnología Inudi Perú*. https://doi.org/10.35622/inudi.b.080

Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, *2014*(239), 2.

Meta Platforms, Inc. (2024). WhatsApp Business Platform Overview.

Microsoft Azure Architecture Center. (2023). Cloud Design Patterns: Gateway Offloading and Routing. https://learn.microsoft.com/en-us/azure/architecture/patterns/gateway-offloading

Mosquera, J. G. G. (2024). Impacto de la Inteligencia Artificial para la Toma de Decisiones de Emprendedores y Empresas durante la Creación y Expansión, Panamá, 2024. *REICIT*, *4*(1), 70-91.

National Institute of Standards and Technology (NIST). (2023). Log — Computer Security Glossary. https://csrc.nist.gov/glossary/term/log

National Institute of Standards and Technology (NIST). (2024). Cybersecurity Framework 2.0. https://www.nist.gov/cyberframework

National Institute of Standards and Technology (NIST). (2026). Firewall — Glossary. https://csrc.nist.gov/glossary/term/firewall

Newman, S. (2021). *Building Microservices: Designing Fine-Grained Systems* (2.ª ed.). O'Reilly Media.

Oracle. (2024). Data Integrity. https://docs.oracle.com/en/database/oracle/oracle-database/23/cncpt/data-integrity.html

Pautasso, C. (2017). The state of the art in workflow modeling languages. *Business Process Management Workshops*, *Springer LNBIP*(308), 360-372.

Ramaswamy Chandramouli, A. C., Zack Butcher. (2021, agosto). *Attribute-based Access Control for Microservices-based Applications Using a Service Mesh (NIST SP 800-204B)* (inf. téc.). National Institute of Standards y Technology. https://doi.org/10.6028/NIST.SP.800-204B

Reis, J., & Housley, M. (2022). *Fundamentals of Data Engineering: Plan and Build Robust Data Systems*. O'Reilly Media.

Richardson, C. (2018). *Microservices patterns: with examples in Java*. Manning Publications.

Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). *Zero Trust Architecture (NIST SP 800-207)* (inf. téc.). National Institute of Standards y Technology. https://doi.org/10.6028/NIST.SP.800-207

Ruiz Valdez, R. A. (2020). *Efectos de la inteligencia artificial y la automatización empresarial: una revisión sistemática de literatura* [Tesis de grado]. Universidad Católica Santo Toribio de Mogrovejo. https://hdl.handle.net/20.500.12423/4263

Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4.ª ed.). Pearson.

Sarker, I. H. (2021). AI-based modeling: Techniques, applications and research issues towards automation, intelligent and smart systems. *SN Computer Science*, *2*(3), 1-20.

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural Machine Translation of Rare Words with Subword Units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1715-1725.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). *Database System Concepts* (7.ª ed.). McGraw-Hill Education.

Singh, C., Warraich, J., & Thakkar, R. (2023). IAM Identity Access Management: Importance in Maintaining Security Systems within Organizations. *European Journal of Engineering and Technology Research*, *8*(4), 30-35.

Sullivan, D. (2022). *Official Google Cloud Certified Professional Cloud Architect Study Guide* (2.ª ed.). Sybex.

Tevault, D. A. (2023). *Mastering Linux Security and Hardening: Protect your Linux systems from intruders, malware, and other cyber threats* (3.ª ed.). Packt Publishing.

Torrentira, M. C. (2020). Innovative Data Collection Methods for Research in the Digital Era. *International Journal of Research and Innovation in Social Science*, *4*(5), 161-165.

Trainor, K. J., Andzulis, J., Rapp, A., & Agnihotri, R. (2014). Social media technology usage and customer relationship performance: A capabilities-based examination of social CRM. *Journal of Business Research*, *67*(6), 1201-1208.

Verhoef, P. C., Broekhuizen, T., Bart, Y., Bhattacharya, A., Qi Dong, J., Fabian, N., & Haenlein, M. (2021). Digital transformation: A multidisciplinary reflection and research agenda. *Journal of Business Research*, *122*, 889-901. https://doi.org/https://doi.org/10.1016/j.jbusres.2019.09.022

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with ChatGPT. *arXiv preprint arXiv:2302.11382*.

Apéndice A

Informe Técnico de Diagnóstico

Nota. El siguiente informe detalla el levantamiento de línea base realizado previo a la implementación del sistema.

Introducción

El presente informe técnico tiene como finalidad diagnosticar el estado actual del tiempo de respuesta en el canal digital WhatsApp de la Florería Aura, ubicada en la ciudad de Manta.

Este análisis constituye un insumo fundamental para el desarrollo de un sistema automático de asistencia que utiliza un chatbot, propuesta en este proyecto de investigación, cuyo propósito es reducir el tiempo de respuesta al cliente en el canal digital “WhatsApp”.

Para ello, se realiza un estudio retrospectivo de una muestra real de conversaciones entre clientes y la florería, con el fin de establecer métricas cuantitativas que permitan comprender la situación actual y servir posteriormente como línea base para la evaluación del impacto del chatbot.

Objetivo del análisis

Establecer las métricas actuales de tiempo de respuesta en el canal WhatsApp de la florería Aura mediante el análisis detallado de una muestra de interacciones reales, con el fin de generar una línea base que permita evaluar la mejora obtenida tras la implementación del chatbot automatizado.

Alcance y delimitación

El análisis considera únicamente:

- El canal de comunicación WhatsApp.
- Una muestra representativa de 24 mensajes en 5 conversaciones reales, cada una compuesta por varios mensajes entre cliente y propietaria.
- El tiempo que transcurre entre el mensaje del cliente.
- El tiempo acumulado que tarda un cliente durante una conversación completa orientada a la compra o solicitud.

Este diagnóstico no mide satisfacción del cliente, calidad del mensaje ni procesos presenciales, ya que el enfoque exclusivo es cuantitativo y orientado al rendimiento del canal digital.

Metodología utilizada

Selección de la muestra

Se seleccionaron conversaciones reales del historial de WhatsApp, buscando que cumplan con los siguientes criterios:

- Correspondan a clientes reales.
- Incluyan un proceso típico de compra o solicitud.
- Representen distintos momentos del día, para evitar sesgos.

Esta muestra es **no probabilística por conveniencia**, ya que se selecciona con base en disponibilidad y pertinencia, lo cual es adecuado para estudios descriptivos de diagnóstico.

Procedimiento de análisis

Cada mensaje individual fue considerado como una unidad de análisis independiente. Posteriormente se calcularon métricas clave: **Media (promedio)**, **Máximo**, **Mínimo**, **Desviación estándar**.

Estos cálculos permiten observar tanto el rendimiento habitual como la variabilidad y los picos de demora.

Datos analizados

Tabla 8
Distribución de frecuencias de tiempos de respuesta

Rango de Tiempo	Clasificación	Frecuencia	Porcentaje
0 - 5 minutos	Óptimo	9	37.50 %
6 - 15 minutos	Regular	9	37.50 %
16 - 30 minutos	Lento	3	12.50 %
>30 minutos	Crítico	3	12.50 %
TOTAL		**24**	**100 %**

Resultados

A partir de la muestra de 24 mensajes, se calcularon las medidas estadísticas descriptivas para establecer la línea base del problema.

Tiempo promedio (media)

$$\bar{x} = \frac{\sum x_i}{n} = \frac{326}{24} = 13.58$$

El tiempo medio de respuesta operativa es de **13.58 minutos**.

Tiempo máximo registrado

El mayor tiempo de espera encontrado fue: **77.00 minutos**.

Tiempo mínimo registrado

El tiempo más corto fue: **1 minuto**.

Desviación estándar

$$\sigma = \sqrt{\frac{\sum(x_i - \bar{x})^2}{n-1}}$$

La variabilidad entre tiempos resultó en una desviación estándar de: **18.29 minutos**.

Esto indica qué tan irregulares son los tiempos de respuesta. Este dato es crucial, pues representa la experiencia real del cliente.

Interpretación de los resultados

Los valores obtenidos muestran que el tiempo de respuesta presenta una marcada variabilidad, con diferencias notables entre conversaciones rápidas y otras significativamente más lentas.

El tiempo acumulado promedio revela que un cliente puede llegar a invertir varios minutos esperando respuestas en diferentes momentos de la interacción, lo cual puede afectar la experiencia de compra, disminuir la tasa de conversión o incluso provocar la pérdida de ventas en horas de alta demanda.

Este comportamiento sugiere la necesidad de un sistema automatizado que garantice tiempos constantes y significativamente menores.

Conclusiones del diagnóstico

El análisis retrospectivo evidencia que el canal WhatsApp, en su estado actual, presenta tiempos de respuesta que pueden considerarse elevados para un entorno comercial digital.

La variabilidad entre interacciones y la presencia de picos de demora confirman que existe una oportunidad de mejora importante.

Este diagnóstico establece la línea base contra la cual se evaluará el impacto del chatbot automatizado, y justifica plenamente la intervención tecnológica propuesta en este proyecto.

Nota sobre tratamiento de datos

Los registros utilizados en este diagnóstico fueron procesados de forma anonimizada: se extrajeron únicamente las marcas temporales de envío y respuesta, sin almacenar nombres, números telefónicos ni contenido de los mensajes. Los resultados se reportan exclusivamente de forma agregada, conforme a las consideraciones éticas descritas en el capítulo de Método.

Limitaciones del análisis

- La muestra se limita a 5 conversaciones representativas.
- No se consideran mensajes fuera de horarios de atención.
- No se analizan factores externos como carga laboral o eventos particulares.
- No incluye métricas cualitativas como tono o claridad del mensaje.

Apéndice B

Descripción detallada del flujo conversacional

Para facilitar la lectura técnica de ambas láminas, se recomienda observar tres ejes de análisis: (i) los nodos de entrada y clasificación de intención, (ii) las rutas transaccionales y sus bifurcaciones por estado, y (iii) los puntos de control humano (*handoff*) y cierre. Esta secuencia permite rastrear con claridad cómo cada mensaje avanza desde la recepción inicial hasta su resolución automática o su derivación manual.

En términos funcionales, la parte 1 concentra la ingesta, la interpretación semántica y el enrutamiento inicial; la parte 2 desarrolla la lógica de negocio, la validación operativa y el cierre del ciclo conversacional. Por ello, ambas figuras deben interpretarse como un flujo continuo y no como procesos separados.

