UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

**FACULTAD DE INGENIERÍA**
**CARRERA DE COMPUTACIÓN**

**TEMA:**

# Desarrollo de un asistente virtual inteligente basado en arquitectura RAG para la gestión de información académica y logística en la Facultad De Ingeniería en la Carrera de Ciencias de la Computación.


**AUTOR:**
**Tapia Alvarez, Victor Hugo**


**Proyecto de tecnologías de información previo a la obtención del título de INGENIERO EN CIENCIAS DE LA COMPUTACIÓN**


**TUTOR:**
**Ing. García Sánchez, Roberto**


**Guayaquil, Ecuador**
**31 de agosto de 2026**

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

**FACULTAD DE INGENIERÍA**
**CARRERA DE COMPUTACIÓN**

## CERTIFICACIÓN

Certifico que el presente proyecto de tecnologías de información fue realizado en su totalidad por **Tapia Alvarez, Victor Hugo,** como requerimiento para la obtención del título de **Ingeniero en Ciencias de la Computación.**

f._____
**García Sánchez, Roberto**

**Guayaquil, 31 de agosto del año 2026**

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

**FACULTAD DE INGENIERÍA**
**CARRERA DE COMPUTACIÓN**

**DECLARACIÓN DE RESPONSABILIDAD**

**Yo, Tapia Alvarez, Victor Hugo**

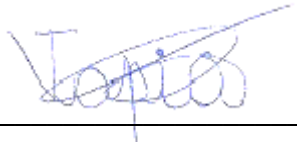
**DECLARO QUE:**

El Proyecto de tecnologías de información, **Desarrollo de un asistente virtual inteligente basado en arquitectura RAG para la gestión de información académica y logística en la facultad de ingeniería en la Carrera de Ciencias de la Computación,** previo a la obtención del título de Ingeniero en Ciencias de la Computación, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este proyecto es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Proyecto de tecnologías de información referido.

**Guayaquil, 31 de agosto del año 2026**

**EL AUTOR:**



**f.**_____

**Tapia Alvarez, Victor Hugo**

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

**FACULTAD DE INGENIERÍA**
**CARRERA DE COMPUTACIÓN**

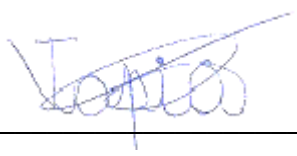
**AUTORIZACIÓN**

**Yo, Tapia Alvarez, Victor Hugo**

Autorizo a la Universidad Católica de Santiago de Guayaquil a la **publicación** en la biblioteca de la institución del Proyecto de tecnologías de información, **Desarrollo de un asistente virtual inteligente basado en arquitectura RAG para la gestión de información académica y logística en la facultad de ingeniería en la Carrera de Ciencias de la Computación**, cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y total autoría.

**Guayaquil, 31 de agosto del año 2026**

**EL AUTOR:**



**f.**_____

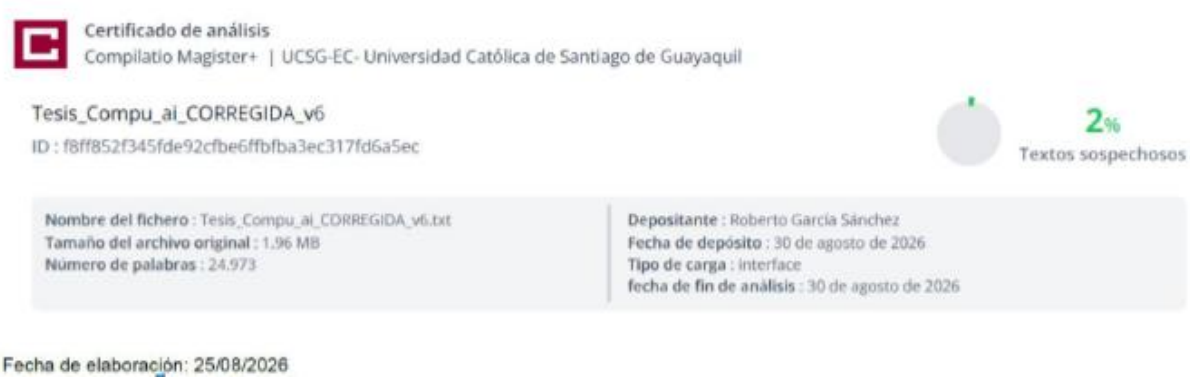
**Tapia Alvarez, Victor Hugo**

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

**FACULTAD DE INGENIERÍA**

**CARRERA DE COMPUTACIÓN**

**REPORTE ANTIPLAGIO**




Fecha de elaboración: 25/08/2026

f._____

**García Sánchez, Roberto**

## AGRADECIMIENTO

En primer lugar, agradezco a Dios, sin él no podría estar donde estoy hoy en día, agradezco por su guía, por todas las veces que me ha dado ánimos para seguir adelante y no rendirme, y darme una luz de esperanza cuando todo se veía perdido.

Además de a mi madre y mi padre, por su afecto, ayuda y sostenibilidad económica que me proporcionaron a lo largo de todos mis estudios.

También agradezco a mi Tutor el Ing. Roberto Garcia Sanchez y a todas las amistades que tengo y tuve a lo largo de mi carrera universitaria, he pasado buenos momentos con ellos y así mismo nos hemos ayudado unos a otros cuando más lo necesitábamos.

## DEDICATORIA

Este logro está dedicado, en primer lugar, a Dios, por darme las fuerzas para levantarme en cada tropiezo. De manera muy especial, a mis padres: gracias por su amor constante, por enseñarme el valor del esfuerzo y por sostenerme en cada etapa de esta carrera. A mis amigos y compañeros de travesía, que se convirtieron en hermanos de este camino, por compartir conmigo los mejores momentos y hacer más ligero el peso de las dificultades.

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

**FACULTAD DE INGENIERÍA**
**CARRERA DE COMPUTACIÓN**

**TRIBUNAL DE SUSTENTACIÓN**

**f.**_____

**ING. ROSA MACIAS MARTINEZ, MGS**

**DOCENTE DE CARRERA**

**f.**_____

**ING. JOSE ERAZO AYON, MGS**

**DELEGADO DIRECTORA DE CARRERA**

**f.**_____

**ING. ANA CAMACHO CORONEL, MGS**

**OPONENTE**

# ÍNDICE

RESUMEN ..... XIV
ABSTRACT ..... XV
INTRODUCCIÓN ..... 2
Capítulo 1. Planteamiento del problema ..... 4
1.1 Anatomía del silo informativo institucional ..... 4
1.2 Demanda sobre la atención presencial en Secretaría ..... 4
1.3 Hacia una interfaz conversacional complementaria ..... 5
1.4 Objetivos ..... 6
1.5 Alcance y limitaciones ..... 6
1.6 Descripción de la solución propuesta ..... 7
2.1 Generación aumentada por recuperación: el paradigma y su estratificación ..... 8
2.2 Generación aumentada con recuperación estructurada: la precisión relacional sobre el dominio transaccional ..... 9
2.3 Failover automático y degradación elegante del motor cognitivo ..... 12
3.1 Tipo, enfoque y diseño de la investigación ..... 14
3.2 Fases del desarrollo ..... 15
3.3 Fuentes de datos y procedimientos de extracción ..... 17
3.4 Entorno de despliegue y condiciones de reproducibilidad ..... 18
3.5 Criterios de validación y métricas ..... 19
3.6 Consideraciones éticas y tratamiento de datos ..... 20
3.7 Limitaciones metodológicas ..... 21
3.8 El patrón híbrido: modularidad, externalización y frontera de confianza ..... 22
3.9 El orquestador FastAPI: recepción de webhooks y disociación entre respuesta HTTP y cómputo ..... 23
3.10 Estructura de microservicios contenerizados ..... 24
3.11 El motor NLP y el enrutador en cascada de cinco niveles ..... 26
3.12 Externalización del cómputo y huella de infraestructura ..... 27
3.13 Análisis de viabilidad económica y límites operativos ..... 28
3.13.1 Estructura de costos de los tres motores en la nube ..... 28
3.13.2 La política de precios de WhatsApp Cloud API ..... 29
3.13.3 Análisis comparativo: nube frente a infraestructura propia ..... 30

3.14 Justificación de la selección tecnológica: análisis comparativo de proveedores 31
3.14.1 Latencia y arquitectura de hardware ..... 32
3.14.2 Soberanía tecnológica y modelos de pesos abiertos..... 33
3.14.3 Roles estratégicos de respaldo..... 34
3.14.4 Proveedores evaluados y descartados ..... 34
Capítulo 4. Desarrollo: escudos deterministas y cortocircuitos lógicos..... 36
4.1 Cortocircuito lógico: fast-paths deterministas antes de cualquier inferencia ..... 36
4.2 El escudo anti-alucinaciones..... 37
4.3 Blindaje estructural del SQL generado ..... 39
4.4 Privacidad de datos y aritmética de reloj para recordatorios ..... 39
4.4.1 Tres capas independientes de privacidad ..... 39
4.4.2 Aritmética de reloj para recordatorios absolutos y relativos..... 40
Capítulo 5. Resultados, validación y conclusiones..... 43
5.1 Diseño del banco de pruebas ..... 43
5.2 Huella de infraestructura..... 45
5.3 Latencia por nivel de la cascada ..... 45
5.4 Continuidad del servicio ante fallo de proveedor ..... 46
5.4.1 Prueba de carga concurrente ..... 47
5.5 Frescura y trazabilidad del dato extraído de Instagram ..... 48
5.5.1 Justificación del diseño de la prueba..... 48
5.5.2 Extracción de la fecha declarada en el texto, no de la fecha de publicación ..... 48
5.5.3 Propagación de una edición posterior a la ingesta ..... 49
5.5.4 Alcance y limitaciones de esta verificación ..... 50
5.6 Verificación del escudo anti-alucinaciones ..... 51
5.7 Resolución de entidades y corrección fonética..... 52
5.8 Poda dinámica del esquema..... 52
5.9 Cumplimiento de los objetivos específicos ..... 53

5.10 Sobre la eliminación del costo de inferencia mediante cortocircuitos deterministas ..... 54
5.11 La disponibilidad ganada por el enrutador en cascada es, sobre todo, arquitectónica ..... 55
5.12 Sobre la precisión determinista de la generación aumentada con recuperación estructurada. ..... 55
5.13 Sobre la eficiencia de infraestructura y la delegación cognitiva ..... 56
5.14 Cobertura de la capa de similitud léxica sobre el conocimiento institucional ..... 57
5.15 Prueba de concepto: recuperación vectorial sobre los fallos de ordenamiento ..... 61
5.16 Limitaciones reconocidas ..... 63
5.17 Trabajo futuro ..... 64
Referencias ..... 65
Anexos ..... 70
Anexo A. Interfaz del panel administrativo ..... 70
Anexo B. Pruebas de interacción en WhatsApp ..... 72

## TABLAS

**Tabla 1** Fuentes de datos, mecanismo de extracción y periodicidad de actualización .. 17
**Tabla 2** Criterios de validación, instrumento y condición de aceptación ..... 20
**Tabla 3** Servicios contenerizados de Compu IA y su exposición de red..... 24
**Tabla 4** Consumo de memoria RAM medido en producción ..... 28
**Tabla 5** Costos por millón de tokens y condiciones de capa gratuita de los motores configurados ..... 29
**Tabla 6** Estimación de costo mensual de inferencia por volumen de consultas (nivel 1 de la cascada)..... 31
**Tabla 7** Criterios de evaluación de proveedores de inferencia y resultado de la selección ..... 32
**Tabla 8** Composición del banco de pruebas por categoría de consulta ..... 43
**Tabla 9** Latencia observada por nivel del enrutador en cascada ..... 45
**Tabla 10** Resultados de la verificación de frescura y trazabilidad del dato extraído .... 49
**Tabla 11** Verificación del comportamiento ante datos ausentes y contradictorios ..... 51
**Tabla 12** Trazabilidad entre objetivos específicos, evidencia y grado de cumplimiento ..... 53
**Tabla 13** Cobertura del banco de consultas antes y después de la capa de similitud léxica ..... 57
**Tabla 14** Clasificación de las consultas no resueltas por la línea base, según la causa del fallo..... 57
**Tabla 15** Configuraciones evaluadas de la capa de similitud y resultado de la selección ..... 58
**Tabla 16** Estabilidad de la resolución ante variaciones de escritura ..... 58
**Tabla 17** Caracterización del corpus de conocimiento institucional ..... 59
**Tabla 18** Sensibilidad del mínimo de raíces coincidentes, con los demás parámetros fijos ..... 59
**Tabla 19** Resultados de la prueba de estrés dirigida sobre la búsqueda de publicaciones ..... 60
**Tabla 20** Frecuencia de las palabras de mayor coincidencia en los pies de foto de la cuenta ..... 60
**Tabla 21** Latencia por capa de resolución ..... 61
**Tabla 22** Resolución de los fallos de ordenamiento por similitud vectorial..... 61
**Tabla 23** Detalle de los casos no resueltos por similitud vectorial..... 62
**Tabla 24** Similitud de la ficha más cercana en consultas del dominio y ajenas a él ..... 62

## ILUSTRACIONES

**Ilustración 1** Circuito de ingesta de los datos institucionales, desde el reporte oficial hasta las tablas de consulta ..... 18
**Ilustración 2** Arquitectura de despliegue de Compu IA sobre servidor privado virtual compartido, con servicios contenerizados, puertos internos y proveedores de inferencia externos. ..... 23
**Ilustración 3** Arquitectura de despliegue de los servicios contenerizados y del enrutador en cascada de cinco niveles ..... 25
**Ilustración 4** Recorrido de un mensaje por las capas deterministas previas a cualquier inferencia ..... 27
**Ilustración 5** Intercepción del escudo anti-alucinaciones registrada en los registrosdel sistema. ..... 38
**Ilustración 6** Flujo de consentimiento previo a la entrega de datos de contacto y confirmación de recordatorio ..... 42
**Ilustración 7** Evidencia documental de la propagación de una edición posterior a la ingesta. ..... 50
**Ilustración 8** Panel administrativo: selección del motor de inferencia activo ..... 70
**Ilustración 9** Gestión del padrón docente con datos de demostración ..... 71
**Ilustración 10** Editor de información de aulas y carga de imágenes de ubicación ..... 71
**Ilustración 11** Consulta de horario resuelta mediante RAG estructurado ..... 72
**Ilustración 12** Fast-path de ubicación: respuesta con fotografía real del lugar consultado ..... 73

## RESUMEN

El acceso a información académica y logística de la Facultad de Ingeniería en la Carrera de Ciencias de la Computación de la Universidad Católica de Santiago de Guayaquil (UCSG) está distribuido entre varios canales oficiales: el portal académico, documentos de malla curricular, la cuenta institucional de Instagram y el conocimiento del personal de secretaría sobre la ubicación física de aulas y laboratorios. Cada canal cumple su función, pero un estudiante que necesita una respuesta puntual debe saber de antemano en cuál de esos canales buscarla. Este proyecto presenta el desarrollo de Compu IA, un asistente virtual conversacional accesible por WhatsApp, construido para centralizar esa consulta en una sola interfaz.

El sistema combina una arquitectura híbrida compuesta por PostgreSQL con la extensión pgvector, un orquestador desarrollado en FastAPI, un panel administrativo web desarrollado en Next.js y TypeScript, un motor de inferencia local basado en Ollama bajo perfil opcional y procesos automatizados de extracción de información, con un enrutador de inferencia en cascada de cinco niveles (Groq, Gemini, Claude, Ollama con aceleración por GPU y Ollama en CPU), que mantiene el servicio operativo cuando alguno de los proveedores de inteligencia artificial falla o queda sin cuota disponible. La recuperación de información se resolvió mediante una arquitectura de tipo Text-to-SQL en lugar de una búsqueda semántica por similitud vectorial, con el fin de priorizar la exactitud del dato entregado sobre la flexibilidad de la búsqueda. Sobre esa recuperación se implementó un conjunto de validaciones deterministas (filtros de expresiones regulares, verificación estructural del SQL generado y un mecanismo que descarta cualquier respuesta del modelo que contradiga los datos reales recuperados), que reducen la dependencia de la corrección del sistema respecto del comportamiento del modelo de lenguaje.

Se evaluó además la viabilidad económica de la arquitectura, comparando el costo mensual estimado de operación en la nube frente al costo de adquisición y mantenimiento de infraestructura propia con GPU dedicada, y se documentaron las limitaciones del prototipo y las líneas de trabajo futuro.



**Palabras clave:** recuperación aumentada por generación, Text-to-SQL, enrutamiento en cascada, tolerancia a fallos, asistente conversacional, WhatsApp, UCSG.

## ABSTRACT

Academic and logistical information for the Facultad de Ingeniería en la Carrera de Ciencias de la Computación at the Universidad Católica de Santiago de Guayaquil (UCSG) is spread across several official channels: the academic portal, curriculum documents, the institution's Instagram account, and the front desk staff's knowledge of the physical location of classrooms and laboratories. Each channel serves its purpose, but a student looking for a specific answer needs to know beforehand which channel holds it. This work presents the development of Compu IA, a conversational virtual assistant accessible through WhatsApp, built to centralize that lookup in a single interface.

The system combines a hybrid architecture composed of PostgreSQL with the pgvector extension, a FastAPI orchestrator, a web administrative panel built with Next.js and TypeScript, a local inference engine based on Ollama under an optional profile, and automated information extraction processes, with a five-level inference cascade router (Groq, Gemini, Claude, GPU-accelerated Ollama, and CPU-based Ollama), which keeps the service operational when one of the artificial intelligence providers fails or runs out of quota. Information retrieval was resolved through a Text-to-SQL architecture rather than through semantic vector similarity search, in order to prioritize the accuracy of the data delivered over the flexibility of the search. On top of that retrieval layer, a set of deterministic validations was implemented (regular expression filters, structural verification of the generated SQL, and a mechanism that discards any model response contradicting the actual retrieved data), reducing the system's dependence on the language model's behavior for correctness.

The economic viability of the architecture was also evaluated, comparing the estimated monthly cost of cloud operation against the acquisition and maintenance cost of dedicated on-premise GPU infrastructure, and the prototype's limitations and future work were documented.



**Keywords:** retrieval-augmented generation, Text-to-SQL, cascade routing, fault tolerance, conversational assistant, WhatsApp, UCSG.

# INTRODUCCIÓN

La Facultad de Ingeniería en la Carrera de Ciencias de la Computación de la Universidad Católica de Santiago de Guayaquil (UCSG) gestiona su información académica y logística a través de varios canales independientes entre sí: el portal institucional para horarios, documentos descargables para la malla curricular, redes sociales para eventos y comunicados, y la atención presencial para consultas puntuales como la ubicación de un aula. Este proyecto parte de una pregunta concreta: si esa información ya existe y ya se publica, ¿qué haría falta para que un estudiante la obtenga sin necesidad de saber de antemano en qué canal buscarla?

La sobrecarga de información, o infoxicación, es un fenómeno documentado en el entorno universitario latinoamericano: cuando la cantidad de canales y fuentes supera la capacidad del estudiante para procesarlas, el efecto no es más conocimiento sino más confusión y más tiempo perdido buscando dónde preguntar (Ramírez et al., 2026). El fenómeno no es exclusivo de esta Facultad ni de esta universidad: la transformación digital de la educación superior en Ecuador enfrenta un desafío recurrente, la proliferación de canales oficiales de comunicación no resuelve por sí sola el acceso a la información si esos canales permanecen desconectados entre sí (Moyolema Peralvo et al., 2026).

La respuesta que se desarrolla en los capítulos siguientes es un asistente virtual conversacional, accesible por WhatsApp, que centraliza el acceso a esa información dentro de una sola interfaz de mensajería. El proyecto no propone reemplazar los canales existentes, sino añadir una capa de consulta que los complemente y que resuelva, en lenguaje natural, las preguntas que hoy exigen recorrer varias fuentes o esperar la disponibilidad del personal de secretaria.

La decisión técnica central del proyecto fue priorizar la exactitud de la respuesta sobre la flexibilidad de la búsqueda. Un asistente construido sobre un modelo de lenguaje sin restricciones adicionales puede responder con fluidez, pero también puede inventar un dato cuando no lo tiene, y en un contexto donde la respuesta afecta una decisión académica, esa fluidez sin respaldo verificable no es aceptable. Por esa razón, el sistema desarrollado traduce cada pregunta a una consulta sobre la base de datos institucional en lugar de recuperar fragmentos de texto por similitud semántica, y somete tanto esa consulta como la respuesta redactada a un conjunto de validaciones que no dependen del comportamiento del modelo de lenguaje.

El documento se organiza en cinco capítulos. El Capítulo 1 plantea el problema de la dispersión de la información, los objetivos del proyecto y su alcance. El Capítulo 2 desarrolla el marco teórico que fundamenta la elección de una arquitectura de recuperación estructurada (Text-to-SQL) frente a la búsqueda vectorial tradicional, y el diseño del enrutador de inferencia en cascada que sostiene la disponibilidad del sistema. El Capítulo 3 declara primero la metodología seguida (el tipo de investigación, las fases del desarrollo, el entorno de despliegue y los criterios con los que se valida el sistema) y describe después la arquitectura implementada: los microservicios que lo componen, el funcionamiento del enrutador en cascada y la justificación económica y técnica de los proveedores seleccionados. El Capítulo 4 documenta los mecanismos deterministas que validan las consultas generadas y las respuestas entregadas al estudiante. El Capítulo 5 presenta los resultados de la validación del sistema y cierra con las conclusiones del proyecto, sus limitaciones reconocidas y las líneas de continuación posibles.

A lo largo del documento se distingue de forma explícita entre lo que el sistema implementa y ejecuta en producción, y lo que quedó provisionado como capacidad de infraestructura para una etapa posterior (particularmente la búsqueda vectorial sobre pgvector), de modo que cada afirmación técnica corresponda a una decisión verificable en el código del sistema.

## Capítulo 1. Planteamiento del problema

### 1.1 Anatomía del silo informativo institucional

La Facultad de Ingeniería en la Carrera de Ciencias de la Computación de la UCSG difunde su información académica y logística a través de varios canales, cada uno adecuado para el tipo de contenido que gestiona: el portal institucional publica los horarios de clases, la malla curricular se distribuye en documentos descargables, los eventos, charlas y competencias se anuncian en la cuenta oficial de Instagram, y la ubicación física de aulas y laboratorios. Cada canal cumple bien la función para la que fue pensado.

Lo que no existe todavía es un punto único donde esos canales se consulten con la misma pregunta. Un estudiante que necesita saber en qué aula se dicta el paralelo de una materia específica debe saber, de antemano, en cuál de esos canales encontrar esa respuesta. Esa necesidad de conocer la estructura organizativa antes de poder consultarla es, en sí misma, es una oportunidad concreta de mejora: no porque los canales actuales fallen en su propósito, sino porque ninguno de ellos fue diseñado para responder, en lenguaje natural y desde un solo lugar, una pregunta que puede involucrar información de más de una fuente.

