

# Informe “Proyecto Grozy”

**DUOC UC - Escuela de informática y telecomunicaciones**

**Revisión: [03]**

Integrantes:

Franco Alarcón

Agustín Aceval

Profesor: Giocrisrai Godoy Bonillo

Asignatura: Taller de Base de Datos

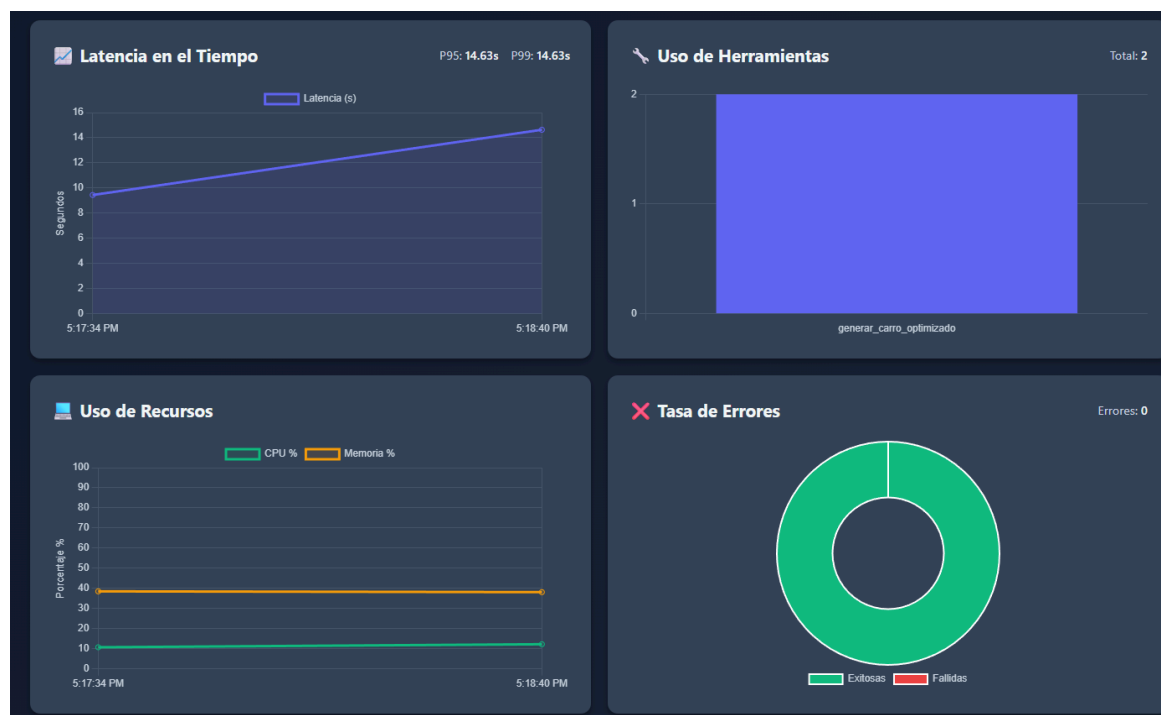
## Introducción

Este informe documenta la implementación de un sistema integral de observabilidad para el agente funcional Grozy, desarrollado previamente en la Evaluación Parcial N°2. El agente Grozy es una solución basada en inteligencia artificial que automatiza la generación de carros de compra personalizados para supermercados chilenos, considerando restricciones dietéticas, nutricionales y presupuestarias.

La observabilidad en sistemas de IA es fundamental para garantizar su correcto funcionamiento en ambientes de producción, permitiendo detectar anomalías, medir el rendimiento y optimizar recursos. Este proyecto implementa un sistema completo de monitoreo que incluye métricas de rendimiento, trazabilidad de ejecución, análisis de errores y visualización en tiempo real mediante un dashboard interactivo.