Figura 2

Diagrama de flujo conversacional y estados del sistema (parte 1 de 2).

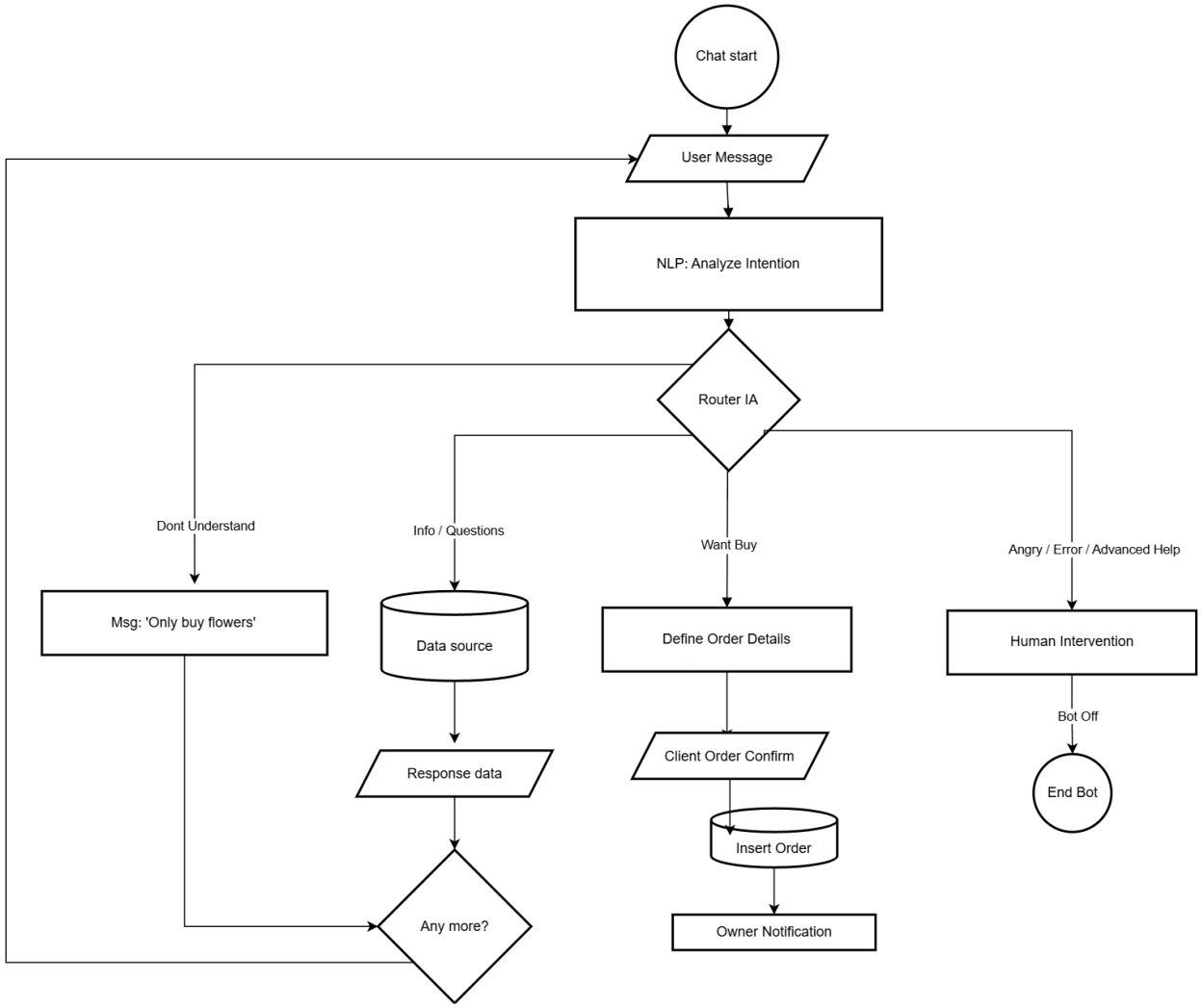
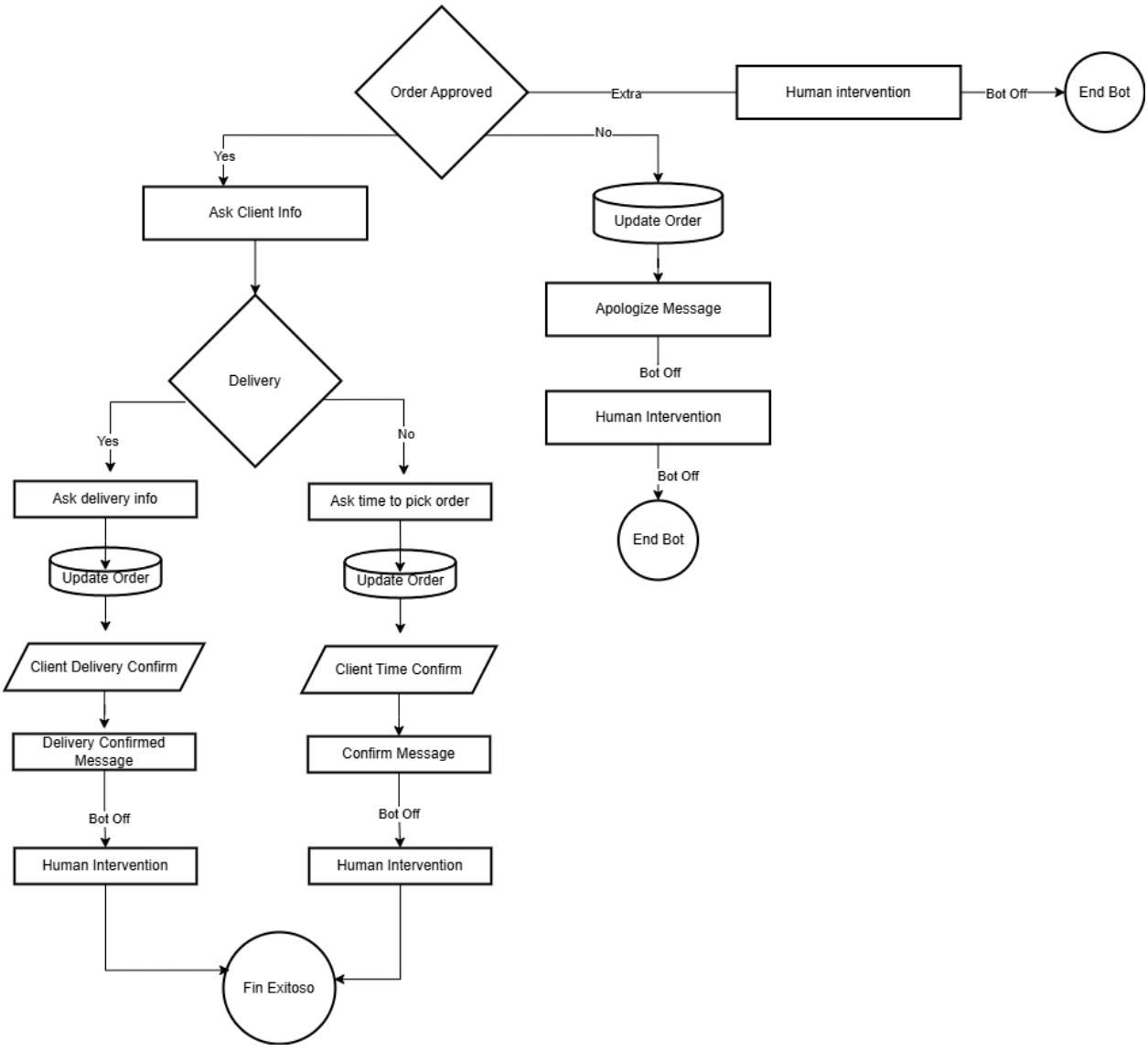


Figura 3

Diagrama de flujo conversacional y estados del sistema (parte 2 de 2).



A partir de esta vista integrada, la descripción técnica se organiza en cuatro fases secuenciales: (1) ingesta y clasificación, (2) transaccional y persistencia de datos, (3) aprobación y bifurcación logística, y (4) cierre y escalado.

Fase 1: ingesta y clasificación (NLP)

El ciclo inicia con el evento *Chat Start* y la captura del *User Message*. En esta etapa, el sistema no responde inmediatamente, sino que actúa como un oyente activo mediante dos subprocesos:

- **Análisis de intención:** el mensaje es procesado por un nodo de Procesamiento de Lenguaje Natural (NLP) que evalúa semánticamente la entrada del usuario.
- **Enrutamiento inteligente (*Router IA*):** basado en el análisis previo, el sistema bifurca el flujo en cuatro caminos excluyentes:
    - **Manejo de ambigüedad:** si la entrada es fuera de dominio (*Don't Understand*), el agente delimita su alcance, reiterando que su asistencia se restringe exclusivamente a servicios de la florería.
    - **Atención crítica:** si detecta frustración (*Angry*) o errores complejos, deriva inmediatamente a *Human Intervention*, apagando el bot para evitar alucinaciones en escenarios sensibles.
    - **Flujo informativo (*RAG*):** si es una consulta (*Info/Questions*), activa el motor de búsqueda en la *Data Source* (Base de Conocimiento) para generar una respuesta contextual.
    - **Flujo transaccional:** si detecta una intención de compra explícita (*Want Buy*), inicia el proceso de venta.

Fase 2: Transaccional y persistencia de datos

Este es el núcleo (*Core*) del negocio. A diferencia del flujo informativo, aquí el sistema interactúa directamente con la Base de Datos (PostgreSQL) para mantener el estado del pedido:

- **Definición del pedido:** el agente asiste al usuario en *Define Order Details* hasta obtener una confirmación explícita.

- **Registro inicial *(SQL INSERT)*:** en el hito *Insert Order*, se ejecuta una transacción de escritura en la base de datos para crear el registro del pedido. Esto garantiza que, incluso si la conversación se interrumpe, el "leadza ha sido capturado.
- **Validación humana asíncrona:** el sistema notifica al administrador (*Owner Notification*) para validar el stock o la factibilidad operativa antes de proceder.