### 1.2 Demanda sobre la atención presencial en Secretaría

La atención presencial en Secretaría absorbe una parte considerable de las consultas que, por su naturaleza, se repiten con frecuencia entre distintos estudiantes. Gran parte de estas dudas, como el contacto de un profesor en específico, la ubicación del auditorio o el aula magna, los horarios de exámenes o la fecha de un evento, corresponden a información que ya se encuentra disponible en algún canal digital de la Universidad. Sin embargo, para un estudiante, especialmente si es nuevo, resulta difícil saber exactamente en qué plataforma buscarla.

Este patrón de saturación en la atención presencial no es exclusivo de esta Facultad. El mercado de chatbots aplicados a educación ha crecido de forma sostenida en los últimos años, y estudios sobre instituciones ecuatorianas reportan que la automatización de consultas repetitivas puede desviar entre el 40% y el 70% de las solicitudes que hoy recaen sobre personal administrativo (Moyolema Peralvo et al., 2026). El diagnóstico que seguirá para el caso de esta carrera, entonces, es una instancia local de un problema que la literatura ya describe a escala regional.

Salvo que el estudiante reciba una instrucción explícita que requiera un trámite personal (por ejemplo, "acércate a Secretaría y solicita ese documento" o "pide en Secretaría que te ayuden con esa gestión" ), el resto de sus consultas suelen ser meramente informativas. No obstante, al acudir físicamente para resolverlas, el estudiante depende de una atención basada en personas, cuya capacidad es finita y su disponibilidad está limitada a un horario. Esto significa que dos personas preguntando al mismo tiempo por la ubicación de un evento requieren el doble de atención para resolver una duda idéntica.

Esta dinámica no es una falla del servicio, sino una característica del soporte presencial, pero se traduce en tiempos de espera innecesarios para el estudiante. Automatizar las consultas más repetitivas y predecibles permitiría que el alumno obtenga la información al instante mediante el asistente virtual. A su vez, esto liberaría gran parte de la carga en Secretaría, permitiendo al personal enfocar su tiempo y capacidad de atención en los trámites administrativos que verdaderamente requieren de su intervención directa y criterio humano.

### 1.3 Hacia una interfaz conversacional complementaria

Actualizar el acceso a esta información no exige reemplazar ninguno de los canales existentes: exige añadir, sobre ellos, una interfaz que los consulte por el estudiante. Tres enfoques posibles se consideraron antes de definir el que se desarrolla en este proyecto, y cada uno aporta algo, aunque también deja una necesidad sin cubrir.

Un portal web centralizado sigue exigiendo que el estudiante sepa navegar hasta la sección correcta; reduce la dispersión, pero no elimina la necesidad de conocer la estructura del sitio. Una sección de preguntas frecuentes es útil para información estable, pero requiere actualización manual constante: un horario reasignado que no se actualiza a tiempo puede generar una respuesta desactualizada, y ese riesgo aumenta con cada canal adicional que haya que mantener al día. Un asistente conversacional basado en menús de opciones predefinidas ordena la interacción, pero traslada al estudiante la tarea de encontrar, dentro del menú, la opción que corresponde a su pregunta, en lugar de dejarle formularla con sus propias palabras.

Un asistente conversacional que reciba la pregunta en lenguaje natural y la resuelva directamente contra la información institucional actualizada resuelve, en conjunto, las limitaciones de los tres enfoques anteriores: no exige conocer una estructura de navegación, se actualiza junto con la fuente de datos y no depende de que el estudiante traduzca su duda a un menú cerrado. Ese es el enfoque que se desarrolla en los capítulos

siguientes, con una condición adicional que se fundamenta en el Capítulo 2: la respuesta entregada debe poder verificarse contra el dato real, y no depender únicamente de la fluidez del modelo de lenguaje que la redacta.

## 1.4 Objetivos

**Objetivo general.** Desarrollar un prototipo de asistente virtual inteligente para la centralización de la oferta informativa y los servicios de apoyo académico, utilizando como caso de estudio la Facultad de Ingeniería en la Carrera de Ciencias de la Computación.

**Objetivos específicos.**

1. Automatizar la captura de datos académicos y extracurriculares mediante procesos de extracción para horarios, cronogramas de exámenes y novedades desde las redes sociales oficiales de la Facultad, y complementar esa base con la información de syllabus que el panel administrativo permite cargar bajo una estructura definida a partir de la plantilla oficial de cada materia.
2. Implementar una arquitectura de datos que permita al asistente responder consultas sobre charlas, competencias, eventos y requisitos académicos de esta facultad de forma precisa y contextualizada.
3. Diseñar un módulo de navegación y orientación que proporcione guías visuales y rutas hacia las aulas y laboratorios específicos de la Facultad de Ingeniería en la Carrera de Ciencias de la Computación.
4. Integrar una lógica de gestión de consultas para servir de enlace entre estudiantes e Ingenieros de la facultad de ingeniería en computación, respetando los criterios de privacidad y disponibilidad establecidos por el Ingeniero.
5. Verificar la operatividad técnica y la precisión de las respuestas del asistente en el entorno de mensajería.

## 1.5 Alcance y limitaciones

- **Público objetivo:** El sistema servirá exclusivamente para estudiantes y personas interesadas en la Facultad de Ingeniería en la Carrera de Ciencias de la Computación.
- **Gestión de consultas:** El asistente responderá dudas sobre horarios de clases, fechas de exámenes, eventos de redes sociales, charlas y competencias de la facultad.

- Actualización de datos: los horarios, exámenes y publicaciones de Instagram se actualizan de forma autónoma mediante procesos programados; el contenido de syllabus se incorpora a través del panel administrativo, siguiendo la estructura fija de la plantilla institucional de syllabus, y no mediante un extractor automático adicional.
- **Asistencia espacial:** El asistente enviará fotografías de las aulas y laboratorios de la facultad, incluyendo los pasos para llegar a ellos desde puntos clave.
- **Protocolo de contacto:** El bot solo entregará el número telefónico de un ingeniero si existe autorización previa; en caso contrario, proporcionará el correo institucional y orientará sobre cómo redactar el mensaje correctamente.
- **Entregable final:** Se desarrollará un prototipo funcional basado en la mejor alternativa técnica disponible, evaluando viabilidad y costos de implementación.

**Limitaciones del sistema:**

- El bot no procesará matrículas, no tendrá acceso a calificaciones privadas y no se integrará con sistemas internos de la universidad que requieran credenciales de acceso no públicas.

### 1.6 Descripción de la solución propuesta

Compu IA es un asistente conversacional que atiende por WhatsApp y responde preguntas sobre la Facultad de Ingeniería en la Carrera de Ciencias de la Computación con datos que ya son públicos, pero que hoy están repartidos en cuatro lugares distintos. La solución no crea información nueva: la reúne, la estructura y la deja disponible en el canal donde el estudiante ya está.

- Sincronización de la información pública. Tres procesos de extracción se ejecutan de forma programada cada madrugada y actualizan la base de datos sin intervención manual: uno recoge horarios y paralelos, otro la información de carrera y la malla, y un tercero las publicaciones de la cuenta institucional de Instagram, de donde salen charlas, eventos y comunicados. La consecuencia práctica es que si la coordinación reasigna un aula el martes, el sistema lo sabe el miércoles a primera hora. El contenido de los syllabus, en cambio, no se extrae de forma automática: se carga desde el panel administrativo mediante un analizador determinista que reconoce la estructura fija de la plantilla oficial, y ese mismo código puede invocarse también por línea de comandos para cargas por lote. Esta distinción importa porque el proyecto no automatiza todo el flujo de información por igual: automatiza lo que cambia con

frecuencia y admite una fuente estructurada estable, y deja bajo control humano, vía panel, la incorporación de contenido que exige revisar cada documento antes de publicarlo.

- **Orientación física dentro del campus.** Preguntar dónde queda un aula o un laboratorio no devuelve una descripción escrita, sino la fotografía real del lugar. El sistema mantiene una galería asociada a cada espacio, de modo que un estudiante de primer ciclo pueda reconocer el sitio antes de llegar, en vez de interpretar una indicación textual.
- **Mediación en el contacto con docentes.** El sistema no entrega el número personal de un profesor por defecto. El docente decide, respondiendo él mismo en una conversación previa, si autoriza compartirlo. Si no responde o no autoriza, el asistente entrega únicamente el correo institucional y orienta al estudiante para redactar una solicitud formal. El valor por defecto ante el silencio nunca es compartir.
- **Decisiones técnicas que sostienen lo anterior.** La recuperación de datos se resuelve traduciendo la pregunta a una consulta SQL sobre la base institucional, y no por similitud semántica, porque en un catálogo de materias la diferencia entre "Cálculo I" y "Cálculo III" tiene que ser exacta y no aproximada. La inferencia se organiza en una cascada de cinco proveedores, de modo que la caída de uno degrade la velocidad y no el servicio. Y sobre la salida del modelo opera un conjunto de validaciones escritas en código, no pedidas por prompt, que descartan cualquier respuesta que contradiga los datos efectivamente recuperados.

La extensión pgvector queda provisionada en la base de datos para una etapa posterior, orientada a normativa redactada en prosa, pero no participa del flujo conversacional en producción. Esta distinción entre lo implementado y lo provisionado se mantiene de forma explícita a lo largo de todo el documento.

## Capítulo 2. Marco teórico

### 2.1 Generación aumentada por recuperación: el paradigma y su estratificación

Puesto que un modelo de lenguaje funciona básicamente como un compresor de información con fecha de corte, todo lo que aprendió queda fijo al momento de su entrenamiento. Ninguna reformulación del prompt puede introducir datos que el modelo nunca procesó. Por ejemplo, si la coordinación reasignó el horario de una asignatura anteayer, para el modelo ese cambio es inexistente o, peor aún, susceptible de ser reemplazado por una invención plausible. La generación aumentada por recuperación

responde a esa limitación desacoplando el conocimiento del parámetro y trasladándolo a una fuente externa consultada en tiempo de inferencia. Gao et al. (2023) sistematizan el paradigma en tres componentes indisociables (recuperación, generación y aumentación) y lo estratifican en tres generaciones sucesivas: Naive RAG, donde la recuperación es un único paso seguido de concatenación al prompt; Advanced RAG, con optimizaciones previas y posteriores a la recuperación; y Modular RAG, donde recuperación y generación se descomponen en módulos intercambiables con rutas de control condicionales. Una revisión posterior en español confirma esa misma estratificación y describe RAG como el mecanismo que ancla la respuesta del modelo en fuentes externas verificables antes de generarla (Godoy Viera y Moreiro González, 2025).

Esa tercera categoría es la que describe con precisión una arquitectura híbrida. La hibridación no consiste en agregar componentes, sino en admitir que un único mecanismo de recuperación resulta insuficiente cuando el corpus institucional es heterogéneo.

### 2.2 Generación aumentada con recuperación estructurada: la precisión relacional sobre el dominio transaccional.

A diferencia de un corpus tradicional, un catálogo de materias opera como una tabla relacional estructurada mediante claves primarias, restricciones de unicidad y llaves foráneas que vinculan cada paralelo a un ciclo, y este a una carrera. Por lo tanto, intentar consultar estos datos mediante proximidad geométrica (proyectando la consulta y las filas a un espacio vectorial para luego ordenarlas por distancia) equivale a aplicar una herramienta probabilística a un problema que ya tiene una solución exacta.

Text-to-SQL no es una alternativa exótica a esa proyección: es un campo con más de siete años de desarrollo metodológico consolidado. Spider estableció el punto de comparación multidominio de referencia (10.181 preguntas y 5.693 consultas SQL únicas sobre 200 bases de datos de 138 dominios) y exige que el modelo generalice a esquemas nunca vistos en lugar de memorizar patrones de una sola base (Yu et al., 2018). Que el problema esté tan estudiado respalda tratarlo como ingeniería madura, no como apuesta.

Madurez del campo no equivale a confiabilidad automática del artefacto generado. BIRD expone la brecha entre el laboratorio y la producción: sobre bases de datos grandes y "sucias" (valores faltantes, columnas ambiguas, ruido real), incluso los modelos más capaces caen muy por debajo del desempeño humano en exactitud de ejecución (Li et al., 2023). Esa brecha es, en sí misma, el argumento contra confiar ciegamente en el SQL que produce un modelo: la traducción de lenguaje natural a

consulta relacional puede fallar, y fallar en silencio si nada la audita después de generada. Es la razón formal, más allá de la conveniencia de implementación, por la que Compu IA no ejecuta el SQL del modelo tal cual sale, sino que lo somete a validación estructural determinista antes de tocar la base de datos, según se detalla en el Capítulo 4.

Biswal et al. (2024), en su propuesta TAG, formalizan por qué ni Text-to-SQL puro ni RAG vectorial puro bastan por sí solos: cada uno cubre un subconjunto de preguntas disjunto del otro. Text-to-SQL resuelve con exactitud lo expresable en álgebra relacional; RAG vectorial resuelve con tolerancia semántica preguntas cuya respuesta vive en texto libre. Ninguno de los dos resuelve bien lo que exige razonar semánticamente sobre datos estructurados a la vez. Esa disyunción es el fundamento de que Compu IA necesite, en principio, más de un mecanismo de recuperación, y es también el motivo por el que la extensión pgvector se aprovisionó en la capa de infraestructura desde el diseño inicial (pgvector contributors, 2025): como reserva deliberada para el subconjunto de preguntas que Text-to-SQL no puede cubrir por construcción: reglamentos, resoluciones y normativa redactada en prosa, donde no existe una traducción limpia a una cláusula WHERE.

Esa capacidad está provisionada, no activada. El motor de recuperación que responde hoy cada consulta de Compu IA es exclusivamente relacional; la vía vectorial permanece como extensión evaluada y disponible, no como dependencia del flujo conversacional en producción.

La evidencia aplicada reciente confirma que enriquecer el contexto recuperado no es gratuito. Gurawa y Dharmik (2025), en un estudio sobre sistemas RAG-Text2SQL, cuantifican que aumentar el tamaño del esquema recuperado para alimentar un generador de SQL mejora la cobertura del esquema, pero incrementa el ruido y el riesgo de alucinación, con umbrales de degradación identificables. Por su parte, RAGTruth aporta un corpus etiquetado a nivel de palabra que permite ubicar dónde exactamente un sistema RAG alucina (Niu et al., 2024). Asimismo, Mala et al. (2025) muestran, comparando estrategias de recuperación densa, dispersa e híbrida, que la recuperación puramente semántica es la más vulnerable a esa alucinación, precisamente cuando el corpus contiene entradas cercanas en significado, pero distintas en el dato que importa. Un catálogo de materias es el escenario de manual para esto: "Cálculo I" y "Cálculo III" son vecinos casi indistinguibles en cualquier espacio de representación razonable, el caso exacto donde la búsqueda densa confunde y la resolución determinista de nombres no tiene margen de error. Las revisiones sobre bases de datos vectoriales aplicadas a modelos de lenguaje

coinciden en esta limitación estructural de la representación densa sobre datos ya estructurados (Han et al., 2024; Jing et al., 2024).

La arquitectura de Compu IA no carece de un mecanismo de enrutamiento previo a la recuperación costosa; solo lo implementa con el instrumento más simple disponible. La literatura reconoce el patrón de anteponer un clasificador barato al mecanismo caro: CRUSH4SQL genera un esquema hipotético para recuperar las tablas y columnas reales más relevantes antes de producir el SQL final (Dhingra et al., 2023); SynTQA decide dinámicamente si conviene resolver por Text-to-SQL o por respuesta a preguntas de extremo a extremo sobre tablas (Zhang et al., 2024); HybGRAG despliega un agente crítico que evalúa si la consulta exige recuperación textual, relacional o ambas (Lee et al., 2024). Los tres comparten la misma arquitectura de dos etapas (primero decidir el camino, después ejecutar el mecanismo elegido) que Compu IA implementa con reglas de expresión regular y un índice de nombres canónicos. La diferencia no es de principio sino de presupuesto computacional: donde esas propuestas pagan una invocación adicional de modelo para decidir el camino, Compu IA paga una coincidencia de patrón.

Compu IA puede compararse con otros asistentes virtuales académicos desarrollados recientemente en universidades ecuatorianas, que comparten el problema pero no la infraestructura ni la decisión de diseño central. Cevallos Minalla (2025) construyó, para la Universidad Técnica de Machala, un asistente virtual sobre tecnologías de OpenAI con fine-tuning, bases de datos vectoriales y conversión de voz a texto, alcanzando una precisión del 96,88% en respuestas estrictamente correctas mediante evaluación por matriz de confusión. Esa arquitectura resuelve el acceso a la información delegando la corrección de la respuesta al propio modelo ajustado, lo que implica que un porcentaje de las respuestas, aunque bajo, queda expuesto a la posibilidad de que el modelo invente o desvíe un dato aun después del ajuste fino. Compu IA se aparta de ese camino: en lugar de mejorar la fiabilidad del modelo mediante entrenamiento adicional, elimina al modelo del camino de decisión siempre que existe una vía determinista, y solo lo invoca para traducir la pregunta a una consulta SQL verificable estructuralmente antes de ejecutarse contra la base de datos institucional. La diferencia no es de grado sino de naturaleza: un asistente con precisión del 96% sigue teniendo, por diseño, un 4% de espacio para el error del modelo; un asistente que resuelve por código determinista y solo recurre al modelo para tareas que no admiten una regla fija reduce ese espacio al subconjunto de preguntas donde el propio proyecto declara que la solución no puede ser exacta por construcción. Otros antecedentes ecuatorianos documentan una tendencia

similar de apoyarse en plataformas de terceros para la comprensión del lenguaje: la Universidad Politécnica Salesiana, sede Cuenca, implementó su asistente sobre IBM Watson Assistant (González Arias, 2022), y en el ámbito universitario iberoamericano en general predomina el patrón de recuperación semántica por similitud vectorial sobre documentos institucionales (Calahorro Jiménez, 2024; Torres Troya, 2024). Ese patrón resulta razonable cuando la fuente es normativa redactada en prosa, pero dado que buena parte de las consultas de este proyecto recaen sobre datos tabulares (horarios, paralelos, fechas de examen), la comparación entre ambos enfoques, desarrollada en el apartado 2.2, es la que finalmente justifica que Compu IA no adopte el camino más común en la literatura reciente.

### 2.3 Failover automático y degradación elegante del motor cognitivo

Un RAG estructurado con validación impecable sigue siendo inútil si el modelo que traduce la pregunta a SQL no responde. La dependencia de un proveedor de inferencia en la nube introduce un punto único de fallo cuya tasa de disponibilidad escapa al control del diseñador, y ese riesgo dejó de ser una hipótesis para convertirse en un dato medido. Chu et al. (2025) caracterizan empíricamente los patrones de interrupción de los servicios públicos de modelos de lenguaje a partir de reportes oficiales de incidentes, y documentan una periodicidad semanal y mensual en las fallas, junto con diferencias sistemáticas entre proveedores en su capacidad para aislar un fallo antes de que se propague. El mismo grupo de investigación, con FAILS, publica un marco de trabajo y un conjunto de datos abierto para automatizar esa recolección de incidentes (Battaglini-Fischer et al., 2025), lo que ofrece una metodología replicable con la que este proyecto podría, en trabajos futuros, cuantificar la tasa de fallo real de sus propios proveedores. El incidente de diciembre de 2024 (una tasa de errores superior al 90 % en un proveedor importante, originada por una falla eléctrica en el centro de datos y resuelta tras varias horas) ilustra este modo de falla en su forma más cruda (OpenAI, 2024).

Jin et al. (2025) trasladan a este dominio el vocabulario formal de la tolerancia a fallos adaptativa en la nube: detección de anomalías, aislamiento de recursos degradados, mitigación de sobrecarga. Ese vocabulario nombra con precisión lo que la cascada de Compu IA implementa en código, descrito en el apartado 3.11.

La familia de trabajos sobre *cascading* y enrutamiento de modelos aporta el andamiaje formal de la estructura en cascada, con una diferencia de motivación que conviene explicitar. Chen et al. (2023) fundaron el estudio de la cascada de modelos con

FrugalGPT, mostrando que los precios de las interfaces de programación difieren en dos órdenes de magnitud y que una secuencia aprendida de invocaciones iguala el desempeño del mejor modelo individual a una fracción del costo. Ong et al. (2024) entrenan el enrutador con datos de preferencia humana y obtienen reducciones superiores al doble sin pérdida de calidad. Ding et al. (2024) ajustan dinámicamente el umbral entre un modelo pequeño y uno grande según la dificultad predicha de la consulta. Aggarwal et al. (2023) sustituyen el clasificador por una autoverificación y gobiernan el escalamiento con un proceso de decisión de Markov parcialmente observable. Por su parte, en un *preprint* reciente, Moslem y Kelleher (2026) unifican este cuerpo de trabajo bajo un marco de tres preguntas: cuándo se decide el enrutamiento, con qué información y cómo se computa la decisión. Este marco permite describir cualquier cascada concreta en términos comparables. Los enfoques de ensamblado, que combinan las salidas de varios modelos en lugar de seleccionar uno solo (Jiang et al., 2023), quedan fuera de este diseño, ya que multiplican el costo por consulta en vez de acotarlo.

El eje dominante en esa literatura es el costo. El eje dominante en el failover de Compu IA es la disponibilidad, y el desplazamiento tiene una consecuencia de diseño concreta: el último eslabón de la cascada no puede ser otro servicio externo. Una secuencia compuesta únicamente por proveedores en la nube comparte el modo de falla que pretende mitigar (correlación por caída de red, agotamiento de cuota, incidente de región) y su disponibilidad agregada sigue acotada por la del enlace.

Chu et al. (2024) formalizan, desde la ingeniería de *software* autoadaptativa, el concepto de degradación elegante: una reducción controlada y especificada del nivel de servicio ante condiciones adversas, en lugar de una simple sustitución binaria entre "funciona" y "no funciona". Bajo este marco, los cinco niveles de la cascada de Compu IA no son intercambiables al azar; constituyen una jerarquía donde cada escalón relaja una dimensión secundaria del servicio (como la latencia o la fluidez de la redacción), mientras conserva intacta la dimensión que no puede negociarse: la veracidad de la respuesta. Park et al. (2024), con LlamaDuo, validan formalmente la migración desde modelos de servicio propietarios hacia modelos locales de menor escala como una estrategia reconocida para mitigar la dependencia operativa y la necesidad de conectividad continua a un tercero.

Que el sistema conozca en qué nivel está operando y actúe en consecuencia es lo que L. Dong et al. (2024) describen bajo la taxonomía de observabilidad de agentes: qué

artefactos conviene trazar (decisión de enrutamiento, nivel activo, latencia) para permitir diagnóstico durante el ciclo de vida del sistema, no solo después de una falla.

Esa degradación solo es aceptable bajo una condición, y aquí la cascada se conecta con la mitigación de alucinaciones. Huang et al. (2023) documentan en su taxonomía que la aumentación por recuperación reduce la probabilidad de alucinación, pero no la elimina, y que la auto-evaluación del propio modelo es una garantía frágil. Y. Dong et al. (2024) sostienen, desde el análisis de soluciones de guardrails, que la validación sistemática de entrada y salida debe residir en una capa externa al modelo, con verificación explícita. Manakul et al. (2023) muestran que la detección por consistencia de muestreo es aplicable incluso sobre modelos de caja negra sin acceso a probabilidades internas, aunque a costa de múltiples invocaciones. La lectura conjunta es una restricción de diseño: cuando la validación de la salida es determinista y externa al generador, la corrección deja de depender del nivel de la cascada que respondió. La validación determinista es lo que hace legítimo al failover, porque garantiza que descender un nivel degrade la elocuencia sin degradar la veracidad.

