

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
SANTIAGO - CHILE



“Diseño e implementación de un asistente conversacional basado en modelos de lenguaje para el análisis de datos de educación superior en Chile”

IGNACIO ANDRÉS ARAYA SALINAS

**MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL
TELEMÁTICO**

PROFESOR GUÍA:

NICOLÁS TORRES RUDLOFF

PROFESORA CORREFERENTE:

PATRICIA MORALES CALVO

OCTUBRE - 2025



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): Memoria o trabajo de título; Tesis de Postgrado;

Título del trabajo: Diseño e implementación de un asistente conversacional basado en modelos de lenguaje para el análisis de datos de educación superior en Chile

Nombre del candidato(a): Ignacio Andrés Araya Salinas

Carrera / Grado: Ingeniería Civil Telemática

Campus: CSSJ **Departamento:** Electrónica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Nicolás Torres Rudloff, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

- 6 meses; 12 meses; 2 años; 3 años; 5 años; 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 29/09/2025

; Firma:

Estudiante o Candidato(a):

Fecha: 29/09/2025

; Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Agradecimientos

A mi familia, gracias por su apoyo incondicional en cada etapa de esta carrera. Su comprensión, paciencia y aliento fueron fundamentales para mantenerme enfocado, incluso en los momentos más exigentes.

A mis amigos de la universidad, gracias por compartir no solo el estudio, sino también las dudas, las risas, los desvelos y las pequeñas grandes victorias que marcaron estos años. Haber recorrido este camino con ustedes hizo toda la diferencia.

A los profesores que formaron parte de mi camino académico, gracias por su disposición a enseñar con paciencia y por transmitir más que contenidos: motivación, curiosidad y visión de futuro.

A la Dirección de Transformación Digital de la Universidad Técnica Federico Santa María, por brindarme la oportunidad de realizar este desarrollo y por el apoyo entregado durante su ejecución.

Y, en especial, quiero dedicar un agradecimiento muy sentido a mi abuelita, cuya presencia fue una parte esencial de mi vida. Su cariño, su compañía y su ejemplo me acompañaron silenciosamente en este proceso, y siempre estarán conmigo.

Finalmente, a todas las personas que, de alguna manera, estuvieron presentes en este trayecto: gracias. Este logro también es de ustedes.

Dedico este trabajo con todo mi cariño a mi abuelita, quien me acompañó durante gran parte de este camino y a quien siempre llevaré en mi corazón.

Tu compañía, tu cariño y todo lo que compartimos mientras viví contigo fueron un pilar en mi vida. Aunque ya no estés físicamente, tu presencia me dio fuerza en los momentos más difíciles de este proceso.

Gracias por tanto.

Resumen

Este trabajo presenta el diseño e implementación de un asistente conversacional basado en modelos de lenguaje, orientado al análisis de datos de educación superior en Chile. El sistema integra datasets de benchmarking en formato CSV, los procesa en memoria mediante la librería Pandas y entrega resultados en forma de tablas, gráficos y explicaciones comprensibles. Para la orquestación del flujo conversacional se empleó LangGraph, lo que permitió estructurar etapas específicas como la detección de intención, la extracción y validación de filtros, el análisis de información y la visualización automática. La representación de resultados se realiza con Matplotlib, mientras que la interacción con el sistema se implementa a través de una API en FastAPI y un front-end en Streamlit, proporcionando una interfaz accesible y amigable para los usuarios.

El asistente fue diseñado para responder consultas en lenguaje natural relacionadas con indicadores clave de educación superior, tales como matrícula, egresados, empleabilidad, rango de ingresos, tasas de retención y programas académicos, entre otros, contribuyendo a la democratización del acceso a la información y al apoyo en la toma de decisiones académicas e institucionales.

Palabras Clave: Asistente conversacional, modelos de lenguaje, consultas en lenguaje natural, educación superior, matrícula universitaria, empleabilidad, tasas de retención, visualización de datos.

Abstract

This work presents the design and implementation of a conversational assistant powered by language models, aimed at analyzing higher-education data in Chile. The system integrates benchmarking datasets in CSV format, processes them in memory using the Pandas library, and delivers results as tables, charts, and comprehensible explanations. LangGraph was employed to orchestrate the conversational flow, enabling the structuring of specific stages such as intent detection, filter extraction and validation, information analysis, and automatic visualization. Results are rendered with Matplotlib, while interaction with the system is implemented through a FastAPI API and a Streamlit front end, providing an accessible and user-friendly interface.

The assistant was designed to answer natural-language queries related to key higher-education indicators—such as enrollment, graduates, employability, income ranges, retention rates, and academic programs, among others—thereby contributing to the democratization of access to information and supporting academic and institutional decision-making.

Keywords: Conversational assistant, language models, natural language queries, higher education, university enrollment, employability, retention rates, data visualization.

Glosario

Asistente conversacional Sistema que permite interactuar en lenguaje natural con los usuarios para responder consultas y generar resultados (tablas, gráficos y explicaciones) a partir de datos.

Benchmarking Estrategia de comparación de indicadores entre instituciones o programas con fines de análisis y mejora.

DataFrame Estructura tabular de datos de dos dimensiones provista por la librería Pandas.

FastAPI Framework de Python para construir APIs de alto rendimiento.

Filtro Restricción aplicada a un conjunto de datos (p. ej., por institución, programa, año o género) para refinar el análisis.

LangGraph Biblioteca para orquestar flujos conversacionales mediante grafos y subgrafos.

Lenguaje natural Forma de comunicación utilizada por las personas en su vida cotidiana, que los sistemas de procesamiento buscan interpretar para permitir interacciones más intuitivas entre usuarios y computadoras.

Matplotlib Librería de Python para la generación de gráficos y visualizaciones.

NL2SQL *Natural Language to SQL*. Subárea de NLIDB enfocada en traducir consultas en lenguaje natural a instrucciones SQL ejecutables sobre bases de datos relacionales..

NL2VIS *Natural Language to Visualization*. Área de investigación que traduce descripciones en lenguaje natural a visualizaciones gráficas (gráficos de barras, líneas, tablas dinámicas, etc.) mediante lenguajes o librerías de visualización..

NLIDB *Natural Language Interface to Databases*. Interfaces que permiten a los usuarios consultar bases de datos utilizando lenguaje natural en lugar de lenguajes formales como SQL..

Pandas Librería de Python para manipulación y análisis de datos, que provee estructuras como *DataFrame*.

Streamlit Framework de Python para crear aplicaciones web interactivas de ciencia de datos.

Tasa de retención Proporción de estudiantes que, tras haber ingresado a una carrera en un año determinado, permanecen matriculados en la misma institución al año siguiente.

Tabla de Contenido

1	Introducción	1
2	Marco General	2
2.1	Definición del Problema	2
2.2	Acercamiento a la Solución	3
2.3	Propuesta del Proyecto	4
2.4	Estado del Arte y Técnica	5
2.4.1	NLIDB / NL2SQL y consultas ejecutables	5
2.4.2	Lenguaje natural a visualización (NL2VIS)	5
2.4.3	Orquestación y control del flujo (ReAct, Toolformer, AutoGen vs. grafos de estado)	6
2.4.4	Confiabilidad: factualidad, consistencia y evaluación	7
2.4.5	Elección del modelo base: uso de un LLM compacto	8
2.4.6	Técnicas de prompting: few-shot prompting	8
2.4.7	Lineamientos identificados en la literatura	9
2.4.8	Síntesis	9
3	Objetivos	11
3.1	Objetivo General	11
3.2	Objetivos Específicos	11
4	Requisitos del Sistema	13
4.1	Requisitos Funcionales	13
4.2	Requisitos No Funcionales	14
4.3	Requisitos de Interfaces	15
4.3.1	Eventos Externos	15
4.3.2	Respuestas del Sistema	17
4.4	Requisitos de Ambiente	19

TABLA DE CONTENIDO

4.4.1	Hardware de Desarrollo	19
4.4.2	Software de Desarrollo	19
4.5	Perfiles de Usuario	20
5	Diseño y Arquitectura del Sistema	22
5.1	Diagrama de Contexto	22
5.2	Diagrama de Arquitectura	23
5.3	Enumeración de Módulos	24
5.3.1	Módulo de Interfaz de Usuario (MIU)	25
5.3.2	Módulo de API y Orquestación (MAO)	25
5.3.3	Módulo de Router de Intención y Datasets (MRID)	26
5.3.4	Módulo de Gestión de Datos (MGD)	26
5.3.5	Módulo de Filtrado y Análisis (MFA)	26
5.3.6	Módulo de Visualización (MVZ)	27
5.3.7	Módulo de Prompts y Lógica LLM (MPL)	27
5.4	Matriz de Requisitos Funcionales y Módulos	28
6	Verificación y Validación	29
6.1	Verificación	29
6.2	Validación	29
6.3	Plan de Testing	30
7	Gestión de Riesgos	31
7.1	Supuestos	31
7.2	Dependencias	31
7.3	Restricciones	32
7.4	Riesgos	32
8	Desarrollo	35
8.1	Conjunto de datos	35

TABLA DE CONTENIDO

8.1.1	Matrícula: <code>Benchmarking_enrollment_program.csv</code>	35
8.1.2	Egresados: <code>Benchmarking_stats_graduate_program.csv</code> .	36
8.1.3	Empleabilidad: <code>Benchmarking_national_employment.csv</code> .	36
8.1.4	Catálogo de programas: <code>Benchmarking_programs.csv</code>	37
8.1.5	Consideraciones de integración	37
8.1.6	Limitaciones	38
8.1.7	Síntesis del Conjunto de Datos	38
8.2	Carga y preparación de datos	39
8.3	Subgrafo de intención y selección de bases de datos	40
8.3.1	Motivación	40
8.3.2	Diseño e implementación	41
8.3.3	Detalles técnicos de implementación	43
8.3.4	Modelos y control de temperatura	43
8.3.5	Política de historial de conversación	44
8.3.6	Validación de salidas	44
8.3.7	Contrato de lectura y escritura de estado	44
8.3.8	Aristas condicionales	45
8.3.9	Evidencias de funcionamiento	45
8.4	Subgrafo de Filtrado	48
8.4.1	Estado y contratos	48
8.4.2	Nodos y flujo	49
8.4.3	Diseño del prompt de extracción (enfoque <i>few-shot</i>)	50
8.4.4	normalización y verificaciones utilizadas	61
8.4.5	Supuestos y limitaciones actuales	61
8.5	Subgrafo de visualización	63
8.5.1	Objetivo y contrato	63
8.5.2	Flujo y nodos	63
8.5.3	Criterios de diseño y robustez	65
8.5.4	Relación con el subgrafo de filtrado	66

TABLA DE CONTENIDO

8.5.5	Limitaciones y consideraciones	66
8.5.6	Ejemplos ilustrativos	66
8.5.7	Ejemplo 2: Barras por género en Ingeniería Civil Telemática (USM, San Joaquín, 2019–2024)	68
8.5.8	Ejemplo 3: Proporción por género (UTFSM–Vitacura, 2024) . .	71
8.5.9	Ejemplo 4: Top 10 programas con más graduados (2023)	73
8.5.10	Ejemplo 5: Listado puntual sin gráfico (tabla)	75
8.5.11	Ejemplo 6: Top 10 universidades con mejor empleabilidad a 1 año en Enfermería (2021)	76
8.5.12	Ejemplo 7: Top-5 de continuidad académica (barra)	78
8.6	Grafo padre	80
8.6.1	Estado orquestado	80
8.6.2	Nodos y responsabilidades	80
8.6.3	Lógica de control	81
8.6.4	Justificación del diseño	81
8.7	API Backend	82
8.7.1	Objetivo y alcance	82
8.7.2	Endpoints y contrato	82
8.7.3	Documentación interactiva y prueba rápida	84
8.7.4	Gestión de sesión e imágenes	85
8.7.5	Caché HTTP de estáticos	86
8.7.6	Normalización de la salida del grafo	86
8.7.7	Invocación del motor conversacional	86
8.7.8	CORS, estáticos y robustez	87
8.7.9	Flujo resumido	87
8.7.10	Consideraciones actuales	87
8.8	Interfaz de Usuario (Frontend)	88
8.8.1	Objetivo y alcance	88
8.8.2	Componentes principales de la interfaz	88

ÍNDICE DE FIGURAS

8.8.3	Gestión de estado y sesión	89
8.8.4	Flujo de interacción con el backend	90
8.8.5	Diseño visual	90
8.8.6	Manejo de tablas e imágenes	91
8.8.7	Experiencia de usuario	92
8.8.8	Ejemplo ilustrativo de interacción	93
8.8.9	Validación y pruebas	95
9	Conclusiones	97
9.1	Trabajo Futuro	98

Índice de figuras

5.1	Diagrama de contexto del sistema y sus componentes principales.	22
5.2	Diagrama de arquitectura del sistema y sus módulos interconectados.	23
8.1	Interfaz del frontend respondiendo a un saludo.	45
8.2	Registro del backend mostrando la clasificación de intención como <code>greeting</code>	46
8.3	Respuesta en frontend ante consultas fuera de dominio.	46
8.4	Registro del backend clasificando la consulta como <code>out_of_domain</code>	46
8.5	Interfaz del frontend mostrando una consulta válida dentro del dominio.	47
8.6	Registro del backend seleccionando el dataset adecuado y clasificando la intención como <code>domain_query</code>	47
8.7	Log del backend mostrando la interacción de <code>extract_filters</code> y <code>apply_filters</code> en una consulta sobre empleabilidad en Derecho (2024).	53
8.8	Caso de activación de <code>clarify</code> ante una consulta sin datos (retención en Derecho, UTFSM).	54
8.9	Ejecución completa de filtrado y análisis sobre el dataset <code>Benchmarking_programs</code>	57
8.10	Ejecución completa de filtrado y análisis sobre el dataset <code>Benchmarking_enrollment_progra</code>	

ÍNDICE DE FIGURAS

8.11 Evidencia extremo a extremo del subgrafo de visualización: desde filtros y plan analítico hasta decisión del tipo de gráfico y render final.	68
8.12 Evidencia del flujo de visualización para el caso de barras por género: desde la decisión del LLM (respeto a la petición explícita del usuario) hasta el render final y la tabla asociada en el frontend.	70
8.13 Evidencia del subgrafo de visualización en un caso de proporciones: filtro por campus, selección de <code>pie</code> por el LLM y render del gráfico con su tabla de soporte.	72
8.14 Evidencia del subgrafo de visualización para un ranking de programas: cambio de base, plan analítico, elección del tipo de gráfico y render final.	74
8.15 Caso sin gráfico: el módulo de visualización degrada a <code>table</code> cuando el usuario pide “solo la columna” (listado puntual).	76
8.16 Caso de empleabilidad: comparación categórica (top-10) → barras, con tabla visible para verificación.	77
8.17 Evidencia del pipeline de visualización para un <i>top-N</i> : el LLM sugiere <code>bar</code> , se construye la <code>chart_spec</code> y se renderiza el gráfico junto a la tabla.	79
8.18 Prueba del endpoint GET <code>/health</code> en la documentación interactiva de FastAPI. Se observa la respuesta <code>{"status": ".ok"}</code> con código 200.	83
8.19 Prueba del endpoint POST <code>/chat</code> con <code>curl</code> : respuesta 200 y JSON con <code>image_url</code> , <code>explanation</code> , <code>table_md</code> y <code>session_id</code>	84
8.20 Portada de <code>/docs</code> generada por FastAPI con los endpoints <code>/health</code> y <code>/chat</code>	84
8.21 Evidencia de gestión de sesión e imágenes: creación de archivos <code>viz_<ts>.png</code> por sesión en <code>static/plots/sessions/<session_id>/</code> y archivo global <code>viz_current.png</code>	85
8.22 Pantalla inicial del frontend. Se observa la barra lateral con la opción <i>Nueva conversación</i> , el título de la aplicación y el mensaje de bienvenida con un ejemplo de consulta.	91

ÍNDICE DE TABLAS

8.23	Flujo de respuesta en el frontend: tras la consulta se muestra el gráfico centrado con botón de descarga, una explicación en viñetas y la tabla de resultados (con opción de descargar CSV).	92
8.24	Respuesta del asistente mostrando un gráfico de barras centrado y el botón <i>Descargar imagen</i> . Esta opción permite exportar la visualización para informes o análisis adicionales. La barra lateral puede ocultarse con el control superior para disponer del área completa del chat.	93
8.25	Explicación breve en viñetas bajo el gráfico. Este acompañamiento textual sintetiza hallazgos y orienta la lectura del resultado sin requerir conocimientos técnicos.	94
8.26	Tabla interactiva presentada dentro de un <i>expander</i> (“Mostrar tabla (primeras filas)”) para no saturar la vista, con la opción de <i>Descargar tabla completa (CSV)</i> . Esto permite inspeccionar los datos de manera rápida y exportarlos para su uso en otras herramientas.	94
8.27	Validación manual con Pandas: cálculo de la tasa de empleabilidad a dos años en programas de Salud de la PUC (2024).	96
8.28	Respuesta generada por el chatbot para la misma consulta, mostrando tabla y explicación en lenguaje natural.	96

Índice de tablas

1	Eventos externos del asistente conversacional	17
2	Respuestas del sistema	18
3	Perfiles de usuario del asistente conversacional para educación superior	21
4	Módulos principales del sistema de análisis conversacional	25
5	Matriz de Requisitos Funcionales y Módulos (MIU: Interfaz de Usuario; MAO: API y Orquestación; MRID: Router de Intención y Datasets; MGD: Gestión de Datos; MFA: Filtrado y Análisis; MVZ: Visualización; MPL: Prompts y Lógica LLM).	28

1. Introducción

El análisis de datos en el ámbito de la educación superior constituye un eje estratégico para comprender tendencias, diagnosticar problemáticas y diseñar políticas que contribuyan al mejoramiento del sistema universitario [16]. En Chile, la expansión y masificación de la educación superior durante las últimas décadas ha dado lugar a una amplia disponibilidad de información relacionada con matrícula, egresos y empleabilidad [21, 15, 7, 8, 19].