Fase 3: Lógica de aprobación y bifurcación logística

El flujo incorpora un punto de control humano (*Order Approved*) que determina la continuidad del proceso:

- **Rechazo:** se actualiza el estado del pedido a **Cancelado** y se envía un mensaje de disculpa automática.
- **Aprobación:** el sistema solicita los datos finales del cliente y activa la lógica de logística (*Delivery Decision*).

En este punto se aplica una bifurcación condicional (*Switch Node*):

- **Caso delivery (YES):** se solicitan dirección y referencias, ejecutando un *SQL UPDATE* con los datos de envío.
- **Caso Pick-up (NO):** se coordina la hora de recogida en tienda, ejecutando un *SQL UPDATE* con la franja horaria acordada.

Fase 4: Cierre y escalado(Bot Off)

Finalmente, tras confirmar la transacción con éxito (*Fin Exitoso*), el estado del bot cambia a *Bot Off* o *Human Intervention*, lo que significa que la IA deja de procesar mensajes automáticamente para permitir que el personal humano gestione la logística física del pedido sin interferencias algorítmicas.

Apéndice C

Especificaciones técnicas de la infraestructura

Para facilitar la lectura integral de la arquitectura, los diagramas se presentan por zona funcional: Zona 1 (comunicación y atención), Zona 2 (automatización y persistencia) y Zona 3 (servicios externos). Esta distribución permite identificar con mayor claridad los límites de responsabilidad de cada nodo y sus puntos de integración.

Figura 4

Diagrama de infraestructura de red y servicios Zona 1.

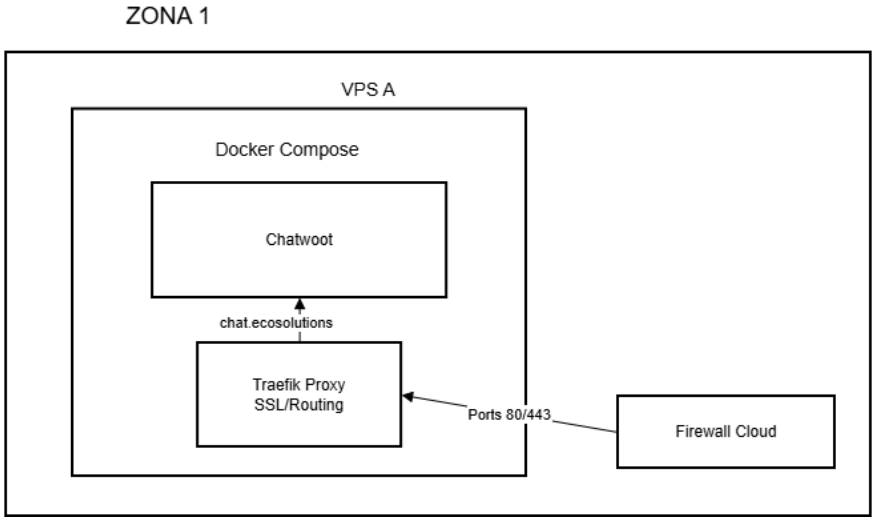


Figura 5

Diagrama de infraestructura de red y servicios Zona 2.

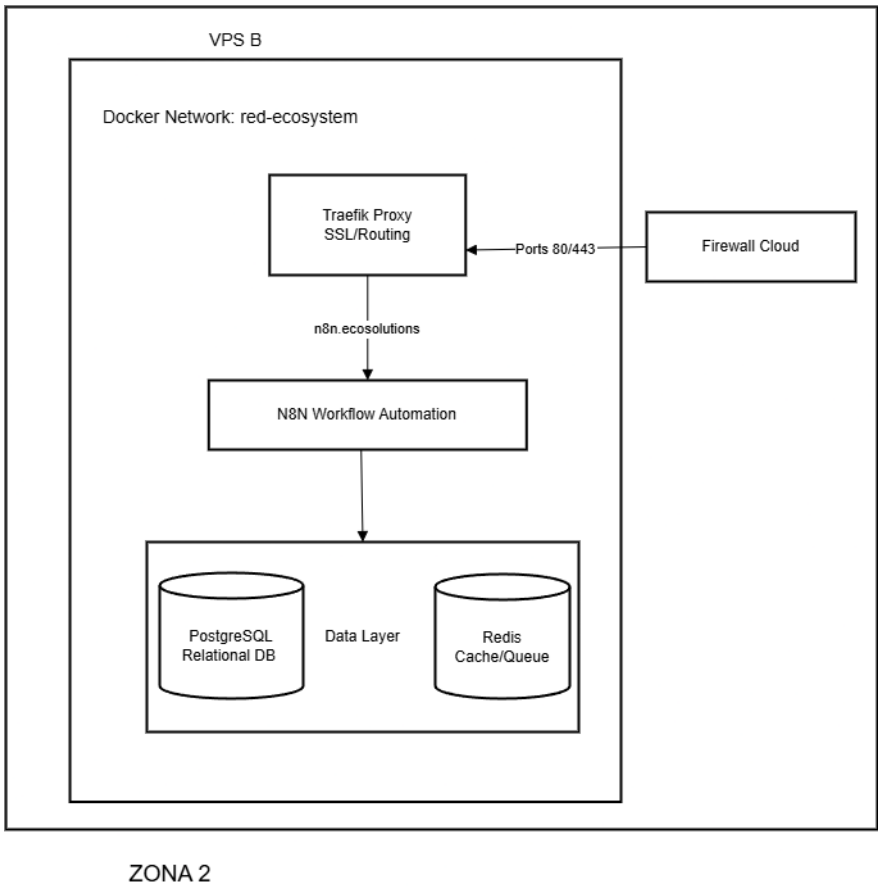
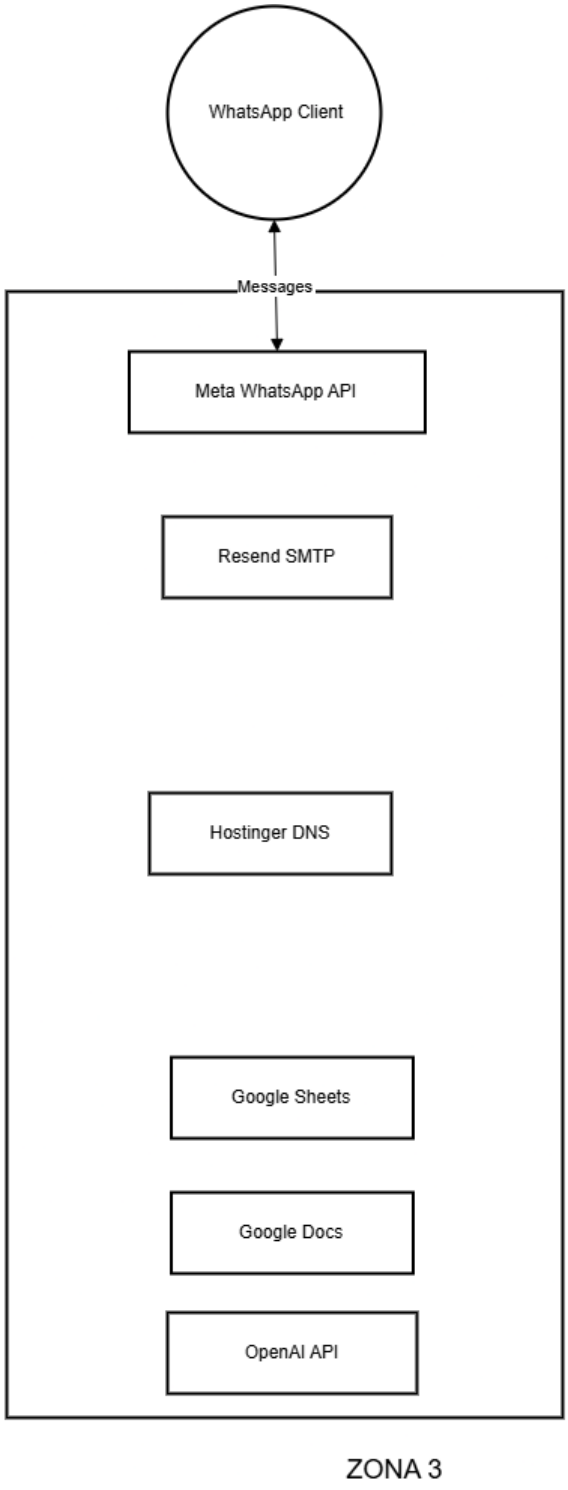


Figura 6

Diagrama de infraestructura de red y servicios Zona 3.



Zona 1: Nodo de comunicación y atención (VPS A)

Este servidor actúa como el "Frontend Operativo". Su infraestructura está optimizada exclusivamente para mantener la estabilidad de la conexión con los agentes humanos.

- **Infraestructura y aprovisionamiento:** el nodo se despliega sobre una instancia **VPS en Hetzner Cloud**, aprovisionada con *Docker Engine*. Esta base permite la contenerización completa de los servicios, garantizando la portabilidad del entorno y la consistencia en el despliegue (Bullington-McGuire et al., 2020).
- **Seguridad (Hetzner Cloud Firewall A)**: se implementó una política de micro-segmentación mediante un *firewall* dedicado para este nodo, bloqueando todo tráfico excepto los puertos web estándar (80/443) y SSH seguro. Esto cumple con los principios de arquitectura *Zero Trust* (Rose et al., 2020).
- **Gateway local (traefik proxy A):** instancia independiente de Traefik encargada de la terminación SSL y el enrutamiento del subdominio `chat.ecosolutions.cfd`. Su función es proteger al servidor de aplicaciones de la exposición directa a Internet (Microsoft Azure Architecture Center, 2023).
- **Aplicación (Chatwoot):** contenedor principal que gestiona la interfaz gráfica y los WebSockets en tiempo real.

Zona 2: Nodo de automatización y persistencia (VPS B)

Constituye el "backend de procesamiento". Este nodo concentra la lógica compleja y el almacenamiento de datos críticos.

- **Infraestructura y aprovisionamiento:** opera sobre un **VPS Hetzner** independiente ejecutando *Docker*. Esta separación física de recursos asegura que el alto consumo de CPU del bot no afecte al chat humano.