Ningún diseño multi-proveedor es gratuito. Reece et al. (2023) advierten que integrar múltiples proveedores heterogéneos introduce vectores de riesgo propios de la integración, no solo la suma de los riesgos individuales de cada uno; y Yu et al. (2024), en su taxonomía de fallos e inyección de fallos en sistemas de inteligencia artificial, señalan que las herramientas actuales de simulación rara vez capturan con fidelidad los modos de falla que ocurren en producción real.

## Capítulo 3. Metodología y diseño de la arquitectura del sistema

Antes de describir el diseño del sistema, este capítulo explicita cómo se abordó el problema desde el punto de vista metodológico: qué tipo de investigación es, qué fases siguió el desarrollo, sobre qué entorno se desplegó y con qué criterios se validará en el Capítulo 5. La segunda mitad del capítulo describe la arquitectura resultante de ese proceso.

### 3.1 Tipo, enfoque y diseño de la investigación

Este trabajo es una investigación aplicada de carácter tecnológico. No busca contrastar una hipótesis sobre un fenómeno preexistente, sino construir un artefacto que resuelva un problema concreto y verificar después si efectivamente lo resuelve. Esa distinción no es formal: define qué cuenta como resultado válido y qué no.

El diseño metodológico corresponde a la investigación en ciencias del diseño, o design science research, en la formulación de Peffers et al. (2007). Ese marco propone seis actividades encadenadas: identificación del problema, definición de los objetivos de la solución, diseño y desarrollo, demostración, evaluación y comunicación. Es el que mejor describe lo que efectivamente se hizo en este proyecto, porque se partió de una dispersión informativa observada, se derivaron objetivos de solución, se construyó un sistema, se lo puso a funcionar sobre un canal real y se lo sometió a verificación.

Complementariamente, la validación se organizó como estudio de caso único, siguiendo las orientaciones de Runeson y Höst (2009) para investigación empírica en ingeniería de software. El caso es la Facultad de Ingeniería en la Carrera de Ciencias de la Computación de la UCSG, y la unidad de análisis es el asistente Compu IA operando en producción. Adoptar explícitamente el marco de estudio de caso obliga a declarar algo que conviene decir de entrada: los resultados de este trabajo son válidos para este contexto y no se generalizan automáticamente a otra facultad ni a otra institución.

El enfoque es predominantemente cuantitativo en lo que se pudo medir (consumo de memoria, latencia, reducción del tamaño del prompt) y cualitativo en lo que solo se pudo observar (comportamiento del sistema ante la caída de un proveedor, corrección de casos límite documentados durante el desarrollo). Esa mezcla no es una debilidad del diseño: es lo que corresponde cuando el objeto de estudio es un artefacto en funcionamiento y no una muestra poblacional.

### 3.2 Fases del desarrollo

El proyecto se ejecutó en cinco fases, correspondientes a las actividades del marco de Peffers et al. (2007).

**Fase 1. Identificación del problema.** Se levantó el mapa de canales por los que la Facultad publica hoy su información: portal institucional, documentos descargables de malla y homologación, cuenta oficial de Instagram y atención presencial en Secretaría. El resultado de esta fase es el diagnóstico del Capítulo 1.

**Fase 2. Definición de los objetivos de la solución.** A partir del diagnóstico se fijaron los cinco objetivos específicos del apartado 1.4 y el alcance del apartado 1.5, incluida la delimitación explícita de lo que el sistema no haría: matrículas, calificaciones privadas e integración con sistemas internos que exijan credenciales no públicas.

**Fase 3. Diseño y desarrollo.** Comprende la definición del esquema relacional, la construcción de los extractores, la implementación del orquestador y el panel

administrativo, y el diseño del enrutador en cascada. Es el contenido de la segunda mitad de este capítulo y del Capítulo 4. El desarrollo fue iterativo y no lineal: varias decisiones documentadas en el Capítulo 4 no salieron de un diseño previo sino de corregir un comportamiento observado en producción, y así están señaladas en el texto.

**Fase 4. Demostración y evaluación.** El sistema se desplegó sobre el canal real de WhatsApp mediante la interfaz oficial de Meta y quedó operativo en ese canal. Las verificaciones descritas en el apartado 3.5 se ejecutaron sobre ese mismo despliegue, invocando la cadena de resolución contra la base de datos de producción, y sus resultados constituyen la primera mitad del Capítulo 5.

**Fase 5. Comunicación.** Corresponde a este documento y a la sustentación.

### 3.2.1 Levantamiento de necesidades: entrevistas a personal y estudiantes

Como insumo adicional a la Fase 1, se realizó un levantamiento de necesidades mediante entrevistas breves a 30 personas de la comunidad universitaria: personal administrativo y estudiantes de distintos ciclos de la carrera. El propósito no fue obtener una muestra estadísticamente representativa, sino identificar de primera mano qué preguntas se repiten con mayor frecuencia en la atención presencial, para contrastar ese hallazgo empírico con el diagnóstico del apartado 1.2. Las consultas más mencionadas coincidieron, en su mayoría, con las ya cubiertas por el diseño original del sistema: ubicación de aulas, contacto de profesores y, en menor medida, información sobre becas. Sin embargo, las entrevistas revelaron un grupo de preguntas frecuentes que el alcance inicial del proyecto (apartado 1.5) no contemplaba, relacionadas con trámites académicos y financieros: cómo resciliar una materia, si es posible resciliar el semestre completo, cómo solicitar una recalificación, dónde revisar pensiones y deudas, y si es posible reprogramar un examen. Sobre este último punto, la propia recolección de datos permitió precisar la respuesta correcta: la reprogramación de un examen solo procede para el primer parcial y para el examen de mejoramiento, nunca para el segundo parcial, conforme al reglamento estudiantil vigente. Las primeras cuatro consultas (resciliación de materia o de semestre, recalificación y revisión de pensiones) tienen como canal de resolución oficial el portal de Servicios en Línea de la universidad, pero su procedencia y condiciones dependen de lo que establece el reglamento en los artículos correspondientes a exámenes de gracia, reprogramación de exámenes y resciliación de materias. Por tratarse de trámites que exigen una solicitud formal y no una simple consulta informativa, quedan fuera del tipo de pregunta que Compu IA resuelve de forma

determinista (apartado 4.1); el aporte del sistema en estos casos se limita a orientar al estudiante hacia el canal correcto y a citar el artículo del reglamento aplicable, replicando el mismo mecanismo ya usado para otras consultas normativas (apartado 3.8). Este hallazgo se registra como una limitación de alcance reconocida y no como una funcionalidad implementada: el prototipo evaluado en el Capítulo 5 no automatiza estos cuatro trámites, y su incorporación queda propuesta como línea de trabajo futuro en el apartado 5.17.

### 3.3 Fuentes de datos y procedimientos de extracción

El sistema no genera información: la recoge de fuentes que ya son públicas. La Tabla 1 resume el origen de cada conjunto de datos, el mecanismo de obtención y su periodicidad.

**Tabla 1**

*Fuentes de datos, mecanismo de extracción y periodicidad de actualización*

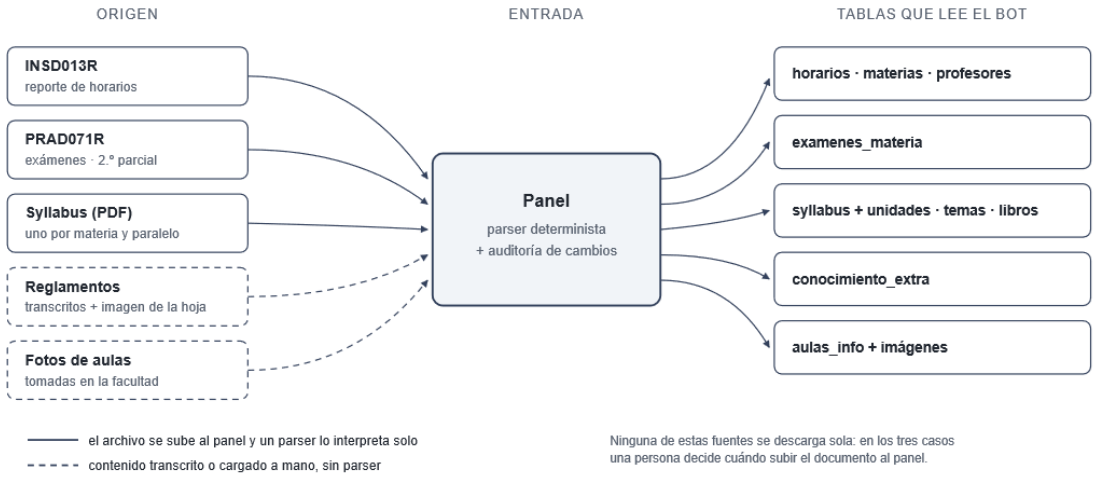
| Conjunto de datos | Fuente | Mecanismo | Periodicidad |
|---|---|---|---|
| Horarios, materias, paralelos | Portal institucional | Extractor automatizado (scraper-horarios) | Diaria, 03:30 |
| Información de carrera y malla | Portal institucional y documentos publicados | Extractor automatizado (scraper-carrera) | Diaria, 03:45 |
| Eventos, charlas y comunicados | Cuenta oficial de Instagram de la Facultad | Interfaz oficial de Instagram (Meta Platforms, 2026b) | Diaria, 04:00 |
| Ubicación de aulas y laboratorios, fotografías | Levantamiento propio en campus | Carga manual por el panel administrativo | Bajo demanda |
| Contactos y consentimiento docente | Declaración del propio docente | Flujo conversacional de consentimiento | Bajo demanda |
| Cronograma de exámenes | Coordinación académica | Carga por el panel administrativo | Por período |

*Nota.* Elaboración propia. Los tres extractores automatizados corren como contenedores de un solo uso disparados por cron: se ejecutan, escriben en la base de datos y terminan.

Es necesario precisar una decisión de diseño sobre la fuente de Instagram. La extracción se realiza a través de la interfaz oficial de la plataforma y no mediante lectura no autorizada del sitio público, lo que evita tanto la fragilidad frente a cambios de maquetado como el incumplimiento de los términos de uso del servicio.

**Ilustración 1**

*Circuito de ingesta de los datos institucionales, desde el reporte oficial hasta las tablas de consulta*




### 3.4 Entorno de despliegue y condiciones de reproducibilidad

Toda medición reportada en el Capítulo 5 corresponde al siguiente entorno, y conviene describirlo con precisión porque varias de las cifras dependen directamente de él.

El sistema opera sobre un Servidor Virtual Privado equipado con 4 vCPU (AMD EPYC) y 7,8 GB de memoria RAM, sin requerir procesamiento gráfico. Este servidor funciona en un entorno de producción real y aloja simultáneamente otros servicios institucionales como correo y DNS. Esta condición obligó a aislar la arquitectura del proyecto y a evitar publicar puertos convencionales para no interferir con procesos existentes, y da mayor validez a las mediciones de rendimiento: el consumo de memoria reportado refleja el comportamiento del sistema bajo carga compartida y no en un entorno de laboratorio aislado.

El sistema combina una arquitectura híbrida compuesta por PostgreSQL con la extensión pgvector, un orquestador desarrollado en FastAPI, un panel administrativo web desarrollado en Next.js y TypeScript, un motor de inferencia local basado en Ollama bajo perfil opcional y procesos automatizados de extracción de información. La orquestación se resuelve con Docker Compose. La exposición pública se delega a un servidor Nginx en el anfitrión, que termina TLS con certificados Let's Encrypt y opera como proxy inverso. El panel administrativo incorpora gestión de cuentas, autenticación multifactor,

auditoría de operaciones, administración de contenidos y herramientas de importación estructurada de syllabus, implementadas dentro de la propia aplicación.

**Condiciones que este despliegue no reproduce, y que quien retome el proyecto debe sustituir.** Este punto merece declararse con claridad, porque afecta la reproducibilidad del trabajo y su eventual adopción institucional.

El servidor y el dominio empleados son recursos personales del autor, reutilizados de proyectos anteriores con el fin explícito de no incurrir en gasto adicional durante el desarrollo del prototipo. Los subdominios que atienden el webhook y el panel administrativo pertenecen a ese dominio personal. En consecuencia, ninguna de las direcciones utilizadas en este trabajo es institucional, y una implementación real dentro de la Facultad exige migrar ambos servicios a infraestructura y a un dominio bajo control de la Universidad. El código no depende de esas direcciones: están parametrizadas por variables de entorno, de modo que la migración es de configuración y no de arquitectura.

La línea telefónica asociada al canal de WhatsApp es igualmente un recurso personal. Se adquirió una línea nueva, distinta de la de uso cotidiano del autor, y está registrada a nombre y con la cédula de identidad del autor, tal como exige el registro de una cuenta en la plataforma de mensajería. Esto tiene una consecuencia que no es técnica sino administrativa y que conviene dejar asentada: el número que atiende hoy a los estudiantes está vinculado a una persona natural y no a la institución. Una implementación institucional debe registrar su propia línea a nombre de la Universidad, tanto para evitar que la continuidad del servicio dependa de una persona que eventualmente se gradúa y se va, como para que la responsabilidad sobre el tratamiento de los mensajes recaiga en la entidad que presta el servicio y no en un particular. Quien retome este trabajo no debería reutilizar la línea del autor bajo ninguna circunstancia.

### 3.5 Criterios de validación y métricas

Verificar un asistente conversacional exige decidir primero qué significa que funcione. Se definieron cinco criterios, cada uno con su instrumento de medición y su condición de aceptación. La distinción entre criterios medidos e indicadores observados es deliberada y se sostiene en todo el Capítulo 5.

**Tabla 2**

*Criterios de validación, instrumento y condición de aceptación*

| Criterio | Qué se verifica | Instrumento | Naturaleza |
|---|---|---|---|
| C1. Exactitud del dato | Que la respuesta entregada coincida con la fila existente en la base institucional | Banco de preguntas con respuesta conocida de antemano | Medido |
| C2. Ausencia de invención | Que ante datos inexistentes el sistema declare la ausencia y no complete con texto plausible | Pruebas dirigidas con premisas falsas | Medido |
| C3. Frescura del dato | Que un cambio en la fuente de origen se refleje en la base tras la siguiente ejecución del extractor | Modificación controlada en la fuente y verificación posterior | Medido |
| C4. Continuidad del servicio | Que la caída de un proveedor degrade la velocidad y no interrumpa la atención | Observación de incidentes reales registrados en la bitácora | Observado |
| C5. Huella de infraestructura | Que el sistema opere dentro de los recursos de un servidor compartido de propósito general | docker stats sobre el despliegue en producción | Medido |

*Nota.* Elaboración propia. La distinción entre criterios "medidos" y "observados" es relevante: C4 no fue sometido a inyección controlada de fallos, de modo que su evidencia es observacional y así se reporta en el apartado 5.4.

### 3.6 Consideraciones éticas y tratamiento de datos

El sistema atiende a personas y maneja datos que las identifican, de modo que corresponde declarar cómo se tratan.

El padrón de acceso almacena únicamente números de cédula, sin nombres asociados. Esa decisión es deliberada: el sistema necesita verificar que quien escribe pertenece a la comunidad de la Facultad, pero no necesita saber quién es, y no almacenar el nombre elimina la posibilidad de construir un perfil por persona. Las métricas de uso se registran de forma anónima, desvinculadas del remitente.

El contacto de docentes se rige por consentimiento expreso otorgado por el propio docente en una conversación previa, nunca por una marca que el administrador asigne en su nombre. Ante la ausencia de respuesta, el valor por defecto es no compartir el número

personal. Los tres mecanismos que implementan esta política se documentan en el apartado 4.4.1.

Las pruebas de validación descritas en el apartado 5.5 se realizaron sobre una cuenta de Instagram personal del autor y no sobre la cuenta oficial de la Facultad. La razón es doble. Metodológicamente, verificar la propagación de cambios exige poder editar y volver a editar una publicación, y el autor no tiene ni debería tener esa capacidad sobre una cuenta institucional. Éticamente, alterar el contenido de un canal oficial de comunicación de la Facultad, aunque fuese de forma transitoria y con fines de prueba, habría afectado a una audiencia real que no consintió participar de un experimento. Usar una cuenta propia resuelve las dos objeciones al mismo tiempo.

### 3.7 Limitaciones metodológicas

Antes de presentar cualquier resultado se declara qué no puede sostenerse con la evidencia recogida. Reconocerlo aquí evita que el Capítulo 5 afirme más de lo que realmente demuestra.

**No hubo inyección controlada de fallos.** La continuidad del servicio ante la caída de un proveedor se observó a partir de incidentes reales registrados en los registros del sistema, no de un experimento diseñado. Yu et al. (2024) documentan que las herramientas de simulación de fallos rara vez reproducen con fidelidad los modos de falla de producción, lo que le da cierto valor a la evidencia observacional, pero no la convierte en medición controlada.

**No hay instrumentación de consumo de tokens.** El registro de uso anota qué motor atendió cada consulta, no cuántos tokens costó. Todas las cifras económicas del apartado 3.13.3 son estimaciones bajo supuestos declarados.

**El alcance evaluado es una sola carrera.** La generalización a la facultad completa está argumentada, no verificada.

**No se midió satisfacción de usuario.** Este trabajo verifica el comportamiento técnico del sistema y no incluye instrumentos de percepción, encuestas ni pruebas de usabilidad con estudiantes. Es una omisión consciente de alcance, no un descuido, y queda señalada como línea de continuación.

**El tamaño de las pruebas de frescura del dato es reducido** en número de casos, aunque no en certeza sobre el mecanismo. Las verificaciones del apartado 5.5 son comprobaciones funcionales dirigidas, diseñadas para exponer los dos riesgos concretos que un extractor de este tipo puede tener (fecha de publicación contra fecha del evento, y propagación de una edición posterior), y en ambos casos el resultado fue el esperado. Lo

que estas pruebas no permiten es traducir ese resultado en una tasa de acierto sobre un volumen alto de publicaciones, porque no fueron diseñadas como una serie estadística: fueron diseñadas para verificar que el mecanismo hace exactamente lo que debe hacer, y eso quedó demostrado.

El modelo de costos depende de la política de precios vigente de Meta para WhatsApp Cloud API, la cual puede cambiar sin previo aviso; el proyecto no incluye un mecanismo de alerta automática ante ese tipo de cambios externos.

### 3.8 El patrón híbrido: modularidad, externalización y frontera de confianza

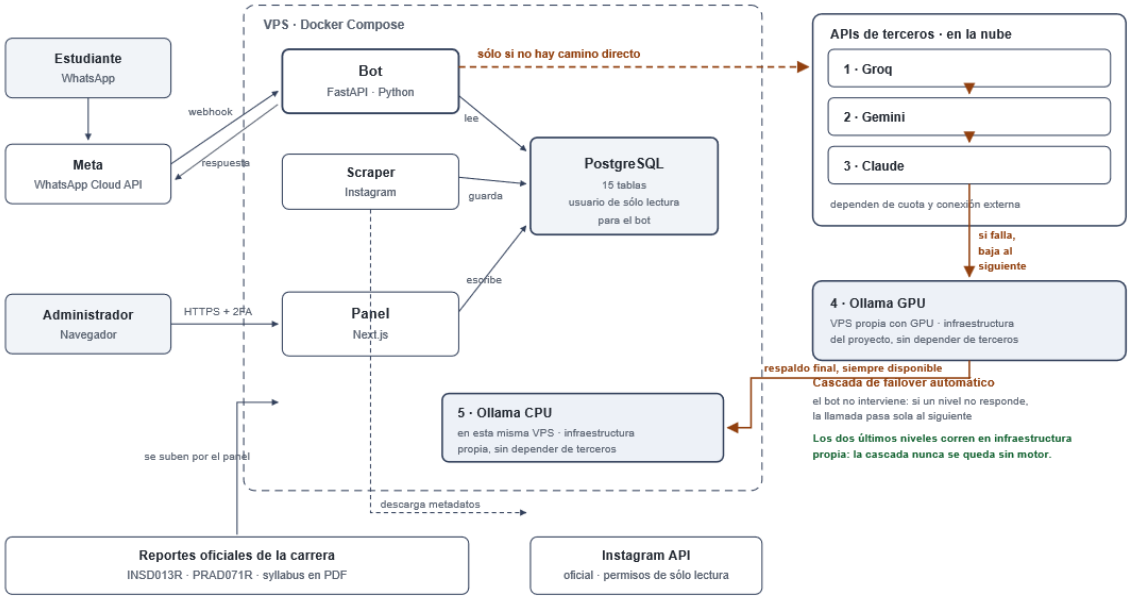
A nivel de arquitectura, Compu IA se aleja del clásico enfoque monolítico que tiene un modelo de lenguaje simplemente integrado. En su lugar, el sistema se diseñó como una composición de cinco servicios contenerizados, donde cada uno asume una responsabilidad única y mantiene su superficie de fallo aislada. Al estar orquestados por Docker Compose y comunicarse de forma exclusiva mediante la red interna, la hibridación opera de manera fluida en dos ejes simultáneos.

El primero es el eje de recuperación de datos, ya fundamentado en el Capítulo 2: Text-to-SQL sobre PostgreSQL como mecanismo primario, exacto por construcción, con pgvector provisionado como reserva de escalabilidad no activada. El segundo eje, objeto de este capítulo, es el de la inferencia: ningún modelo de lenguaje corre dentro del proceso que atiende el webhook. El orquestador FastAPI nunca ejecuta una red neuronal; delega esa carga a un enrutador que decide, en cada invocación, si el cómputo se resuelve en la nube o se externaliza a un motor local. Esa delegación permite que el proceso que sostiene la conversación permanezca liviano con independencia de qué tan pesado sea el modelo que finalmente redacta la respuesta: la arquitectura separa el dónde se atiende la petición del dónde se calcula la inferencia, y esa separación es la que hace posible el offloading analizado en 3.12.

La frontera de confianza no es únicamente perimetral. Se refuerza a nivel de rol de base de datos: el usuario rag_reader, que ejecuta el SQL generado por el modelo, tiene revocado explícitamente el acceso a las tablas padron_cedulas, claves_api, motor_llm, administradores y configuracion_sistema. Un modelo manipulado por inyección de prompt para intentar leer un token de interfaz de programación no lo consigue por permisos de PostgreSQL, no por buena conducta del modelo.

**Ilustración 2**

*Arquitectura de despliegue de Compu IA sobre servidor privado virtual compartido, con servicios contenerizados, puertos internos y proveedores de inferencia externos.*




## 3.9 El orquestador FastAPI: recepción de webhooks y disociación entre respuesta HTTP y cómputo

Meta exige que cualquier webhook confirme la recepción con un código HTTP 200 casi de inmediato; si esa confirmación no llega a tiempo, la plataforma reintenta el envío, y un reintento sobre un sistema que todavía está procesando el mensaje original produce duplicados (Meta Platforms, 2026a). Esa restricción de protocolo gobierna el diseño del endpoint.