No obstante, la existencia de grandes volúmenes de datos no se traduce necesariamente en un acceso efectivo ni en un uso generalizado. Si bien existe abundante información, esta puede ser difícil de obtener o procesar, lo que limita su utilización. La información suele encontrarse fragmentada en distintos repositorios y, en muchos casos, se presenta en formatos que requieren un nivel de especialización técnica para ser procesados adecuadamente. Este escenario genera una brecha significativa entre la disponibilidad de datos y su aprovechamiento real para la toma de decisiones académicas y de gestión institucional [6, 26].

Ante esta situación, surge la necesidad de contar con sistemas que no solo concentren la información, sino que también ofrezcan mecanismos de interacción más naturales e intuitivos. La posibilidad de formular consultas en lenguaje natural y recibir resultados claros en tablas, gráficos y explicaciones, abre la puerta a democratizar el acceso a los datos y ampliar su uso en distintos niveles de la comunidad universitaria [11, 23].

En este contexto, los asistentes conversacionales basados en modelos de lenguaje ofrecen una alternativa innovadora para simplificar el acceso a la información y entregar respuestas claras. Estas tecnologías convierten los datos en herramientas interactivas que apoyan la investigación, la docencia y la gestión, fortaleciendo la capacidad de las instituciones para para enfrentar los desafíos del sistema educativo en constante cambio.

2. Marco General

2.1. Definición del Problema

Las instituciones de educación superior enfrentan un escenario en el que la disponibilidad de datos es amplia, pero su uso efectivo está restringido. La fragmentación de la información, los formatos poco amigables para usuarios no técnicos y la dependencia de especialistas en análisis de datos dificultan que esta información sea utilizada en toda su potencialidad.

En muchos casos, el acceso a información crítica sobre matrícula, egresos, tasas de retención o empleabilidad está mediado por equipos técnicos que procesan las bases de datos originales y generan reportes o visualizaciones [19]. Este proceso, aunque efectivo, puede resultar lento y poco flexible, pues limita la autonomía de autoridades, académicos o estudiantes que deseen explorar directamente los datos.

La ausencia de herramientas accesibles genera una brecha tecnológica y cognitiva que restringe la democratización de la información. Datos que podrían ser un insumo cotidiano para la gestión académica y la toma de decisiones terminan subutilizados o disponibles únicamente en informes estáticos. Esto es especialmente problemático en un contexto donde la agilidad para analizar tendencias y evaluar indicadores se ha convertido en un factor estratégico para la competitividad y sostenibilidad institucional.

En síntesis, el problema que motiva este trabajo radica en la dificultad de acceso y uso de los datos de educación superior por parte de actores no especializados, lo cual limita la capacidad de las instituciones para aprovechar plenamente los recursos informativos disponibles en la planificación y el mejoramiento continuo.

2.2. Acercamiento a la Solución

El enfoque adoptado prioriza una interacción en lenguaje natural que reduzca la carga cognitiva de los usuarios y simplifique el acceso a la información institucional. La idea central es que las consultas expresadas en términos cotidianos puedan resolverse mediante un flujo conversacional que guíe, aclare y traduzca dichas consultas en operaciones analíticas sobre datos estructurados, presentando resultados legibles y contextualizados.

Este acercamiento se basa en un ciclo de diálogo que interpreta la intención de la pregunta y, cuando es necesario, solicita aclaraciones ante ambigüedades. Posteriormente, el sistema verifica la disponibilidad de los datos y la coherencia de los filtros aplicados como institución, programa, año o género, para elaborar una respuesta que combine texto explicativo con representaciones visuales o tabulares.

Asimismo, el diseño del sistema debe orientarse por criterios de usabilidad y pertinencia. Las respuestas deben expresarse en un lenguaje claro y comprensible, con mensajes que orienten al usuario en caso de errores o inconsistencias, y con tiempos de respuesta compatibles con un flujo conversacional natural. La visualización también resulta clave, ya que la elección de gráficos o tablas debe adecuarse al tipo de consulta, ya sea para mostrar tendencias, realizar comparaciones o detallar desagregaciones.

En conjunto, este enfoque permite perfilar un asistente conversacional como una alternativa pertinente al problema identificado. La integración de comprensión de lenguaje, análisis de datos y visualización en un flujo unificado orientado al usuario representa un paso concreto hacia la democratización del acceso a la información en educación superior.

2.3. Propuesta del Proyecto

En este escenario, se plantea el diseño e implementación de un asistente conversacional orientado al análisis de datos de educación superior en Chile. El sistema se concibe con una arquitectura modular que combinará procesamiento de datos, orquestación conversacional y visualización automática. Los datasets de benchmarking, en formato CSV, serán procesados en memoria mediante la librería Pandas, lo que permitirá realizar operaciones rápidas y flexibles sobre grandes volúmenes de registros. La interacción entre el usuario y el sistema se organizará a través de LangGraph, que estructurará el flujo en etapas específicas: detección de intención, extracción y validación de filtros, análisis de información y generación de visualizaciones. Los resultados se presentarán con Matplotlib en gráficos y tablas interpretables, mientras que la comunicación con los usuarios se implementará mediante una API en FastAPI y una interfaz en Streamlit, garantizando una experiencia accesible y amigable.

El alcance del proyecto contempla la resolución de consultas relacionadas con indicadores clave de educación superior, como matrícula, egresados, empleabilidad y tasas de retención, incluyendo la capacidad de realizar comparaciones entre instituciones, programas académicos y variables de género.

Si bien se plantea como un prototipo académico, su diseño modular y flexible abre la posibilidad de escalarlo en el futuro, incorporando nuevas fuentes de información y extendiendo su utilidad a otros contextos o instituciones interesadas en mejorar el acceso y análisis de datos educativos.

2.4. Estado del Arte y Técnica

Este trabajo se ubica en la intersección de: (i) interfaces conversacionales para datos estructurados (NLIDB/NL2SQL), (ii) orquestación de LLMs para flujos multi-paso con control y trazabilidad, y (iii) confiabilidad en razonamiento tabular (factualidad, consistencia y evaluación). El objetivo es habilitar consultas en lenguaje natural sobre datos públicos de educación superior en Chile, ejecutando los análisis en Pandas a partir de filtros estructurados en formato JSON (retornados por el LLM) y entregando tablas y gráficos reproducibles y auditables, con un enfoque de verificación por ejecución que asegura transparencia y trazabilidad en cada paso.

2.4.1. NLIDB / NL2SQL y consultas ejecutables

Las encuestas de Qin [17] y de Katsogiannis-Meimarakis y Koutrika [12] revisan los principales avances en interfaces de lenguaje natural a bases de datos. Entre las técnicas más efectivas se encuentran el *schema linking*, la codificación conjunta de la pregunta con el esquema de la base, la decodificación con restricciones y la validación por ejecución, todas orientadas a mejorar la precisión en consultas complejas con *joins* y agregaciones. En el dominio educativo, el patrón NL → SQL/Pandas → tabla materializada aporta transparencia al exponer la consulta y reproducibilidad de los resultados. En esta tesis, los metadatos de esquema y previsualizaciones de registros se inyectan explícitamente en el *prompt*; el LLM retorna un JSON con filtros y tipo de análisis que luego se ejecuta en Pandas, materializando tablas verificables.

2.4.2. Lenguaje natural a visualización (NL2VIS)

Las interfaces de lenguaje natural orientadas a la visualización buscan traducir descripciones textuales en representaciones gráficas consistentes. El survey de Shen et al. [20] ofrece una revisión exhaustiva del área e identifica desafíos como la ambigüedad semántica, la validación de tipos de datos y cardinalidad, y la necesidad de garantizar

correspondencias correctas entre atributos de datos y canales visuales. En línea con ese énfasis en la corrección semántica y la validación, esta tesis prioriza la fidelidad informativa por sobre la sofisticación visual cuando no hay garantías suficientes.

De manera complementaria, trabajos como Chat2VIS [14] muestran que modelos de propósito general (p. ej., GPT-3, Codex, ChatGPT) pueden generar automáticamente código de visualización si se proporcionan metadatos de esquema y un diseño cuidadoso del prompt. Sin embargo, la variabilidad de las salidas confirma la necesidad de aplicar validaciones y controles antes de la generación final.

Bajo estas premisas, la propuesta de esta tesis adopta un enfoque conservador: la selección de gráficos se rige por reglas verificables y se recurre a tablas como estrategia de respaldo cuando no existen garantías suficientes sobre la corrección de la visualización.

2.4.3. Orquestación y control del flujo (ReAct, Toolformer, AutoGen vs. grafos de estado)

La literatura reciente propone arquitecturas que superan el esquema lineal prompt → respuesta al incorporar memoria, herramientas y control explícito. ReAct intercala razonamiento y uso de herramientas para mitigar alucinaciones [27]; Toolformer explora el autoaprendizaje de llamadas a APIs [18]; y AutoGen formaliza configuraciones multi-agente con roles especializados y registro de estado [24].

En contraste, la orquestación mediante grafos de estado define nodos especializados y bordes condicionales, ofreciendo un flujo determinista, reentrante y trazable, lo que resulta especialmente útil en escenarios que requieren auditoría, como la ejecución de consultas y la generación de resultados verificables. En el ámbito educativo, Chomphooyod et al. (2025) reportan una aplicación de LangGraph para la generación automática de sílabos mediante flujos multi-agente con memoria de estado y ciclos de refinamiento controlados [5]. Esta tesis adopta LangGraph con un enfoque análogo, pero orientado al análisis de datos educativos: se emplean subgrafos encapsulados para

(i) la selección de datasets, (ii) la extracción y aplicación de filtros semánticos, y (iii) la visualización/tablas, todos con contratos de entrada/salida explícitos y registro de trazas para garantizar control y transparencia.

2.4.4. Confiabilidad: factualidad, consistencia y evaluación

En el análisis de indicadores educativos mediante LLMs resulta fundamental garantizar factualidad, consistencia y robustez. FacTool propone un marco unificado para la detección de errores factuales asistido por herramientas externas [4], ejemplo de cómo reducir alucinaciones en sistemas de generación automática.

En el campo de NL2SQL, se ha enfatizado la evaluación por ejecución: juzgar la calidad de un modelo en función del resultado efectivo de la consulta más que de la forma sintáctica de la predicción. Este principio fue empleado y popularizado por Seq2SQL [30] y posteriormente reforzado por Execution-Guided Decoding (EGD), que integra la ejecución parcial como guía durante la decodificación para reducir inconsistencias semánticas [22]. De manera complementaria, benchmarks como Spider [28] se han consolidado como referentes de evaluación estandarizada en NL2SQL, al ofrecer un marco riguroso para medir generalización y exactitud por ejecución.

Aunque estos trabajos se desarrollaron en contextos generales de NLP y bases de datos, sus principios son trasladables al dominio educativo. Este proyecto aplica la misma lógica de evaluación por ejecución a datasets de educación superior en Chile. Además, se añaden mecanismos específicos para reforzar la consistencia de los resultados: normalización de porcentajes (p. ej., $88\% \rightarrow 0.88$), validación de rangos temporales y un nodo de clarificación que se activa cuando un filtro no coincide con el vocabulario de las columnas. Estas medidas reducen errores antes de ejecutar el análisis y fortalecen la confiabilidad del sistema al trabajar con información sensible y de alto impacto en la toma de decisiones educativas.

2.4.5. Elección del modelo base: uso de un LLM compacto

La literatura reciente enfatiza que el rendimiento de los sistemas conversacionales no depende únicamente del tamaño del modelo, sino de la integración de técnicas de orquestación y verificación por ejecución.

En esta línea, Xu et al. [25] presentan un survey sobre *knowledge distillation* en LLMs, destacando cómo estas técnicas permiten reducir modelos de gran escala manteniendo un desempeño competitivo con menor latencia y costo computacional, lo cual es especialmente relevante en aplicaciones interactivas de análisis conversacional.

Por su parte, Katsogiannis-Meimarakis y Koutrika [12] muestran que en NL2SQL las mayores mejoras provienen de técnicas como *schema linking*, decodificación con restricciones y validación por ejecución, más que del simple escalamiento de parámetros.

2.4.6. Técnicas de prompting: few-shot prompting

El diseño del *prompt* ha demostrado ser un factor crítico para guiar el comportamiento de los LLMs. Desde la introducción de GPT-3, Brown et al. [3] evidenciaron que los modelos de lenguaje pueden aprender nuevas tareas a partir de pocos ejemplos en el *prompt*, fenómeno conocido como *few-shot prompting*. Posteriormente, Liu et al. [13] sistematizaron esta técnica en su survey sobre métodos de prompting, destacando que ejemplos cuidadosamente seleccionados pueden mejorar el rendimiento y la consistencia en múltiples tareas. De manera complementaria, Zhao et al. [29] propusieron mecanismos de calibración para escenarios *few-shot*, mostrando que ajustes en el diseño del *prompt* permiten reducir sesgos y mejorar la robustez.

En síntesis, la literatura reconoce al *few-shot prompting* como una estrategia eficaz para inducir comportamientos consistentes en los LLMs. En concordancia, este trabajo adopta plantillas de prompting enriquecidas con ejemplos de consultas y salidas estructuradas esperadas (en formato JSON), de modo que el modelo dispone de guías

explícitas que refuerzan la precisión en la extracción de filtros y configuraciones de análisis.

2.4.7. Lineamientos identificados en la literatura

De la revisión de la literatura se desprenden principios comunes para el diseño de sistemas de análisis conversacional con LLMs. En primer lugar, diversos trabajos coinciden en priorizar resultados ejecutados: privilegiar tablas derivadas de consultas efectivas frente a respuestas puramente textuales, con el fin de garantizar verificabilidad y trazabilidad [12, 17]. Asimismo, se resalta la importancia de inyectar metadatos de esquema y catálogos de valores en los *prompts*, lo que fortalece el enlazado semántico entre la consulta del usuario y las columnas disponibles [12].

Otro eje clave es la orquestación en pasos especializados, donde los LLMs combinan razonamiento con uso de herramientas externas y roles diferenciados, registrando entradas y salidas en cada etapa. Este patrón, presente en propuestas como ReAct, Toolformer y AutoGen, ha demostrado reducir alucinaciones y aumentar la transparencia del flujo de decisión [27, 18, 24].

Finalmente, en el ámbito de la visualización, se plantea la aplicación de principios de NL2VIS para guiar la generación automática de gráficos, recurriendo a tablas cuando no existen garantías suficientes sobre la corrección visual [20, 14].