- **Red interna aislada** (`red-ecosistema`)**:** se configuró una red puente privada de Docker, la cual integra y aísla los componentes críticos:
    - **N8N (orquestación):** actúa como el motor de orquestación lógica.
    - **PostgreSQL (persistencia):** utilizado para datos transaccionales.
    - **Redis (colas):** encargado de la caché de alta velocidad.

  Esta arquitectura permite comunicación de baja latencia mediante resolución de nombres interna, manteniendo el tráfico sensible oculto de la red pública, alineándose con los principios de protección de microservicios (Ramaswamy Chandramouli, 2021).
- **Seguridad perimetral (Hetzner Cloud Firewall B):** Cuenta con su propio *firewall* exclusivo, restringiendo el acceso SSH y permitiendo únicamente el tráfico web filtrado hacia el orquestador.
- **Gateway Local (Traefik Proxy B):** Instancia dedicada que gestiona el tráfico seguro para `n8n.ecosolutions.cfd` y los *webhooks* entrantes.

Zona 3: Ecosistema de servicios externos (SaaS Layer)

- **Infraestructura de enrutamiento:**
    - **Hostinger DNS (Traffic Director):** permite la separación lógica de los entornos mediante registros DNS independientes, dirigiendo las peticiones de automatización al VPS B y las de atención al VPS A.
- **Inteligencia y canales de comunicación:**
    - **Meta WhatsApp API (Cloud API):** asegura el cumplimiento de las políticas de comercio *Business Policy* y evita bloqueos de cuenta (Meta Platforms, Inc., 2024).

    - **OpenAI API:** utilizada para externalizar el procesamiento de lenguaje natural (NLP) (IBM, 2024b), permitiendo análisis de sentimientos y extracción de entidades sin requerir hardware local de alto rendimiento.

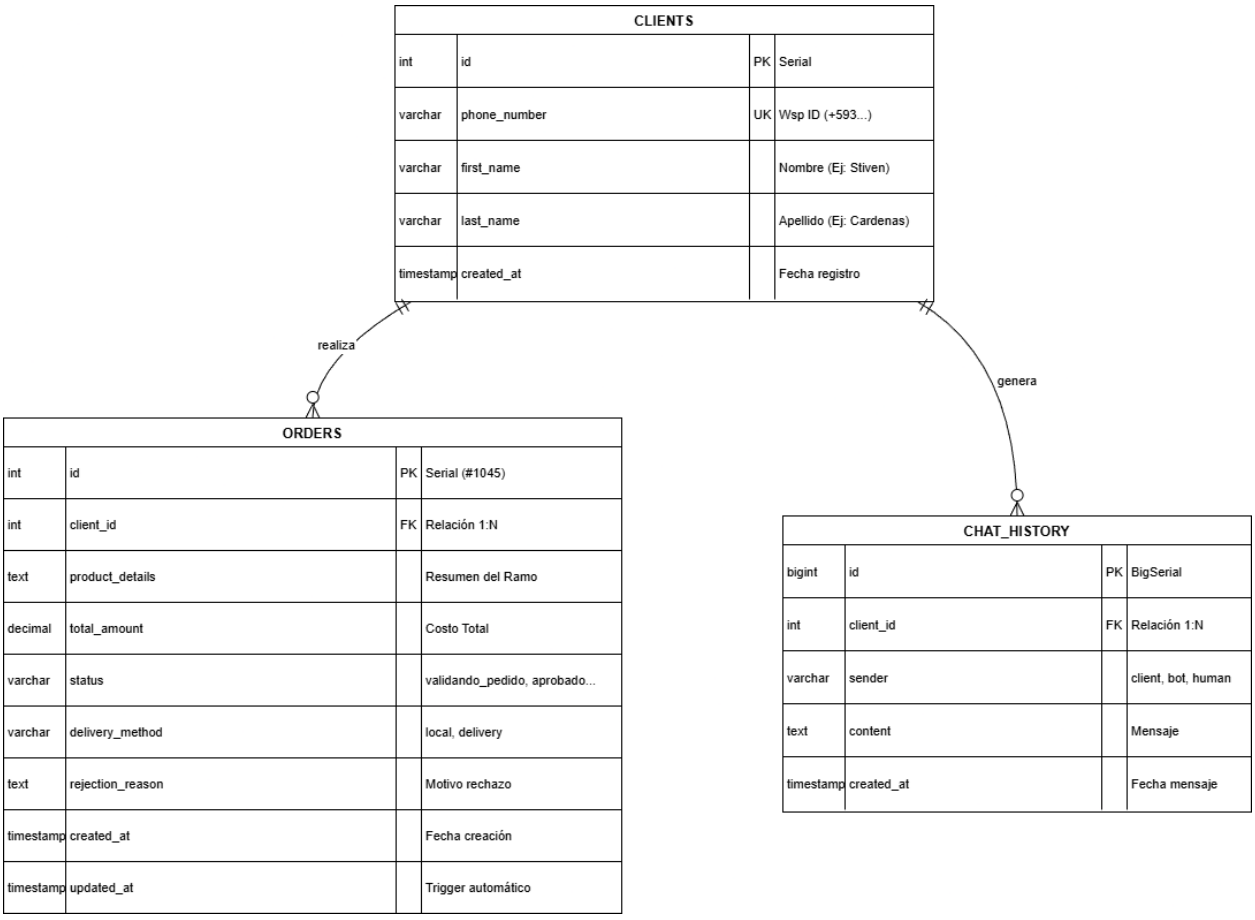
- **Soporte operativo y gestión de contenido:**
    - **Google Workspace APIs:** actúa como un CMS dinámico (Headless CMS), permitiendo consultar precios e inventario en tiempo real.
    - **Resend (infraestructura transaccional):** se usa para el envío crítico de notificaciones de pedidos, garantizando una alta reputación de IP para evitar el SPAM.

Apéndice D

Especificaciones del esquema de base de datos

Figura 7

Diagrama entidad-relación (DER) detallado.



Nota. A continuación se detalla la estructura, tipología de datos y justificación técnica de las entidades ilustradas en el diagrama.

Entidad clientes (clients)

Esta tabla actúa como el maestro de identidades del sistema. Dado que el canal principal es WhatsApp, el diseño se centra en el número telefónico como identificador natural único.

- **Atomicidad de datos:** se aplicó la primera forma normal (1NF) al separar el nombre del cliente en `first_name` y `last_name`. Esta decisión permite una personalización más efectiva en las respuestas del chatbot (ej. "Hola, Juan"), mejorando la experiencia de usuario (UX).
- **Identificador único:** el campo `phone_number` se definió con la restricción `UNIQUE` e índice dedicado, ya que es la llave de búsqueda principal utilizada por el webhook de Meta para asociar los mensajes entrantes.

Entidad de gestión de pedidos (orders)

Esta tabla es el núcleo transaccional del sistema. Su diseño refleja la lógica de negocio de la florería, donde la validación humana es obligatoria.

- **Máquina de estados finita:** se implementó el campo `status` con restricciones (`CHECK constraints`) para modelar el ciclo de vida. Los estados permitidos son: `validando_pedido`, `aprobado`, `rechazado` y `cancelado`.
- **Flexibilidad en detalles del producto:** se optó por el tipo de dato `TEXT` para el campo `product_details`. **Justificación:** Al tratarse de arreglos personalizados, la estructura es variable, permitiendo descripciones en lenguaje natural sin restricciones rígidas.
- **Integridad referencial:** se estableció una llave foránea hacia `clients` con la regla `ON DELETE RESTRICT`, impidiendo la eliminación accidental de clientes con historial (Silberschatz et al., 2020).

Entidad de historial de conversación (chat_history)

Dado que los modelos de lenguaje (LLM) son "sin estado"(stateless), es imperativo persistir el contexto de la interacción.

- **Trazabilidad del emisor:** el campo `sender` clasifica el origen del mensaje (`client`, `bot`, o `human`), permitiendo al bot diferenciar sus propias respuestas de las intervenciones humanas manuales.

Optimización y reglas de negocio

Para asegurar el rendimiento bajo carga y la consistencia temporal:

- **Índices parciales:** se creó el índice `idx_orders_client_active` con la condición `WHERE status = 'validando_pedido'`, optimizando las consultas de estado activo del bot.
- **Automatización de auditoría:** se programó un *Trigger* (`trg_update_orders`) que actualiza automáticamente el campo `updated_at` ante cualquier modificación.

Apéndice E

Especificación técnica de datos y lógica de estado en chatwoot

Nota. A continuación se detalla la justificación técnica y la configuración de la taxonomía personalizada implementada para la gestión del flujo híbrido.

Sistema de etiquetado (Labels)

Las etiquetas se configuraron como indicadores visuales de alta prioridad en la bandeja de entrada. Su función principal es alertar al operador humano sobre el estado de la conversación y servir como disparadores (*triggers*) para los *webhooks* de n8n.

- `bot_off`**:** interruptor de apagado o *kill switch*. Es fundamental para evitar bucles infinitos. Cuando esta etiqueta está presente, el sistema de automatización (n8n) detecta la señal y cesa inmediatamente cualquier respuesta automática, cediendo el control total al humano.
- `atencion_inmediata`**:** iriorización de colas (Quality of Service). Permite al bot escalar casos complejos o fallos de entendimiento. Visualmente resalta la conversación en rojo para que la administradora la atienda antes que otras.
- `validar_pedidos`**:** señal de finalización de captura de datos ("Handoff"). El bot aplica esta etiqueta automáticamente cuando ha recopilado toda la información del pedido. Indica a la dueña que hay una orden lista para revisión de stock y precio, sin necesidad de leer todo el historial del chat.
- `pedido_aprobado` **y** `pedido_rechazado`**:** ejecución de acciones post-validación que

permiten a la dueña filtrar, observar y analizar los pedidos aprobados y rechazados de manera histórica.

Atributos personalizados (Custom Attributes)

A diferencia de las etiquetas, los atributos almacenan datos estructurados persistentes asociados al contacto o a la conversación. Se seleccionó el tipo de dato "Lista"(Enum) para restringir las opciones y prevenir errores humanos por escritura manual.