El manejador del webhook realiza exactamente dos operaciones de forma síncrona dentro del bucle de eventos de FastAPI: verifica la firma X-Hub-Signature-256, un HMAC-SHA256 del cuerpo crudo de la petición contra el secreto de aplicación de Meta, comparado con una función resistente a ataques de temporización, y, si es válida, delega el payload completo a un ThreadPoolExecutor de cuatro trabajadores. La verificación de firma falla cerrada: si el secreto no está configurado, el webhook rechaza todo en lugar de aceptar payloads sin validar. Ni una consulta a la base de datos se ejecuta antes de confirmar que el remitente es realmente Meta.

El resto del flujo (consulta del usuario, transcripción de audio, clasificación de intención, generación de SQL y redacción final) corre íntegramente fuera del bucle de eventos. La razón es estructural: la mayor parte de ese código es bloqueante por naturaleza (peticiones HTTP salientes, SQLAlchemy síncrono sobre psycopg2), de modo que una sola nota de voz en transcripción congelaba el bucle completo, y ninguna otra petición recibía respuesta mientras tanto. Sacar el trabajo pesado del bucle no acelera el procesamiento de un mensaje individual; garantiza que procesar uno no bloquee la capacidad de aceptar los demás.

El orden de filtrado dentro del procesador es estrictamente secuencial, y cada paso corta el flujo si aplica: limitación de tasa, consulta o registro de usuario, flujo de consentimiento docente, registro bloqueante de estudiante nuevo, comandos explícitos, detección de recordatorio en lenguaje natural, anti-duplicados, verificación del estado de la cascada, y encolado del RAG asíncron

### 3.10 Estructura de microservicios contenerizados

**Tabla 3**

*Servicios contenerizados de Compu IA y su exposición de red*

| Servicio | Imagen / build | Rol | Exposición |
|---|---|---|---|
| db | ankane/pgvector:latest | PostgreSQL con pgvector; único almacén de estado persistente | 127.0.0.1:55432→ 5432 |
| app | Dockerfile raíz | Orquestador FastAPI: webhook, flujos, RAG, cascada | 127.0.0.1:18000→ 8000 |
| panel | panel/Dockerfile | Administración Streamlit: motor activo, modo demo, difusión, aulas | 127.0.0.1:18501→ 8501 |
| ollama | ollama/ollama:latest | Motor local de inferencia; perfil opcional | 127.0.0.1:11434→ 11434 |

| | | | |
|---|---|---|---|
| scraper-* (×3) | /Dockerfile | ETL de un solo uso: horarios, carrera, Instagram | Sin puerto; disparado por cron |

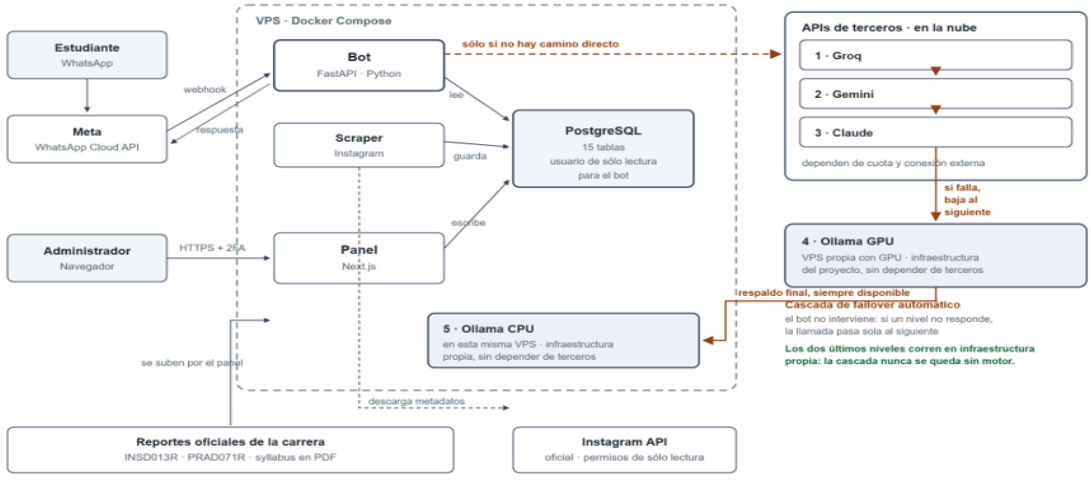
Ningún servicio publica un puerto fuera de la interfaz de bucle local: la exposición pública es responsabilidad exclusiva de un servidor Nginx en el anfitrión, que termina TLS y opera como proxy inverso hacia el puerto interno correspondiente. Los puertos de anfitrión no son los convencionales porque, en el despliegue real, el servidor no está dedicado al proyecto: es un servidor compartido que ya ejecuta correo, DNS y otros sitios sobre los puertos estándar, y evitar la colisión sin alterar servicios preexistentes fue una restricción de despliegue.

El servicio ollama no arranca con una invocación normal de Docker Compose: vive bajo un perfil opcional, de modo que un despliegue sin intención de ejecutar inferencia local no reserva memoria para un contenedor que no va a usar. Los tres servicios de extracción comparten un único Dockerfile y solo difieren en el comando ejecutado; corren una vez, escriben en la base de datos y terminan. Su disparo diario mediante cron es lo que mantiene horarios, información de carrera y publicaciones sincronizados sin intervención manual.

La Ilustración 2 sitúa esos contenedores sobre el servidor y traza las dos rutas que puede seguir una consulta: la resolución directa contra la base de datos y el descenso por la cascada de inferencia cuando ninguna ruta directa aplica.

**Ilustración 3**

*Arquitectura de despliegue de los servicios contenerizados y del enrutador en cascada de cinco niveles*



### 3.11 El motor NLP y el enrutador en cascada de cinco niveles

Toda invocación a un modelo, tanto el que traduce la pregunta a SQL como el que redacta la respuesta, pasa por una única función que abstrae al resto del sistema del proveedor efectivamente usado. Detrás de ella opera el orden canónico de la cascada: **Groq → Gemini → Claude → Ollama GPU → Ollama CPU**.

El orden refleja una jerarquía de costo y latencia crecientes hacia la soberanía: Groq y Gemini como primeras opciones comerciales, Claude como tercer nivel, y dos instancias de Ollama como los dos últimos escalones: una sobre GPU en un servidor remoto acelerado y otra sobre CPU en el propio servidor de aplicación. La quinta posición es la que rompe la correlación de fallo compartida por cualquier secuencia compuesta únicamente por proveedores externos: CPU local no depende de cuota, facturación ni disponibilidad de red hacia un tercero.

Disponibilidad verificada antes de invocar. Una función de comprobación determina, sin ejecutar ninguna inferencia, si hay clave vigente para los proveedores comerciales o si el sondeo de salud de la instancia de Ollama correspondiente respondió dentro de la ventana de caché. Un nivel no disponible ni siquiera se intenta.

Manejo de fallos: circuit breaker genérico. Cada intento está envuelto en un manejador de excepciones de propósito general. El sistema no inspecciona el código de estado HTTP para distinguir un 429 (límite de tasa excedido) de un tiempo de espera agotado o una clave revocada: cualquier excepción que el proveedor propague, y una respuesta 429 se propaga como excepción, dispara el mismo camino. Se marca el motor como caído durante 60 segundos y la ejecución continúa al siguiente nivel dentro de la misma petición del usuario, sin reintentar el que acaba de fallar y sin que el estudiante perciba el fallo intermedio. Esa generalidad es una fortaleza: cubre límite de tasa, tiempo de espera, error del servidor y cuota agotada bajo un único mecanismo de reacción.

El sondeo de salud de cada instancia de Ollama se cachea **30 segundos**, de modo que una ráfaga de mensajes no produce una ráfaga equivalente de verificaciones de red. El orden efectivo, además, no está fijo en tiempo de compilación: el panel administrativo permite promover manualmente un proveedor a prioridad 1, y esa elección se relee con una caché de vida corta sin reiniciar el contenedor.

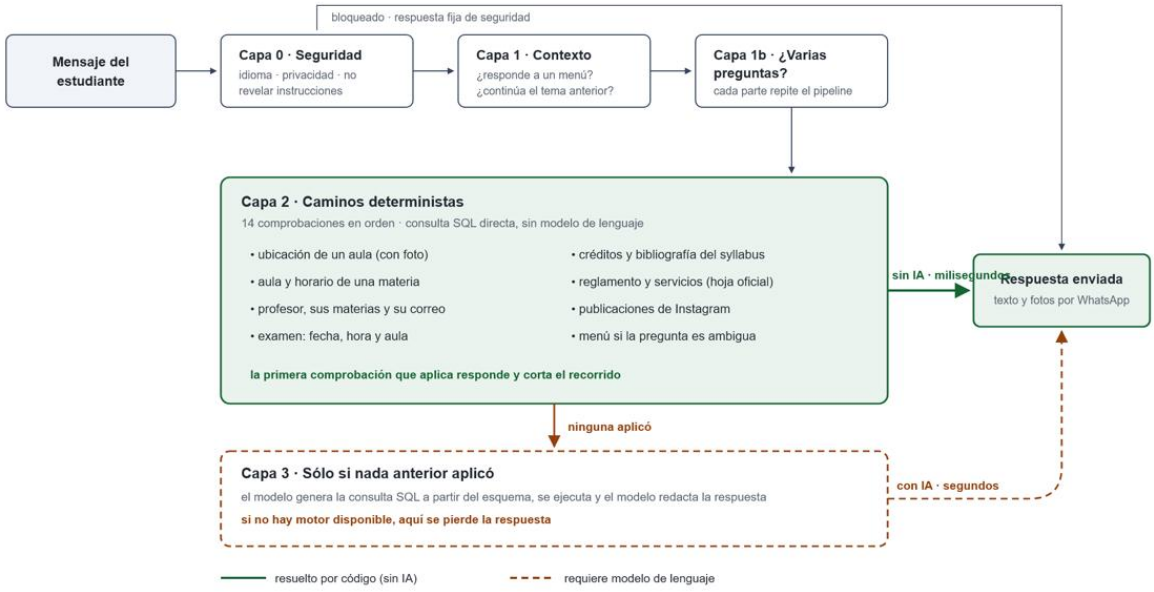
El sistema conoce su propio nivel de degradación antes de encolar trabajo. Una función de predicción determina, sin inferir, a qué nivel caería la próxima consulta. Si el único disponible es CPU local, se advierte al estudiante de una demora mayor y se dispara una alerta administrativa; si no queda ningún nivel, la alerta se dispara igual. Esa alerta

está limitada a una notificación cada 12 horas, reclamada mediante una actualización condicional atómica sobre una fila única: si dos procesos detectan la degradación simultáneamente, solo uno consigue emitir el aviso.

El costo del último escalón está medido, no estimado: una consulta aislada contra Ollama en CPU tarda entre 20 y 90 segundos; con dos consultas concurrentes compitiendo por los mismos núcleos, cada una se vuelve varias veces más lenta y el tiempo total resulta peor que procesarlas en serie, evidencia de que la inferencia en CPU está limitada por cómputo, no por entrada/salida. Esa medición fija, aguas abajo, el diseño de la cola: dos hilos de trabajo y un tope de 20 consultas pendientes, donde la persona en la posición N espera aproximadamente N × 25 segundos.

**Ilustración 4**

*Recorrido de un mensaje por las capas deterministas previas a cualquier inferencia*




La Ilustración 3 ordena ese recorrido. Cada comprobación determinista que aplica corta el flujo y responde; solo un mensaje que no encaja en ninguna llega al modelo de lenguaje.

### 3.12 Externalización del cómputo y huella de infraestructura

La Ilustración 4 resume el circuito completo de los datos descritos en el apartado 3.3, desde el documento oficial hasta las tablas que el asistente consulta en cada respuesta.

Ninguna de esas fuentes se descarga sola: en todos los casos una persona decide cuándo cargar el documento en el panel.

Ningún proceso de la aplicación carga jamás un modelo de lenguaje en su propia memoria. Cuando el motor activo es un proveedor comercial, la inferencia ocurre íntegramente fuera de la infraestructura del proyecto: el contenedor abre una conexión saliente, envía el prompt y espera texto; ningún peso de modelo reside en el servidor propio.

Medido en producción mediante docker stats sobre el despliegue real (4 vCPU, 7,8 GB de memoria total, compartido con correo, DNS y otros sitios del mismo operador), los tres contenedores que sostienen el servicio consumen en conjunto 350,74 MiB de memoria RAM:

**Tabla 4**

*Consumo de memoria RAM medido en producción*

| Contenedor | Rol funcional | CPU (%) | RAM (uso) | RAM (%) |
|---|---|---|---|---|
| universidad_app | API REST y orquestación NLP/RAG | 0.41% | 125 MiB | 1.57% |
| universidad_panel | Panel administrativo | 1.10% | 179.2 MiB | 2.26% |
| universidad_db_vector | Base de datos (PostgreSQL + pgvector) | 0.00% | 46.54 MiB | 0.59% |
| Total atribuible al sistema | | 1.51% | 350.74 MiB | 4.42% |

**Nota:** Medición tomada con docker stats --no-stream el 27/07/2026 (04:00:57 UTC), sobre el despliegue en producción en estado de reposo (sin consultas en curso). El consumo bajo carga concurrente no fue instrumentado.

Frente a los cerca de 8 GB que exigía mantener Ollama cargado localmente antes de migrar el motor primario a la nube, la diferencia de más de un orden de magnitud no proviene de optimizar el software del bot, sino de trasladar el costo de memoria del modelo al proveedor que lo sirve. El offloading no reduce el trabajo computacional que exige responder una pregunta: lo traslada fuera del perímetro que el proyecto debe aprovisionar, pagar y escalar.

### 3.13 Análisis de viabilidad económica y límites operativos

#### 3.13.1 Estructura de costos de los tres motores en la nube

La cascada no invoca modelos genéricos sino modelos específicos, configurados por defecto en el propio código, y es sobre esos modelos que corresponde calcular el costo real.

**Tabla 5**

*Costos por millón de tokens y condiciones de capa gratuita de los motores configurados*

| Nivel | Motor / modelo | Input (USD/MTok) | Output (USD/MTok) | Capa gratuita |
|---|---|---|---|---|
| 1 | Groq, gpt-oss-120b (rol SQL) | 0,15 | 0,60 | Sin capa a costo cero; el límite es de cuota (30 RPM / 1K RPD / 8K TPM / 200K TPD), no de facturación (Groq, 2026a, 2026b) |
| 1 | Groq, gpt-oss-20b (rol redacción) | 0,075 | 0,30 | Misma condición; cuota más holgada (500K TPD) |
| 2 | Gemini, gemini-3.6-flash | 1,50 | 7,50 | Sí; capa gratuita a costo cero acotadas por solicitudes diarias. El contenido de la capa gratuita se usa para mejorar los productos del proveedor (Google, 2026) |
| 3 | Claude, claude-opus-4-8 | 5,00 | 25,00 | Ninguna; facturación por token desde la primera solicitud (Anthropic, 2026) |

La tabla expone la asimetría que el propio orden de la cascada ya anticipaba: Groq es, con margen, el motor más barato por token, diez veces menos que Gemini en entrada y más de treinta veces menos que Claude en salida, y ocupa el primer nivel. La cascada no solo resuelve disponibilidad: en el orden en que está escrita, también resuelve costo, degradando hacia proveedores más caros únicamente cuando el más barato no responde.

### 3.13.2 La política de precios de WhatsApp Cloud API

Desde julio de 2025, Meta factura exclusivamente mensajes de plantilla; cualquier mensaje de texto o imagen libre enviado dentro de una ventana de servicio abierta por el

propio usuario es gratuito (Meta Platforms, 2026c). Esa ventana se abre automáticamente cada vez que el estudiante escribe y permanece abierta 24 horas desde su último mensaje.

Compu IA nunca sale de esa ventana en su flujo conversacional. La función de envío invocada desde el webhook, desde el procesador asíncrono, desde los recordatorios y desde las alertas administrativas construye exclusivamente payloads de tipo texto e imagen. La función capaz de enviar plantillas facturables existe, pero tiene un único punto de invocación en todo el proyecto: la difusión masiva administrativa del panel, una acción deliberada del operador y no un paso del pipeline conversacional. En términos de costo de mensajería, la atención a cada estudiante individual es estructuralmente gratuita.

Esa gratuidad no es accidental: es una restricción impuesta activamente. El subsistema de recordatorios rechaza en tres niveles independientes cualquier solicitud que exigiría entregar un mensaje fuera de la ventana de 24 horas. Compu IA sacrifica deliberadamente una función de producto, recordatorios a más de un día, para permanecer dentro del segmento que Meta no factura.

Esta condición de gratuidad depende enteramente de la política vigente de Meta y no está garantizada a futuro. Meta ha modificado su modelo de precios de WhatsApp Cloud API en el pasado (incluyendo el cambio de julio de 2025 que elimina el cobro por conversación y lo traslada a plantillas), por lo que una revisión periódica de los anuncios oficiales del proveedor es necesaria para verificar que la arquitectura de costos descrita en este apartado siga vigente en producción.

### 3.13.3 Análisis comparativo: nube frente a infraestructura propia

No existe en el código ningún contador de tokens consumidos; el registro de uso anota qué motor atendió cada consulta, no cuántos tokens costó. Cualquier cifra mensual es, por tanto, una estimación bajo supuestos declarados, no una medición de producción.

Tomando un presupuesto deliberadamente generoso por consulta resuelta, esquema podado, reglas y pregunta en el paso de SQL (≈1.200 tokens de entrada, ≈80 de salida) más pregunta y resultados formateados en el paso de redacción (≈400 de entrada, ≈150 de salida), el costo de una consulta resuelta íntegramente por el nivel 1 es de aproximadamente 0,0003 USD.

**Tabla 6**

*Estimación de costo mensual de inferencia por volumen de consultas (nivel 1 de la cascada)*

| Escenario | Consultas/día | Costo mensual estimado (USD) |
|---|---|---|
| Bajo (carrera piloto) | ≈ 50 | ≈ 0,45 |
| Medio | ≈ 500 | ≈ 4,50 |
| Alto (facultad completa) | ≈ 1.000 | ≈ 9,00 |

Incluso asumiendo que una fracción significativa degrade a Gemini o Claude, el gasto total se mantiene muy por debajo de los 50 USD mensuales en cualquier escenario razonable de adopción de una sola carrera.

Optar por una alternativa local implica cambiar la métrica de costo: en lugar de pagar una factura mensual, se inmoviliza capital antes de procesar la primera consulta. Para dimensionarlo, una tarjeta gráfica de consumo con memoria suficiente para servir modelos con latencia aceptable (24 GB de VRAM) cuesta entre 700 y 2.000 USD, y una tarjeta de servidor de 40 a 80 GB multiplica esa cifra varias veces. A ese desembolso inicial deben sumarse rubros que ninguna interfaz comercial factura: el chasis, la fuente de poder, la refrigeración, la depreciación del equipo y el consumo eléctrico. Este último ronda los 350 a 450 vatios bajo carga sostenida y se acumula existan o no consultas, a diferencia del costo por token, que baja a cero sin tráfico. Tampoco entra en la comparación el tiempo del personal que debe mantener y monitorear ese servidor, que en una facultad rara vez es tiempo disponible.

El diseño actual no descarta esta vía: la reserva. Los niveles 4 y 5 de la cascada ya están implementados y probados; lo que falta para activarlos a escala no es ingeniería, es exclusivamente el capital de la GPU dedicada.

### 3.14 Justificación de la selección tecnológica: análisis comparativo de proveedores

Ningún proveedor entra a la cascada por popularidad de marca. Cada uno ocupa su nivel porque resuelve una restricción de ingeniería que los demás dejan abierta. Los tres apartados siguientes desarrollan los criterios que ordenaron esa selección (latencia, soberanía tecnológica y rol de respaldo), y el apartado 3.14.4 documenta los proveedores que se evaluaron y quedaron fuera, entre ellos OpenAI y Grok, cuya ausencia responde a una decisión argumentada y no a una omisión. La Tabla 7 resume el resultado del proceso.

**Tabla 7**

*Criterios de evaluación de proveedores de inferencia y resultado de la selección*

| **Proveedor** | **Criterio decisivo evaluado** | **Pesos del modelo** | **Rol asignado** | **Motivo de la decisión** |
|---|---|---|---|---|
| Groq | Latencia hasta el primer token | Abiertos (familia gpt-oss) | Nivel 1 | Hardware de inferencia determinista (Abts et al., 2020); ≈0,6 s medidos en el despliegue |
| Gemini | Costo marginal y disponibilidad | Cerrados | Nivel 2 | Capa gratuita acotada por solicitudes diarias (Google, 2026), que funciona como colchón económico |
| Claude | Adherencia a instrucciones complejas | Cerrados | Nivel 3 | Último recurso comercial antes de la inferencia local (Anthropic, 2026) |
| Ollama (GPU) | Soberanía del dato | Abiertos | Nivel 4 | Elimina el envío de prompts a terceros; implementado y probado, sin GPU en el despliegue actual |
| Ollama (CPU) | Independencia de red y de cuota | Abiertos | Nivel 5 | Rompe la correlación de fallo de los niveles 1 a 3; 20 a 90 s por consulta |
| OpenAI | Latencia y pesos del modelo | Cerrados | **Descartado** | No aporta ninguna dimensión que otro nivel no cubra ya, y sus pesos no son públicos |
| Grok | Costo y estabilidad de términos | Cerrados | **Descartado** | Sin capa gratuita utilizable y con condiciones comerciales de menor estabilidad para una institución educativa |
| Mistral / DeepSeek | Pesos abiertos | Abiertos | **Descartado como proveedor** | Sus modelos abiertos son ejecutables en los niveles 4 y 5; contratar además su servicio en la nube duplicaría la dependencia sin agregar tolerancia a fallos |

*Nota.* Elaboración propia a partir de las condiciones publicadas por cada proveedor y de las mediciones descritas en el apartado 3.14.1, con las mediciones del apartado 5.3.

### 3.14.1 Latencia y arquitectura de hardware

WhatsApp no tolera el patrón de streaming progresivo al que un usuario de navegador está acostumbrado: el estudiante espera una respuesta completa en una ventana

que no muestra texto apareciendo palabra por palabra. En ese contexto, el *Time-to-First-Token* deja de ser una métrica matizable y se convierte en el techo de latencia percibida del sistema completo.

Groq resuelve esa restricción desde el hardware, y conviene explicar de dónde sale la diferencia. Su LPU (Language Processing Unit) desciende del Tensor Streaming Processor descrito por Abts et al. (2020): un diseño de silicio con planificación estática y ejecución determinista, en el que el compilador conoce de antemano el grafo completo de cómputo y elimina toda la orquestación dinámica que una GPU de propósito general debe resolver en tiempo de ejecución. Los autores eliminan deliberadamente del hardware los elementos reactivos, árbitros y cachés incluidos, y ese es justamente el origen de la latencia predecible. La misma línea de trabajo reporta latencias de cola muy estrechas sobre modelos tipo transformador (Abts et al., 2022).

Lo que interesa acá no es la cifra del fabricante sino la medida en el propio despliegue: sobre Compu IA, una consulta resuelta en el nivel 1 devuelve respuesta en aproximadamente 0,6 segundos, frente a los 5 a 10 segundos que tomaba esa misma consulta sobre inferencia en GPU convencional. Casi un orden de magnitud. Para el primer nivel de una cascada donde cada consulta espera respuesta síncrona, ese es el criterio que decide qué proveedor puede ocupar la posición 1, y no la reputación de la marca.