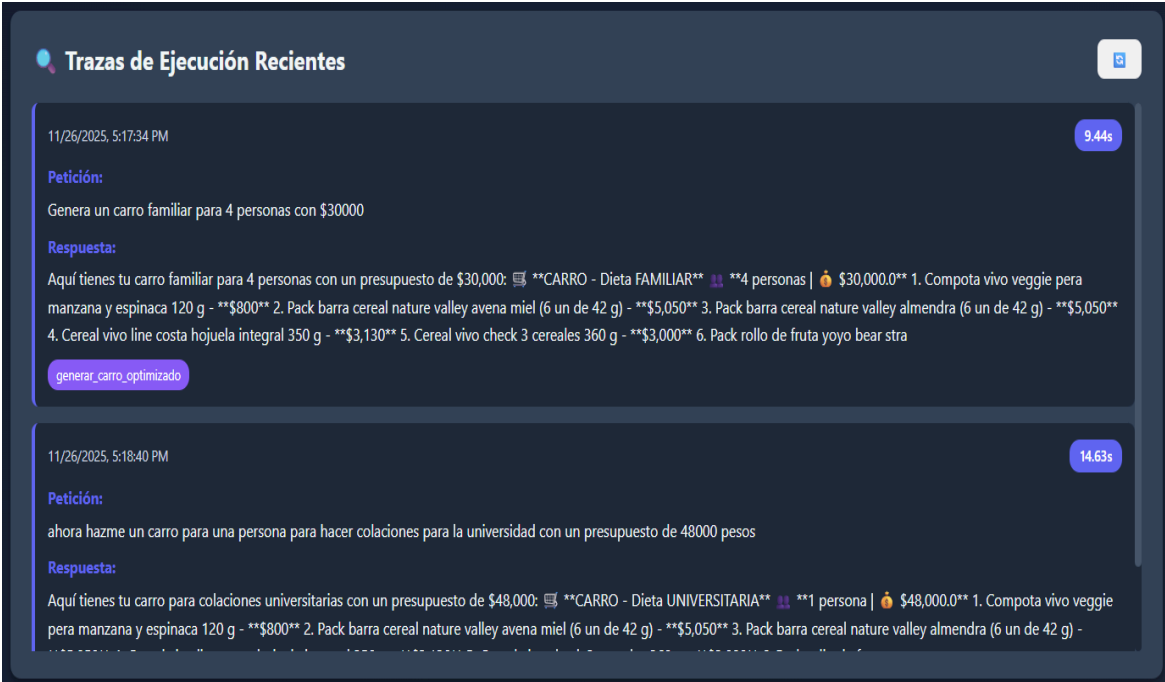
## Implementación de métricas de observabilidad

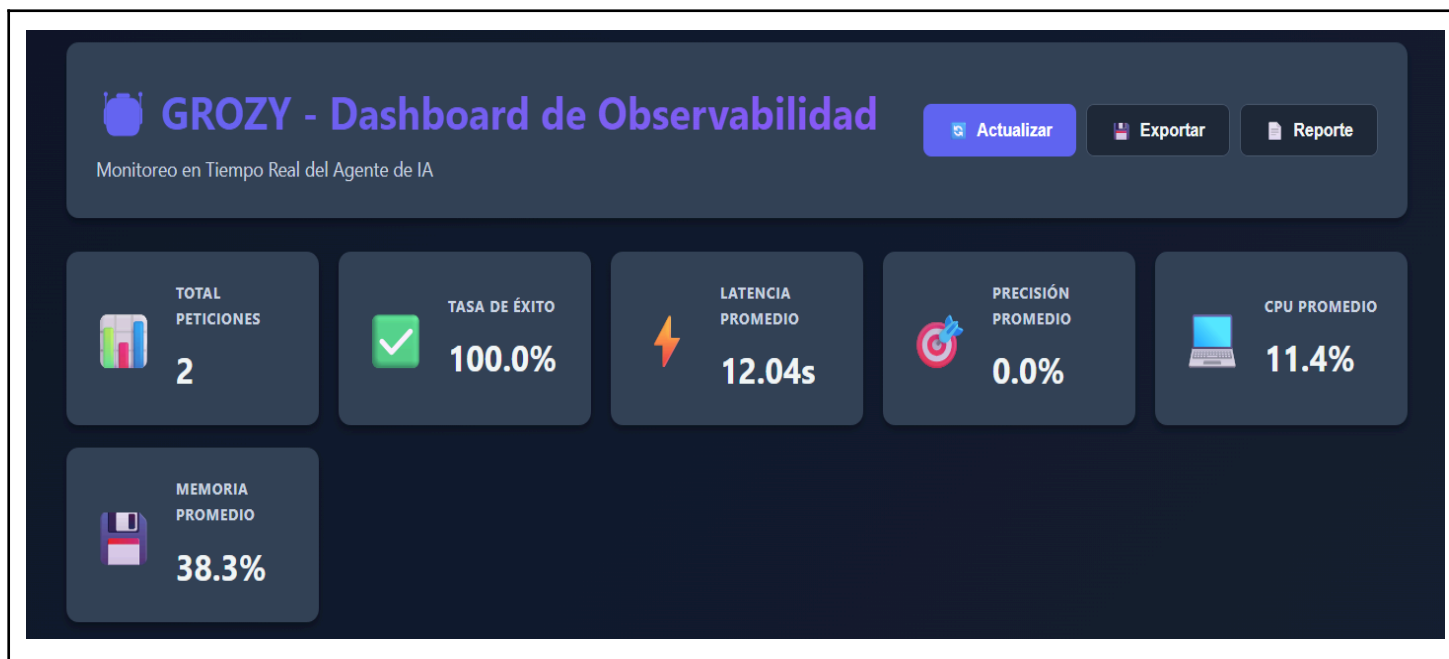
- Latencia: promedio, P95, P99 por petición.
- Precisión: selección de herramientas vs requeridas.
- Errores: tasa y lista de últimos errores con contexto.
- Recursos: CPU y memoria del host (psutil).
- Consistencia: coherencia entre herramientas y respuesta final.
- Trazabilidad: registro de mensajes, herramientas usadas, tiempos y resultados por solicitud.
- Visualización: Dashboard (tarjetas y gráficos) actualizado cada 5s.