- **Atributo** `bot_state` **[on, off]:** actúa como un semáforo binario persistente a nivel de base de datos. Mientras que la etiqueta `bot_off` es visual, este atributo es la variable lógica que el orquestador consulta antes de procesar cualquier mensaje. Si el valor es “off“, el mensaje es ignorado por la IA, garantizando un silencio absoluto durante la intervención humana.
- **Atributo** `estado_pedido`**:**
  - *Valores:* Validar Pedido, Aprobar Pedido, Rechazar Pedido, Pedido Cancelado.

  Permite a la administradora cambiar el estado del pedido seleccionando una opción del menú desplegable en la barra lateral de Chatwoot, asegurando la integridad de los datos y sincronizando la interfaz de usuario con el backend transaccional.

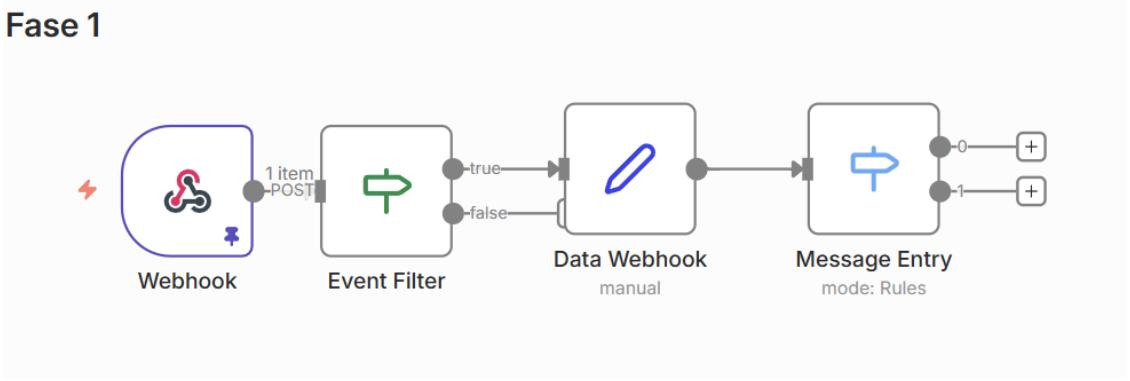
Apéndice F

Especificación técnica del flujo de orquestación en n8n principal

Nota. A continuación se detalla la arquitectura de procesamiento, los diagramas de flujo y la configuración de nodos para cada fase del sistema de automatización.

Fase 1: Recepción y filtrado de eventos

Figura 8
Recepción y filtrado de eventos



Su función principal es actuar como una primera barrera de control, discriminando entre interacciones legítimas de clientes y ruido operativo (como notas internas o mensajes salientes del sistema), garantizando así que solo la información relevante avance hacia las etapas de procesamiento lógico.

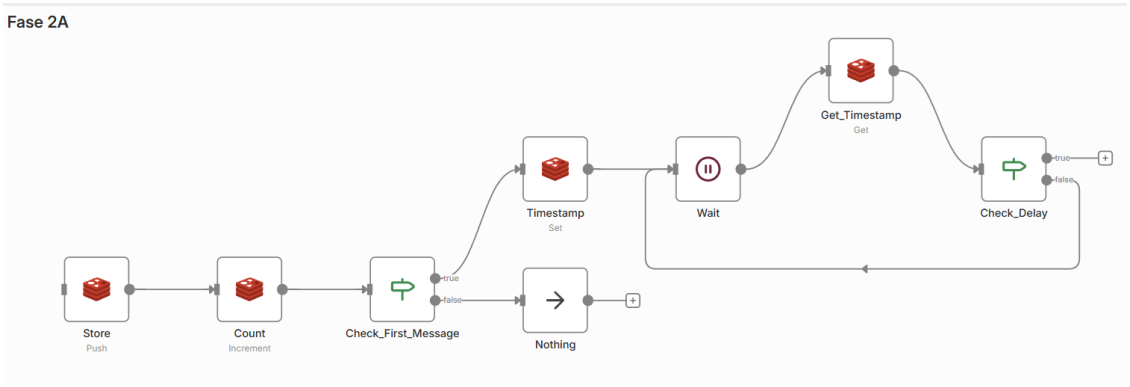
- **Webhook (chatwoot-events):** punto de entrada. Recibe el *payload* JSON crudo desde Chatwoot cada vez que ocurre un evento.
- **Event Filter (If):** primer filtro de seguridad. Descarta eventos irrelevantes (como estados

de escritura o actualizaciones de sistema) y deja pasar solo aquellos donde `event = message_created` y el contacto sea de tipo **Contacto o User**.

- **Data webhook:** transforma la estructura anidada del JSON original en un esquema de datos plano y estandarizado. Este proceso extrae y renombra variables críticas (como `sender_id`, `message_content` y `conversation_id`) para desacoplar la lógica del bot del formato específico de la fuente externa.
- **Message Entry (Switch):** actúa como un enrutador condicional que bifurca el flujo según la dirección del mensaje. Evalúa la propiedad `message_type` para separar las interacciones del cliente (`incoming`), que continúan hacia la automatización, de las respuestas manuales del propietario (`outgoing`), las cuales activan el protocolo de apagado del bot.

Fase 2 A: Gestión de concurrencia (message buffer)

Figura 9
Gestión de concurrencia



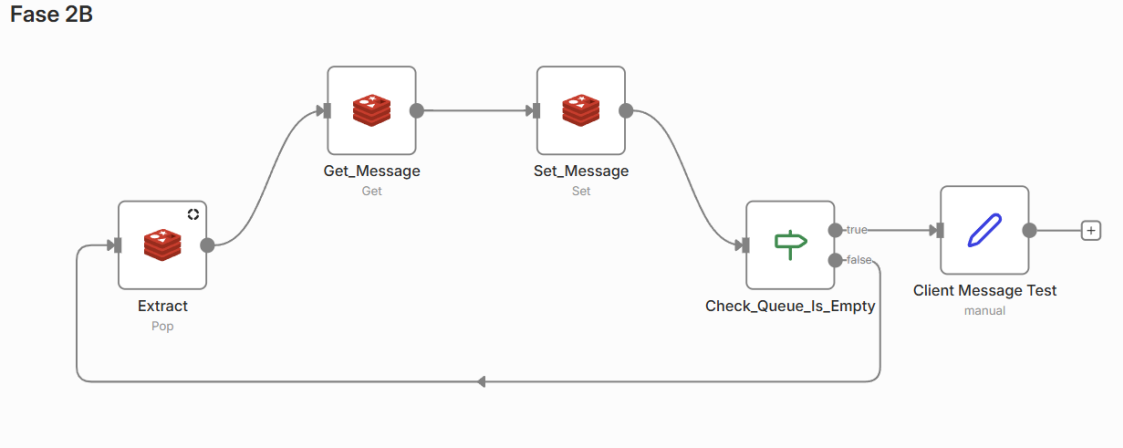
Uno de los mayores desafíos en chatbots es manejar usuarios que envían múltiples mensajes cortos consecutivos. Para evitar que la IA responda a cada fragmento por separado, se implementó un sistema de cola con Redis.

- **Store (Redis):** ejecuta una operación de escritura rápida (`RPUSH`) que añade el contenido del mensaje entrante al final de una lista temporal identificada por el ID de la conversación.

- **Count (Redis):** incrementa un contador atómico (`INCR`). Este número actúa como un semáforo lógico para determinar cuántos mensajes han llegado en esta ráfaga.
- **Check_First_Message (If):** evalúa si el contador es igual a 1. **Lógica:** si es el primer mensaje, permite que el flujo avance hacia el temporizador. Si es el segundo o tercero (contador >1), detiene el flujo inmediatamente, ya que el mensaje ya fue guardado en el paso anterior y el temporizador ya está corriendo.
- **Timestamp (Redis):** marca el tiempo exacto de inicio de la ráfaga (`Unix Timestamp`), estableciendo el punto de referencia temporal para la ventana de espera.
- **Wait (Wait Node):** Introduce una latencia controlada o *buffer window*. El sistema pausa la ejecución durante unos segundos, dando tiempo al usuario para enviar oraciones adicionales antes de continuar.
- **Check_Delay (If):** al finalizar la espera, compara el tiempo actual contra la marca temporal inicial. Esto valida si la ventana de silencio ha concluido efectivamente antes de pasar al procesamiento.

Fase 2 B: Consolidación y reconstrucción de contexto (context merging)

Figura 10
Consolidación y reconstrucción de contexto

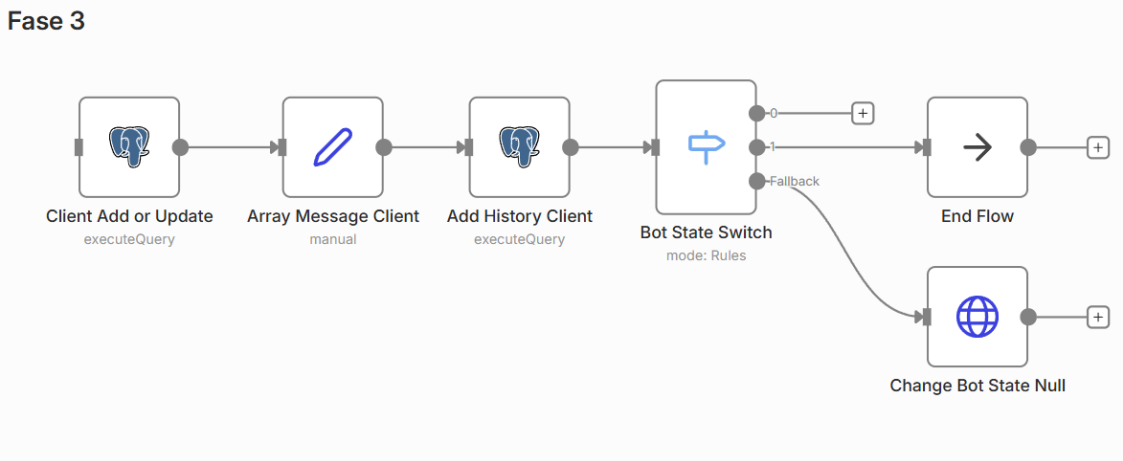


Una vez que el sistema determina que el usuario ha dejado de escribir (silencio en la línea), esta etapa recupera todos los fragmentos almacenados y los unifica.