### 3.14.2 Soberanía tecnológica y modelos de pesos abiertos

Una interfaz cerrada no vende solo inferencia: vende una dependencia sin salida. Si el proveedor sube el precio, depreca el modelo integrado o cambia los términos de uso de los datos enviados, la institución no tiene camino de continuidad, porque el modelo detrás de esa interfaz es propiedad exclusiva del proveedor y sus pesos nunca se publican.

Groq no vende un modelo propio: vende infraestructura de inferencia para modelos cuyos pesos son públicos. Los modelos configurados por defecto pertenecen a la familia de pesos abiertos, descargable y ejecutable fuera de cualquier interfaz propietaria. Esa es la propiedad que ninguna interfaz cerrada ofrece: si el proveedor cambiara sus condiciones de forma inaceptable, el modelo puede desplegarse sobre infraestructura propia sin autorización de terceros. La dependencia es de infraestructura de cómputo, no de propiedad intelectual del modelo, y esa distinción es exactamente lo que separa a Compu IA de un sistema con dependencia irreversible de proveedor.

Conviene declarar una asimetría real del estado actual: el motor local de Compu IA está configurado con una familia de modelo distinta de la que ejecuta el proveedor de nivel 1. Migrar hoy no sería mover el mismo modelo a otro motor, sino cambiar de familia al mismo tiempo que de proveedor. El argumento de soberanía sigue siendo válido como propiedad arquitectónica, ambas familias son de pesos abiertos y el catálogo del proveedor incluye familias equivalentes, de modo que la continuidad exacta es alcanzable con un cambio de configuración y no de arquitectura, pero afirmar que ya ocurre sin fricción sería impreciso.

### 3.14.3 Roles estratégicos de respaldo

Gemini ocupa el segundo nivel por una combinación que ningún otro proveedor comercial replica al mismo costo: alta disponibilidad de infraestructura y una capa de uso gratuito con costo cero por token dentro de un límite de solicitudes diarias (Google, 2026), condición de sostenibilidad que una institución educativa con recursos acotados puede aprovechar como colchón económico real.

Claude ocupa el tercer nivel, el último proveedor comercial antes de caer a inferencia local. No entra ahí por costo: entra por capacidad de razonamiento. Cuando el sistema ya perdió sus dos opciones más rápidas y económicas, la consulta que llega a Claude es, casi por definición, una que ya generó fricción en algún punto anterior: el nivel donde se justifica pagar más por un modelo con mayor capacidad de seguir instrucciones sin desviarse. Corresponde precisar el alcance del argumento: la corrección final de cualquier respuesta no depende de qué tan bien razone el modelo que la redactó, sino de los escudos deterministas del Capítulo 4, que auditan la salida sin importar el proveedor. La capacidad de Claude no garantiza que la respuesta sea correcta; reduce la probabilidad de que necesite ser descartada y sustituida por el formato de respaldo.

### 3.14.4 Proveedores evaluados y descartados

Enumerar los proveedores elegidos sin explicar los rechazados deja el argumento a medias, así que corresponde cerrarlo.

OpenAI fue el caso más discutido, y no se descartó por calidad. Se descartó porque, evaluado contra los tres criterios que ordenan la cascada, no gana ninguno: en latencia no alcanza a la inferencia sobre hardware dedicado, en costo marginal no compite con una capa gratuita, y en soberanía queda por debajo de cualquier alternativa de pesos

públicos, porque los modelos detrás de su interfaz no se publican. Un cuarto nivel comercial que no aporta una dimensión nueva no agrega tolerancia a fallos: agrega superficie de dependencia.

Grok se evaluó por su costo aparente y quedó fuera por dos motivos. No ofrece una capa gratuita aprovechable en el rango de volumen de este proyecto, y sus condiciones comerciales han cambiado con una frecuencia que una institución educativa, que planifica por período académico y no por trimestre, no puede absorber sin riesgo.

Mistral y DeepSeek merecen una precisión distinta, porque no se descartaron sus modelos: se descartó contratarlos como servicio. Sus pesos son públicos, de modo que ya son candidatos legítimos para los niveles 4 y 5 de la cascada con un solo cambio de configuración. Sumarlos además como proveedores en la nube duplicaría la dependencia de red sin romper ninguna correlación de fallo nueva, que es justamente lo que la cascada busca evitar.

El criterio general, entonces, es que ningún nivel se agrega por prestigio. Un nivel entra a la cascada solo si cierra un modo de falla que los anteriores dejan abierto. Bajo esa regla, tres proveedores comerciales y dos motores locales bastan, y un cuarto proveedor comercial sería redundancia sin cobertura adicional.

## Capítulo 4. Desarrollo: escudos deterministas y cortocircuitos lógicos

El diseño de Compu IA invierte la relación habitual entre modelo y regla. En este sistema, la regla decide primero, y el modelo solo interviene donde la regla no alcanza. Este capítulo documenta los mecanismos que materializan esa inversión y que sostienen, en la práctica, la restricción de diseño establecida al final del Capítulo 2: que la veracidad no dependa del nivel de la cascada que respondió.

### 4.1 Cortocircuito lógico: fast-paths deterministas antes de cualquier inferencia

No toda pregunta necesita un modelo de lenguaje para resolverse, y confundir "necesita lenguaje natural para formularse" con "necesita un modelo para responderse" es el error que esta capa corrige. La pregunta por la ubicación de un aula tiene una respuesta que vive íntegramente en una fila de la tabla de aulas: no hay ambigüedad semántica que resolver, solo un identificador que extraer y una fotografía que adjuntar. Tratarla como candidata a dos rondas de inferencia, una para traducirla a SQL, otra para redactar, es gastar segundos en un problema que una expresión regular resuelve en microsegundos.

El fast-path de ubicación reutiliza deliberadamente el patrón de verbos del clasificador de intención, para no mantener dos inventarios distintos de expresiones como "cómo llego", "dónde queda" o "en qué edificio". Sobre ese primer filtro, tres patrones adicionales desambiguan el tipo de lugar: uno extrae el número de aula o laboratorio tolerando ambos órdenes de palabras; otro reconoce el auditorio y la asociación de estudiantes como casos especiales; y un tercero resuelve nombres no numéricos comparando contra la tabla real con tolerancia a conectores gramaticales y un diccionario de sinónimos institucionales.

La iteración que llevó a este diseño está documentada en el propio código. Antes, resolver una ubicación dependía de que el modelo de SQL generara la consulta correcta *y* de que el modelo de redacción mencionara el identificador exacto en su texto, para que un tercer mecanismo pudiera engancharlo por expresión regular después: tres eslabones probabilísticos para una operación que es, en esencia, una búsqueda por clave. La versión actual resuelve todo en código y construye el texto mencionando el identificador exacto tal como figura en la base de datos, no por naturalidad, sino porque el adjuntador de imágenes escanea ese texto con una coincidencia literal para decidir qué fotografía enviar. La coincidencia se busca primero en la pregunta del estudiante y solo se recurre a la respuesta generada cuando la pregunta no produjo ninguna imagen: ajuste posterior a un falso positivo real de producción, donde preguntar por las autoridades enganchaba de

rebote la fotografía de un edificio porque el texto de esa respuesta mencionaba el nombre de la facultad correspondiente.

Lo que recibe el estudiante es una fotografía real del lugar, servida como archivo estático y adjuntada al mensaje como contenido multimedia. Ningún modelo la generó ni la describió. Lo que cambió con esta capa no es cómo se sirve la imagen, sino cuántas invocaciones median entre la pregunta y el envío: pasaron de dos a cero.

### 4.2 El escudo anti-alucinaciones

La defensa central de Compu IA contra la alucinación no consiste en pedirle al modelo que no invente. Consiste en no darle la oportunidad cuando la verdad ya está disponible en código, y en descartar por completo su salida cuando la contradice.

Primera intercepción: nunca se interroga al modelo sobre un vacío. Cuando la consulta SQL devuelve cero filas, el sistema no invoca al modelo de redacción para que explique la ausencia: corta antes, con un mensaje fijo. La razón es empírica, no teórica: se comprobó en pruebas en vivo que, si el estudiante afirma algo falso como si fuera un hecho establecido, el modelo de redacción a veces *confirma* la afirmación en lugar de negarla, el modo de falla que Huang et al. (2023) describen dentro de su taxonomía de alucinación. Cortar la invocación en código hace ese modo de falla estructuralmente imposible: no hay generación que corromper si no hay generación. Antes de rendirse, el sistema intenta una última corrección determinista: si la búsqueda fue sobre nombres de docentes y no coincidió con nada, reintenta una vez con corrección fonética, cubriendo el caso de que la ausencia sea un error de transcripción de audio y no una ausencia real de datos.

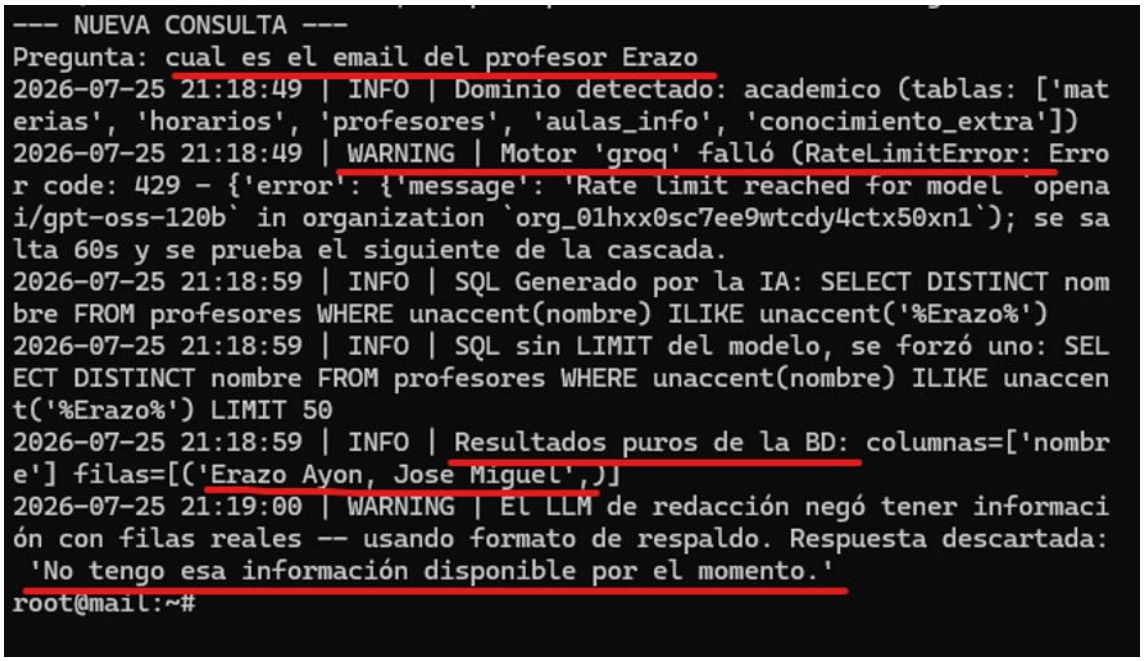
Segunda intercepción: la salida se audita después de generarse y se descarta por completo si falla. Cuando el conjunto de resultados no está vacío y se invoca al modelo de redacción, el texto devuelto pasa por un filtro que detecta patrones de negación. Si el modelo genera una negativa a pesar de contar con los datos, la respuesta se descarta íntegramente. En lugar de aplicar parches o correcciones parciales, el texto se sustituye por un formato construido en código utilizando las mismas filas que el modelo tuvo a su disposición y no supo procesar. Esta decisión de diseño es clave, ya que el escudo de validación no confía en la estructura del modelo, sino en la presencia real del dato en la fuente. En definitiva, la existencia del dato primario siempre tiene la razón frente a las alucinaciones del modelo.

Dos variantes completan el escudo. El modelo de SQL puede emitir un valor centinela cuando detecta que la pregunta es demasiado vaga para resolverse sin volcar una tabla completa; se corta ahí, sin gastar el modelo de redacción en sintetizar un volcado que nunca debió generarse. Y cuando la pregunta solicita contacto de forma explícita, el modelo de redacción se omite por completo: la capa determinista construye la tarjeta real, y dejar que el modelo redacte en su lugar abre la puerta a que invente un cierre de cortesía que derive al estudiante a otra instancia en lugar de entregar el dato que el sistema ya localizó.

Esta arquitectura es la implementación concreta del argumento de Y. Dong et al. (2024) sobre *guardrails*: la validación sistemática de la salida debe residir en una capa externa al modelo, con verificación explícita, y no en la instrucción de prompt que el modelo puede incumplir de forma intermitente.

**Ilustración 5**

*Intercepción del escudo anti-alucinaciones registrada en los registrosdel sistema.*




**Nota.** La fecha cuando se realizó esta consulta el modelo de redacción recibió únicamente el nombre del profesor (la tabla consultada no tenía en aquel momento el correo), y al no encontrar el dato específico que se le pidió, generó una negación genérica que hubiera confundido al estudiante haciéndole creer que el profesor no existe en el sistema. El escudo determinista detectó ese patrón de negación pese a que sí había una fila real, descartó la respuesta del modelo, y la reemplazó por el mensaje de respaldo sin

necesidad de saber la causa exacta del error del modelo, solo la evidencia de que había datos reales que no debían negarse.

### 4.3 Blindaje estructural del SQL generado

El SQL que produce el modelo se somete a cuatro controles independientes de su voluntad. Debe comenzar con una sentencia de selección; no debe contener punto y coma interno que permitiría apilar una segunda sentencia arbitraria en la misma llamada al conector de base de datos; no debe contener ninguna palabra clave de escritura o de definición de datos; y no debe presentar ambigüedad de precedencia entre conjunción y disyunción al mismo nivel de paréntesis dentro de la cláusula de filtrado, patrón que en la práctica hace que un filtro obligatorio se anule por una disyunción posterior. Si la consulta no trae límite de filas, se le agrega uno en código: no se le pide al modelo que lo recuerde, se garantiza después de que generó la consulta.

A esos controles se anteponen tres correcciones deterministas que reparan la consulta sin volver a invocar al modelo: normalización simétrica de acentos, insensibilidad a mayúsculas en el campo de paralelo, y corrección del nombre de columna de ciclo. Y antes de todo ello opera la resolución de entidades: un índice de nombres de materias construido desde la propia base de datos empareja la mención del usuario contra el nombre canónico y lo inyecta en la pregunta que el modelo recibe; si el emparejamiento arroja múltiples candidatos, se ofrece un menú de desambiguación sin gastar una sola invocación. El modelo recibe una entidad ya resuelta, no una cadena que debe adivinar.

La lógica compartida por todas estas capas es la misma: la instrucción en lenguaje natural dentro del prompt es una preferencia que el modelo puede incumplir; la regla en código es una garantía que no depende de su cooperación.

### 4.4 Privacidad de datos y aritmética de reloj para recordatorios

#### 4.4.1 Tres capas independientes de privacidad

La primera capa opera sobre lo que el estudiante pregunta de sí mismo: un patrón intercepta, antes de cualquier modelo, consultas sobre datos personales propios y responde con un mensaje fijo que declara que el sistema no almacena perfiles ni historiales.

La segunda capa gobierna el contacto con terceros, y es donde vive la lógica más elaborada del sistema. El consentimiento no es una casilla que el administrador marca en nombre del docente: es un flujo conversacional donde el propio profesor responde si

autoriza compartir su número personal. Ante el silencio, el valor por defecto nunca es compartir, sino entregar solo el correo institucional.

Sobre ese consentimiento se apilan tres guardas adicionales, y cada una nació de un falso positivo real detectado en producción. La primera exige intención explícita de contacto en la pregunta antes de adjuntar cualquier tarjeta, porque antes bastaba con mencionar un apellido de pasada para que el sistema entregara los datos, un efecto invasivo que apareció en una prueba en vivo. La segunda compara solo por apellido y no por nombre de pila, porque un nombre común compartido por varios docentes producía colisiones reales entre personas sin ninguna relación. La tercera impide adjuntar una tarjeta bajo una respuesta que ya declaró ambigüedad o ausencia de información, para que el sistema no se contradiga dentro del mismo mensaje.

La tercera capa es la revocación a nivel de rol de PostgreSQL ya descrita en 3.8. Las tres no son redundantes entre sí: cada una cierra un vector de fuga distinto: la intención expresada por el estudiante, la lógica de negocio sobre el consentimiento y el permiso bruto a nivel de base de datos. Este esquema de consentimiento y revocación inmediata por WhatsApp implementa, en la práctica, los derechos de acceso, rectificación y oposición reconocidos en los artículos 7, 9 y 14 de la Ley Orgánica de Protección de Datos Personales del Ecuador (LOPDP), y las solicitudes de eliminación total o de copia de datos que excedan lo resoluble por el propio flujo conversacional se canalizan al responsable del tratamiento, quien debe atenderlas dentro del plazo de 15 días hábiles previsto por dicha ley.

### 4.4.2 Aritmética de reloj para recordatorios absolutos y relativos

El reconocimiento temporal distingue tres formas de expresión y las resuelve con lógica distinta. El tiempo relativo acepta tanto dígitos como números escritos en palabras, porque una nota de voz transcrita produce la forma literal y no la numérica, y un patrón que solo buscara dígitos no coincidiría con ese caso real. El tiempo absoluto resuelve la ambigüedad de una hora entre 1 y 11 sin marca de meridiano con la misma heurística que aplicaría una persona: calcula ambos candidatos del día y elige la primera ocurrencia futura; si ambas ya pasaron, cae al día siguiente.

La ventana de 24 horas que impone la plataforma de mensajería se hace cumplir en tres niveles deliberadamente redundantes: un rechazo léxico inmediato para expresiones que evidentemente la exceden, sin necesidad de calcular ningún delta; un

rechazo numérico para deltas que ninguna palabra clave delata; y una tercera verificación dentro de la propia función de agendamiento, como respaldo ante cualquier otro punto de entrada. Cada capa cierra un hueco que la anterior no puede ver por su naturaleza, léxica una, aritmética la otra.

Dos correcciones de robustez completan el subsistema. Un defecto real de producción hacía que el texto de confirmación mostrara la hora del sistema del contenedor como si fuera la hora local del estudiante, con un desfase constante; el envío siempre fue correcto porque se calcula con zona horaria consciente, y la corrección convierte explícitamente a la zona local antes de formatear el mensaje. Y porque el almacén de trabajos del planificador reside en memoria del proceso, un reinicio del contenedor borraba cualquier recordatorio programado sin borrar la fila que lo respalda: una función de recarga se ejecuta en el arranque, relee los recordatorios pendientes y decide para cada uno si aún cabe dentro de la ventana o debe marcarse como expirado.

**Ilustración 6**

*Flujo de consentimiento previo a la entrega de datos de contacto y confirmación de recordatorio.*

## Capítulo 5. Resultados, validación y conclusiones

Este capítulo presenta primero la evidencia recogida durante la validación del sistema, organizada según los criterios definidos en el apartado 3.5, y cierra con la interpretación de esos resultados: lo que el trabajo permite concluir, sus limitaciones reconocidas y las líneas de continuación posibles.

### 5.1 Diseño del banco de pruebas

La verificación de exactitud se realizó sobre un conjunto de preguntas cuya respuesta correcta se conocía de antemano, obtenida directamente de la fuente institucional. Las consultas se ejecutaron invocando la cadena de resolución del sistema dentro del contenedor de producción, contra la base de datos real y con el mismo recorrido de decisión que atiende un mensaje entrante. No se enviaron a través de la interfaz de la plataforma de mensajería, de modo que la prueba cubre el motor que resuelve la consulta y no la entrega del mensaje. De cada caso se registró la respuesta del asistente y se comparó contra la fila correspondiente en la base de datos.

Las preguntas se agruparon en las categorías que el sistema efectivamente atiende, con un criterio adicional: cada categoría incluye al menos una formulación deliberadamente imperfecta, con error ortográfico, sin tildes o con el nombre de la materia incompleto, porque así es como escriben las personas y no como escriben los casos de prueba.

**Tabla 8**

*Composición del banco de pruebas por categoría de consulta*

| **Categoría** | **Mecanismo que la resuelve** | **Consultas probadas** | **Respuestas correctas** |
|---|---|---|---|
| Ubicación de aulas y laboratorios | Fast-path determinista | 10 | 10 |
| Horarios y paralelos | RAG estructurado (Text-to-SQL) | 5 | 5 |
| Cronograma de exámenes | Fast-path determinista | 3 | 3 |
| Contacto docente | Vía determinista con consentimiento | 5 | 5 |
| Eventos y charlas | RAG estructurado sobre datos de Instagram | 4 | 4 |
| Preguntas sin respuesta en la base | Corte por cero filas | 6 | 6 |
| Total | | 32 | 32 |

**Nota.** Elaboración propia. Pruebas ejecutadas el 27 de julio de 2026 sobre el despliegue en producción, invocando directamente la cadena de resolución del sistema contra la base de datos real. Una consulta de la última categoría se excluyó del cómputo porque el dato por el que preguntaba sí existía en la base, de modo que no correspondía a la categoría asignada; el conteo se reporta sobre las 17 consultas válidas restantes. En esa última categoría la respuesta correcta es la declaración explícita de que el sistema no dispone de esa información.

El resultado agregado es de 32 respuestas correctas sobre 32 consultas válidas. Este resultado no corresponde a la primera ejecución del banco de pruebas: una ronda inicial, más reducida, detectó cuatro fallas puntuales que llevaron a dos correcciones concretas en el código antes de repetir la validación.

Tres de esas fallas compartían la misma causa. El reconocimiento de los verbos que disparan las vías deterministas toleraba la ausencia de tildes, pero no el error de tipeo dentro de la propia palabra clave. Formulaciones como "komo llego al aula 203", "donde qeda el bano de hombres" o "cuando son las vacasiones" no activaban el fast-path correspondiente y derivaban en una solicitud genérica de aclaración. El dato existía en la base y el mecanismo que debía servirlo funcionaba: lo que fallaba era la puerta de entrada. La corrección consistió en extender al verbo disparador la misma coincidencia aproximada de cadenas que ya operaba sobre los nombres propios, descrita en el apartado 5.7.

La cuarta falla era de otra naturaleza y resultó más instructiva. Ante la pregunta "hay hackaton este ciclo", la palabra "ciclo" llevaba al clasificador de intención a asignar dominio académico, con lo cual la tabla de publicaciones de Instagram quedaba fuera del esquema podado antes de que el generador de SQL pudiera considerarla. El evento existía en la base y era perfectamente recuperable, pero la poda lo dejaba inalcanzable. La corrección ajustó el criterio de clasificación de dominio para que esa palabra no excluya por error la fuente de Instagram del esquema disponible.

Tras aplicar ambas correcciones, se repitió el banco de pruebas con un volumen mayor de consultas por categoría, incluyendo nuevamente formulaciones con error ortográfico deliberado, y ninguna de las cuatro fallas originales volvió a presentarse.

Reportar estas cuatro fallas es preferible a omitirlas. Un banco de pruebas que arroja acierto pleno en todas las categorías suele indicar que las preguntas se escribieron conociendo la implementación, y en ese caso no mide la robustez del sistema sino la habilidad de quien redactó las preguntas.