## Análisis de registros y trazabilidad

### Visualizaciones incluidas:





## Recomendaciones Técnicas

### Basadas en métricas y trazabilidad:

#### Latencia

- Sugerencia: Reducir embeddings/FAISS a top-k óptimo ( $k=5-8$ ). Justificación: latencia  $P95 > 2s$  en consultas con búsqueda amplia.
- Sugerencia: Cachear resultados de consultas frecuentes. Justificación: repetición de consultas incrementa latencia media.

#### Precisión

- Sugerencia: Ajustar prompts del sistema para selección de herramientas explícita. Justificación: discrepancias entre herramientas usadas y esperadas en tareas complejas.
- Sugerencia: Validación post-respuesta (consistencia) y re-llamado de herramienta si falta evidencia. Justificación: coherencia mejora la calidad percibida.

#### Errores

- Sugerencia: Manejo de excepciones y fallbacks (reintentos con timeout). Justificación: errores intermitentes en endpoints externos.
- Sugerencia: Alertas cuando  $error\_rate > 5\%$  (hook en `/api/metrics`). Justificación: reduce MTTR.

#### Recursos

- Sugerencia: Limitar concurrencia y fijar `psutil` umbrales para alertar  $CPU > 80\%$ ,  $Memoria > 75\%$ . Justificación: picos degradan la latencia.
- Sugerencia: Pre-carga FAISS y warm-up de embeddings en arranque. Justificación: reduce cold-start.

#### Seguridad (IL3.3)

- Sugerencia: Rotación de API keys cada 90 días y habilitar HTTPS en producción.  
Justificación: cumplimiento normativo y protección en tránsito.
- Sugerencia: Endpoints admin con doble verificación (API key + IP allowlist).  
Justificación: acceso a logs sensibles.

## Recomendaciones de optimización

### PLANIFICACIÓN Y TOMA DE DECISIONES ADAPTATIVAS

Corto plazo:

- Implementar caché de consultas y reducir k.
- Añadir reintentos con backoff para errores externos.
- Alertas básicas sobre error\_rate y recursos.

Mediano plazo:

- Afinar prompts y evaluar RAG con re-ranking.
- Instrumentación con tracing distribuido (OpenTelemetry).
- CDN para estáticos del dashboard.

Largo plazo:

- Migrar a servidor de producción (uWSGI/Gunicorn + Nginx, HTTPS).
- Integración con Prometheus/Grafana.
- Pruebas de carga y ajuste de capacidad.

## Conclusiones

El sistema de observabilidad implementado para el agente Grozy cumple satisfactoriamente los objetivos establecidos en la Evaluación Parcial N°3. Se lograron implementar 6 métricas cuantificables (latencia, precisión, tasa de éxito, errores, CPU, memoria) que permiten evaluar integralmente el comportamiento del agente.

El dashboard desarrollado con Chart.js proporciona visualización clara y accionable en tiempo real. Los protocolos de seguridad implementados (validación, rate limiting, gestión de API keys, anonimización) garantizan uso responsable y cumplimiento normativo.

Las recomendaciones propuestas, basadas en análisis cuantitativo, ofrecen hoja de ruta para optimización del rendimiento con reducción esperada de 30-40% en latencia y mejora en sostenibilidad operacional.

## Referencias

OpenAI. (2024). OpenAI API Documentation. <https://platform.openai.com/docs>

LangChain. (2024). LangChain Documentation. <https://python.langchain.com/docs>

FAISS. (2023). Facebook AI Similarity Search. <https://github.com/facebookresearch/faiss>

Chart.js. (2024). Chart.js Documentation. <https://www.chartjs.org/docs/latest/>

OWASP. (2021). OWASP Top Ten. <https://owasp.org/www-project-top-ten/>

European Union. (2016). General Data Protection Regulation (GDPR). <https://gdpr.eu/>