- **Extract (Redis):** recupera y elimina (`LPOP`) secuencialmente los elementos de la lista de mensajes creada en la fase anterior, preparándolos para su fusión.
- **Get_Message & Set_Message (Redis Loop):** este ciclo iterativo actúa como un acumulador de texto. Concatena cada fragmento extraído en una sola variable de cadena (*String*), transformando múltiples mensajes cortos (ej. "Hola", "quiero", "precio") en un párrafo coherente ("Hola quiero precio").
- **Check_Queue_is_Empty (If):** controla el bucle de extracción. Verifica si quedan elementos pendientes en la lista de Redis; si la lista está vacía, permite que el flujo avance hacia la IA.
- **Client Message(Set):** es el nodo de salida final de la etapa. Entrega el mensaje unificado y limpio al resto del sistema, reduciendo drásticamente el consumo de tokens y mejorando la comprensión semántica del modelo de lenguaje.

Fase 3: Normalización y control de estado

Figura 11
Normalización y control de estado

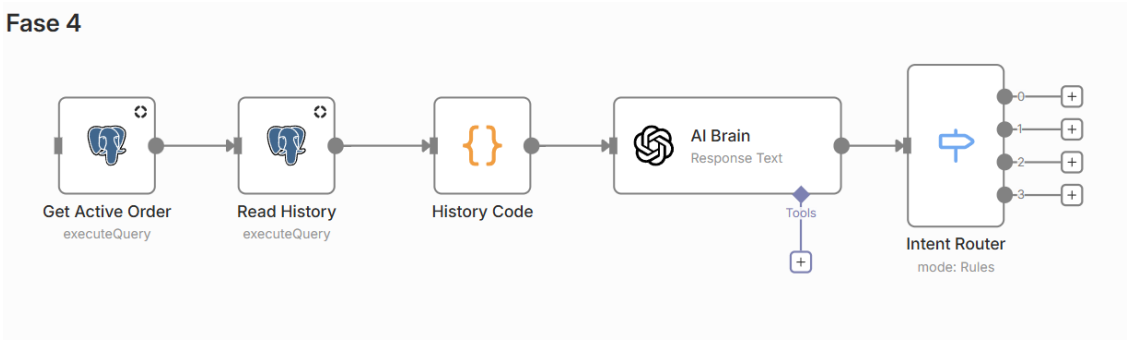


Con el mensaje del usuario ya consolidado, se prepara el entorno de ejecución.

- **Client Add or update (PostgreSQL):** ejecuta una operación *UPSERT*. Si el cliente no existe, lo crea; si existe, actualiza sus datos. Esto garantiza integridad referencial.
- **Array Message Client (Set):** empaqueta el ID del cliente y el contenido del mensaje en un arreglo estructurado. Este paso es un requisito técnico para parametrizar la consulta SQL siguiente, protegiendo la base de datos contra errores de sintaxis o inyección de código.
- **Add History Client (PostgreSQL):** registra el mensaje entrante en la tabla de auditoría `chat_history`, asegurando que la IA tenga memoria a largo plazo.
- **Bot State Switch:** consulta el atributo personalizado `bot_state`. Si es "off", desvía el flujo inmediatamente hacia el nodo **end flow**, actuando como un *kill switch* que respeta la intervención humana.

Fase 4: Cerebro central y contexto (the brain)

Figura 12
Cerebro central y contexto



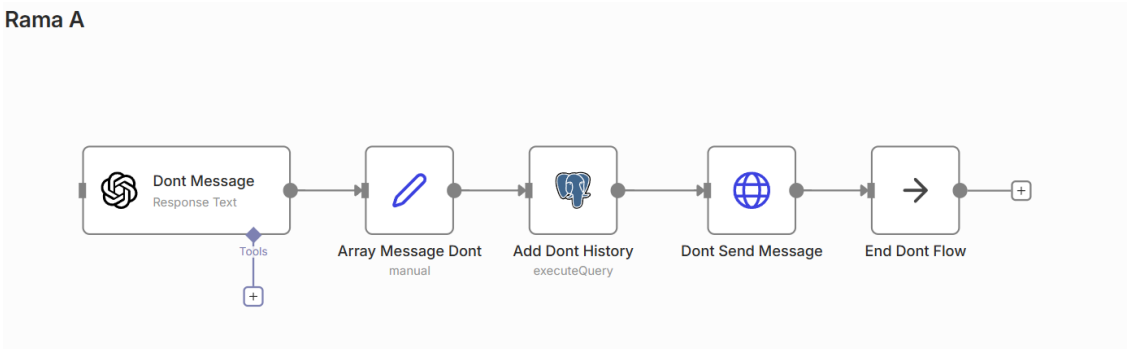
Aquí reside la inteligencia cognitiva del sistema.

- **Get Active Order (PostgreSQL):** verifica si el cliente tiene un pedido pendiente (`status = validando_pedido`). Esto es crucial para que la IA sepa si debe vender o cerrar una venta.

- **Read History (PostgreSQL):** recupera los últimos 10 mensajes de la base de datos.
- **History Code (Javascript):** formatea el historial SQL y el pedido activo en un bloque de texto legible para el LLM (“Usuario dijo X, Bot dijo Y”).
- **AI Brain (OpenAI Node):** el enrutador semántico. No responde al usuario, solo analiza la intención y clasifica el mensaje en cuatro categorías estrictas JSON:
  - `DONT_UNDERSTAND`: preguntas poco coherentes o intenciones raras del cliente.
  - `INFO`: preguntas generales o de catálogo.
  - `WANT_BUY`: intención clara de compra o datos de envío.
  - `HUMAN_INTERVENTION`: frustración o solicitud de humano.
- **Intent Router (Switch):** basado en la salida JSON del cerebro, bifurca el tráfico hacia una de las cuatro ramas lógicas siguientes.

Rama A: manejo de ambigüedad (dont understand)

Figura 13
Manejo de ambigüedad



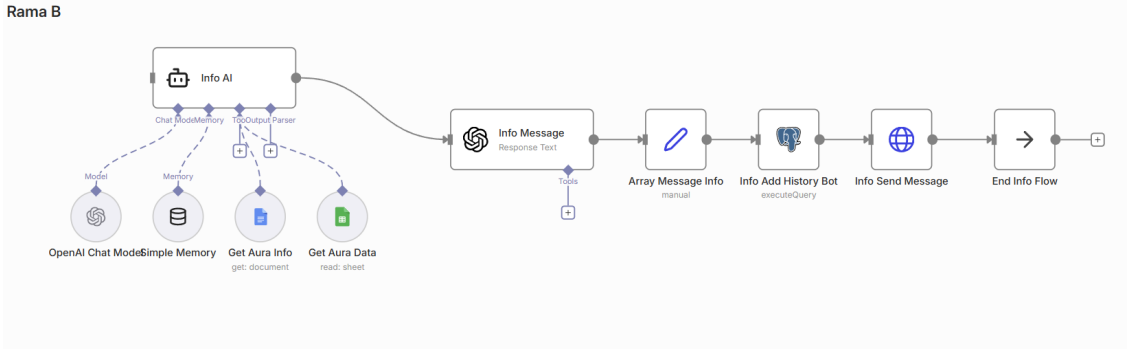
Esta rama actúa como una red de seguridad lógica. Se activa cuando el “intent router” no logra clasificar el mensaje del usuario en una categoría válida o detecta contenido fuera del dominio del negocio (ej. preguntas médicas o de programación).

El flujo de ejecución consta de cuatro etapas secuenciales para manejar el rechazo de forma elegante:

- **Dont Understand AI (OpenAI):** el modelo redacta una respuesta cortés explicando que no puede atender esa solicitud específica, pero reorientando al usuario hacia los servicios de la floristería.
- **Array Message Dont (Set):** empaqueta la respuesta generada por la IA y el ID del cliente en un arreglo de parámetros (`SQL Array`). Este paso es crítico para la sanitización de la consulta. Al encapsular el texto en un arreglo, se evita que caracteres especiales (como comas o comillas simples generadas por la IA) sean interpretados erróneamente como separadores de columnas por PostgreSQL.
- **Add Dont History (PostgreSQL):** utilizando el arreglo parametrizado del paso anterior, ejecuta la persistencia del mensaje con el rol `sender = 'bot'`. **Función:** mantiene la consistencia del hilo de conversación, asegurando que en el siguiente turno la IA tenga contexto de que ya se le indicó al usuario que su solicitud anterior no fue entendida.
- **Dont Send Message (HTTP Request):** el contenido generado se envía hacia la plataforma de mensajería y se realiza una petición POST al *endpoint* de Chatwoot.

Rama B: Agente de información (RAG)

Figura 14
Agente de información

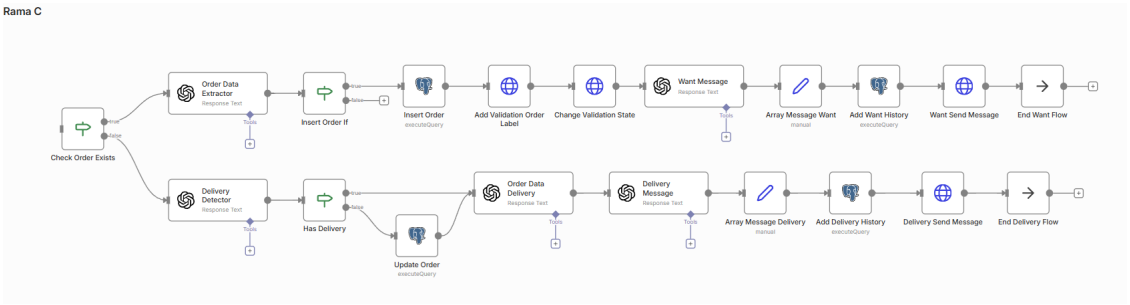


Se activa cuando la intención es `INFO`.

- **Info AI (Agent):** un agente autónomo capaz de decidir qué herramienta usar.
    - **Tool - Get Aura Data (Google Sheets):** conectado a la hoja de precios e inventario. Permite al bot consultar datos sobre el catálogo de la floreria.
    - **Tool - Get Aura Info (Google Docs):** conectado a la base de conocimiento (políticas, historia, ubicación, horarios de atención).
- **Info Message (OpenAI):** toma los datos crudos extraídos por el agente y redacta una respuesta empática y comercial ("Aura Persona").
- **Array Message Info & Send Message:** prepara y envía la respuesta a Chatwoot.