### 5.2 Huella de infraestructura

El consumo de memoria se midió con docker stats sobre el despliegue en producción descrito en el apartado 3.4. El resultado, presentado en la Tabla 4 del apartado 3.12, es de 350,74 MiB para los tres contenedores que sostienen el servicio de forma permanente.

Lo relevante de esa cifra no es su magnitud absoluta sino su origen. Ningún proceso del sistema carga jamás un modelo de lenguaje en memoria propia: cuando el motor activo es un proveedor comercial, el contenedor abre una conexión saliente, envía el prompt y espera texto. Frente a los cerca de 8 GB que exigía mantener Ollama residente antes de migrar el motor primario a la nube, la diferencia de más de un orden de magnitud no proviene de haber optimizado el software, sino de haber trasladado el costo de memoria del modelo al proveedor que lo sirve.

El resultado práctico es que el sistema completo cabe holgadamente en un servidor compartido de propósito general que además atiende correo, DNS y otros sitios. Para una facultad, eso significa que la barrera de entrada no es la infraestructura.

### 5.3 Latencia por nivel de la cascada

La latencia se midió como tiempo transcurrido entre la recepción del mensaje y el envío de la respuesta, sobre consultas resueltas por RAG estructurado, que son las que atraviesan dos invocaciones al modelo.

**Tabla 9**

*Latencia observada por nivel del enrutador en cascada*

| Nivel | Motor | Latencia observada | Origen del dato |
|---|---|---|---|
| 1 | Groq (LPU) | ≈0,4 s (mediana, n=30) | Medición en el despliegue |
| 2 | Gemini | 2 a 4 s | El registro solo anota el motor cuando falla; las invocaciones exitosas no dejan traza temporal |
| 3 | Claude | 2 a 4 s | El registro solo anota el motor cuando falla; las invocaciones exitosas no dejan traza temporal |

| Nivel | Motor | Latencia observada | Origen del dato |
|---|---|---|---|
| 4 | Ollama sobre GPU | Depende del tipo y cuanta ram de GPU se tenga a disposición | La velocidad dependerá de cuanta ram tenga la GPU |
| 5 | Ollama sobre CPU | 20 a 90 s (consulta aislada) | Medición en el despliegue |
| - | Inferencia sobre GPU convencional | 5 a 10 s | Medición previa a la migración a la nube |

**Nota.** Elaboración propia. Los niveles 2 y 3 no se reportan con cifra porque el sistema solo deja registro del motor empleado cuando esta falla: las invocaciones exitosas no dejan traza con marca de tiempo que permita aislar su latencia, de modo que no existe medición defendible. El nivel 4 está implementado y probado, pero no puede medirse en el despliegue actual por ausencia de hardware acelerado. La medición del nivel 5 corresponde a una consulta aislada; bajo concurrencia el comportamiento se degrada de forma no lineal, como se detalla a continuación.

El comportamiento del último escalón merece un comentario, porque condicionó el diseño de la cola. Con dos consultas concurrentes compitiendo por los mismos núcleos, cada una se vuelve varias veces más lenta y el tiempo total resulta peor que procesarlas en serie. Esa observación es evidencia de que la inferencia en CPU está limitada por cómputo y no por entrada y salida, y de ahí salió el dimensionamiento de la cola: dos hilos de trabajo y un tope de 20 consultas pendientes, con una espera aproximada de N × 25 segundos para quien ocupa la posición N.

Hay un caso adicional que conviene reportar porque es el ahorro más grande que produjo el proyecto. Antes de que el comando de continuación de listado se resolviera de forma determinista, esa entrada obligaba al modelo de SQL a generar un menú completo, con un tiempo medido en producción de aproximadamente 108 segundos para una consulta que no tenía nada que responder. Al convertirla en fast-path, ese tiempo pasó a ser del orden de los microsegundos y el costo de inferencia asociado desapareció por completo. No se redujo, solo dejó de existir.

### 5.4 Continuidad del servicio ante fallo de proveedor

Este apartado reporta evidencia observacional, no experimental, y conviene decirlo antes que cualquier resultado. La cascada no fue sometida a inyección controlada

de fallos. Lo que sigue proviene de incidentes reales registrados en los registrosdel sistema durante el período de operación.

Durante la ventana de observación se registró un único tipo de transición: el agotamiento del límite de solicitudes por minuto del proveedor de nivel 1, que devuelve un error 429 sobre el modelo destinado a la generación de SQL. El comportamiento fue en todos los casos el mismo: el interruptor de circuito marcó el motor como caído durante 60 segundos, la ejecución continuó al siguiente nivel dentro de la misma petición del estudiante, y el estudiante recibió su respuesta sin percibir el fallo intermedio. No se registró ninguna transición originada en ausencia de clave vigente ni en agotamiento de la cuota diaria, condiciones que el código contempla pero que no se presentaron en el período observado.

Entre el 24 y el 27 de julio de 2026, sobre un registro del sistema continua de aproximadamente tres días, se registraron 47 transiciones de nivel, todas por la causa descrita. Corresponde acotar el alcance de esa cifra. Los registros del sistema no distinguen entre las consultas de operación y las consultas ejecutadas durante la propia validación, de modo que el número describe con qué frecuencia se activó el mecanismo en el período observado y no una tasa de fallo del proveedor atribuible solo a tráfico de estudiantes. La ventana tampoco es lo bastante extensa para sostener una proyección anual.

Lo que esta evidencia sostiene es que el punto único de fallo quedó estructuralmente eliminado. Lo que no sostiene es una cifra de disponibilidad continua sobre un período prolongado, porque el sistema no fue instrumentado para calcularla. La afirmación defendible es arquitectónica: cualquier integración basada en un solo proveedor se cae cuando ese proveedor se cae, y esta no. Esa lectura se apoya además en evidencia externa, ya que Chu et al. (2025) documentan que los proveedores de modelos de lenguaje en producción fallan con periodicidad medible.

### 5.4.1 Prueba de carga concurrente

Para evaluar el comportamiento del sistema ante un escenario de uso simultáneo, como el de varios miembros de un tribunal consultando el asistente al mismo tiempo, se ejecutó una prueba deliberada de carga concurrente el 29 de julio de 2026, invocando directamente la misma función que procesa cada mensaje real, sin pasar por la interfaz de mensajería, contra la base de datos de producción y con las claves de API reales.

Se lanzaron ráfagas de 12 solicitudes simultáneas por cada una de ocho preguntas representativas, y una ráfaga combinada de 150 solicitudes simultáneas mezclando las

ocho. Las 150 solicitudes de la ráfaga combinada obtuvieron respuesta en 47.3 segundos totales, sin un solo caso de agotamiento completo de la cascada. El proveedor de nivel 1 sostuvo apenas una consulta exitosa bajo esa carga antes de entrar en su período de enfriamiento, y el nivel 2 absorbió el resto sin registrar fallas, lo cual confirma en condiciones de estrés real el mismo mecanismo de circuit breaker descrito en el apartado 3.11. El nivel 3 no fue invocado en esta prueba, porque el nivel 2 nunca llegó a agotarse; una prueba anterior con una ráfaga más concentrada sí había hecho fallar al nivel 2 en dos ocasiones sobre 150, lo que confirma que ese límite existe y es alcanzable, aunque esta ronda no llegó a ese punto.

## 5.5 Frescura y trazabilidad del dato extraído de Instagram

Este apartado verifica el criterio C3 y aporta la evidencia empírica más específica del trabajo, porque toca el punto donde un asistente institucional falla con más frecuencia: no cuando no sabe algo, sino cuando responde con confianza un dato que ya cambió.

### 5.5.1 Justificación del diseño de la prueba

Verificar la frescura del dato exige poder modificar la fuente y observar si el cambio se propaga. La cuenta oficial de Instagram de la Facultad no permite eso: el autor no tiene permisos de edición sobre ella y, aunque los tuviera, alterar el contenido de un canal oficial de comunicación (incluso de forma transitoria) habría afectado a una audiencia real ajena al experimento. La prueba se ejecutó, por lo tanto, redirigiendo temporalmente el extractor hacia una cuenta personal del autor, con la misma interfaz oficial de la plataforma y el mismo código de extracción que corre en producción. Se modificó únicamente el identificador de la cuenta objetivo, parametrizado por variable de entorno.

Esta decisión, además de resolver la objeción ética, permitió probar un escenario que la cuenta institucional no habría permitido observar en ningún caso.

### 5.5.2 Extracción de la fecha declarada en el texto, no de la fecha de publicación

El riesgo específico que se quiso verificar es este: una publicación tiene una fecha propia, la de su publicación, y puede mencionar en su descripción una fecha distinta, la del evento que anuncia. Un extractor ingenuo toma la primera y descarta la segunda, con lo cual el asistente termina anunciando eventos con fechas equivocadas.

Se utilizó una publicación existente en la cuenta personal, publicada originalmente en 2021, y se editó su descripción para que anunciara un evento con fecha en 2026. Se ejecutó el extractor y se verificó el registro resultante en la base de datos.

**Resultado:** el sistema reconoció como válida la información del evento de 2026 y no descartó la publicación por la antigüedad de su fecha de publicación. Al formular después la consulta por WhatsApp, el asistente respondió con la fecha del evento y no con la fecha de la publicación.

Este resultado no es trivial. Confirma que la vigencia de la información se determina por el contenido declarado en el texto y no por la antigüedad del registro que lo contiene, que es exactamente la distinción que una publicación institucional real exige. Una charla anunciada en una publicación de hace tres semanas sigue siendo válida hasta que ocurra, y una publicación reciente sobre un evento ya pasado no lo es.

### 5.5.3 Propagación de una edición posterior a la ingesta

El segundo riesgo verificado es el opuesto: que el sistema capture correctamente la primera vez y después quede congelado, sirviendo indefinidamente una versión desactualizada de un contenido que en la fuente ya cambió.

Se editó la descripción de una publicación ya ingerida en una ejecución anterior del extractor y se disparó una nueva ejecución.

**Resultado:** el registro correspondiente en la base de datos se actualizó con el contenido nuevo, sin generar un registro duplicado. La consulta posterior por WhatsApp devolvió la información corregida.

Esto verifica que la escritura del extractor opera por actualización sobre el registro existente y no por inserción incondicional. Esa propiedad es necesaria para que una corrección publicada por la Facultad, como una charla que cambia de aula o un evento que se reprograma, llegue efectivamente al estudiante en la siguiente ejecución diaria y no en el siguiente período académico.

**Tabla 10**

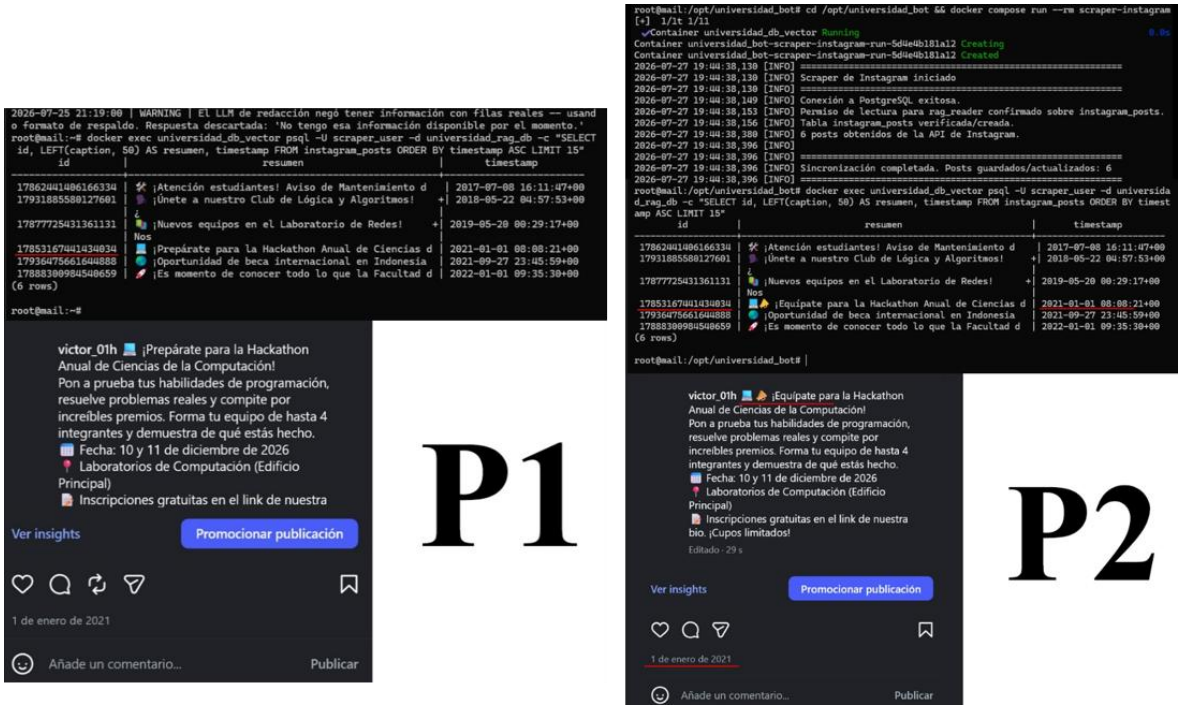
*Resultados de la verificación de frescura y trazabilidad del dato extraído*

| **Prueba** | **Condición evaluada** | **Resultado esperado** | **Resultado obtenido** |
|---|---|---|---|
| P1 | Publicación de 2021 con evento declarado en 2026 en su descripción | El sistema toma la fecha del evento, no la de publicación | Cumplido |

| Prueba | Condición evaluada | Resultado esperado | Resultado obtenido |
|---|---|---|---|
| P2 | Edición de la descripción de una publicación ya ingerida | El registro se actualiza sin duplicarse | Cumplido |

**Nota.** Elaboración propia. Pruebas ejecutadas sobre una cuenta personal del autor, con el mismo código de extracción en producción y a través de la interfaz oficial de la plataforma (Meta Platforms, 2026b).

**Ilustración 7**
*Evidencia documental de la propagación de una edición posterior a la ingesta.*




### 5.5.4 Alcance y limitaciones de esta verificación

Estas dos pruebas demuestran que los mecanismos funcionan, y lo hacen de forma concluyente para los dos riesgos que se propusieron verificar: la distinción entre fecha de publicación y fecha del evento, y la propagación correcta de una edición posterior. Lo que estas pruebas no estiman, porque no fueron diseñadas para eso, es una tasa de acierto sobre un volumen alto de publicaciones; son comprobaciones funcionales dirigidas y no una serie con potencia estadística, y así deben leerse. Además, una cuenta personal tiene un volumen y una estructura de publicación distintos de los de una cuenta institucional,

de modo que estas pruebas no dicen nada sobre el comportamiento del extractor frente a los límites de tasa de la plataforma cuando el volumen de publicaciones crece. Ambas condiciones se recogen en el apartado 5.14 como líneas de continuación y no como duda sobre el resultado ya obtenido.

### 5.6 Verificación del escudo anti-alucinaciones

El criterio C2 se verificó con pruebas dirigidas en las que el estudiante afirma como hecho establecido algo que no existe en la base de datos.

El caso documentado durante el desarrollo es el que motivó la primera intercepción del escudo. Ante una afirmación falsa formulada como premisa, según la cual un docente determinado dictaba una asignatura específica en un día y una hora concretos, el modelo de redacción, invocado sobre un conjunto de resultados vacío, confirmó la afirmación en lugar de negarla. Es el modo de falla que Huang et al. (2023) describen dentro de su taxonomía de alucinación, y apareció en una prueba en vivo, no en un escenario construido.

La corrección no consistió en ajustar el prompt. Consistió en cortar la invocación: cuando la consulta SQL devuelve cero filas, el sistema no llama al modelo de redacción y responde con un mensaje fijo que declara la ausencia de información. El modo de falla se volvió estructuralmente imposible, porque no hay generación que corromper si no hay generación.

Una segunda intercepción cubre el caso inverso: cuando sí hay filas pero el modelo redacta una negativa. En esa situación la respuesta se descarta por completo y se sustituye por un formato construido en código a partir de las mismas filas que el modelo tuvo disponibles.

**Tabla 11**

*Verificación del comportamiento ante datos ausentes y contradictorios*

| Escenario probado | Comportamiento esperado | Resultado |
|---|---|---|
| Consulta sobre un dato inexistente en la base | Declarar explícitamente que no dispone de la información | 1 de 1 |
| Afirmación falsa del estudiante formulada como premisa | No confirmarla; declarar ausencia de información | 2 de 2 |
| Respuesta del modelo que niega pese a existir filas | Descartar el texto y sustituirlo por el formato determinista | 2 de 2 |
| Pregunta demasiado vaga para resolverse sin volcar una tabla | Cortar antes de la redacción y solicitar precisión | 2 de 2 |

**Nota.** Elaboración propia. Pruebas ejecutadas el 27 de julio de 2026 sobre el despliegue en producción. El primer escenario se reporta sobre una sola consulta válida porque una segunda formulación prevista resultó inaplicable: el dato por el que preguntaba sí existía en la base. En el tercer escenario los registros del sistema muestran que actuaron dos capas deterministas encadenadas, la corrección fonética del apellido y el corte por desambiguación, de modo que la consulta nunca llegó al modelo de redacción.

### 5.7 Resolución de entidades y corrección fonética

Un asistente que recibe notas de voz debe tolerar transcripciones imperfectas de nombres propios, que es donde más se equivoca cualquier sistema de reconocimiento. El sistema resuelve esto comparando el nombre recibido contra el padrón real de docentes mediante coincidencia aproximada de cadenas.

El umbral de similitud se calibró de forma empírica contra el conjunto real de aproximadamente 41 nombres del padrón, buscando el punto donde el sistema corrige errores de transcripción sin confundir a dos personas distintas. El valor adoptado es 0,72, y con él no se produjo ningún falso positivo sobre ese conjunto. El caso que fijó el umbral es ilustrativo: transcripciones como "Eraso" o "Brazo" deben resolver a "Erazo", pero no debe resolverse a un docente cualquiera un apellido que simplemente comparte algunas letras.

Conviene declarar el alcance de este resultado: cero falsos positivos sobre 41 nombres es un resultado válido para el padrón actual de esta carrera. No garantiza el mismo comportamiento sobre un padrón mayor, donde la densidad de apellidos parecidos aumenta y el umbral probablemente deba recalibrarse. Está señalado como línea de continuación.

### 5.8 Poda dinámica del esquema

El tamaño del contexto que se envía al generador de SQL tiene un efecto documentado sobre la calidad del resultado: Gurawa y Dharmik (2025) muestran que ampliar el esquema recuperado mejora la cobertura, pero incrementa el ruido y el riesgo de alucinación.

El sistema poda el esquema según el dominio inferido de la pregunta antes de construir el prompt de SQL. La reducción del tamaño del esquema enviado, comparando la representación completa contra la podada en cada dominio, se sitúa entre el 32 % y el 53 %. Conviene precisar que se trata de una comparación estática entre ambas representaciones y no de una métrica calculada consulta por consulta en tiempo de

ejecución, porque el sistema no instrumenta ese cálculo. Cuando la señal de dominio resulta ambigua, el sistema envía el esquema completo en lugar de arriesgar una poda incorrecta, decisión que privilegia la exactitud sobre el ahorro.

Los registros del sistema sí permite dimensionar con qué frecuencia interviene el mecanismo. De 667 consultas registradas en la ventana de observación, 353 llegaron a la etapa donde se decide la poda; las restantes se resolvieron antes, por fast-path o por el clasificador de intención, sin construir nunca un prompt de SQL. De esas 353, el clasificador asignó dominio académico en 139 casos y dominio de Instagram en 26, mientras que en 188 no obtuvo una señal suficiente y se envió el esquema completo. El reparto muestra que la poda opera sobre algo menos de la mitad de las consultas que alcanzan esa etapa y que la vía conservadora sigue siendo la más frecuente, lo cual es coherente con la prioridad declarada.

### 5.9 Cumplimiento de los objetivos específicos

La Tabla 12 relaciona cada objetivo específico con la evidencia que lo sustenta y con el apartado donde esa evidencia se presenta. La última columna declara el grado de cumplimiento sin redondear hacia arriba.

**Tabla 12**

*Trazabilidad entre objetivos específicos, evidencia y grado de cumplimiento*

| Objetivo específico | Evidencia | Apartado | Cumplimiento |
|---|---|---|---|
| OE1. Automatizar la captura de datos académicos y extracurriculares | Tres extractores programados por cron; verificación de frescura y de propagación de cambios | 3.3, 5.5 | Cumplido |
| OE2. Implementar una arquitectura de datos que responda de forma precisa y contextualizada | RAG estructurado con validación estructural del SQL, escudo anti-alucinaciones y poda dinámica de esquema | 4.2, 4.3, 5.6, 5.8 | Cumplido |
| OE3. Diseñar un módulo de navegación con guías visuales hacia aulas y laboratorios | Fast-path de ubicación con galería fotográfica asociada a cada espacio | 4.1, 5.1, Anexo B | Cumplido con observación |
| OE4. Integrar una lógica de enlace con docentes respetando privacidad y disponibilidad | Tres capas independientes de privacidad, con consentimiento declarado por el propio docente | 4.4.1 | Cumplido |

| Objetivo específico | Evidencia | Apartado | Cumplimiento |
|---|---|---|---|
| OE5. Verificar la operatividad técnica y la precisión de las respuestas | Banco de pruebas, mediciones de huella y latencia, verificación de frescura y de comportamiento ante datos ausentes | 5.1–5.9 completo | Cumplido parcialmente |

**Nota**. Elaboración propia. La observación sobre el OE3 corresponde a las fallas de reconocimiento del verbo disparador documentadas en el apartado 5.1: el módulo resuelve correctamente cuando se activa, pero su detección no tolera errores de tipeo en la palabra clave. El cumplimiento parcial del OE5 responde a que la verificación se ejecutó sobre una sola carrera, sin inyección controlada de fallos, sin instrumentación de consumo de tokens y sin medición de satisfacción de usuario. Esas ausencias se detallan en el apartado 3.7 y se retoman como líneas de continuación en el apartado 5.15.

### 5.10 Sobre la eliminación del costo de inferencia mediante cortocircuitos deterministas

Conviene empezar por el hallazgo más contundente de este trabajo: para el subconjunto de tráfico que un fast-path logra interceptar, el costo de inferencia no se reduce, desaparece. Una pregunta por la ubicación de un aula, la paginación de un listado, las materias de un ciclo o un saludo nunca genera un prompt, nunca abre una conexión hacia un proveedor y nunca consume un token: se resuelve por coincidencia de patrón contra una estructura ya cargada en memoria, en microsegundos.

El caso documentado durante el desarrollo ilustra la magnitud del problema corregido. Antes de que el comando de continuación de listado se resolviera de forma determinista, esa misma entrada obligaba al modelo de SQL a generar un menú completo, con un tiempo de respuesta medido en producción de aproximadamente 108 segundos para una consulta que, en rigor, no tenía nada que responder. Ese tiempo, junto con el gasto de tokens asociado, se eliminó al introducir la regla determinista. El ahorro no es proporcional al tamaño del prompt evitado, es binario, y su impacto agregado depende de qué fracción del tráfico real cae dentro de ese patrón repetitivo y predecible, algo esperable en un dominio institucional acotado.