Estos lineamientos inspiran el diseño del sistema propuesto en esta tesis, que operacionaliza dichas prácticas en el dominio educativo chileno mediante subgrafos especializados para filtrado, ejecución y visualización en el marco de LangGraph.

2.4.8. Síntesis

La convergencia entre NLIDB/NL2SQL, orquestación controlada por grafos de estado y verificación por ejecución configura un patrón prometedor para el análisis de datos educativos con requisitos de trazabilidad. Sin embargo, persisten vacíos en la in-

CAPÍTULO 2 : MARCO GENERAL

tegración de estas piezas dentro de flujos conversacionales auditables, especialmente en dominios sensibles como la educación superior, desde la interpretación hasta la visualización con controles semánticos. Ello justifica la propuesta metodológica desarrollada en este trabajo.

3. Objetivos

3.1. Objetivo General

El objetivo general de este trabajo es diseñar e implementar un asistente conversacional, basado en modelos de lenguaje y técnicas de orquestación modular, orientado a responder consultas en lenguaje natural sobre indicadores clave de la educación superior en Chile. El sistema busca integrar datos de *benchmarking* y entregar resultados en forma de tablas, gráficos y explicaciones comprensibles, reduciendo la dependencia de especialistas técnicos para acceder y analizar la información. De esta manera, se persigue democratizar el acceso a los datos educativos y apoyar la toma de decisiones en ámbitos académicos e institucionales.

3.2. Objetivos Específicos

Para alcanzar el objetivo general, se plantean los siguientes objetivos específicos:

- Diseñar una arquitectura modular de orquestación de subgrafos con LangGraph, que estructure el flujo conversacional en etapas diferenciadas: detección de intención, extracción y validación de filtros, análisis y visualización de resultados.
- Implementar mecanismos de procesamiento de lenguaje natural capaces de: (i) identificar la intención de la consulta, (ii) validar si está dentro del dominio, (iii) seleccionar los datasets relevantes y (iv) extraer y normalizar filtros de manera automática a partir de preguntas en lenguaje natural, garantizando robustez frente a consultas ambiguas o incompletas.
- Integrar y normalizar los datasets de *benchmarking* de educación superior en Chile, almacenados en archivos CSV y representados como DataFrames de Pandas, incluyendo información sobre matrícula, egresados, programas académicos y tasas de empleabilidad.

CAPÍTULO 3 : OBJETIVOS

- Desarrollar un submódulo de filtrado y análisis de datos que permita ejecutar operaciones avanzadas como agregaciones, tendencias temporales, distribuciones, comparaciones entre categorías, cálculos de retención y resúmenes estadísticos dinámicos.
- Diseñar un módulo de visualización automática, encargado de generar tablas en formato Markdown y gráficos dinámicos con `Matplotlib`, acompañados de explicaciones textuales producidas por el modelo de lenguaje.
- Implementar una interfaz de programación de aplicaciones (API) con `FastAPI`, que facilite la interacción con el asistente de forma sencilla, extensible y escalable, asegurando la interoperabilidad con otros sistemas.
- Incorporar un sistema de gestión de sesiones y persistencia temporal de imágenes, que garantice una experiencia de usuario robusta, evitando conflictos en consultas concurrentes y gestionando la limpieza automática de archivos.
- Desarrollar un *front-end* interactivo con `Streamlit`, que proporcione una interfaz amigable y accesible para los usuarios finales, permitiendo la interacción directa con el asistente mediante un entorno conversacional enriquecido con visualizaciones y descargas de datos.
- Incorporar mecanismos de robustez y clarificación automática de consultas, que detecten y comuniquen inconsistencias en filtros, propongan correcciones guiadas y apliquen estrategias de *fallback* para mantener la continuidad y calidad del análisis.

4. Requisitos del Sistema

4.1. Requisitos Funcionales

RF1: El sistema debe permitir a los usuarios formular consultas en lenguaje natural relacionadas con indicadores de educación superior en Chile (matrícula, egresados, empleabilidad, tasas de retención).

RF2: El sistema debe validar si la consulta está dentro del dominio soportado; en caso contrario, debe responder con un mensaje aclaratorio.

RF3: El sistema debe identificar y ajustar automáticamente los datasets relevantes para cada consulta, considerando el contexto de interacciones anteriores.

RF4: Para las consultas válidas, el sistema debe detectar la intención analítica e identificar los filtros relevantes (por ejemplo: institución, programa, año, género, área, campus).

RF5: El sistema debe procesar los datos en memoria utilizando la librería Pandas, generando resultados consistentes a partir de los datasets de benchmarking en formato CSV.

RF6: El sistema debe entregar resultados en tres modalidades complementarias: texto explicativo, tablas en formato Markdown y gráficos generados con Matplotlib.

RF7: El sistema debe permitir comparaciones entre instituciones, programas académicos y variables de género cuando la consulta lo requiera.

RF8: El sistema debe implementar un ciclo de diálogo, solicitando aclaraciones al usuario cuando la consulta sea ambigua o incompleta.

RF9: El sistema debe ofrecer una interfaz conversacional accesible mediante un frontend en Streamlit, conectado a una API en FastAPI.

RF10: El sistema debe gestionar sesiones de usuario, asegurando que cada consulta mantenga coherencia con el historial de interacción.

RF11: El sistema debe proporcionar explicaciones en lenguaje natural que acom-

pañen a las tablas y gráficos, facilitando la comprensión de los resultados por parte de usuarios no técnicos.

RF12: El sistema debe permitir la descarga de los resultados presentados, incluyendo las tablas en formato CSV y los gráficos en formato de imagen, para su uso posterior por parte de los usuarios.

4.2. Requisitos No Funcionales

RNF1 Usabilidad: La interfaz debe ser clara, comprensible y orientada a usuarios no especializados en análisis de datos o programación.

RNF2 Rendimiento: Las consultas deben resolverse en tiempos compatibles con un flujo conversacional natural (respuesta en pocos segundos).

RNF3 Escalabilidad: La arquitectura modular del sistema debe permitir incorporar nuevos datasets o funcionalidades sin reestructurar el núcleo de la aplicación.

RNF4 Mantenibilidad: El código debe estar organizado en módulos (procesamiento de datos, visualización, interfaz) que faciliten su actualización y extensión.

RNF5 Confiabilidad: El sistema debe manejar consultas incompletas, errores de formato y datos faltantes, entregando mensajes orientativos en lugar de fallar.

RNF6 Portabilidad: El sistema debe poder ejecutarse en distintos entornos (local, servidor institucional o contenedores Docker) sin necesidad de grandes cambios.

RNF7 Seguridad: El sistema no debe exponer información sensible y debe limitar el acceso únicamente a los datasets públicos o autorizados.

RNF8 Compatibilidad: El sistema debe integrarse con librerías y frameworks estándar de Python (Pandas, Matplotlib, FastAPI, Streamlit, LangGraph), garantizando interoperabilidad.

RNF9 Consistencia visual: Los gráficos y tablas deben mantener formatos numéricos uniformes (porcentajes, separadores de miles) y etiquetas claras y legibles.

4.3. Requisitos de Interfaces

4.3.1. Eventos Externos

En el asistente conversacional desarrollado, los eventos externos corresponden a las interacciones entre el usuario (front-end en Streamlit) y la API (FastAPI). Estas interacciones incluyen el inicio de una conversación, el envío de consultas en lenguaje natural, la exploración de valores, la aplicación de filtros, la solicitud de gráficos o tablas, y la descarga de resultados. Cada evento refleja una funcionalidad específica y mantiene una trazabilidad clara hacia la respuesta generada por el sistema.

La Tabla 1 detalla estos eventos, indicando su descripción, el iniciador, los parámetros involucrados y la respuesta esperada del sistema.

Evento	Descripción	Iniciador	Parámetros	Respuesta
Inicio de conversación	Visualización del mensaje de bienvenida y preparación del estado de sesión.	Usuario	—	Mensaje inicial mostrado en la interfaz.
Envío de consulta	El usuario ingresa una pregunta en lenguaje natural (ej.: evolución de matrícula, tasa de empleo).	Usuario	Texto libre de la consulta + historial	Petición enviada al endpoint /chat; el backend procesa y retorna resultados estructurados.
Exploración de valores únicos	Solicitud de listar valores posibles para una columna (ej.: instituciones, programas).	Usuario / Sistema	Nombre de columna y dataset	Lista de valores únicos presentada como tabla o mensaje.

CAPÍTULO 4 : REQUISITOS DEL SISTEMA

Evento	Descripción	Iniciador	Parámetros	Respuesta
Ajuste de filtros desde texto	El usuario especifica criterios en lenguaje natural (institución, área, años, género, campus).	Usuario	Cadenas con restricciones y rangos	Filtros normalizados y aplicados sobre los DataFrame.
Solicitud de gráfico/tablas	El usuario pide ver resultados como gráfico y/o tabla. El sistema decide el tipo de visualización más adecuado.	Usuario	Preferencias implícitas o explícitas (métrica/dimensiones)	Render de imagen (URL en /static/...), tabla en Markdown y explicación textual.
Explicación de resultados	El sistema genera una explicación textual que acompaña los gráficos y tablas para facilitar la interpretación.	Sistema	Resultado del análisis	Texto explicativo mostrado en la interfaz.
Descarga de tabla	Descarga de la tabla actualmente mostrada.	Usuario	—	Botón de descarga (CSV) habilitado en el front-end.
Descarga de imagen	Descarga del gráfico actualmente mostrado.	Usuario	—	Botón de descarga (PNG) habilitado en el front-end.
gestión de imágenes	La imagen de visualización se copia a un directorio de sesión con nombre versionado para evitar colisiones.	Sistema	Identificador de sesión	URL estable por sesión (/static/plots/sessions/...); limpieza automática de archivos antiguos.

Evento	Descripción	Iniciador	Parámetros	Respuesta
Limpieza por TTL de recursos	Eliminación periódica de imágenes antiguas y control del número máximo por sesión.	Sistema	Parámetros internos (TTL, límite por sesión)	Recursos obsoletos eliminados; sin acción del usuario.
Nueva conversación	Reinicia el historial y el contexto de la sesión actual en la interfaz.	Usuario	—	Estado reiniciado; se borra la historia en el front-end.
Error de validación de filtros	Filtros inválidos (columna inexistente, tipo no compatible, rango incorrecto).	Sistema	Filtros recibidos	Mensaje de error con indicación de corrección (ej.: columnas válidas).
Error de modelo/tiempo de espera	Fallo del modelo o timeout del backend/LLM al procesar la consulta.	Sistema	—	Mensaje de error y recomendación de reintento; no se pierde el historial.
Error de conexión	Fallo de red o indisponibilidad del backend.	Sistema	—	Mensaje de error visible en el front-end.

Tabla 1: Eventos externos del asistente conversacional

4.3.2. Respuestas del Sistema

En esta subsección se describen las respuestas del sistema ante los eventos previos. El flujo incluye detección de la validez de la consulta y de los datasets relevantes, extracción y aplicación de filtros, generación opcional de aclaraciones, y la construcción de salidas (tabla y/o gráfico con explicación) que se devuelven al front-end. La Tabla 2 resume cada respuesta.

CAPÍTULO 4 : REQUISITOS DEL SISTEMA

Respuesta	Descripción
Mensaje de bienvenida	El front-end muestra el saludo inicial y deja el campo de entrada habilitado para recibir la primera consulta.
Validación de intención	El sistema determina si la consulta corresponde a un saludo, una pregunta válida en dominio o un tema fuera de alcance. En caso de ser fuera de dominio, devuelve un mensaje explicativo.
Selección de datasets	Se identifican las bases de datos relevantes (matrícula, egresados, programas, empleabilidad) para procesar la consulta; si se trata de un seguimiento, se ajusta la selección anterior.
Extracción de filtros y análisis	El backend procesa el texto, detecta columnas y filtros (p. ej., institución, área, años), e infiere el tipo de análisis (tendencia, agregación, comparación).
Aplicación de filtros y resumen	Se normalizan tasas y fechas cuando aplica; se computa un resumen y/o vista previa de registros para confirmar el alcance de los datos filtrados.
Aclaración guiada	Si los filtros no devuelven registros o hay valores inválidos, el sistema genera un mensaje de clarificación con sugerencias de columnas/valores.
Generación de gráfico	Se renderiza una imagen (PNG) con la visualización solicitada y se expone vía ruta estática para su visualización y descarga.
Generación de tabla	Se construye una tabla en formato Markdown/GitHub y se entrega al front-end, que agrega el botón de descarga en CSV.
Explicación de resultados	El sistema acompaña tablas y gráficos con una interpretación textual en viñetas que facilita la comprensión de los hallazgos.
Gestión de recursos gráficos	Las imágenes se promueven a carpetas de sesión con nombre versionado y se eliminan automáticamente tras un tiempo de vida (TTL).
Reinicio de sesión	Al iniciar una nueva conversación, se limpia el estado del grafo y del front-end.
Manejo de errores	Ante un fallo de red o del backend, se devuelve un mensaje de error legible en la interfaz.

Tabla 2: Respuestas del sistema

4.4. Requisitos de Ambiente

Esta sección describe el hardware y software necesarios para el desarrollo y despliegue del asistente conversacional. Se listan únicamente las tecnologías principales utilizadas en el proyecto.

4.4.1. Hardware de Desarrollo

- Equipo con CPU multinúcleo.
- Memoria suficiente para cargar y manipular simultáneamente los cuatro datasets en memoria.
- Almacenamiento en SSD para mejorar tiempos de lectura/escritura.
- No se requiere GPU local; el procesamiento de lenguaje natural se delega al servicio externo.

4.4.2. Software de Desarrollo

- **Lenguaje base:** Python 3.x
- **Backend:** FastAPI
- **Orquestación conversacional:** LangGraph (y LangChain)
- **Procesamiento de datos:** Pandas, NumPy
- **Visualización:** Matplotlib
- **Frontend:** Streamlit

4.5. Perfiles de Usuario

El asistente conversacional para análisis de datos de educación superior está diseñado para satisfacer las necesidades de diferentes grupos de usuarios, cada uno con características y competencias específicas. Estos perfiles han sido definidos para garantizar que las funcionalidades del sistema respondan a los requerimientos técnicos y prácticos de su uso.

La Tabla 3 presenta una clasificación de los principales perfiles de usuario, abarcando directivos universitarios, analistas institucionales, investigadores, unidades de aseguramiento de la calidad y usuarios con fines pedagógicos. Esta segmentación asegura que el sistema sea accesible y útil para una amplia gama de contextos académicos y de gestión.

Perfil	Socioeconómico y Cultural	Ocupacional	Etario	Competencias y Características
Directivos y Decanos Universitarios	Formación profesional universitaria; acceso a plataformas e informes institucionales.	Gobierno universitario y planificación estratégica.	35–65 años	Requieren visualizaciones claras y resúmenes ejecutivos; no necesitan manejo técnico de datos.
Analistas Institucionales y de Planificación	Cultura de datos y calidad; experiencia en reportes y auditorías internas.	Acreditación y benchmarking interinstitucional.	25–55 años	Manejan filtros y cruces; requieren exportar tablas y trazabilidad de variables y periodos.
Investigadores y Académicos en Educación Superior	Comunidad académica con acceso a datos abiertos y bibliografía especializada.	Investigación aplicada y docencia universitaria.	30–65 años	Necesitan resultados auditables, series históricas y definiciones consistentes de indicadores.

CAPÍTULO 4 : REQUISITOS DEL SISTEMA

Perfil	Socioeconómico y Cultural	Ocupacional	Etario	Competencias y Características
Unidades de Aseguramiento de la Calidad y Acreditación	Equipos con formación en gestión de calidad y marcos normativos.	Auto-evaluaciones y preparación de evidencias para acreditación.	28–60 años	Énfasis en consistencia metodológica, documentación de fuentes y replicabilidad de consultas.
Docentes y Estudiantes (uso pedagógico)	Usuarios de contextos formativos universitarios, de perfil heterogéneo.	Apoyo a asignaturas de gestión, estadística y educación superior.	18–65 años	Requieren respuestas comprensibles y ejemplos guiados; valoran tablas y gráficos listos para clase.