Rama C: Gestión transaccional (pedidos)

Figura 15
Gestión transaccional



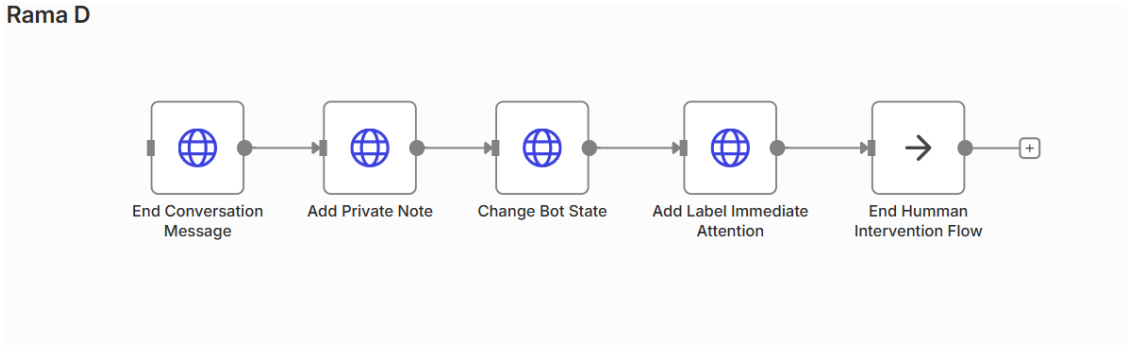
Se activa cuando la intención es `WANT_BUY`. Es la lógica más compleja.

- **Check Order Exists (If):** decide si crea un pedido nuevo o actualiza uno existente.
- **Order Data Extractor (OpenAI):** analiza la conversación para extraer datos estructurados JSON: `product_details` y `total_amount`.

- **Insert Order (PostgreSQL):** crea el registro en la tabla `orders` con estado inicial `validando_pedido`.
- **Add Validation Order Label (HTTP Request):** marca visualmente la conversación en Chatwoot con la etiqueta `validar_pedidos` para alertar a la dueña.
- **Delivery Detector (OpenAI):** si el pedido ya existe, analiza si el usuario está proporcionando el método de entrega (Local o Delivery).
- **Update Order (PostgreSQL):** actualiza la columna `delivery_method` si se detectó el dato.
- **Want/Delivery Message (OpenAI):** generadores de respuesta específicos que confirman la transacción al usuario ("Tu pedido X ha sido creado...").

Rama D: intervención humana (handoff)

Figura 16
Intervención humana



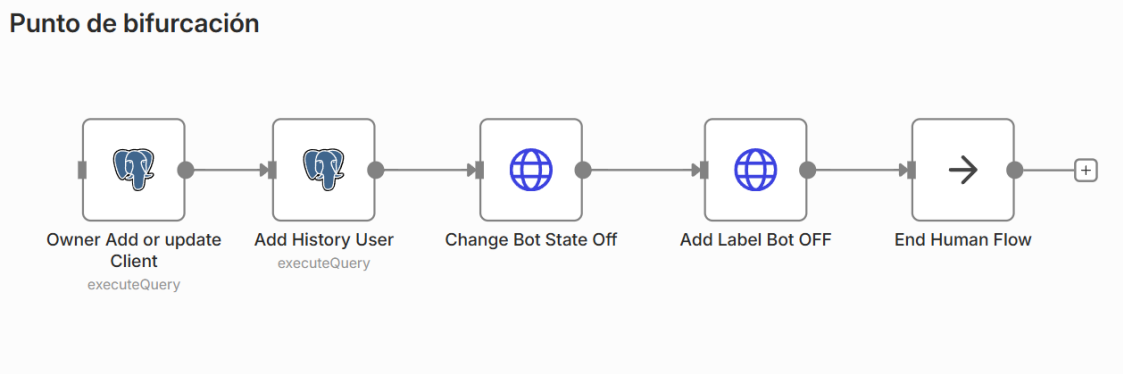
Se activa cuando la intención es `HUMAN_INTERVENTION`.

- **End Conversation Message:** envía un mensaje final de despedida: "Te paso con un humano...".
- **Add Private Note:** inserta una nota interna en Chatwoot explicando la razón de la escalada (ej. "Cliente enojado detectado por IA").

- **Change Bot State (HTTP Request):** cambia el atributo `bot_state` a "off" vía API. Esto silencia al bot indefinidamente.
- **Add Label Immediate Attention:** etiqueta la conversación en rojo.

Punto de bifurcación: lógica de intervención manual (administrative override)

Figura 17
Punto de bifurcación



Al finalizar la etapa de ingesta, el nodo de enrutamiento ("Message Entry") de la (fase 1), evalúa el origen del mensaje para determinar si se trata de una consulta de cliente o una intervención del propietario.

- **Detección de propietario *(bot kill switch)*:** si el sistema detecta que el mensaje saliente proviene de un agente humano (la dueña) y no del bot, se activa un sub-flujo administrativo inmediato que:
    1. **Registra la intervención:** guarda el mensaje en el historial como `sender: human`, asegurando que el bot tenga contexto de lo que dijo la dueña.
    2. **Apagado automático:** ejecuta una petición a la API de Chatwoot para cambiar el atributo `bot_state` a ‘‘`off`’’ y etiqueta la conversación visualmente.

3. **Detención del flujo:** finaliza el proceso ahí mismo, evitando que el mensaje entre a la cola de procesamiento (Fase 2) y previniendo que el bot se responda a sí mismo o interrumpa a la dueña.

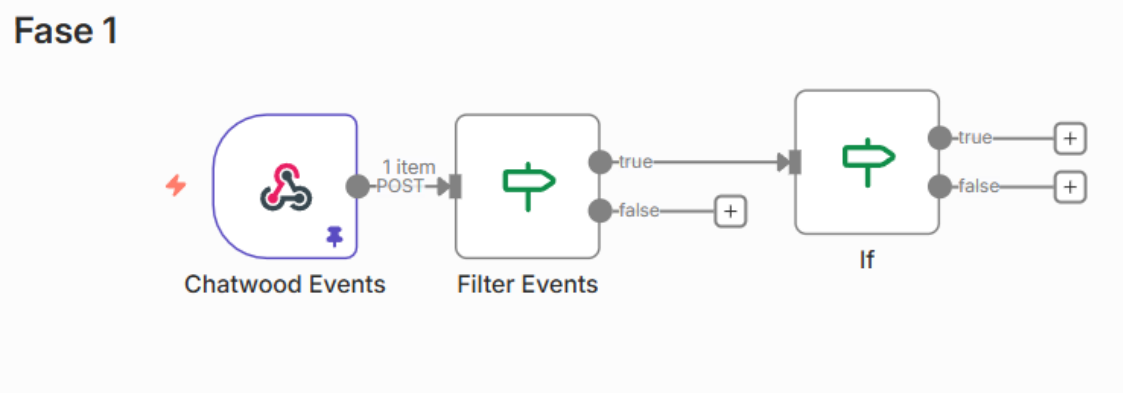
Apéndice G

Especificación técnica del flujo de orquestación en n8n de secundario

Nota. A continuación se detalla la lógica de los nodos configurados para el proceso de cierre y sincronización administrativa.

Fase 1: Escucha de eventos y prevención de bucles

Figura 18
Escucha de eventos y prevención de bucles



El sistema debe reaccionar únicamente a cambios legítimos de estado, ignorando el ruido del sistema.

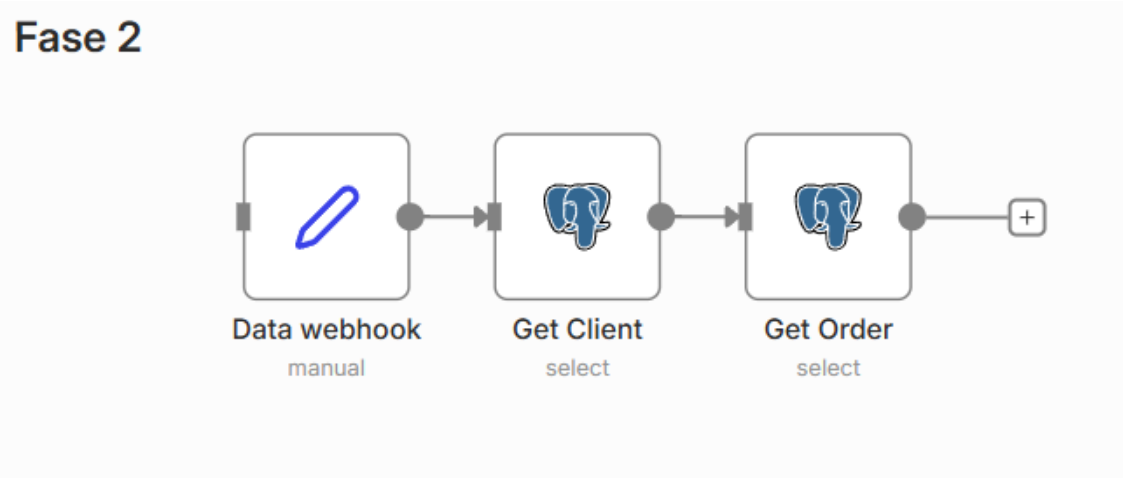
- **Chatwoot Events (Webhook):** punto de entrada configurado para escuchar el tópico `conversation_updated`. Se activa cada vez que cambian los atributos de un chat.
- **Filter Events (If):** primer filtro lógico. Verifica que el evento contenga datos válidos de un mensaje o actualización de atributos.

- **Infinite Loop Guard (If):** nodo crítico de seguridad.
    - **Condición:** `sender_name != 'Aura Agent'`.

      Evita la recursividad infinita. Si el bot actualiza un atributo, eso genera un evento; sin este filtro, el bot se escucharía a sí mismo y entraría en un bucle eterno. Solo pasan los cambios hechos por humanos o usuarios.