### 5.11 La disponibilidad ganada por el enrutador en cascada es, sobre todo, arquitectónica

Quizá el aporte menos visible pero más importante del diseño sea que el enrutador de cinco niveles (Groq, Gemini, Claude, Ollama con aceleración por GPU y Ollama en CPU) convierte la caída de un proveedor individual en una simple degradación de velocidad, no en una interrupción del servicio. Estos cinco niveles no son alternativas intercambiables al azar, sino una jerarquía: cada uno se verifica antes de invocarse, y cada fallo activa un interruptor de circuito de 60 segundos que evita insistir contra un proveedor ya identificado como caído, sin que el estudiante perciba el fallo intermedio.

Cada transición relevante (operación normal, degradación al secundario, y el escenario límite de ausencia total de proveedores en la nube) se validó explícitamente contra el despliegue real. Este punto admite matiz: no se trata de una cifra de disponibilidad continua medida sobre un período prolongado, algo que este proyecto no instrumentó, sino de algo distinto y, en cierto sentido, más sólido: la eliminación estructural del punto único de fallo que tendría cualquier integración basada en un solo proveedor. Esa lectura se apoya en evidencia empírica externa: los proveedores de modelos de lenguaje en producción fallan con periodicidad medible, como documentan Chu et al. (2025).

### 5.12 Sobre la precisión determinista de la generación aumentada con recuperación estructurada.

Sobre este punto conviene ser preciso y no exagerar el hallazgo. El RAG estructurado basado en Text-to-SQL alcanza, en este dominio, una precisión que la recuperación semántica por similitud vectorial no puede igualar, y la razón no tiene que ver con la calidad lingüística del modelo. Tiene que ver con que existe una cadena de validación posterior a la generación, y esa cadena es verificable. Explicado en términos más simples: no importa si el modelo "entiende bien" la pregunta, importa si lo que produce pasa o no los filtros que vienen después. No es que el SQL generado sea infalible (la literatura sobre bases de datos reales documenta que ni los modelos más capaces lo son, según Li et al., 2023), sino que el sistema nunca confía ciegamente en esa salida: cada consulta pasa por rechazo de escritura y de ambigüedad de precedencia, por imposición forzada de un límite de filas, y por un escudo posterior que descarta cualquier redacción que contradiga las filas realmente recuperadas.

La búsqueda semántica por similitud, en contraste, no admite un mecanismo equivalente de verificación posterior, porque su salida (un ordenamiento por proximidad) carece de una noción binaria de correcto o incorrecto. La consulta relacional sí la tiene, y es esa propiedad, no la elocuencia del modelo que redacta la respuesta, la que sostiene la confiabilidad del sistema en un dominio donde un dato erróneo puede tener consecuencia administrativa real.

### 5.13 Sobre la eficiencia de infraestructura y la delegación cognitiva

Hubo un resultado que no esperaba al empezar. La eficiencia de infraestructura de Compu IA no depende de la escala del hardware disponible, sino del patrón arquitectónico elegido. Basta con 350,74 MiB de memoria RAM sobre un servidor compartido de propósito general para sostener el servicio completo, porque ningún modelo de lenguaje reside jamás en ese servidor. Frente a los cerca de 8 GB que exigía mantener Ollama cargado localmente antes de migrar el motor primario a la nube, la diferencia de más de un orden de magnitud no proviene de haber optimizado el software del bot, sino de haber trasladado el costo de memoria del modelo al proveedor que lo sirve. En un sentido parecido, la seguridad perimetral (terminación de TLS, exposición pública) se delega a un servidor Nginx externo al conjunto de microservicios, liberando a la aplicación de una responsabilidad que no le corresponde resolver por sí misma.

Para una institución educativa de escala promedio, la comparación económica no deja demasiado espacio para la duda: un gasto mensual de entre 0,45 y 9,00 USD, sin capital inicial ni personal dedicado a operar un servidor de GPU, frente a un desembolso de miles de dólares que además se deprecia con el tiempo. Bajo cualquier análisis de costo total de propiedad razonable, la delegación cognitiva a la nube resulta la decisión financieramente más sensata para este caso de uso, aunque no es la única variable en juego.

La única condición que revertiría esa conclusión no tiene que ver con rendimiento ni disponibilidad, ya resueltos por la cascada sin gasto de capital adicional, sino con la soberanía del dato: una política institucional que exigiera que las consultas no salgan de infraestructura controlada por la universidad haría inaceptable, por motivos ajenos al costo, seguir enviando prompts a terceros. Es exactamente el escenario para el que ya existen, y ya fueron probados, los niveles de inferencia local de la cascada: no están activos a escala, pero tampoco cuestan nada mientras no se necesiten.

### 5.14 Cobertura de la capa de similitud léxica sobre el conocimiento institucional

El atajo de palabras clave descrito en el apartado 4.1 localiza una ficha de conocimiento únicamente cuando el administrador anticipó el término que emplearía el estudiante. Para dimensionar ese límite se construyó un banco de 96 consultas: 76 redactadas en el registro que usa un estudiante al escribir por mensajería y 20 ajenas al dominio académico. Las mediciones de este apartado se ejecutaron con la cascada de inferencia deshabilitada, de modo que describen el comportamiento determinista y no el del modelo de lenguaje.

**Tabla 13**

*Cobertura del banco de consultas antes y después de la capa de similitud léxica*

| **Indicador** | **Valor** |
|---|---|
| Consultas evaluadas | 96 (76 en registro de estudiante y 20 ajenas al dominio) |
| Resueltas por el atajo de palabras clave | 30/76 (39 %) |
| Sin respuesta en la línea base | 46/76 (61 %) |
| Resueltas por la capa de similitud léxica | 11/46 (24 %) |
| Respuestas incorrectas introducidas | 0/46 |
| Falsos positivos sobre consultas ajenas | 0/20 |
| Regresión sobre los casos ya resueltos | 30/30 |
| Cobertura total resultante | 41/76 (54 %) |

La clasificación de los 46 casos que la línea base no resuelve separa lo que es una brecha de vocabulario de lo que es un problema de ordenamiento.

**Tabla 14**

*Clasificación de las consultas no resueltas por la línea base, según la causa del fallo*

| **Categoría** | **Descripción** | **Casos** |
|---|---|---|
| A | Ninguna raíz de la consulta existe en el corpus | 1/46 |
| B | La ficha correcta no comparte ninguna raíz con la consulta | 3/46 |
| C | La ficha correcta comparte raíces, pero otra obtiene mayor puntaje | 24/46 |
| D | La ficha correcta gana con menos de dos raíces coincidentes | 6/46 |
| E | La ficha correcta gana sin alcanzar el margen exigido sobre la segunda | 1/46 |

| Categoría | Descripción | Casos |
|---|---|---|
| F | Resuelta por la capa de similitud | 11/46 |

Las categorías D y E reúnen once consultas en las que la ficha correcta ya ocupa el primer lugar y solo la retienen los dos umbrales de seguridad. La brecha de vocabulario, que abarca las categorías A y B, explica cuatro casos. El límite real del método está en la categoría C, con 24 consultas en las que la ficha correcta pierde el ordenamiento frente a otra que comparte más términos.

La selección de parámetros se resolvió por barrido sobre el mismo banco, bajo la restricción de no admitir ninguna respuesta a consultas ajenas al dominio.

**Tabla 15**

*Configuraciones evaluadas de la capa de similitud y resultado de la selección*

| Configuración | Aciertos | Incorrectas | Falsos positivos |
|---|---|---|---|
| Estricta: margen 2,0× sin lista de exclusión | 8/46 | 0/46 | 0/20 |
| Amplia: margen 1,2× sin lista de exclusión | 13/46 | 0/46 | 1/20 |
| Amplia con exclusión de raíces genéricas (adoptada) | 11/46 | 0/46 | 0/20 |

La configuración adoptada supera a la estricta en tres consultas y elimina el único falso positivo de la variante amplia, originado en la coincidencia de dos raíces vacías de contenido semántico. Ese caso aislado no constituye una tasa de error de la variante.

La sensibilidad del método a la forma en que el estudiante escribe se evaluó reescribiendo el banco completo con cuatro variantes de tecleo.

**Tabla 16**

*Estabilidad de la resolución ante variaciones de escritura*

| Variante | Aciertos | Incorrectas | Falsos positivos | Línea base intacta |
|---|---|---|---|---|
| Original | 11/46 | 0/46 | 0/20 | 30/30 |
| Con tildes | 11/46 | 0/46 | 0/20 | 30/30 |
| Mayúsculas sostenidas | 11/46 | 0/46 | 0/20 | 30/30 |
| «q» en lugar de «que» | 11/46 | 0/46 | 0/20 | 29/30 |

| Variante | Aciertos | Incorrectas | Falsos positivos | Línea base intacta |
|---|---|---|---|---|
| Transposición de caracteres | 4/46 | 1/46 | 0/20 | 12/30 |

Las tildes, las mayúsculas y la abreviatura «q» no alteran el resultado. La transposición de caracteres degrada por igual a las dos capas léxicas, incluida la de palabras clave. La prueba inyecta el error en la palabra más larga de cada consulta, que es el peor caso posible; la limitación queda recogida en el apartado 5.16.

El criterio de rareza y el mínimo de raíces coincidentes se sostienen sobre las magnitudes del corpus que se detallan a continuación.

**Tabla 17**

*Caracterización del corpus de conocimiento institucional*

| Medida | Valor |
|---|---|
| Fichas | 49 |
| Raíces distintas tras lematización y supresión de tildes | 1 260 |
| Frecuencia documental media y máxima de una raíz | 2,63 y 33 |
| Raíces bajo el umbral de rareza de 18 fichas | 1 251/1 260 (99 %) |
| Raíces descartadas por frecuentes | 9/1 260 |
| Raíces descartadas por la lista de exclusión semántica | 7 |

Las tablas de este apartado se basan en el estado del corpus con 49 fichas de conocimiento institucional, que fue el que se utilizó para la validación cuantitativa. Posteriormente se añadieron tres fichas nuevas sobre resciliación de semestre, resciliación de materia y revisión de calificaciones, junto con una ampliación de la ficha de pensión diferenciada, lo que lleva el corpus actual a 52 fichas. Estas incorporaciones mejoran la cobertura del asistente sobre trámites reglamentarios, pero no se reflejan en las métricas reportadas aquí, que describen el estado medido durante la validación.

**Tabla 18**

*Sensibilidad del mínimo de raíces coincidentes, con los demás parámetros fijos*

| Mínimo de raíces | Aciertos | Incorrectas | Falsos positivos |
|---|---|---|---|
| 1 | 17/46 | 17/46 | 7/20 |
| 2 (adoptado) | 11/46 | 0/46 | 0/20 |
| 3 | 2/46 | 0/46 | 0/20 |

Con una sola raíz la mitad de las respuestas emitidas es incorrecta y siete consultas ajenas al dominio reciben respuesta. Con tres, la cobertura cae a dos casos. El valor

adoptado es el único que combina cobertura aprovechable con ausencia de falsos positivos.

La búsqueda de publicaciones institucionales por contenido se sometió a una prueba de estrés dirigida: diez consultas construidas de forma deliberada para compartir dos palabras largas con algún pie de foto. Las proporciones que siguen corresponden a ese diseño adversarial sobre un conjunto reducido y no a una tasa de error esperada en operación.

**Tabla 19**

*Resultados de la prueba de estrés dirigida sobre la búsqueda de publicaciones*

| Lote de prueba | Antes del ajuste | Después del ajuste |
|---|---|---|
| Consultas legítimas de publicaciones (6) | 6/6 correctas | 6/6 correctas |
| Consultas ajenas al dominio (6) | 0/6 devuelven publicaciones | 0/6 |
| Consultas adversariales (10) | 5/10 devuelven publicaciones | 4/10 |

De las cuatro consultas adversariales que aún reciben publicaciones, dos constituyen respuestas pertinentes: la publicación del convenio ante una consulta por convenios y la del descuento con carnet ante una consulta por descuentos. La causa del sobre-disparo se midió sobre el vocabulario de las propias publicaciones.

**Tabla 20**

*Frecuencia de las palabras de mayor coincidencia en los pies de foto de la cuenta*

| Palabra | Publicaciones que la contienen |
|---|---|
| computación | 11/20 |
| ingeniería | 11/20 |
| asociación | 11/20 |
| estudiantil | 10/20 |
| mantenimiento | 7/20 |
| firma | 2/20 |
| electrónica | 2/20 |

Las palabras que dominan la cuenta institucional coinciden con las de las consultas académicas, de modo que ningún umbral de frecuencia las separa por sí solo. La mitigación combina el descarte de términos presentes en más de tres publicaciones con un filtro que impide responder con una publicación a consultas de ubicación, horario o contacto.

El costo temporal de la capa añadida se midió sobre la instancia de producción sin carga concurrente.

**Tabla 21**

*Latencia por capa de resolución*

| Capa | Media | Mediana | p95 | Rango | n |
|---|---|---|---|---|---|
| Atajo de palabras clave | 167,8 ms | 128,4 ms | 159,0 ms | 107,9 – 3 476,4 ms | 152 |
| Similitud léxica | 24,8 ms | 24,2 ms | 31,2 ms | 18,4 – 63,9 ms | 152 |
| Cascada de inferencia (nivel 1, redacción corta) | 403,3 ms | 394,7 ms | 501,5 ms | 179,5 – 616,7 ms | 30 |

Las dos primeras filas corresponden a las 76 consultas del banco ejecutadas en dos rondas consecutivas sobre la infraestructura y los datos de producción; no son tráfico de usuarios registrados. La tercera se obtuvo con invocaciones espaciadas doce segundos para respetar el límite de tokens por minuto del proveedor y corresponde al caso más favorable de la cascada: una redacción corta sobre datos ya recuperados. Aun con ese espaciamiento, tres de treinta y tres intentos fueron rechazados por cuota. La resolución por similitud léxica consume una quinta parte del tiempo del atajo de palabras clave y alrededor de una dieciseisava parte del nivel más rápido de la cascada.

### 5.15 Prueba de concepto: recuperación vectorial sobre los fallos de ordenamiento

La categoría C del apartado anterior reúne 24 consultas en las que la ficha correcta comparte vocabulario con la pregunta pero otra obtiene mayor puntaje. Para determinar si una representación semántica resolvería ese ordenamiento se ejecutó una prueba acotada, sin integrarla al flujo del asistente: se generaron representaciones vectoriales de las 49 fichas y de las 24 consultas con un modelo multilingüe servido por la instancia local de inferencia, se cargaron en una tabla temporal con la extensión pgvector y se ordenaron por distancia coseno. La extensión y la tabla se revirtieron al terminar la medición.

**Tabla 22**

*Resolución de los fallos de ordenamiento por similitud vectorial*

| Método | Ficha correcta en primera posición | Entre las tres primeras |
|---|---|---|
| Capa léxica desplegada | 0/24 | — |
| Similitud vectorial por distancia coseno | 18/24 (75 %) | 20/24 (83 %) |

El resultado de la capa léxica es 0/24 por construcción: la categoría C agrupa exactamente los casos en que esa capa ordena primero una ficha incorrecta. Los cuatro casos que la vía vectorial tampoco sitúa entre los tres primeros comparten patrón.

**Tabla 23**

*Detalle de los casos no resueltos por similitud vectorial*

| **Consulta** | **Ficha elegida (similitud)** | **Ficha correcta (similitud)** | **Posición** |
|---|---|---|---|
| cuánto cuesta pasar una materia de otra carrera | Homologar materias de otra carrera (0,588) | Cuánto cuesta homologar una materia (0,484) | 6 |
| se puede diferir el pago | Pensión diferenciada (0,525) | Pago de deuda con tarjeta de crédito (0,493) | 5 |
| quiero entrar al equipo de fútbol | Estar al día en pagos (0,464) | Deportes, cultura y guardería (0,414) | 8 |
| cómo busco información para mi tesis | Página de Servicios en Línea (0,517) | Alfabetización Informacional (0,462) | 8 |

En los cuatro, la ficha elegida es una vecina temática legítima de la consulta y la diferencia de similitud con la correcta va de tres a diez centésimas. En el tercero, la similitud de la ficha elegida queda por debajo de la mediana de los aciertos, de modo que un umbral de abstención lo convertiría en una consulta sin respuesta en lugar de en una respuesta equivocada. La viabilidad de ese umbral se estimó contrastando la similitud de la ficha más cercana en los aciertos y en las consultas ajenas al dominio.

**Tabla 24**

*Similitud de la ficha más cercana en consultas del dominio y ajenas a él*

| **Grupo** | **n** | **Mínimo** | **Mediana** | **Máximo** |
|---|---|---|---|---|
| Aciertos con la ficha correcta en primera posición | 18 | 0,466 | 0,594 | 0,699 |
| Consultas ajenas al dominio | 20 | 0,289 | 0,380 | 0,574 |

Las distribuciones se separan, pero se solapan: tres de las veinte consultas ajenas superan el mínimo de los aciertos, y las tres por coincidencia de un solo término entre la consulta y la ficha. Un umbral situado en 0,47 conserva los dieciocho aciertos y admite esos tres casos; situado en 0,58 excluye la totalidad del ruido y pierde nueve aciertos. La calibración es viable, aunque no inmediata, y el patrón de los solapamientos sugiere

combinar el umbral con los filtros de intención deterministas ya existentes en lugar de sustituirlos. Ambas mediciones corresponden a una prueba de concepto sobre los bancos descritos; la vía vectorial no participa del flujo del asistente y su integración se recoge en el apartado 5.17.

**5.16 Limitaciones reconocidas**

Corresponde declarar con honestidad lo que este trabajo no llegó a demostrar. En la práctica, varias validaciones resultaron menos limpias de lo previsto. La cascada no fue sometida a inyección controlada de fallos: la validación disponible es observacional, basada en incidentes reales de cuota agotada o proveedor caído, no en un experimento de ingeniería del caos propiamente diseñado (Yu et al., 2024). Tampoco existe instrumentación de consumo de tokens, de modo que las cifras de costo del apartado 3.13.3 son estimaciones bajo supuestos declarados y no mediciones; no es posible afirmar con los datos disponibles que el gasto real coincida con esas proyecciones, aunque el orden de magnitud parece razonablemente sólido.

A esto se suman otras limitaciones de alcance. Los extractores de datos están acoplados a la estructura de las fuentes de origen y se romperán si esas fuentes cambian su formato. La capacidad vectorial sobre pgvector permanece provisionada pero no activada. Y el alcance evaluado se limita a una sola carrera piloto, por lo que la generalización a una facultad completa está argumentada pero no verificada empíricamente. Este último punto admite discusión: la resolución determinista de entidades debería escalar, pero afirmarlo sin evidencia directa sería ir más allá de lo que los datos permiten sostener.

A lo anterior se añaden dos limitaciones funcionales que la validación puso a la vista. El reconocimiento de los verbos que activan las vías deterministas no tolera errores de tipeo, de modo que una consulta escrita con la palabra clave mal tecleada no alcanza el mecanismo que sí tiene el dato. Y la poda del esquema, cuando el clasificador de intención asigna un dominio equivocado, deja fuera del alcance del generador de SQL una tabla que contenía la respuesta. Ninguna de las dos compromete la arquitectura, pero ambas afectan la tasa de acierto observada y quedan reportadas en el apartado 5.1.

Hay además dos limitaciones de contexto que conviene declarar porque condicionan cualquier adopción institucional. La primera es que el servidor y el dominio empleados son recursos personales del autor, reutilizados para no incurrir en gasto durante el desarrollo del prototipo; ninguna dirección utilizada en este trabajo es institucional, y una implementación real exige migrar ambos servicios a infraestructura bajo control de

la Universidad. La segunda es que la línea telefónica del canal de WhatsApp está registrada a nombre del autor, con su cédula de identidad, tal como exige la plataforma. Eso significa que hoy el servicio depende de una persona natural y no de la institución, lo cual es aceptable para un prototipo académico y no lo es para una operación permanente. Ambas condiciones son de configuración y no de arquitectura, pero deben resolverse antes de cualquier despliegue institucional.

### 5.17 Trabajo futuro

El desarrollo de Compu IA dejó abiertas varias líneas de trabajo que no se ejecutaron en esta etapa, pero que pueden consolidar el prototipo en una solución institucional más completa.

La primera línea es la integración plena de la búsqueda vectorial sobre normativa redactada en prosa. La extensión pgvector ya está provisionada en la base de datos, pero hoy no participa del flujo conversacional; una etapa posterior debería incorporar recuperación semántica sobre reglamentos y resoluciones, calibrando un umbral de similitud que mantenga la restricción de cero falsos positivos y combinando ese umbral con los filtros de intención deterministas que el sistema ya utiliza.

La segunda línea es la instrumentación directa del consumo de tokens y de la degradación de los proveedores de inferencia. Las cifras económicas del apartado 3.13.3 se basan en estimaciones razonables, pero no en mediciones de producción; un registro fino de tokens por consulta y de eventos de fallo permitiría ajustar esos cálculos y afinar los márgenes de seguridad de la cascada.

Finalmente, los resultados se obtuvieron sobre la Carrera de Ciencias de la Computación. En caso de que la Facultad decida extender el sistema, sería necesario repetir las pruebas sobre las nuevas cohortes para confirmar que las tasas de acierto y los riesgos de alucinación se mantienen dentro de los márgenes aceptables.