Tabla 3: Perfiles de usuario del asistente conversacional para educación superior

5. Diseño y Arquitectura del Sistema

5.1. Diagrama de Contexto

El diagrama de contexto proporciona una visión clara de las interacciones entre los actores y componentes que participan en el sistema de análisis conversacional. La Figura 5.1 muestra cómo el usuario, el backend orquestado por grafos de estado, las bases de datos y el módulo de visualización se relacionan para entregar resultados tabulares y gráficos.

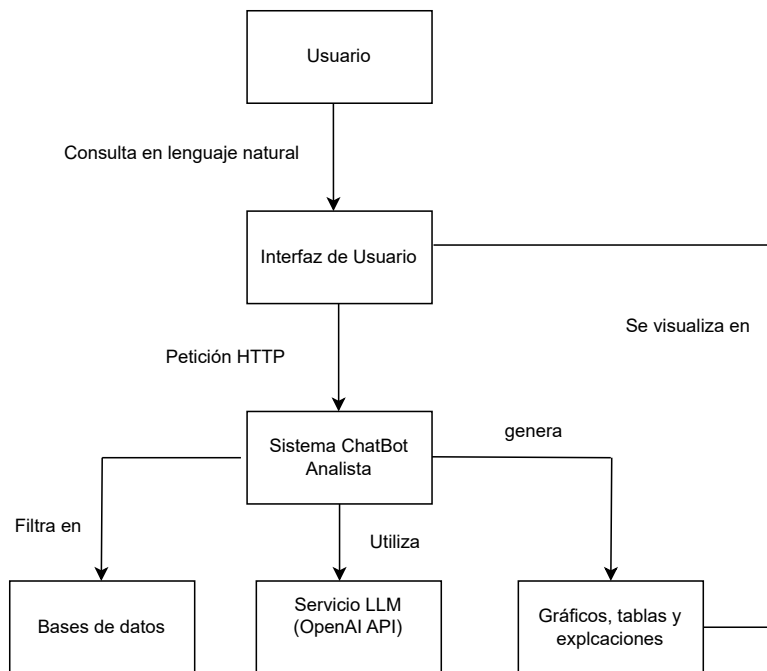


Figura 5.1: Diagrama de contexto del sistema y sus componentes principales.

Fuente: Elaboración propia.

5.2. Diagrama de Arquitectura

La arquitectura del sistema se diseñó de tal manera que cada componente se encargue de una parte específica del proceso, desde la captura de los datos hasta la interfaz de usuario. La Figura 5.2 ilustra cómo se interconectan los distintos módulos que componen el sistema y cómo se integran entre sí para garantizar un flujo de trabajo eficiente.

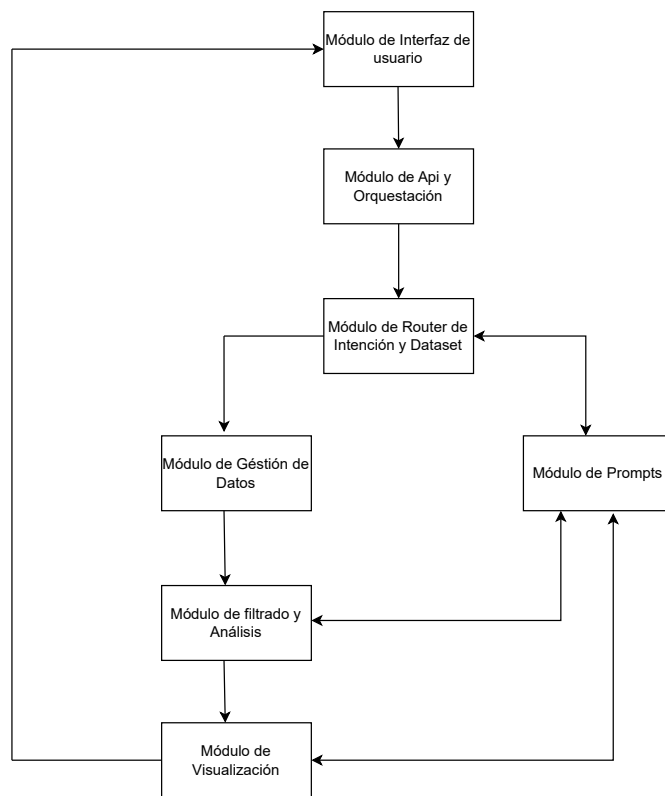


Figura 5.2: Diagrama de arquitectura del sistema y sus módulos interconectados.

Fuente: Elaboración propia.

5.3. Enumeración de Módulos

El sistema de análisis conversacional se compone de siete módulos principales, cada uno diseñado para cumplir una función específica dentro del flujo de trabajo. Estos módulos aseguran la cohesión y el correcto funcionamiento del sistema, desde la recepción de la consulta en lenguaje natural hasta la entrega de resultados en forma tabular, gráfica y textual interpretada.

La Tabla 4 resume el nombre de cada módulo, su propósito fundamental y la sección donde se describen con mayor detalle.

Módulo	Propósito	Sección
Módulo de Interfaz de Usuario (MIU)	Gestiona la interacción directa con el usuario, recibiendo las consultas en lenguaje natural y mostrando los resultados en forma de texto, tablas y gráficos.	5.3.1
Módulo de API y Orquestación (MAO)	Coordina el flujo de procesamiento a través de grafos de estado, gestionando la comunicación entre el frontend y el backend, así como la interacción entre los distintos módulos.	5.3.2
Módulo de Router de Intención y Datasets (MRID)	Clasifica la intención del usuario (saludo, fuera de dominio o consulta válida) y determina los datasets relevantes para dar respuesta a la consulta.	5.3.3
Módulo de Gestión de Datos (MGD)	Carga, limpia y normaliza los datasets de matrícula, egresados, empleabilidad y programas, dejándolos preparados para su análisis posterior.	5.3.4
Módulo de Filtrado y Análisis (MFA)	Extrae y aplica filtros a partir de la consulta en lenguaje natural, y ejecuta operaciones analíticas como agregaciones, tendencias, comparaciones y cálculos de retención.	5.3.5
Módulo de Visualización (MVZ)	Genera tablas y gráficos a partir de los resultados analíticos, selecciona el tipo de visualización más adecuado y proporciona explicaciones interpretativas.	5.3.6

Módulo	Propósito	Sección
Módulo de Prompts y Lógica LLM (MPL)	Gestiona las plantillas de prompts que guían al modelo de lenguaje en cada fase del flujo: clasificación de intención, selección de datasets, aplicación de filtros y generación de visualizaciones.	5.3.7

Tabla 4: Módulos principales del sistema de análisis conversacional

5.3.1. Módulo de Interfaz de Usuario (MIU)

- **Propósito:** Facilitar la comunicación entre el usuario y el sistema mediante una interfaz gráfica desarrollada en `Streamlit`.
- **Alcance:** Recibir consultas en lenguaje natural y presentar los resultados en forma de texto, tablas exportables y gráficos descargables.
- **Dependencias:** Módulo de API y Orquestación (MAO).
- **Estructura general:**
 - **Entrada:** Consulta en lenguaje natural ingresada por el usuario.
 - **Salida:** Respuesta procesada con explicaciones, tablas y gráficos interactivos.

5.3.2. Módulo de API y Orquestación (MAO)

- **Propósito:** Coordinar el flujo de información entre el frontend y los módulos internos del backend.
- **Alcance:** Implementar el grafo de orquestación que organiza la ejecución de los subgrafos (router, filtrado y visualización) y devolver la respuesta final lista para ser mostrada.
- **Dependencias:** Todos los módulos internos del sistema.
- **Estructura general:**
 - **Entrada:** Mensajes de usuario enviados desde la interfaz.

- **Salida:** Respuestas enriquecidas en formato estructurado (texto, tabla y/o gráfico).

5.3.3. Módulo de Router de Intención y Datasets (MRID)

- **Propósito:** Identificar la intención del usuario y determinar los datasets relevantes para responder a la consulta.
- **Alcance:** Clasificar las consultas como saludo, fuera de dominio o consulta válida; seleccionar y ajustar los datasets según el contexto conversacional.
- **Dependencias:** Módulo de Prompts y Lógica LLM (MPL), Módulo de Gestión de Datos (MGD).

5.3.4. Módulo de Gestión de Datos (MGD)

- **Propósito:** Administrar y preparar los datasets que sustentan el análisis.
- **Alcance:** Cargar los archivos de matrícula, egresados, empleabilidad y programas en formato CSV/Parquet; limpiar columnas de texto; transformar datos numéricos y temporales para su correcto procesamiento.
- **Dependencias:** Archivos de datos oficiales utilizados en el proyecto.

5.3.5. Módulo de Filtrado y Análisis (MFA)

- **Propósito:** Traducir la consulta del usuario en filtros y ejecutar operaciones analíticas sobre los datasets.
- **Alcance:** Aplicar filtros por institución, año, programa, región o área de conocimiento; realizar agregaciones, tendencias, comparaciones entre categorías y métricas de retención.
- **Dependencias:** Módulo de Gestión de Datos (MGD), Módulo de Prompts y Lógica LLM (MPL).

5.3.6. Módulo de Visualización (MVZ)

- **Propósito:** Presentar los resultados analíticos en formatos visuales comprensibles.
- **Alcance:** Generar tablas y gráficos adaptados al tipo de consulta; aplicar heurísticas para seleccionar la visualización más adecuada; proporcionar explicaciones interpretativas.
- **Dependencias:** Módulo de Filtrado y Análisis (MFA), Módulo de Prompts y Lógica LLM (MPL).

5.3.7. Módulo de Prompts y Lógica LLM (MPL)

- **Propósito:** Guiar al modelo de lenguaje mediante prompts cuidadosamente diseñados para cada fase del flujo.
- **Alcance:** Estandarizar la comunicación con el LLM en tareas críticas como clasificación de intención, selección de datasets, aplicación de filtros y generación de visualizaciones.
- **Dependencias:** Módulo de Router de Intención y Datasets (MRID), Módulo de Filtrado y Análisis (MFA), Módulo de Visualización (MVZ).

5.4. Matriz de Requisitos Funcionales y Módulos

Para garantizar una correcta alineación entre los requisitos funcionales del sistema y los módulos definidos para su implementación, se presenta a continuación la Tabla 5. Esta matriz muestra cómo cada módulo de la solución (MIU, MAO, MRID, MGD, MFA, MVZ y MPL) contribuye al cumplimiento de los objetivos funcionales.

	MIU	MAO	MRID	MGD	MFA	MVZ	MPL
RF1	X	X	X		X		X
RF2		X	X				X
RF3		X	X	X			X
RF4		X	X		X		X
RF5		X		X	X		
RF6	X	X			X	X	X
RF7		X	X		X	X	X
RF8	X	X	X		X		X
RF9	X	X					
RF10	X	X					
RF11						X	X
RF12	X	X				X	

Tabla 5: Matriz de Requisitos Funcionales y Módulos (MIU: Interfaz de Usuario; MAO: API y Orquestación; MRID: Router de Intención y Data-sets; MGD: Gestión de Datos; MFA: Filtrado y Análisis; MVZ: Visualización; MPL: Prompts y Lógica LLM).

6. Verificación y Validación

Esta sección establece el procedimiento utilizado para comprobar la coherencia de los resultados del asistente. La idea central es simple: por cada consulta en lenguaje natural realizada al sistema, se replica el filtrado con código determinista en Pandas y se comparan los resultados. El propósito es confirmar que el asistente entrega tablas consistentes con un análisis reproducible sobre los cuatro datasets públicos, incluyendo consultas con múltiples filtros, agregaciones y comparaciones dentro de cada dataset.

6.1. Verificación

- **Respuesta del asistente:** La consulta retorna una tabla y, cuando aplica, una imagen; la tabla se conserva como evidencia.
- **Reproducibilidad manual:** La misma consulta se replica con código en Pandas, aplicando filtros y agregaciones equivalentes.
- **Tiempo de respuesta:** Se comprueba que el asistente entregue resultados en un rango aceptable (segundos) incluso en consultas complejas con múltiples filtros y agregaciones.
- **Compatibilidad de ejecución:** Se valida que el asistente funcione de manera consistente en distintos navegadores y sistemas operativos.

6.2. Validación

- **Metodología de comparación:**
 - Ejecutar la consulta en el asistente y guardar la tabla devuelta.
 - Replicar la consulta con código manual en Pandas sobre los mismos datos.
 - Comparar ambas tablas tras normalizar formato (orden de filas/columnas y tipos de datos).
- **Criterios de aceptación:**
 - **Conteos y sumas:** coincidencia exacta.

- **Proporciones y tasas:** tolerancia absoluta $\leq 1e-6$.
 - **Estructura:** mismo número de filas y columnas y mismo orden cuando se solicita un Top-N u orden específico.
 - **Integridad de la tabla devuelta:** verificar que al parsear la salida del asistente (Markdown a DataFrame) no se pierdan registros ni se alteren valores.
- **Cobertura mínima de casos:**
- **Por dataset (4 en total):** al menos una serie temporal y un corte por entidad (institución, programa, área o campus, según corresponda).
 - **Consultas complejas:** al menos dos consultas con múltiples filtros y agregaciones, validando que la tabla del asistente coincide con la obtenida manualmente.
 - **Casos sin datos o valores inexistentes:** verificar que el asistente devuelve tabla vacía o mensaje de aclaración, y que el código manual produce el mismo resultado vacío.

6.3. Plan de Testing

- **Paso 1: Selección de consultas representativas** (por dataset y dos casos complejos con múltiples filtros).
- **Paso 2: Ejecución en el asistente** y registro de la tabla resultante.
- **Paso 3: Replicación en Pandas** con filtros y agregaciones equivalentes.
- **Paso 4: Comparación** contra los criterios de aceptación definidos.
- **Paso 5: Evidencia** (capturas y tablas).
- **Paso 6: Registro de discrepancias** Documentar diferencias entre asistente y código manual, indicando la causa probable (redondeo, normalización, ambigüedad) y la resolución adoptada.

7. Gestión de Riesgos

La gestión de riesgos de este proyecto se centra en identificar y mitigar los problemas más probables en el desarrollo e implementación del asistente conversacional para el análisis de datos de educación superior en Chile. El sistema combina un backend en FastAPI, un frontend en Streamlit, orquestación con LangGraph, procesamiento en memoria con Pandas, visualización con Matplotlib y el uso de un modelo de lenguaje externo. Este análisis aborda los supuestos, dependencias, restricciones y riesgos principales, junto con medidas de mitigación para garantizar la viabilidad y eficiencia del sistema.

7.1. Supuestos

- **Acceso a datasets públicos:** Se asume que los cuatro archivos de benchmarking estarán disponibles en formato tabular y sin cambios críticos en su estructura.
- **Procesamiento en memoria:** Se considera que el tamaño de los datasets es manejable en RAM, sin requerir cómputo distribuido.
- **Entorno de ejecución:** Se presupone un entorno estable con versiones compatibles de Python y librerías (Pandas, FastAPI, Streamlit, LangGraph, Matplotlib).
- **Disponibilidad del LLM:** Se asume que la API del proveedor de LLM estará operativa y con cuota suficiente.
- **Clientes compatibles:** Se espera que los usuarios accedan desde navegadores modernos y con conexión estable.

7.2. Dependencias

- **Proveedor de LLM:** La disponibilidad y el costo de la API son críticos para el funcionamiento del sistema.
- **Fuentes de datos:** La calidad y estabilidad de los datasets públicos condicionan la precisión de los análisis.

- **Infraestructura de despliegue:** El sistema depende de un servidor con CPU y RAM suficientes, hosting para backend y frontend, y configuración de seguridad básica (TLS, CORS).
- **Ecosistema de librerías:** Es necesario mantener compatibilidad entre versiones de Pandas, Matplotlib, FastAPI, Streamlit y LangGraph.
- **Parámetros de ejecución:** La configuración correcta de rutas de datos, URL del backend, credenciales del LLM y política de gestión de imágenes es indispensable para el funcionamiento del sistema.

7.3. Restricciones

- **Tiempo de respuesta:** El sistema debe entregar resultados en segundos, condicionado por la latencia del LLM y las operaciones de filtrado.
- **Recursos en memoria:** El rendimiento está limitado por la RAM disponible en el servidor, dado que el procesamiento se realiza en memoria.
- **Cuotas y costos del LLM:** El uso del modelo está restringido por la cuota asignada y el costo de la API.
- **Seguridad de archivos:** Solo se exponen imágenes bajo una ruta pública controlada, sin acceso a rutas internas ni a datos sensibles.
- **Datos procesados:** Únicamente se utilizan datasets públicos, sin manejo de información personal.