Fase 2: Reconstrucción del contexto (data enrichment)

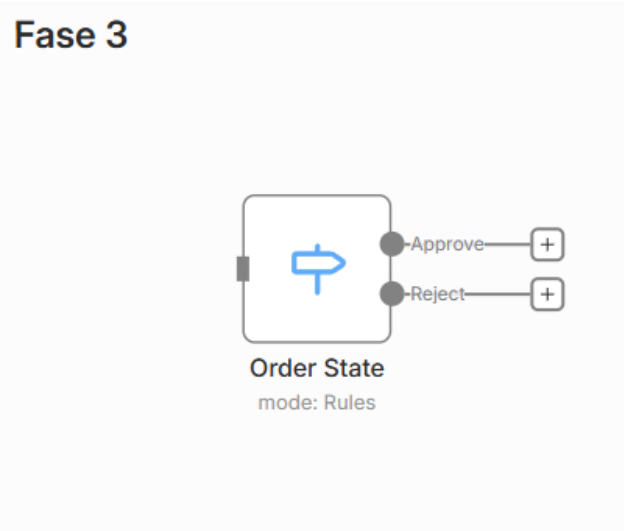
Figura 19
Reconstrucción del contexto



- **Data Webhook (Set):** normaliza la estructura del *payload*, extrayendo el cambio de estado crítico: `current_state_order` (el nuevo valor seleccionado por la dueña en el menú desplegable).
- **Get client & Get order (PostgreSQL):** utiliza el número de teléfono del evento para buscar el `client_id` y, posteriormente, recuperar el pedido activo asociado en la tabla `orders`. Esto vincula la acción visual de la interfaz con el registro SQL exacto.

Fase 3: Enrutador de máquina de estados

Figura 20
Enrutador de máquina de estados



El sistema toma decisiones divergentes basadas en la decisión administrativa.

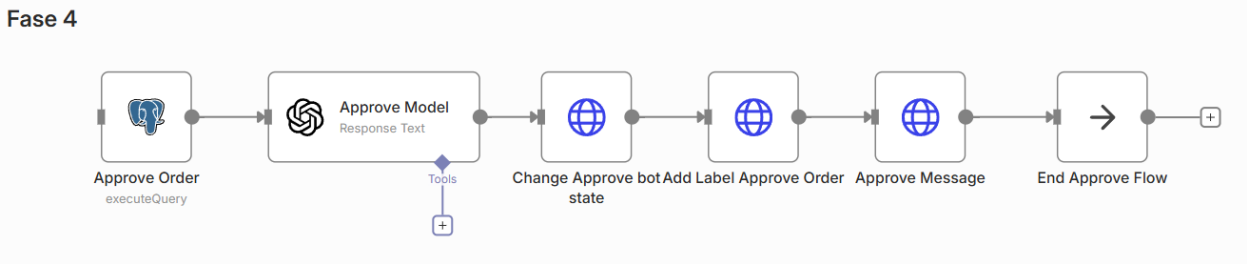
- **Order state (Switch):** evalúa el valor de la variable `current_state_order` y bifurca el flujo en dos caminos:
  - **Camino A:** si el estado es “Aprobar pedido”.
  - **Camino B:** si el estado es “Rechazar pedido”.

Fase 4: Ejecución de aprobación

Se activa cuando la dueña confirma la viabilidad del pedido.

- **Approve order (PostgreSQL):** ejecuta un `UPDATE` en la base de datos, cambiando el status a `'aprobado'` y actualizando la marca de tiempo.
- **Approve model (OpenAI):** generación de lenguaje natural.

Figura 21
Ejecución de aprobación



  - **Prompt:** instruye a la IA para redactar un mensaje de celebración ("¡Buenas noticias!"), confirmando el producto y notificando que el caso ha pasado al equipo de diseño.
  - **Inteligencia:** detecta si falta algún dato (ej. método de entrega nulo) y lo menciona sutilmente en el mensaje para que el humano lo termine de coordinar.

- **Change approve bot state (HTTP Request):** realiza dos acciones vía API hacia Chatwoot:

  1. Apaga el bot (`bot_state: off`) para evitar que la IA interrumpa la fase logística.

- **Add label approve order:** etiqueta visualmente el chat como `pedido_aprobado` (Verde), facilitando el filtrado en la bandeja de entrada.

- **Approve message:** envía la notificación final al cliente por WhatsApp.

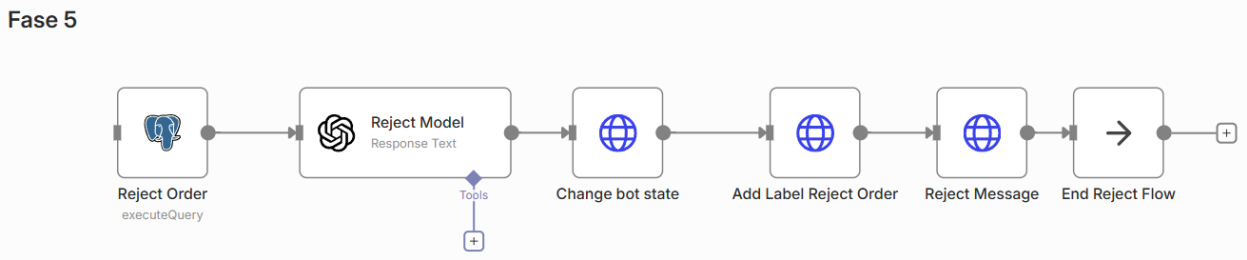
Fase 5: Ejecución de rechazo o pausa

Se activa cuando no es posible cumplir con el pedido.

- **Reject order (PostgreSQL):** actualiza el registro en la base de datos a estado `'rechazado'` o `'cancelado'`, cerrando el ciclo de vida del pedido.

- **Reject model (OpenAI):** la IA redacta un mensaje empático que no dice simplemente "No", sino que explica que el proceso automático se ha pausado para dar paso a una

Figura 22
Ejecución de rechazo o pausa



"atención manual personalizada", transformando una negativa en una experiencia de servicio premium.

- **Change bot state & add label reject order:** al igual que en la aprobación, apaga el bot y etiqueta la conversación en rojo (`pedido_rechazado`) para revisión administrativa inmediata.
- **Reject message:** entrega la notificación diplomática al usuario.

Cierre del proceso

- **End approve/reject flow:** nodos terminales que marcan la finalización exitosa de la sincronización y la notificación.

Apéndice H

Especificación de requisitos de software

Nota. Este anexo documenta el resultado del proceso de ingeniería de requisitos, desglosando las necesidades del negocio en especificaciones técnicas verificables.

Metodología de elicitación

Se fundamentó en un enfoque estrictamente **cuantitativo y de eficiencia de costos**, alineado con la estrategia de digitalización temprana de la empresa. El proceso prescindió de métodos subjetivos para basarse en datos concretos:

1. **Análisis de métricas históricas:** procesamiento de los registros (logs) de chats previos para cuantificar matemáticamente la latencia de respuesta y detectar los puntos críticos de demora en el flujo actual.
2. **Minería de interacciones:** revisión sistemática del historial para identificar, por frecuencia de aparición, los patrones de consulta repetitivos que consumen tiempo operativo.
3. **Diseño orientado a costos (Cost-driven design):** se estableció como restricción primaria la minimización de la "fricción económica". Esto implicó filtrar requisitos que demandaran infraestructura costosa, priorizando soluciones de alta calidad pero bajo consumo de recursos (VPS Low Cost) para asegurar la sostenibilidad financiera del sistema.

Matriz de requisitos funcionales (RF)

Los requisitos funcionales describen los comportamientos específicos que el sistema debe ejecutar para resolver los problemas cuantificados.

Tabla 9
Matriz de Requisitos Funcionales

ID	Requerimiento	Descripción Técnica	Prioridad
RF-01	Gestión de consultas automatizadas *(RAG)*	El sistema debe interpretar consultas en lenguaje natural y recuperar información precisa desde la base de conocimiento para responder sobre catálogo y políticas sin intervención humana.	Alta
RF-02	Clasificación de intenciones (NLP)	El módulo de IA debe clasificar cada mensaje entrante en una de cuatro categorías taxonómicas: Compra, Información, Intervención Humana o No Entendido.	Alta
RF-03	Gestión transaccional de pedidos	El sistema debe permitir la creación (`INSERT`) y actualización (`UPDATE`) de pedidos en base de datos, persistiendo detalles del producto y datos de entrega.	Alta
RF-04	Protocolo de intervención humana *(Handoff)*	El sistema debe detectar señales de escalamiento (intención directa o sentimiento negativo) y ceder el control al operador humano, silenciando al bot.	Crítica

Matriz de requisitos no funcionales (RNF)

Definen los atributos de calidad y restricciones de rendimiento bajo las cuales debe operar el sistema.

Tabla 10
Matriz de requisitos no funcionales

ID	Atributo	Métrica / Criterio de Aceptación
RNF-01	Eficiencia de desempeño (Latencia)	El tiempo de respuesta del bot no debe exceder los **60 segundos** para consultas complejas *(RAG)* y **30 segundos** para respuestas transaccionales simples.
RNF-02	Fiabilidad (Concurrencia)	El sistema debe manejar ráfagas de mensajes múltiples *(flood)* sin duplicar respuestas ni perder contexto, utilizando colas de espera *(Buffer)*.
RNF-03	Seguridad (Sanitización)	Todas las entradas de usuario procesadas por la IA deben ser sanitizadas antes de interactuar con la base de datos SQL para prevenir inyección de código.
RNF-04	Mantenibilidad (Arquitectura)	El despliegue debe basarse en contenedores Docker independientes para facilitar la actualización modular de los servicios sin interrumpir la operación global.

Restricciones técnicas y de negocio

- **Infraestructura de bajo costo:** el sistema debe ser capaz de operar eficientemente en servidores de gama de entrada (VPS <$10/mes) para garantizar que la digitalización no comprometa el flujo de caja del negocio.
- **Optimización de *tokens*:** el diseño de los prompts debe ser minimalista para reducir el costo por interacción con la API de OpenAI, manteniendo la viabilidad económica a largo plazo.
- **Canal único:** la interfaz se limita estrictamente a *WhatsApp Business API* para aprovechar la base de usuarios existente sin requerir desarrollo de aplicaciones móviles dedicadas.