## Referencias


Aggarwal, P., Madaan, A., Anand, A., Potharaju, S. P., Mishra, S., Zhou, P., Gupta, A., Rajagopal, D., Kappaganthu, K., Yang, Y., Upadhyay, S., Faruqui, M., & Mausam. (2023). *AutoMix: Automatically mixing language models*. arXiv. https://arxiv.org/abs/2310.12963

Anthropic. (2026). *Pricing* [Documentación técnica]. https://claude.com/pricing

Battaglini-Fischer, S., Srinivasan, N., Szarvas, B. L., Chu, X., & Iosup, A. (2025). *FAILS: A framework for automated collection and analysis of LLM service incidents*. arXiv. https://arxiv.org/html/2503.12185

Biswal, A., Patel, L., Jha, S., Kamsetty, A., Liu, S., Gonzalez, J. E., Guestrin, C., & Zaharia, M. (2024). *Text2SQL is not enough: Unifying AI and databases with TAG*. arXiv. http://arxiv.org/pdf/2408.14717.pdf

Buitrago Roa, L. M. (2025). *Sistema inteligente de búsqueda y coincidencia basado en Vector Stores* [Tesis de maestría, Universidad Europea]. Repositorio institucional Titula. https://titula.universidadeuropea.com/handle/20.500.12880/13618

Calahorro Jiménez, P. (2024). UJA-GPT: Un chatbot con generación aumentada por recuperación *para la comunidad universitaria* [Trabajo de fin de grado, Universidad de Jaén]. Repositorio institucional CREA. https://crea.ujaen.es/items/bc091df8-8c37-412e-ae87-d0e3fc8dba2f

Cevallos Minalla, C. J. (2025). Arquitectura e implementación de un Asistente Virtual basado en tecnologías OpenAI para el acceso eficiente a información académica. Revista Científica Arbitrada Multidisciplinaria PENTACIENCIAS, 7(3), 196–213. https://doi.org/10.59169/pentaciencias.v7i3.1485

Chen, L., Zaharia, M., & Zou, J. (2023). FrugalGPT: How to use large language models while reducing cost and improving performance. arXiv. https://arxiv.org/abs/2305.05176

Chu, S., Koe, J., Garlan, D., & Kang, E. (2024). *Integrating graceful degradation and recovery through requirement-driven adaptation*. arXiv. http://arxiv.org/pdf/2401.09678.pdf

Chu, X., Talluri, S., Lu, Q., & Iosup, A. (2025). An empirical characterization of outages and incidents in public services for large language models. *Proceedings of the 16th ACM/SPEC International Conference on Performance Engineering (ICPE '25)*. https://arxiv.org/pdf/2501.12469.pdf

Dhingra, D., Kothyari, M., Chakrabarti, S., & Sarawagi, S. (2023). CRUSH4SQL: Collective retrieval using schema hallucination for Text2SQL. En *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing* (pp. 14054–14066). Association for Computational Linguistics. https://aclanthology.org/2023.emnlp-main.868.pdf

Ding, D., Mallick, A., Wang, C., Sim, R., Mukherjee, S., Ruhle, V., Lakshmanan, L. V. S., & Awadallah, A. H. (2024). *Hybrid LLM: Cost-efficient and quality-aware query routing*. arXiv. https://arxiv.org/abs/2404.14618

Dong, L., Lu, Q., & Zhu, L. (2024). *AgentOps: Enabling observability of LLM agents*. arXiv. https://arxiv.org/pdf/2411.05285.pdf

Dong, Y., Mu, R., Jin, G., Qi, Y., Hu, J., Zhao, X., Meng, J., Ruan, W., & Huang, X. (2024). *Building guardrails for large language models*. arXiv. https://arxiv.org/abs/2402.01822

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (2023). *Retrieval-augmented generation for large language models: A survey*. arXiv. https://arxiv.org/abs/2312.10997

Godoy Viera, A. F., & Moreiro-González, J. A. (2026). Inteligencia Artificial Generativa: análisis conceptual y práctico del Modelo Generación Aumentada de Recuperación. Bibliotecas. Anales de Investigación, 21(Monográfico), 1–14. https://revistasbnjm.sld.cu/index.php/BAI/article/view/1133

González Arias, P. M. (2022). Diseño, desarrollo e implementación de una asistente virtual para la resolución de dudas sobre los procesos académicos de la Universidad Politécnica Salesiana – sede Cuenca utilizando inteligencia artificial y procesamiento de lenguaje natural [Trabajo de titulación, Universidad Politécnica Salesiana]. Repositorio institucional UPS. https://dspace.ups.edu.ec/handle/123456789/22027

Google. (2026). *Gemini API pricing* [Documentación técnica]. https://ai.google.dev/gemini-api/docs/pricing

Groq. (2026a). *Rate limits* [Documentación técnica]. https://console.groq.com/docs/rate-limits

Groq. (2026b). *Pricing* [Página de precios]. https://groq.com/pricing

Gurawa, P., & Dharmik, A. (2025). *Balancing content size in RAG-Text2SQL system* [Preprint]. arXiv. http://arxiv.org/pdf/2502.15723.pdf

Han, Y., Liu, C., & Wang, P. (2024). *Vector database management systems: Fundamental concepts, use-cases, and current challenges*. arXiv. https://arxiv.org/pdf/2309.11322.pdf

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004).Design Science in Information Systems Research. *MIS Quarterly* 1 March 2004; 28 (1): 75–105. https://doi.org/10.2307/25148625

Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., & Liu, T. (2023). *A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions*. arXiv. https://arxiv.org/abs/2311.05232

Jiang, D., Ren, X., & Lin, B. Y. (2023). LLM-Blender: Ensembling large language models with pairwise ranking and generative fusion. En *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023)*. https://arxiv.org/abs/2306.02561

Jin, Y., Yang, Z., Xu, X., Zhang, Y., & Ji, S. (2025). *Adaptive fault tolerance mechanisms of large language models in cloud computing environments*. arXiv. http://arxiv.org/pdf/2503.12228.pdf

Jing, Z., Su, Y., & Han, Y. (2024). *When large language models meet vector databases: A survey*. arXiv. https://arxiv.org/pdf/2402.01763.pdf

Lee, M.-C., Zhu, Q., Mavromatis, C., Han, Z., Adeshina, S., Ioannidis, V. N., Rangwala, H., & Faloutsos, C. (2024). *HybGRAG: Hybrid retrieval-augmented generation on textual and relational knowledge bases*. arXiv. https://arxiv.org/html/2412.16311

Li, J., Hui, B., Qu, G., Yang, J., Li, B., Li, B., Wang, B., Qin, B., Cao, R., Geng, R., Huo, N., Zhou, X., Ma, C., Li, G., Chang, K. C.-C., Huang, F., Cheng, R., & Li, Y. (2023). *Can LLM already serve as a database interface? A BIg bench for large-scale database grounded Text-to-SQLs*. arXiv. https://arxiv.org/pdf/2305.03111.pdf

Mala, C. S., Gezici, G., & Giannotti, F. (2025). *Hybrid retrieval for hallucination mitigation in large language models: A comparative analysis*. arXiv. https://arxiv.org/pdf/2504.05324.pdf

Manakul, P., Liusie, A., & Gales, M. J. F. (2023). SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. En *Proceedings*

*of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023)*. https://arxiv.org/abs/2303.08896

Meta Platforms. (2026a). Webhooks para la plataforma de WhatsApp Business [Documentación técnica]. https://developers.facebook.com/docs/whatsapp/cloud-api/guides/set-up-webhooks

Meta Platforms. (2026b). Instagram Platform [Documentación técnica]. https://developers.facebook.com/docs/instagram-platform

Meta Platforms. (2026c). Precios de la plataforma de WhatsApp Business [Documentación técnica]. https://developers.facebook.com/documentation/business-messaging/whatsapp/pricing

Moslem, Y., & Kelleher, J. D. (2026). *Dynamic model routing and cascading for efficient LLM inference: A survey* [Preprint]. HAL Open Science. https://hal.science/hal-05528300v1/file/Routing-Cascades-Survey-Feb2026.pdf

Moyolema Peralvo, K. A., Zambrano Vega, P. V., Simbaña Aguirre, R. A., & Molina Chalacán, L. J. (2026). Transformación educativa mediante inteligencia artificial conversacional en instituciones de educación superior ecuatorianas. Revista Conrado, 22(108), e5203. https://conrado.ucf.edu.cu/index.php/conrado/article/download/5203/4540/15237

Niu, C., Wu, Y., Zhu, J., Xu, S., Shum, K., Zhong, R., Song, J., & Zhang, T. (2024). *RAGTruth: A hallucination corpus for developing trustworthy retrieval-augmented language models*. arXiv. https://arxiv.org/pdf/2401.00396.pdf

Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., & Stoica, I. (2024). *RouteLLM: Learning to route LLMs with preference data*. arXiv. https://arxiv.org/abs/2406.18665

OpenAI. (2024, 26 de diciembre). *High error rates for ChatGPT* [Reporte de incidente]. OpenAI Status. https://status.openai.com/incidents/6bwlxnvdncnm

Park, C., Jiang, J., Wang, F., Paul, S., & Tang, J. (2024). *LlamaDuo: LLMOps pipeline for seamless migration from service LLMs to small-scale local LLMs*. arXiv. http://arxiv.org/pdf/2408.13467.pdf

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. Journal of Management

Information Systems, 24(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

pgvector contributors. (2025). *pgvector: Open-source vector similarity search for Postgres* [Repositorio de software]. GitHub. https://github.com/pgvector/pgvector

Ramírez, J., Torres, M., & Salazar, P. (2026). Infoxicación en los estudiantes de la facultad de Educación. Revista Educare et Natura, 4(1). https://revistas.uncp.edu.pe/index.php/educanatura/article/view/2661

Reece, M., Lander, T. E., Stoffolano, M., Sampson, A., Dykstra, J., Mittal, S., & Rastogi, N. (2023). *Systemic risk and vulnerability analysis of multi-cloud environments*. arXiv. https://arxiv.org/pdf/2306.01862.pdf

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering, 14(2), 131–164. https://doi.org/10.1007/s10664-008-9102-8

Torres Troya, J. D. (2024). Implementación de un Chatbot para Mejorar la Experiencia Estudiantil mediante Retrieval-Augmented Generation [Trabajo de fin de grado, Universidad Politécnica de Madrid]. Repositorio institucional UPM. https://oa.upm.es/90623/1/TFG_JOSEPH_DAVID_TORRES_TROYA.pdf

Yu, G., Tan, G., Huang, H., Zhang, Z., Chen, P., Natella, R., & Zheng, Z. (2024). *A survey on failure analysis and fault injection in AI systems*. arXiv. http://arxiv.org/pdf/2407.00125.pdf

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., & Radev, D. (2018). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and Text-to-SQL task. En *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 3911–3921). Association for Computational Linguistics. https://aclanthology.org/D18-1425.pdf

Zhang, S., Luu, A. T., & Zhao, C. (2024). *SynTQA: Synergistic table-based question answering via mixture of Text-to-SQL and E2E TQA.* arXiv. http://arxiv.org/pdf/2409.16682v2.pdf

# Anexos

## Anexo A. Interfaz del panel administrativo

El panel administrativo constituye el plano de control del sistema e incorpora gestión de cuentas, autenticación multifactor, auditoría de operaciones, administración de contenidos y herramientas de importación estructurada de syllabus. Permite consultar el estado del motor de inferencia activo, promover manualmente un proveedor a prioridad 1 dentro de la cascada, activar el modo de demostración, gestionar el padrón de cédulas y editar la información de aulas sin necesidad de acceso al servidor.

**Ilustración 8**

*Panel administrativo: selección del motor de inferencia activo.*

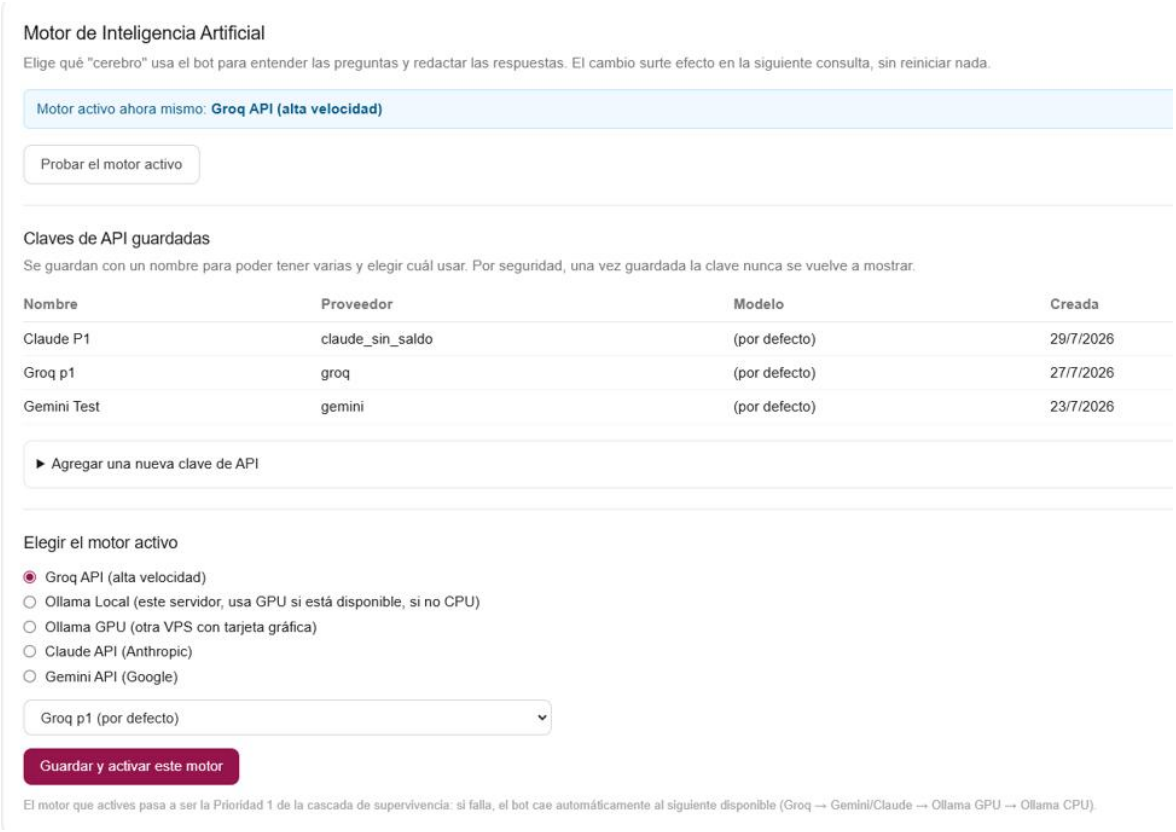


**Ilustración 9**

*Gestión del padrón docente con datos de demostración*

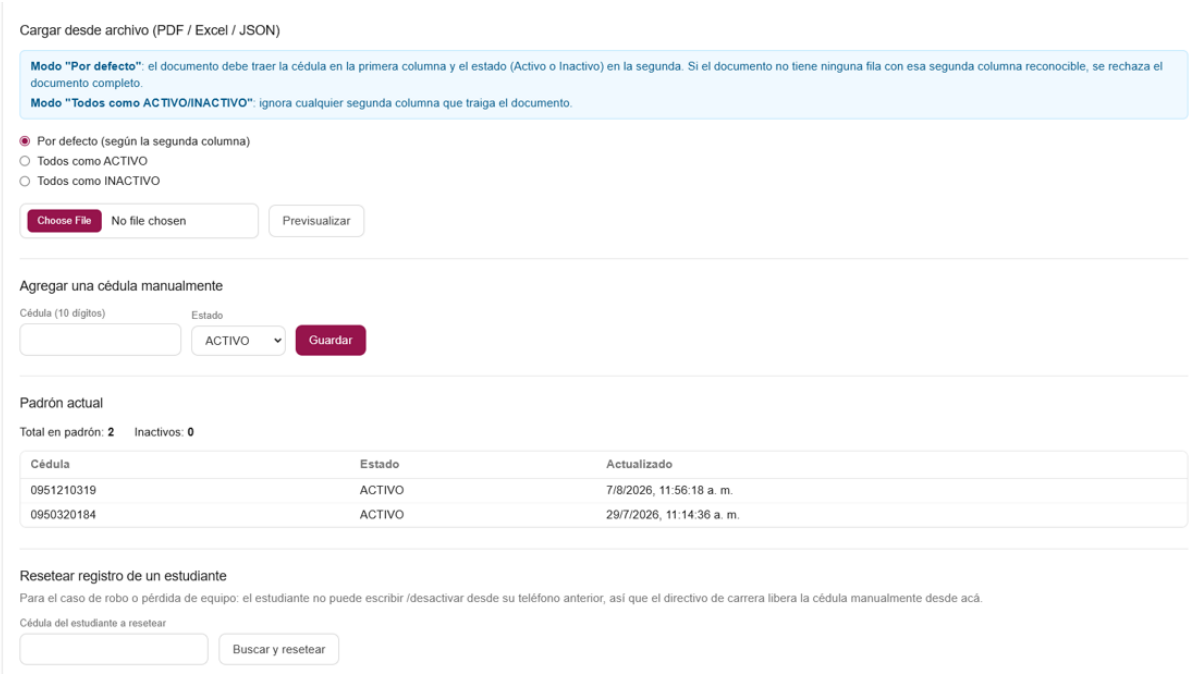
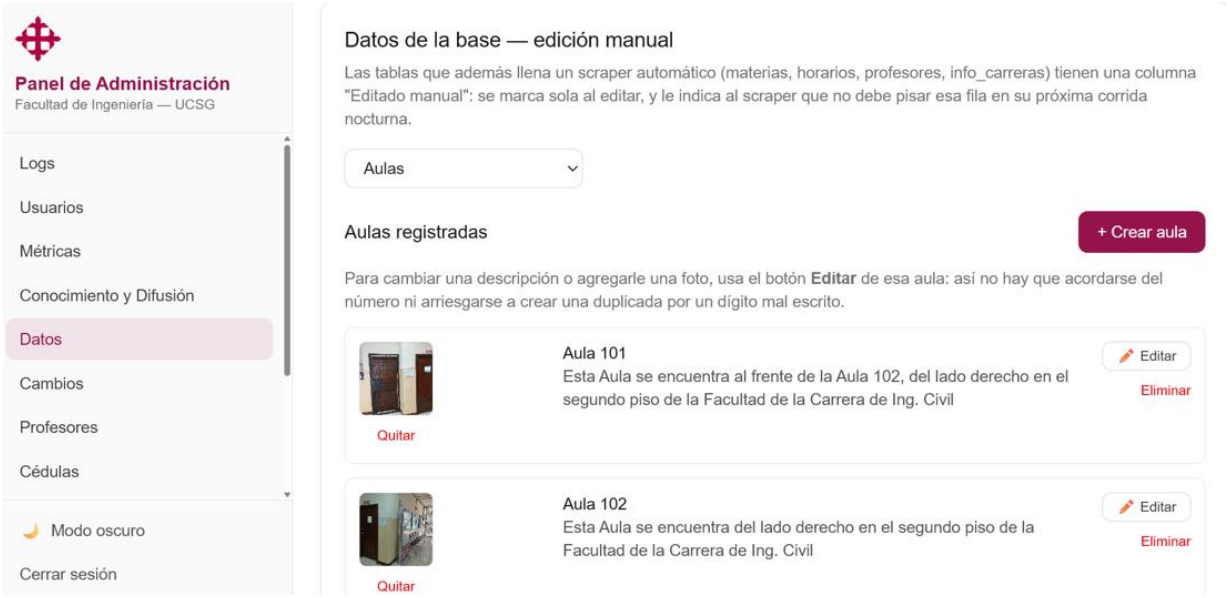



**Ilustración 10**

*Editor de información de aulas y carga de imágenes de ubicación*



## Anexo B. Pruebas de interacción en WhatsApp

Las siguientes capturas documentan el comportamiento del sistema en producción sobre el canal real de mensajería, tanto en el flujo de consulta académica como en la respuesta determinista de los fast-paths.

**Ilustración 11**

*Consulta de horario resuelta mediante RAG estructurado*

**Ilustración 12**
*Fast-path de ubicación: respuesta con fotografía real del lugar consultado*

# DECLARACIÓN Y AUTORIZACIÓN

Yo, **Tapia Alvarez**, **Victor Hugo** con C.C: # **0950320184** autor/a del Proyecto de Tecnologías de Información: "**Desarrollo de un asistente virtual inteligente basado en arquitectura RAG para la gestión de información académica y logística en la facultad de ingeniería en la Carrera de Ciencias de la Computación."** requerido para la obtención del título de **Ingeniero en Ciencias de la Computación** en la Universidad Católica de Santiago de Guayaquil.

1.- Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Guayaquil, 31 de agosto del 2026

Nombre: **Tapia Alvarez, Victor Hugo**

C.C: **0950320184**

# REPOSITORIO NACIONAL EN CIENCIA Y TECNOLOGÍA

## FICHA DE REGISTRO DE TESIS/TRABAJO DE TITULACIÓN

| | | | |
|---|---|---|---|
| **TEMA Y SUBTEMA:** | Desarrollo de un asistente virtual inteligente basado en arquitectura RAG para la gestión de información académica y logística en la facultad de ingeniería en la Carrera de Ciencias de la Computación. | | |
| **AUTOR(ES)** | Tapia Alvarez, Victor Hugo | | |
| **REVISOR(ES)/TUTOR(ES)** | García Sánchez, Roberto | | |
| **INSTITUCIÓN:** | Universidad Católica de Santiago de Guayaquil | | |
| **FACULTAD:** | Ingeniería | | |
| **CARRERA:** | Ingeniería en Ciencias de la Computación | | |
| **TITULO OBTENIDO:** | Ingeniero en Ciencias de la Computación | | |
| **FECHA DE PUBLICACIÓN:** | 31 de agosto del 2026 | **No. DE PÁGINAS:** | 72 p. |
| **ÁREAS TEMÁTICAS:** | Inteligencia artificial, Tecnología de la información, Servicios de información, Enseñanza superior. | | |
| **PALABRAS CLAVES/ KEYWORDS:** | Recuperación aumentada por generación, Text-to-SQL, enrutamiento en cascada, tolerancia a fallos, asistente conversacional, WhatsApp, UCSG. | | |


**Resumen:**

El acceso a información académica y logística de la Facultad de Ingeniería en la Carrera de Ciencias de la Computación de la Universidad Católica de Santiago de Guayaquil (UCSG) está distribuido entre varios canales oficiales: el portal académico, documentos de malla curricular, la cuenta institucional de Instagram y el conocimiento del personal de secretaría sobre la ubicación física de aulas y laboratorios. Cada canal cumple su función, pero un estudiante que necesita una respuesta puntual debe saber de antemano en cuál de esos canales buscarla. Este proyecto presenta el desarrollo de Compu IA, un asistente virtual conversacional accesible por WhatsApp, construido para centralizar esa consulta en una sola interfaz.El sistema combina una arquitectura híbrida compuesta por PostgreSQL con la extensión pgvector, un orquestador desarrollado en FastAPI, un panel administrativo web desarrollado en Next.js y TypeScript, un motor de inferencia local basado en Ollama bajo perfil opcional y procesos automatizados de extracción de información, con un enrutador de inferencia en cascada de cinco niveles (Groq, Gemini, Claude, Ollama con aceleración por GPU y Ollama en CPU), que mantiene el servicio operativo cuando alguno de los proveedores de inteligencia artificial falla o queda sin cuota disponible. La recuperación de información se resolvió mediante una arquitectura de tipo Text-to-SQL en lugar de una búsqueda semántica por similitud vectorial, con el fin de priorizar la exactitud del dato entregado sobre la flexibilidad de la búsqueda. Sobre esa recuperación se implementó un conjunto de validaciones deterministas (filtros de expresiones regulares, verificación estructural del SQL generado y un mecanismo que descarta cualquier respuesta del modelo que contradiga los datos reales recuperados), que reducen la dependencia de la corrección del sistema respecto del comportamiento del modelo de lenguaje.Se evaluó además la viabilidad económica de la arquitectura, comparando el costo mensual estimado de operación en la nube frente al costo de adquisición y mantenimiento de infraestructura propia con GPU dedicada, y se documentaron las limitaciones del prototipo y las líneas de trabajo futuro.


| | | |
|---|---|---|
| **ADJUNTO PDF:** | ☐ SI | ☒ NO |
| **CONTACTO CON AUTOR/ES:** | **Teléfono:**+593-999-616326 | E-mail: **victortapia0001@gmail.com** |
| **CONTACTO CON LA INSTITUCIÓN (C00RDINADOR DEL PROCESO UTE)::** | **Toala Quimí, Edison José** | |
| | **Teléfono:** +593-990-976776 | |
| | **E-mail:** edison.toala@cu.ucsg.edu.ec | |

| **SECCIÓN PARA USO DE BIBLIOTECA** | |
|---|---|
| **Nº. DE REGISTRO (en base a datos):** | |
| **Nº. DE CLASIFICACIÓN:** | |
| **DIRECCIÓN URL (tesis en la web):** | |