7.4. Riesgos

- **Consultas fuera de dominio o ambiguas:** El usuario puede formular preguntas vagas o ajenas al ámbito de educación superior chilena.
Mitigación: Clasificación de intención y flujo de clarificación antes de ejecutar filtros.
- **Clasificación y propuesta de filtros/datasets por el LLM:** El modelo puede clasificar mal la intención (in-domain/follow-up), elegir datasets no pertinentes o proponer filtros

incompletos o erróneos (institución, programa, año, campus).

Mitigación: Temperatura baja para reducir variabilidad; plantillas de prompt estables; validación contra catálogos reales (valores únicos por columna); normalización de texto; umbral para “abstenerse” y pedir aclaración cuando la confianza sea baja; registro de decisiones (logging) y revisión de discrepancias detectadas en la fase de verificación con Pandas.

- **Errores en extracción de filtros:** Aun con intención correcta, el LLM puede devolver filtros mal formados o valores inexistentes.

Mitigación: Validación sintáctica y semántica de filtros, sugerencia de valores válidos, reintentos con backoff y mensajes de corrección guiada.

- **Cambios en esquemas de datos:** Diferencias en nombres o tipos de columnas entre versiones de datasets pueden romper el flujo.

- **Mitigación:** Mapas de columnas canónicos, chequeo de contratos al cargar, pruebas de regresión al actualizar datos.

- **Calidad de datos:** Filas duplicadas, nulos o formatos heterogéneos afectan resultados.

Mitigación: Limpieza sistemática (coerción numérica, normalización de porcentajes, eliminación de duplicados).

- **Consistencia entre datasets:** Diferencias en codificación temporal o en denominaciones (p. ej., year vs year_aux, nombres de instituciones/programas) pueden generar resultados no comparables entre bases.

Mitigación: Alineación de codificación temporal, normalización de nombres y uso de catálogos canónicos; validación de cobertura antes de cualquier comparación entre resultados provenientes de distintas bases.

- **Rendimiento insuficiente:** Agrupaciones con alta cardinalidad o gráficos pesados degradan tiempos de respuesta.

Mitigación: Límites de filas, caché donde corresponda y uso de formatos más eficientes.

- **Selección de visualización no representativa:** El tipo de gráfico elegido puede no reflejar la naturaleza de los datos.
Mitigación: Reglas mínimas para validar ejes y métricas; fallback a tabla cuando falten condiciones; verificación previa de columnas clave.
- **Errores del LLM:** Respuestas irrelevantes o JSON mal formados pueden romper el flujo.
Mitigación: Extracción robusta de JSON, validación de salida y reintentos con backoff.
- **Gestión de sesiones y concurrencia:** Acumulación de archivos públicos o accesos simultáneos pueden saturar almacenamiento o colisionar en recursos compartidos.
Mitigación: Versionado por sesión, límites por usuario y limpieza periódica automática.
- **Dependencia del proveedor de LLM:** Caídas del servicio o cambios de precios impactan la operación.
Mitigación: Manejo de errores claro, monitorización y alternativas de modelo como contingencia.

8. Desarrollo

8.1. Conjunto de datos

El asistente conversacional se apoya en un conjunto de bases de datos públicas de educación superior en Chile, obtenidas de procesos de *benchmarking* institucional. Estas bases fueron normalizadas en formato CSV y contienen información sobre matrícula, egresados, empleabilidad e identificación de programas. En conjunto, permiten articular una visión integrada del ciclo de formación académica, desde el ingreso de los estudiantes hasta su inserción laboral.

8.1.1. Matrícula: `Benchmarking_enrollment_program.csv`

Este dataset concentra indicadores de matrícula y nuevos ingresos por programa, institución, sede y año. Posee aproximadamente 225,885 registros y 31 columnas, abarcando el periodo 2007–2024. Sus columnas principales incluyen:

Identificación institucional: `institution_name`, `campus_name`, `region`, `province`, `city`.

Características del programa: `program_name`, `global_level`, `field_knowledge`,
`program_plan_type`, `teaching_mode`, `session_name`, `program_accred`,
`institution_accred`.

Variables de matrícula: `total_enrolled`, `total_new_enrolled`, `female_enrolled`,
`male_enrolled`, `female_new_enrolled`, `male_new_enrolled`.

Temporalidad: `year_aux` (*también existe* `year`).

Este dataset permite caracterizar el perfil de ingreso y permanencia de los estudiantes, analizar diferencias de género, contrastar dinámicas entre campus y regiones, y estudiar tendencias históricas de matrícula por área de conocimiento o tipo de programa.

8.1.2. Egresados: `Benchmarking_stats_graduate_program.csv`

Contiene información sobre egresados y titulados en programas de pregrado y posgrado. Incluye 181,409 registros y 24 columnas, correspondientes al periodo 2007–2023. Sus columnas principales se organizan en:

Identificación institucional y curricular: `institution_name`, `program_name`, `global_level`, `program_class_1`, `teaching_mode`.

Desagregación por género: `total_graduates`, `total_females`, `total_males`, `total_nb` (*casi sin datos en el periodo*).

Localización: `campus_name`, `region`, `province`, `city`.

Temporalidad: `year`.

Este dataset es fundamental para analizar los resultados de titulación, identificar brechas de género en la finalización de estudios y estudiar la evolución temporal del egreso en distintas áreas de conocimiento. Además, posibilita vincular las tasas de titulación con la matrícula inicial y explorar trayectorias académicas por modalidad o nivel formativo.

8.1.3. Empleabilidad: `Benchmarking_national_employment.csv`

Este dataset reúne indicadores de inserción laboral y retorno económico de los egresados. Contiene 9,181 registros y 17 columnas, cubriendo el periodo 2019–2024. Sus variables principales se organizan en:

Identificación institucional: `institution_type`, `institution_name`, `area`, `accred_status`.

Identificación del programa: `program_gen_name`, `program_name`, `program_institution`, `program_gen_name_normalized`.

Empleabilidad: `emp_rate_1yr`, `emp_rate_2yr`.

Continuidad: `cont_rate` (continuidad de estudios) y `retention_rate_1yr` (retención estudiantil al primer año).

Resultados económicos: `income_brackets` (con frecuencia s/i) y `annual_tuition` (alta ausencia de datos).

Temporalidad: `year`.

Además de medir inserción laboral, esta base permite analizar *continuidad de estudios* y *retención estudiantil al primer año* mediante `cont_rate` y `retention_rate_1yr`. Estas tasas se reportan principalmente en escala [0,1] y, en algunos casos, presentan valores faltantes o la marca s/i, que debe tratarse como ausencia de información. Dado que la cobertura temporal es 2019–2024, los análisis longitudinales de continuidad y retención deben considerar dicha ventana y posibles rezagos de actualización.

Nota: En `annual_tuition` faltan datos en ~67,45 % de los registros y en `emp_rate_2yr` en ~37,87 %. Los valores s/i deben tratarse como faltantes lógicos (sin información).

8.1.4. Catálogo de programas: `Benchmarking_programs.csv`

Opera como catálogo maestro para estandarizar la denominación de programas. Consta de 6,159 registros y 13 columnas (sin dimensión temporal). Incluye, entre otras, las siguientes variables:

Identificadores y normalización: `program_sies_code`, `program_institution`,
`std_program_name_normalized`.

Información académica: `program_name`, `institution_name`, `global_level`, `field_knowledge`.

Este catálogo asegura la interoperabilidad entre las demás bases, unificando programas con denominaciones heterogéneas en matrícula, egresados y empleabilidad. Es indispensable para realizar cruces consistentes y comparaciones válidas entre fuentes.

8.1.5. Consideraciones de integración

Las cuatro bases comparten llaves como `institution_name`, `program_institution` y `program_name`, lo que posibilita su integración en consultas cruzadas. La base de matrícula utiliza principalmente `year_aux` (y también dispone de `year`), mientras que egresados y

empleabilidad emplean year. La clave `std_program_name_normalized` no está presente en empleabilidad, por lo que no es universal.

8.1.6. Limitaciones

La base de empleabilidad solo está disponible desde 2019, lo que restringe los análisis longitudinales en comparación con matrícula y egresados (2007 en adelante). El catálogo de programas no incluye dimensión temporal y cumple una función de referencia. Asimismo, los registros recientes pueden estar sujetos a rezagos de actualización oficial. En empleabilidad, las variables económicas presentan alta proporción de valores faltantes (`annual_tuition` ~67,45 %) y `emp_rate_2yr` presenta ausencia parcial de datos (~37,87 %). En egresados, la variable `total_nb` carece de representatividad (casi sin datos).

8.1.7. Síntesis del Conjunto de Datos

En conjunto, los cuatro datasets constituyen el insumo empírico central del asistente conversacional. Cada consulta en lenguaje natural se traduce en operaciones de filtrado, agregación y comparación sobre estas bases, generando respuestas en forma de tablas, gráficos y explicaciones. De esta manera, se articula un análisis integral, transparente y trazable del sistema de educación superior chileno.

8.2. Carga y preparación de datos

Una vez definidos los conjuntos de datos, el sistema requiere cargarlos en memoria de manera estructurada para que puedan ser consultados por el asistente conversacional. Para ello, se implementó un módulo específico denominado `loader.py`, encargado de la lectura, limpieza y normalización de los archivos.

El proceso sigue las siguientes etapas:

- **Localización de archivos:** el módulo identifica automáticamente todos los ficheros `.csv` almacenados en el directorio `data/`, cuya ruta se obtiene desde el archivo de configuración `.env`.
- **Limpieza de columnas de texto:** variables clave como `institution_name` y `program_institution` son normalizadas mediante funciones auxiliares para eliminar inconsistencias (espacios, acentos, mayúsculas/minúsculas).
- **Conversión de variables numéricas:** indicadores de matrícula, tasas de empleabilidad y aranceles (`emp_rate_1yr`, `annual_tuition`, `total_enrolled`, entre otros) son transformados al tipo `float`, unificando formatos decimales, y gestionando valores nulos.
- **Normalización temporal:** las columnas `year` y `year_aux` son convertidas a valores numéricos, garantizando consistencia entre bases que utilizan denominaciones diferentes para la dimensión temporal.

El resultado de este procedimiento se organiza en un diccionario de Python denominado `dfs`, donde cada clave corresponde al nombre de un dataset y cada valor es un `DataFrame` de Pandas. Por ejemplo, `dfs["Benchmarking_enrollment_program"]` contiene la base de matrícula y `dfs["Benchmarking_national_employment"]` la de empleabilidad.

De esta manera, los cuatro conjuntos de datos quedan disponibles en memoria para ser reutilizados en las etapas posteriores de selección de datasets, filtrado, análisis y visualización, conformando una capa de persistencia en memoria que facilita la integración con el flujo conversacional del asistente.

8.3. Subgrafo de intención y selección de bases de datos

La primera etapa de la arquitectura corresponde al subgrafo de intención y selección de bases de datos, cuya función principal es iniciar el flujo conversacional validando la pertinencia de las consultas y determinando el camino que debe seguir la interacción. Este componente actúa como una puerta de entrada estricta: ninguna consulta puede avanzar hacia los módulos de filtrado, análisis o visualización sin haber sido previamente clasificada como perteneciente al dominio de trabajo definido.

En este sentido, el subgrafo cumple un rol de control inicial que garantiza que únicamente las preguntas relevantes sean procesadas en etapas posteriores. Su operación permite distinguir con claridad entre consultas útiles y pertinentes, y aquellas que resultan irrelevantes o se encuentran fuera del alcance del sistema, evitando así un uso ineficiente de recursos computacionales y lógicos.

8.3.1. Motivación

El diseño de este subgrafo responde a dos motivaciones centrales. La primera es la necesidad de proteger al sistema frente a consultas irrelevantes o ajenas al dominio establecido. Si dichas consultas no fueran interceptadas de manera temprana, podrían propagarse hacia los módulos de análisis y generar errores, respuestas inconsistentes o un consumo innecesario de recursos. Preguntas relacionadas con política, clima, deportes o instituciones extranjeras constituyen ejemplos claros de interacciones que no deben ser procesadas por el asistente. El subgrafo actúa entonces como un filtro inicial, capaz de identificar estas situaciones y entregar una respuesta aclaratoria inmediata, interrumpiendo el flujo antes de que se activen procesos más costosos o inútiles.

La segunda motivación se vincula con la necesidad de preservar la coherencia conversacional. Es habitual que los usuarios formulen preguntas dependientes del contexto previo, como “¿y en 2022?” o “¿y en la PUC?”, las cuales carecen de sentido si se analizan de manera aislada. El subgrafo reconoce estos casos como consultas de seguimiento y ajusta la interacción en función del historial de conversación, asegurando que el diálogo se mantenga continuo, comprensible y contextualizado.

De este modo, el módulo no debe interpretarse como un complemento accesorio, sino como un filtro estructural que establece la frontera entre lo que el sistema está en condiciones de procesar y lo que debe descartarse. Solo cuando una consulta es clasificada como dentro del dominio, el flujo puede avanzar hacia la selección de datos y el análisis. Este enfoque no solo optimiza el uso de recursos, sino que también garantiza la trazabilidad de las decisiones y refuerza la fiabilidad global del asistente.

8.3.2. Diseño e implementación

El subgrafo fue implementado con la librería LangGraph y mantiene un estado denominado `AgentState`. Dicho estado actúa como la memoria persistente del subgrafo: se transmite de un nodo a otro, permitiendo que cada componente acceda al contexto conversacional y lo actualice según su función. En LangGraph, el estado es la unidad fundamental de comunicación: todo nodo recibe como entrada un objeto de estado, lo procesa y devuelve una versión enriquecida o modificada, que luego se propaga al resto del grafo. Esto asegura continuidad entre turnos y coherencia en las decisiones.

En este caso, `AgentState` conserva las variables necesarias para dar continuidad a la conversación:

- `last_query`: la última consulta del usuario.
- `previous_query`: la consulta inmediatamente anterior.
- `selected_datasets`: los conjuntos de datos que se han elegido para responder la pregunta actual.
- `previous_datasets`: los conjuntos utilizados en la interacción previa.
- `followup_context`: un indicador booleano que señala si la consulta corresponde a un seguimiento.

Sobre esta base se definieron tres nodos principales y la ruta es:

$$\text{router_intent} \rightarrow \begin{cases} \text{end} & \text{si } \text{greeting} \vee \text{out_of_domain} \\ \text{adjust_datasets_node} \rightarrow \text{end} & \text{si } \text{domain_query} \wedge \text{followup} \\ \text{select_dataset_node} \rightarrow \text{end} & \text{si } \text{domain_query} \wedge \neg \text{followup} \end{cases}$$

Descripción de los nodos:

- **router_intent**: clasifica la intención en tres categorías posibles: saludo, fuera de dominio o consulta válida (`domain_query`). Para ello construye un mensaje a partir de plantillas de prompt y el historial de conversación. Además, evalúa si la consulta es un seguimiento, comparándola con la interacción anterior.
- **adjust_datasets_node**: se activa únicamente en los casos de seguimiento. Su propósito es mantener los conjuntos previamente seleccionados y ajustarlos al nuevo contexto, evitando que el usuario deba repetir información.
- **select_dataset_node**: corresponde a consultas nuevas. Utiliza un prompt específico para identificar desde cero los conjuntos de datos relevantes para continuar con el análisis.

El diseño se apoya en plantillas de prompt almacenadas como archivos de texto independientes (`is_in_domain.txt`, `select_dataset.txt`, `is_followup.txt`, `adjust_datasets.txt`, entre otros). Estas plantillas encapsulan instrucciones claras y ejemplos mediante la técnica de *few-shot prompting*, lo que permite guiar al modelo para entregar siempre salidas en formato JSON, fácilmente validadas por el sistema.

Además, la función `safe_invoke` implementa un mecanismo de invocación robusto hacia el modelo de lenguaje, con reintentos y retroceso exponencial en caso de error. Este mecanismo garantiza resiliencia: incluso si ocurre una falla temporal, el sistema puede reintentar la ejecución y mantener el flujo estable.

Finalmente, el montaje del grafo se realiza con `StateGraph`. El nodo de entrada es `router_intent`, desde el cual se definen **aristas condicionales** (*conditional edges*) que

permiten decidir dinámicamente el camino a seguir según el contenido del estado. En este caso:

- Si la intención es un saludo o una consulta fuera de dominio, la arista conduce directamente al final del grafo.
- Si la consulta es válida y corresponde a un seguimiento, la arista lleva hacia `adjust_datasets_node`.
- Si la consulta es válida y corresponde a una nueva pregunta, la arista conecta con `select_dataset_node`.

Este uso de aristas condicionales dota al sistema de flexibilidad y evita bifurcaciones rígidas. A su vez, el grafo se compila con `MemorySaver` como mecanismo de checkpointing, que conserva el estado entre turnos de la conversación, permitiendo continuidad en sesiones largas y auditoría de los pasos intermedios.

Cabe destacar que este subgrafo puede ser integrado como un **subgrafo** dentro de un grafo mayor. En `LangGraph`, un subgrafo corresponde a un grafo encapsulado que se comporta como un nodo en otro grafo. Esto permite reutilizar su lógica, mantenerlo desacoplado del resto de la arquitectura y facilitar el trabajo modular. La comunicación entre el grafo padre y el subgrafo ocurre mediante el traspaso del estado: si ambos comparten las mismas claves de estado, la integración es directa; si no, es posible definir funciones de transformación para adaptar la información entrante y saliente.

8.3.3. Detalles técnicos de implementación

8.3.4. Modelos y control de temperatura

Se instancian dos clientes del modelo de lenguaje con propósitos diferenciados: `greeting_llm` con temperatura 0.5 para producir saludos naturales y ligeramente variados, y `llm` con temperatura 0.0 para pasos deterministas que requieren salidas en JSON (clasificación de intención y selección/ajuste de datasets). Esta separación reduce la entropía en etapas críticas y preserva naturalidad en respuestas no transaccionales.

8.3.5. Política de historial de conversación

Para controlar el contexto, el historial de mensajes se recorta a `MAX_HISTORY = 15`. Con ello se atenúan efectos no deseados de turnos muy antiguos y se contienen los costos de cómputo. De forma explícita, el estado almacena `last_query` y `previous_query`, lo que simplifica la detección de seguimientos y la generación de prompts que comparan consultas consecutivas sin depender únicamente del buffer de mensajes.

8.3.6. Validación de salidas

Las respuestas del modelo se esperan en JSON. Dado que algunos modelos pueden intercambiar texto libre, se extrae el primer bloque JSON mediante una expresión regular laxa (`{.*?}`) y luego se parsea con `json.loads`. Si el JSON es inválido o está ausente, el sistema activa rutas seguras: por ejemplo, en `router_intent` se mantiene `intent = domain_query` como valor por defecto y, en los nodos de selección/ajuste, se retorna una lista vacía y se guía al usuario con un mensaje explícito. Esta normalización de salidas evita estados incoherentes y facilita auditorías.

8.3.7. Contrato de lectura y escritura de estado

Cada nodo respeta un contrato explícito sobre `AgentState`:

- `router_intent`: lee `messages`, `previous_datasets`, `previous_query` y `last_query`; escribe `intent` y `followup_context`. Si determina saludo o fuera de dominio, adjunta un `AIMessage` con la respuesta y retorna.
- `adjust_datasets_node`: lee `previous_datasets`, `previous_query` y `last_query`; escribe `selected_datasets` y sincroniza `previous_datasets` con la nueva selección.
- `select_dataset_node`: lee `last_query`; escribe `selected_datasets`; ante selección vacía, inserta un mensaje instructivo para que el usuario precise el requerimiento.

Este contrato explícito minimiza efectos laterales, simplifica pruebas unitarias por nodo y facilita la evolución del grafo.

8.3.8. Aristas condicionales

La función `next_after_router` concentra la lógica de transición tras `router_intent`. Si la intención es saludo o fuera de dominio, el flujo concluye; si existe `followup_context` verdadero, el control pasa a `adjust_datasets_node`; en caso contrario, a `select_dataset_node`. Al externalizar la lógica de enrutamiento, el cuerpo del nodo se mantiene enfocado en la clasificación y se mejora la legibilidad del grafo.

8.3.9. Evidencias de funcionamiento

Para validar el correcto funcionamiento del subgrafo, se realizaron pruebas controladas con tres tipos de consultas: saludos, preguntas fuera de dominio y preguntas válidas dentro del dominio. A continuación se presentan ejemplos gráficos que ilustran cómo el sistema clasifica y responde en cada caso.

Caso 1: saludo. Cuando el usuario ingresa un saludo simple (ejemplo: “Hola”, “¿Cómo estás?”), el subgrafo clasifica la intención como `greeting` y retorna un mensaje cordial sin activar módulos posteriores.

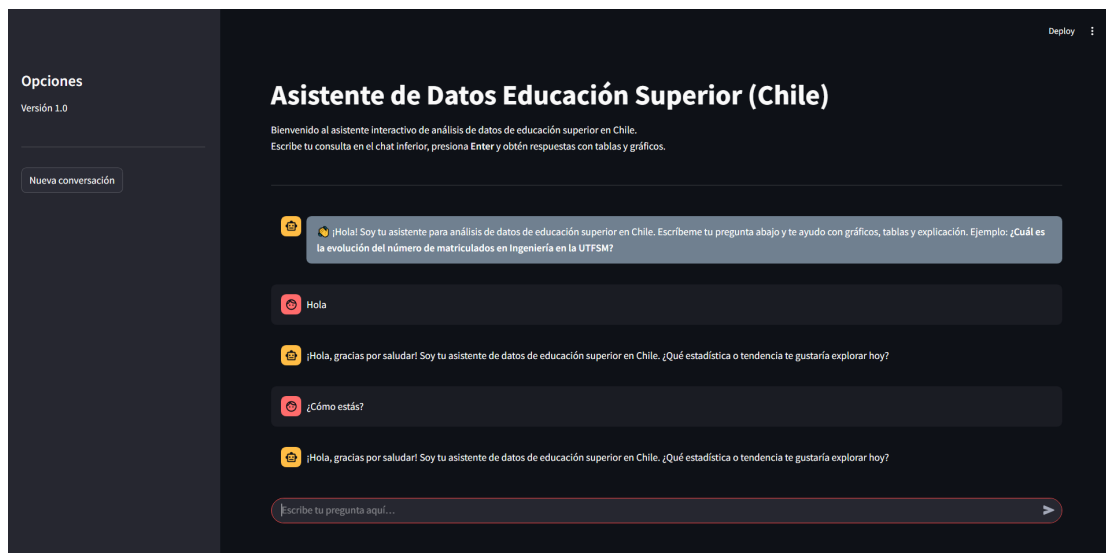


Figura 8.1: Interfaz del frontend respondiendo a un saludo.

Fuente: Elaboración propia.

CAPÍTULO 8 : DESARROLLO

```
2025-09-20 22:21:11,044 INFO [router_intent] last_query='Hola' previous_query='Hola'
2025-09-20 22:21:12,196 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-20 22:21:12,203 INFO router_intent intent=greeting
2025-09-20 22:21:13,549 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-20 22:21:13,552 INFO [/chat] session=3264bd895224b4dc image_url=no table_md=no expl_len=140
172.17.0.1:57118 - "POST /chat HTTP/1.1" 200
2025-09-20 22:21:20,119 INFO [router_intent] last_query='¿Cómo estás?' previous_query='¿Cómo estás?'
2025-09-20 22:21:21,365 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-20 22:21:21,366 INFO router_intent intent=greeting
2025-09-20 22:21:22,675 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-20 22:21:22,677 INFO [/chat] session=3264bd895224b4dc image_url=no table_md=no expl_len=140
```

Figura 8.2: Registro del backend mostrando la clasificación de intención como greeting.

Fuente: Elaboración propia.

Caso 2: fuera de dominio. Consultas como “¿Cuál es la mejor universidad de Latinoamérica?” o “¿Quién será el presidente de Chile este año?” se clasifican como `out_of_domain`. El sistema responde aclarando su especialidad y bloquea el acceso a las fases de análisis.



Figura 8.3: Respuesta en frontend ante consultas fuera de dominio.

Fuente: Elaboración propia.

```
2025-09-20 22:24:50,880 INFO [router_intent] last_query='¿Cuál es la mejor universidad de Latinoamérica?' previous_query='¿Cuál es la mejor universidad de Latinoamérica?'
2025-09-20 22:24:51,896 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-20 22:24:51,902 INFO router_intent intent=out_of_domain
2025-09-20 22:24:53,370 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-20 22:24:53,372 INFO [/chat] session=3264bd895224b4dc image_url=no table_md=no expl_len=233
172.17.0.1:51164 - "POST /chat HTTP/1.1" 200
127.0.0.1:52038 - "GET /health HTTP/1.1" 200
2025-09-20 22:25:05,471 INFO [router_intent] last_query='¿Quién será el presidente de Chile este año?' previous_query='¿Quién será el presidente de Chile este año?'
2025-09-20 22:25:06,232 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-20 22:25:06,239 INFO router_intent intent=out_of_domain
2025-09-20 22:25:07,170 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
```

Figura 8.4: Registro del backend clasificando la consulta como `out_of_domain`.

Fuente: Elaboración propia.

Caso 3: dentro del dominio. En una consulta válida como “¿Cuál es la tasa de empleo a dos años para Ingeniería Comercial en la UTFSM?”, el subgrafo la clasifica como

CAPÍTULO 8 : DESARROLLO

domain_query, selecciona el dataset correspondiente y habilita el paso hacia la etapa de filtrado y análisis.



Figura 8.5: Interfaz del frontend mostrando una consulta válida dentro del dominio.

Fuente: Elaboración propia.

```
127.0.0.1:4104 - GET /health HTTP/1.1 200
2025-09-20 22:13:51,688 INFO [router_intent] last_query='¿Cuál es la tasa de empleo promedio a dos años para Ingeniería Comercial en la Universidad Técnica Federico Santa María?' previous_query='¿Cuál es la tasa de empleo promedio a dos años para Ingeniería Comercial en la Universidad Técnica Federico Santa María?'
2025-09-20 22:13:52,657 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-20 22:13:52,661 INFO router_intent intent=domain_query
2025-09-20 22:13:52,661 INFO IN select_dataset_node: selected_datasets=None
2025-09-20 22:13:53,842 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-20 22:13:53,843 INFO select_dataset_node selected_datasets=['Benchmarking_national_employment']
```

Figura 8.6: Registro del backend seleccionando el dataset adecuado y clasificando la intención como domain_query.

Fuente: Elaboración propia.

8.4. Subgrafo de Filtrado

El subgrafo de filtrado y análisis constituye el núcleo operativo de la arquitectura conversacional. A partir de la consulta validada por el subgrafo de intención y de los conjuntos de datos seleccionados, realiza tres acciones encadenadas: (i) traduce el lenguaje natural a filtros estructurados, (ii) aplica esos filtros sobre los `DataFrame` en memoria (`dfs`) y (iii) ejecuta el análisis solicitado (agregaciones, tendencias, distribuciones o comparaciones). Genera tablas y mensajes explicativos listos para visualización o respuesta textual. Además, incorpora rutas de clarificación cuando no hay coincidencias y mecanismos de robustez (normalización de porcentajes, manejo de columnas temporales, verificación de columnas requeridas y planes alternativos) que sostienen la coherencia del flujo conversacional.

8.4.1. Estado y contratos

El subgrafo opera sobre un estado tipado `FilterState` que se transfiere entre nodos. Sus claves principales son:

<code>selected_datasets</code> (list[str])	conjuntos de datos activos a consultar
<code>last_query</code> (str)	consulta actual del usuario en lenguaje natural
<code>messages</code>	historial de mensajes, preservado para coherencia con el modelo
<code>filters</code> (dict)	filtros estructurados por conjunto de datos
<code>filtered_data</code> (dict)	registros resultantes tras aplicar los filtros (serializados)
<code>analysis_config</code> (dict)	plan de análisis reproducible (tipo, métricas, orden, límite)
<code>analysis_result</code> (Any)	salida final del análisis, lista para visualización o respuesta
<code>list_col</code> (str)	indicador para el modo de exploración de valores únicos

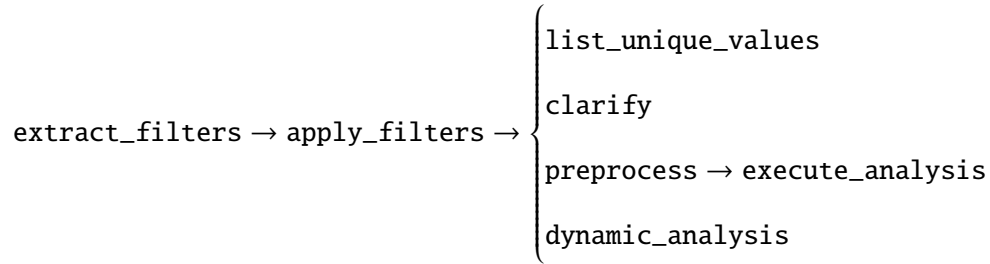
Cada nodo lee y modifica únicamente la parte del estado que le corresponde. Esta separación de responsabilidades asegura claridad en el flujo, facilita la verificación y permite auditar las decisiones del sistema.

8.4.2. Nodos y flujo

El subgrafo se implementa con `StateGraph` (`LangGraph`), que representa el proceso como un grafo dirigido de *nodos* unidos por *aristas*. Cada nodo es una función $f : \text{FilterState} \rightarrow \text{FilterState}$: recibe el estado compartido, actualiza sólo las claves que le corresponden (p. ej., `filters`, `analysis_config`, `filtered_data`) y devuelve el estado actualizado. Las aristas fijan el orden de ejecución y pueden ser *condicionales*: según el contenido del estado, el flujo se enruta al siguiente nodo apropiado (p. ej., aclaración, listado de valores o ejecución de análisis).

Para trazar y poder reanudar el flujo, usamos un *checkpoint*. En este trabajo se emplea `MemorySaver`, que almacena una “instantánea” del estado tras cada nodo en la memoria del proceso. Con ello es posible (i) auditar paso a paso las decisiones tomadas y (ii) retomar la ejecución desde un punto conocido. En producción, este componente puede cambiarse por un backend persistente.

La ruta principal es:



Nodo `extract_filters`. Este nodo traduce la consulta a una representación formal compuesta por filtros y un plan de análisis. Para ello, construye un *prompt* con los datasets activos, sus esquemas de columnas y ejemplos de registros; lo envía al modelo y recibe una respuesta estrictamente en JSON con filtros por dataset y una sección `analysis`. A continuación, valida y normaliza los filtros contra los datos reales, estandariza las métricas y aplica ajustes automáticos para robustecer el flujo.

8.4.3. Diseño del prompt de extracción (enfoque *few-shot*)

El *prompt* del nodo `extract_filters` está diseñado para obtener, de forma consistente, un JSON con: (i) filtros por dataset y (ii) una configuración de análisis estructurada. El mensaje sitúa al modelo como “*extractor de filtros y configurador de análisis*” y siempre incluye: `datasets`, `schemas`, `previews` y la `query`.

Componentes del prompt (qué son y para qué sirven):

- `datasets`: identificadores de los conjuntos de datos disponibles. *Acota* dónde debe buscar el modelo.
- `schemas`: para cada dataset, la lista de columnas (nombres exactos y, cuando aplica, tipo). *Obliga* a usar solo columnas reales.
- `previews`: primeras filas de cada dataset como muestra de valores. *Da contexto* sobre categorías y rangos.
- `query`: la consulta en lenguaje natural del usuario. *Aporta la intención* (filtros, métrica, periodo, orden, límite).

Se emplea **few-shot prompting**: ejemplos breves de entrada–salida que fijan el formato (solo JSON), el uso estricto de columnas presentes en `schemas/previews` y reglas frecuentes (p. ej., en `trend` no fijar `year_aux` salvo rango explícito; en `rankings`, incluir `order_by` y `limit`; estructura específica para `retention`).

Ejemplo concreto (pregunta → JSON). Consulta en lenguaje natural: “*Muéstrame el top 5 de programas del área Salud con más alumnos nuevos en 2023, comparando por institución.*”

Salida JSON esperada:

```
{
  "Benchmarking_enrollment_program": {
    "field_knowledge": "Salud",
    "year_aux": 2023
```

```
},
"analysis": {
  "type": "group_and_aggregate",
  "group_by": ["institution_name", "program_name"],
  "metrics": { "total_new_enrolled": "sum" },
  "order_by": { "total_new_enrolled": "desc" },
  "limit": 5,
  "compare_by": "institution_name"
}
}
```

Este ejemplo (incluido como *few-shot*) muestra cómo la consulta libre se convierte en filtros explícitos (`field_knowledge`, `year_aux`) y en un plan reproducible (tipo, agrupaciones, métrica a agregar, ordenamiento, límite y variable de comparación), reforzando el formato y las restricciones que el modelo debe respetar.

Procesamiento del JSON devuelto. La respuesta del modelo se recorta para extraer el primer bloque JSON entre llaves y luego se parsea. Si ese JSON no es válido, el nodo devuelve el estado sin cambios y registra el incidente en el log. En ese caso no se aplican valores por defecto y, al permanecer sin filtros, el enrutamiento posterior conduce a `clarify`. Si es válido, se verifica primero si se solicitó el modo de exploración (`explore_column`); en ese caso, se activa `list_col` y el nodo finaliza. En caso contrario, los filtros por dataset se normalizan contra `dfs` (corrigiendo tipos y valores fuera de dominio) y la sección `analysis` se limpia y completa (incluida la estandarización de métricas). A continuación, se aplican ajustes de robustez: convertir `group_and_aggregate` sin métricas en `distribution`; proponer un ranking por tasas de retención cuando la consulta lo sugiere y existen columnas disponibles; añadir `compare_by` si hay múltiples categorías seleccionadas; y retirar `year_aux` en análisis de tipo `trend` para no truncar la serie temporal. El resultado queda almacenado en `state["filters"]` y `state["analysis_config"]`, listo para la etapa de aplicación de filtros.

Nodo `apply_filters`. Este nodo toma los criterios de filtrado ya definidos y los aplica sobre *copias* de los `DataFrame` originales, de modo que la configuración en `state.filters` no se modifica. Su objetivo es dejar los datos listos para el análisis, con unidades y tipos consistentes.

Primero homogeneiza tasas y porcentajes: cuando una columna es (o se comporta como) una tasa, sus valores se convierten a razón en $[0, 1]$ y cualquier umbral `min/max` del filtro se reescala a esa misma unidad, evitando comparaciones entre `%` y razón. Luego, si el filtro indica un *rango* temporal (años o periodos), la columna correspondiente se trata como numérica para impedir comparaciones por texto.

A continuación, aplica secuencialmente cada condición de filtrado sobre la copia del `DataFrame` hasta obtener el subconjunto solicitado. Durante el proceso registra cuántas filas resultan por conjunto de datos (mensajes breves para trazabilidad).

`state["filtered_data"]` almacena, por dataset, los registros ya filtrados y listos para los nodos de preprocesamiento y análisis, sin efectos colaterales sobre la configuración original de filtros.

Ejemplo ilustrativo: interacción de `extract_filters` y `apply_filters`. La Figura 8.7 muestra un caso en que el sistema procesa la consulta “*Promedio de empleabilidad a 1 año por universidad, en Derecho 2024. Dame el top 10*”.

En el nodo `extract_filters`, el modelo propone filtros explícitos (`year = 2024`, `program_gen_name_normalized = DERECHO`) y un `analysis_config` de tipo `group_and_aggregate`, con agrupación por `institution_name`, cálculo del promedio de `emp_rate_1yr`, orden descendente y límite de 10 resultados.

Luego, en el nodo `apply_filters`, esos criterios se aplican sobre el dataset `Benchmarking_national_employment`, generando 30 registros válidos que cumplen con las condiciones. De esta manera, el flujo continúa hacia la ejecución del análisis, asegurando que los filtros sugeridos por el LLM se traduzcan en subconjuntos de datos coherentes y listos para agregación.

```

2025-09-21 02:00:30.162 INFO [router_intent] last_query="Promedio de empleabilidad a 1 año por universidad, en derecho 2024. Dame el top 10" previous_query="Promedio de empleabilidad a 1 año por universidad, en derecho 2024. Dame el top 10
2025-09-21 02:00:32.551 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-21 02:00:32.552 INFO router_intent intent-domain_query
2025-09-21 02:00:32.557 INFO fn select_dataset_mode: selected_datasets=None
2025-09-21 02:00:33.855 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-21 02:00:33.866 INFO select_dataset_mode selected_datasets=[Benchmarking_national_employment]
2025-09-21 02:00:33.868 INFO [extract_filters] (reset list_col) Entrando con selected_datasets=[Benchmarking_national_employment]
2025-09-21 02:00:35.038 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-21 02:00:35.042 INFO [normalize_and_validate] Dataset=Benchmarking_national_employment, raw_filters={'year': 2024, 'program_gen_name_normalized': 'DERECHO'}
2025-09-21 02:00:35.042 INFO [normalize_and_validate] Normalized filters for Benchmarking_national_employment: {'year': 2024, 'program_gen_name_normalized': 'DERECHO'}
2025-09-21 02:00:35.042 INFO [extract_filters] Filtros extraídos: {'Benchmarking_national_employment': {'year': 2024, 'program_gen_name_normalized': 'DERECHO'}}
2025-09-21 02:00:35.043 INFO [extract_filters] Configuración de análisis: {'type': 'group_and_aggregate', 'group_by': ['institution_name'], 'metrics': {'emp_rate_1yr': 'mean'}, 'order_by': {'emp_rate_1yr': 'desc'}, 'limit': 10}
2025-09-21 02:00:35.043 INFO [apply_filters] Aplicando filtros
2025-09-21 02:00:35.070 INFO [apply_filters] 30 registros en 'Benchmarking_national_employment'
2025-09-21 02:00:35.090 INFO [execute_analysis] Ejecutando análisis generico
2025-09-21 02:00:35.091 INFO [execute_analysis] DataFrame para Benchmarking_national_employment: columns=['year', 'institution_type', 'institution_name', 'area', 'accred_status', 'program_gen_name', 'program_name', 'cont_rate', 'retenti
on_rate_1yr', 'program_duration_sens', 'emp_rate_1yr', 'emp_rate_2yr', 'income_brackets', 'annual_tuition', 'institution_name_normalized', 'program_gen_name_normalized', 'program_institution'], files=30
2025-09-21 02:00:35.041 DEBUG [prepare_visual_of] DataFrame shape: (10, 2)

```

Figura 8.7: Log del backend mostrando la interacción de `extract_filters` y `apply_filters` en una consulta sobre empleabilidad en Derecho (2024).

Fuente: Elaboración propia.

Nodo clarify. Este nodo se activa cuando, tras aplicar los filtros, algún dataset queda sin registros. Su objetivo es detener el flujo antes de un análisis vacío y entregar al usuario un único mensaje de corrección, claro y accionable.

Primero limpia resultados previos (`filtered_data`, `analysis_config`, `analysis_result`) pero conserva los filtros originales. Luego detecta en qué columnas hay problemas (nombres inexistentes o valores que no aparecen) normalizando texto (mayúsculas/minúsculas, tildes) y comparando contra los valores reales del dataset; con ello construye una lista de faltantes acompañada de una muestra acotada de valores válidos.

Con esa evidencia genera un prompt a partir de la plantilla `clarify.txt`, interpolando: el nombre del dataset, los filtros originales, los faltantes detectados, el listado de columnas disponibles y un conjunto breve de columnas sugeridas (por ejemplo, Año, Institución, Programa). La plantilla exige una salida breve, en un único texto y sin detalles técnicos. El modelo devuelve ese mensaje de aclaración (por ejemplo, “corrige este nombre”, “elige entre estas opciones”, “agrega un rango de años”), que se añade al historial y **cierra esta rama del flujo** a la espera de la nueva instrucción del usuario.

Resultado. Un solo mensaje de aclaración, específico y accionable, que indica qué parte del filtro no coincide con los datos y propone alternativas válidas, evitando respuestas vacías y guiando al usuario de vuelta a un flujo válido.

Ejemplo ilustrativo. La Figura 8.8 muestra un caso de consulta sin resultados: “¿Cuál es la tasa de retención de Derecho en la Universidad Técnica Federico Santa María?”. El sistema construyó correctamente los filtros y un plan de análisis con `group_and_aggregate`, pero

CAPÍTULO 8 : DESARROLLO

al aplicar los criterios el dataset quedó vacío. En consecuencia, se activó el nodo `clarify`, que generó un mensaje breve y accionable para que el usuario verificara los nombres de las columnas (*Institución, Programa*) y corrigiera su petición.



(a) *Frontend: el sistema informa que no se encontraron resultados e invita a revisar las columnas disponibles.*

```
2025-09-21 01:49:55,540 INFO [router_intent] last_query='¿Cuál es la tasa de retención de Derecho en la Universidad Técnica Federico Santa María?' previous_query='¿Cuál es la tasa de retención de Derecho en la Universidad Técnica Federico Santa María?'
2025-09-21 01:49:56,480 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-21 01:49:56,410 INFO router_intent intent-domain_query
2025-09-21 01:49:56,411 INFO IN select_dataset_node: selected_datasets=None
2025-09-21 01:49:57,105 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-21 01:49:57,201 INFO select_dataset_node selected_datasets=['Benchmarking_national_employment']
2025-09-21 01:49:57,200 INFO [extract_filters] (reset list_col) Entrando con selected_datasets=['Benchmarking_national_employment']
2025-09-21 01:49:58,335 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-21 01:49:58,340 INFO [normalize_and_validate] dataset=Benchmarking_national_employment, raw_filters={'program_gen_name_normalized': 'DERECHO', 'institution_name': 'Universidad Técnica Federico Santa María'}
2025-09-21 01:49:58,341 INFO [normalize_and_validate] normalized_filters_for_benchmarking_national_employment: {'program_gen_name_normalized': 'DERECHO', 'institution_name': 'Universidad Técnica Federico Santa María'}
2025-09-21 01:49:58,342 INFO [extract_filters] Filtros extraídos: {'benchmarking_national_employment': {'program_gen_name_normalized': 'DERECHO', 'institution_name': 'Universidad Técnica Federico Santa María'}}
2025-09-21 01:49:58,342 INFO [extract_filters] Configuración de análisis: {'type': 'group_and_aggregate', 'group_by': {'program_name'}, 'metrics': {'retention_rate_1yr': 'mean'}, 'order_by': {'retention_rate_1yr': 'desc'}, 'limit': 10}
2025-09-21 01:49:58,343 INFO [apply_filters] Aplicando filtros
2025-09-21 01:49:58,350 INFO [apply_filters] 0 registros en 'Benchmarking_national_employment'
2025-09-21 01:49:58,357 INFO [clarify] Iniciando clasificación por estos vacíos
2025-09-21 01:49:59,666 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-21 01:49:59,663 MESSAGE [clarify] LLM response: No se encontraron resultados para tu consulta actual. Te sugiero intentar filtrar utilizando las columnas "Institución" o "Programa" para obtener datos disponibles. Por favor, verifica que los nombres estén escritos exactamente como aparecen en el dataset.
```

(b) *Backend: el log muestra la extracción de filtros, la configuración de `analysis_config`, la aplicación de filtros sin resultados y la activación del nodo `clarify`.*

Figura 8.8: Caso de activación de `clarify` ante una consulta sin datos (retención en Derecho, UTFSM).

Fuente: Elaboración propia.

Nodo `list_unique_values`. Este nodo se activa cuando el flujo requiere explorar los valores disponibles de una columna concreta, señalada previamente en `state["list_col"]` (por ejemplo, tras una solicitud del tipo “¿qué instituciones hay?”). Su función es simple: tomar el *dataset* activo, verificar que la columna existe y extraer el conjunto de valores únicos, ordenados y sin nulos.

El resultado se guarda en la clave del estado `analysis_result` como una lista de cadenas, y se añade un mensaje breve con el listado para que el usuario pueda elegir opciones válidas antes de continuar. Si la columna indicada no existe en el *dataset* seleccionado, se informa explícitamente del error y no se altera el resto del estado.

Resultado. Un inventario conciso de valores disponibles en la columna consultada, útil para acotar la consulta, guiar al usuario y evitar errores por nombres inexistentes.

Nodo `preprocess_dtypes`. Antes de ejecutar `group_and_aggregate`, este nodo realiza un preprocesamiento breve para evitar errores de tipo y agrupaciones incorrectas. En particular: (i) convierte a numéricas solo las columnas declaradas como métricas, manteniendo las dimensiones categóricas; (ii) depura `group_by` cuando incluye columnas numéricas que no son categorías (se aceptan `year` y `year_aux` por su carácter temporal). Al finalizar, actualiza `state["filtered_data"]` con los registros saneados y `state['analysis_config']` con el `group_by` corregido, asegurando que las agregaciones posteriores sean consistentes con la intención analítica.

Nodo `execute_analysis`. Este nodo toma `analysis_config` ya normalizada y ejecuta el análisis sobre los datos filtrados. Su objetivo es producir resultados consistentes y listos para visualización, verificando primero que existan las columnas mínimas requeridas y aplicando ajustes prudentes cuando faltan piezas.

Flujo general. Para cada *dataset*, se valida que las columnas exigidas por el tipo de análisis estén presentes; si no lo están, se informa con un mensaje breve. Cuando hay columnas de porcentaje, sus valores se normalizan a razón en $[0, 1]$ antes de cualquier agregación. Luego se ejecuta el caso indicado en `analysis_config`:

- **`group_and_aggregate`:** se agrupa por `group_by` y se aplican `metrics`; si `group_by` está vacío, se entregan agregados globales.
- **`trend`:** se agrupa por la dimensión temporal indicada en `by` y, cuando corresponde, se añaden comparaciones por `compare_by`. La salida es una tabla “larga”, apta para series de tiempo.

- **distribution:** se devuelve la lista de valores observados de una columna (útil para explorar categorías).
- **compare_categories:** si hay `metric`, se agrupa por las categorías y se calcula la agregación solicitada; si no hay `metric` pero las categorías incluyen columnas numéricas, se activa un modo de “series” que reestructura a formato largo y agrega por tiempo y categoría si existe una columna temporal detectable. En este caso se ajusta `analysis_config` para facilitar la visualización posterior.
- **retention:** se calcula la retención a N años alineando periodos t y $t+N$ y evaluando `enrolled_t1/enrolled_t` con las llaves de agrupación definidas. Si faltan columnas clave para cohortes pero el dataset incluye una tasa de retención ya calculada, se aplica un plan alternativo: se promedia esa tasa, se ordena de forma descendente, se limita el resultado y se registra un mensaje explicativo.

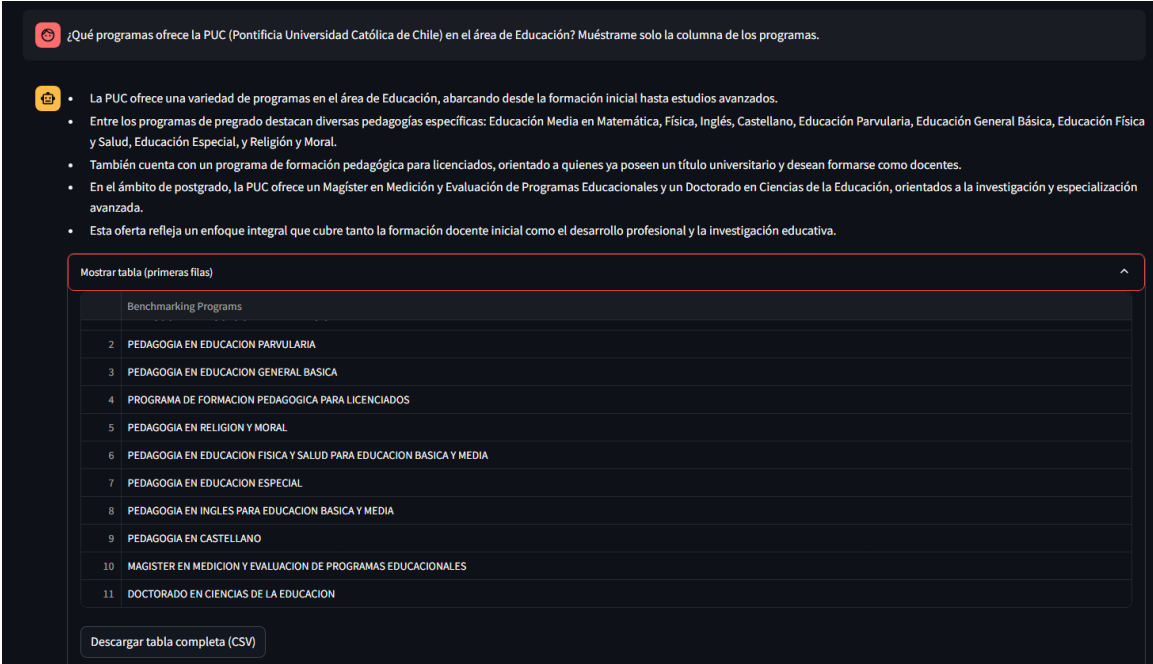
Al finalizar, si `analysis_config` define `order_by` o `limit`, las salidas se ordenan y recortan con esos parámetros. El resultado se serializa y se deja en `state["analysis_result"]`, junto con mensajes breves que documentan ajustes o faltantes relevantes.

Ejemplos ilustrativos. La Figura 8.9 y la Figura 8.10 muestran dos casos concretos de cómo el sistema procesa una consulta de usuario. Cada ejemplo incluye: (i) la vista de frontend que recibe el usuario, con los resultados listos para explorar; y (ii) el log del backend, donde se aprecia paso a paso la extracción de filtros, su normalización y la configuración del análisis (`analysis_config`). Finalmente, el nodo `execute_analysis` toma esta configuración normalizada y ejecuta el plan definido, validando columnas requeridas, aplicando normalizaciones de tasas cuando corresponde y generando la tabla de salida lista para visualización.

En el primer caso (Figura 8.9), el usuario pregunta qué programas ofrece la Pontificia Universidad Católica de Chile en el área de Educación. El nodo `extract_filters` propone como filtros `institution_name = PUC` y `field_knowledge = Educación`, además de un análisis de tipo `distribution` sobre la columna `program_name`. El log muestra cómo estos filtros se validan contra el dataset `Benchmarking_programs`, resultando en 12 registros. Luego, `execute_analysis` aplica el plan `distribution` y devuelve al frontend la lista de programas de esa

CAPÍTULO 8 : DESARROLLO

institución y área, que se visualiza en forma de tabla.



¿Qué programas ofrece la PUC (Pontificia Universidad Católica de Chile) en el área de Educación? Muéstrame solo la columna de los programas.

- La PUC ofrece una variedad de programas en el área de Educación, abarcando desde la formación inicial hasta estudios avanzados.
- Entre los programas de pregrado destacan diversas pedagogías específicas: Educación Media en Matemática, Física, Inglés, Castellano, Educación Parvularia, Educación General Básica, Educación Física y Salud, Educación Especial, y Religión y Moral.
- También cuenta con un programa de formación pedagógica para licenciados, orientado a quienes ya poseen un título universitario y desean formarse como docentes.
- En el ámbito de postgrado, la PUC ofrece un Magíster en Medición y Evaluación de Programas Educativos y un Doctorado en Ciencias de la Educación, orientados a la investigación y especialización avanzada.
- Esta oferta refleja un enfoque integral que cubre tanto la formación docente inicial como el desarrollo profesional y la investigación educativa.

Mostrar tabla (primeras filas)

Benchmarking Programs	
2	PEDAGOGÍA EN EDUCACION PARVULARIA
3	PEDAGOGÍA EN EDUCACION GENERAL BASICA
4	PROGRAMA DE FORMACION PEDAGOGICA PARA LICENCIADOS
5	PEDAGOGÍA EN RELIGION Y MORAL
6	PEDAGOGÍA EN EDUCACION FISICA Y SALUD PARA EDUCACION BASICA Y MEDIA
7	PEDAGOGÍA EN EDUCACION ESPECIAL
8	PEDAGOGÍA EN INGLES PARA EDUCACION BASICA Y MEDIA
9	PEDAGOGÍA EN CASTELLANO
10	MAGISTER EN MEDICION Y EVALUACION DE PROGRAMAS EDUCACIONALES
11	DOCTORADO EN CIENCIAS DE LA EDUCACION

Descargar tabla completa (CSV)

(a) Frontend: programas del área Educación en la PUC (Benchmarking_programs).

```
177.0.0.1:42420 - GET /health HTTP/1.1 300
2025-09-21 00:58:23.381 INFO [router_intent] last_query='¿Qué programas ofrece la PUC (Pontificia Universidad Católica de Chile) en el área de Educación? Muéstrame solo la columna de los programas.' previous_query='¿Qué programas ofrece
2025-09-21 00:58:26.853 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-21 00:58:26.867 INFO router_intent Intent-domain_query
2025-09-21 00:58:26.869 INFO IN select_dataset_node selected_datasets=None
2025-09-21 00:58:28.218 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-21 00:58:28.221 INFO select_dataset_node selected_datasets=[Benchmarking_programs]
2025-09-21 00:58:29.224 INFO [extract_filters] (reset list_col) Entiendo con selected_datasets=['Benchmarking_programs']
2025-09-21 00:58:30.184 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-21 00:58:30.302 INFO [normalize_and_validate] Dataset=benchmarking_programs, raw_filters={'institution_name': 'Pontificia Universidad Católica de Chile', 'field_knowledge': 'Educación'}
2025-09-21 00:58:30.389 INFO [normalize_and_validate] Normalized filters for Benchmarking_programs: {'institution_name': 'Pontificia Universidad Católica de Chile', 'field_knowledge': 'Educación'}
2025-09-21 00:58:30.391 INFO [extract_filters] Filtros extraídos: {'benchmarking_programs': {'institution_name': 'Pontificia Universidad Católica de Chile', 'field_knowledge': 'Educación'}}
2025-09-21 00:58:30.391 INFO [extract_filters] Configuración de análisis: {'type': 'distribution', 'column': 'program_name'}
2025-09-21 00:58:30.392 INFO [apply_filters] Aplicando filtros
2025-09-21 00:58:30.400 INFO [apply_filters] 12 registros en 'Benchmarking_programs'
2025-09-21 00:58:30.482 INFO [execute_analysis] Ejecutando análisis genérico
2025-09-21 00:58:30.486 INFO [execute_analysis] Dataframe para benchmarking_programs: columns=['institution_class', 'institution_code', 'institution_name', 'program_name', 'field_knowledge', 'std_program_name', 'global_level', 'progra
+class', 'program_size_code', 'order', 'institution_group', 'std_program_name_normalized', 'program_institution'], files=12
2025-09-21 00:58:30.522 DEBUG [prepare_visual_df] DataFrame shape: (12, 1)
```

(b) Backend: log con filtros aplicados y análisis de tipo distribution.

Figura 8.9: Ejecución completa de filtrado y análisis sobre el dataset `Benchmarking_programs`.

Fuente: Elaboración propia.

En el segundo caso (Figura 8.10), la consulta pide la proporción de nuevos ingresos por género en Ingeniería Comercial durante 2024, comparando universidades. El nodo `extract_filters` identifica como filtros `program_name = Ingeniería Comercial`, `institution_name` en un conjunto de universidades y `year_aux = 2024`. Además, construye un plan de análisis de tipo `group_and_aggregate`, con `group_by = institution_name` y métricas agregadas sobre los nuevos ingresos (`total_new_enrolled`, `female_new_enrolled`, `male_new_enrolled`). El log confirma que se aplicaron estos filtros y que `execute_analysis`

Cuándo se ejecuta. Tras aplicar los filtros, si no hay una configuración analítica específica. Si un dataset queda sin filas, se informa con un mensaje breve y no se intenta resumir.

Qué produce para cada dataset con datos:

- **Resumen numérico:** para cada columna numérica calcula suma, promedio, conteo, desviación estándar, mínimo, máximo y mediana, ofreciendo una lectura rápida de magnitudes y dispersión.
- **Resumen categórico:** para cada columna no numérica calcula la frecuencia de cada categoría y su porcentaje relativo, permitiendo detectar categorías dominantes o poco frecuentes (tratando nulos de forma explícita).
- **Vista de ejemplo (preview):** muestra hasta 10 filas representativas con columnas informativas priorizadas (si existen) como `program_name`, `institution_name`, `campus_name`, `field_knowledge`, `year_aux` y `region`; en su defecto, utiliza las primeras columnas disponibles. La vista elimina duplicados para mejorar la legibilidad.

Salida y estado. El resultado se guarda en `state[.analysis_result]` con la forma:

```
{
  "<dataset>": {
    "resumen": { ...estadísticos numéricos y categóricos... },
    "preview": [ ...hasta 10 filas representativas... ]
  },
  ...
}
```

Además, se añade al historial un mensaje conciso que indica los filtros aplicados, las columnas mostradas en la `preview` y una invitación a solicitar un análisis específico. Esto mantiene la trazabilidad y guía al usuario para refinar su consulta en el siguiente paso.

Enrutamiento condicional y grafo del subproceso. Después de aplicar los filtros, la función `route_after_filters` decide a qué nodo ir. Lee el estado (`FilterState`) y evalúa

CAPÍTULO 8 : DESARROLLO

reglas en un orden fijo. La primera que se cumple define el camino. Esto evita análisis vacíos y respeta lo que pidió el usuario.

Reglas (en orden):

1. **No hay filtros** (`state["filters"]` vacío) \Rightarrow `clarify`.
Motivo: no hay criterios para buscar; se pide al usuario que precise la consulta.
2. **Algún dataset quedó sin filas** (hay listas vacías en `state["filtered_data"]`) \Rightarrow `clarify`.
Motivo: no se avanza a un análisis sin datos; se entregan opciones válidas para corregir.
3. **Explorar valores de una columna** (`list_col` activo) y **no hay plan de análisis** (`analysis_config` ausente) \Rightarrow `list_unique_values`.
Motivo: el usuario quiere ver qué valores existen antes de seguir.
4. **Hay plan de análisis** (`analysis_config` presente) \Rightarrow `preprocess`.
Motivo: preparar tipos y `group_by` para que el análisis no falle.

Si ninguna regla aplica, se usa el **caso por defecto**: `dynamic_analysis`.

Motivo: no hay plan, pero sí datos; se muestra un resumen numérico y categórico con una *preview*.

Flujo completo.

1. `START` \rightarrow `extract_filters`: extrae filtros y arma el análisis desde la consulta.
2. `extract_filters` \rightarrow `apply_filters`: aplica los filtros sobre copias de los `DataFrame`.
3. Desde `apply_filters`, `route_after_filters` decide entre:
 - `clarify`: mensaje para corregir filtros (nombres/valores/columnas).
 - `list_unique_values`: lista los valores disponibles de la columna pedida.
 - `preprocess` \rightarrow `execute_analysis`: limpia tipos y `group_by`, y luego ejecuta el análisis configurado.
 - `dynamic_analysis`: si no hay plan, entrega un resumen y una *preview* útil.

4. Nodos de término: `clarify` → END, `list_unique_values` → END, `execute_analysis` → END, `dynamic_analysis` → END.

Por qué el orden importa.

- Primero se verifica que haya filtros y datos: evita correr análisis vacíos.
- Se respeta la intención del usuario: si pidió explorar valores, se lista; si definió un plan, se prepara y se ejecuta.
- Si no hay plan, igual se entrega valor: un resumen claro del resultado de los filtros.

8.4.4. normalización y verificaciones utilizadas

- **Porcentajes a razón:** cualquier columna identificada como tasa o porcentaje se homogeneiza a razones en el intervalo $[0, 1]$. Los filtros definidos por el usuario (`min/max` o valores escalares) se transforman con el mismo criterio. Así se comparan sin ambigüedad columnas que llegan como “88 %”, “0,88” u “88”.
- **Temporalidad numérica:** cuando se filtra por rangos de año, la columna temporal se convierte a numérico para evitar comparaciones alfabéticas. De este modo, el rango 2018–2022 se interpreta correctamente.
- **Validación de columnas requeridas:** para cada tipo de análisis se construye un conjunto mínimo de columnas esperadas. Si falta alguna en el subconjunto filtrado, se advierte al usuario. En el caso de retención, se dispone de un plan alternativo basado en tasas precomputadas, si están disponibles.
- **Mensajes trazables:** cada paso añade mensajes breves y verificables (conteo de filas filtradas, ajustes y aclaraciones), lo que facilita el seguimiento y la auditoría del sistema.

8.4.5. Supuestos y limitaciones actuales

- Cada consulta trabaja con un único conjunto de datos; este subgrafo no realiza uniones entre fuentes. La integración multi-dataset se planifica como un subgrafo específico con

su propio proceso de extracción de filtros para cruces y un nodo dedicado a ejecutar análisis con unión de tablas.

- Se asume que los `DataFrame` han sido cargados y normalizados previamente por el módulo `loader.py` (nombres y tipos consistentes, columnas temporales alineadas).
- Si el modelo devuelve un JSON mal formado, se aplican valores por defecto conservadores y se guía al usuario con mensajes explícitos para recuperar el flujo.

8.5. Subgrafo de visualización

El subgrafo de visualización constituye la etapa final del sistema conversacional. A partir de los resultados del subgrafo de filtrado (`analysis_result` y, en su defecto, `filtered_data`), genera una salida comprensible para el usuario en forma de gráfico, tabla y explicación en lenguaje natural. De este modo se completa el ciclo que va desde la consulta en lenguaje natural hasta una presentación interpretable de la información.

8.5.1. Objetivo y contrato

El objetivo es doble: (i) escoger el tipo de gráfico adecuado según los datos y la configuración analítica; (ii) generar un producto final consistente (imagen, tabla y explicación). El subgrafo garantiza que, aun en situaciones ambiguas, siempre habrá al menos una tabla coherente como respuesta.

El estado utilizado extiende al de filtrado:

- `visualization_df`: DataFrame listo para graficar.
- `chart_type`: tipo de visualización sugerido (línea, barra, torta o tabla).
- `chart_spec`: especificación estructurada (ejes, series, títulos, leyendas).
- `current_table`: tabla ordenada para mostrarse.
- `image_path` y `chart_reason`: ruta de la imagen generada y justificación del tipo de gráfico.

8.5.2. Flujo y nodos

El subproceso se implementa con `StateGraph` y sigue un flujo determinista (este subgrafo se compila sin `checkpoint` porque no requiere reanudación paso a paso):

```
prepare_visual_df → llm_visualization_type → add_proportion_columns  
→ generate_chart_spec → clear_previous_image  
→ render_chart → explain_chart
```

prepare_visual_df. Unifica la salida del análisis en un único conjunto tabular listo para graficar. Tolera diferentes formatos de entrada (tablas, listas de registros o diccionarios) y aplica normalización. Cuando los resultados incluyen simultáneamente datos de empleabilidad y de egresados, intenta una unión ligera mediante claves comunes como año, institución y programa. Si no existe un análisis formal, utiliza directamente los datos filtrados. Además, cuando hay una sola columna con diccionarios, la aplanar para simplificar la estructura.

llm_visualization_type. Determina el tipo de gráfico más adecuado con apoyo de un modelo de lenguaje. Para ello invoca la plantilla `prompts/decide_chart_type.txt`, que recibe tres entradas interpoladas: (i) la última consulta del usuario, (ii) la configuración analítica vigente y (iii) un perfil reducido de los datos (número de filas y conteo de valores únicos en columnas relevantes). El *prompt* solicita explícitamente que la salida sea estrictamente JSON con las claves `chart_type` y `reason`; no obstante, por seguridad el parser extrae el último bloque `{...}` que contiene `chart_type` y tolera texto adicional antes o después. Si el JSON no se puede parsear, el sistema degrada de forma conservadora a `chart_type="table"`.

add_proportion_columns. Actúa únicamente en los gráficos de torta relacionados con género. Si el conjunto incluye columnas estándar de mujeres y hombres y existe una única fila, calcula automáticamente los porcentajes y agrega estas variables al conjunto de datos. Esto facilita la interpretación sin intervención adicional del usuario.

generate_chart_spec. Construye la especificación detallada del gráfico, definiendo ejes, series, etiquetas y título. Aplica heurísticas prácticas: realiza agregaciones locales si los datos llegan sin procesar y hay duplicados sobre `group_by`; pivota la tabla para series multi-institución en gráficos de línea o barra cuando hay columna temporal y `compare_by`; genera rankings Top-N en métricas de retención cuando existen columnas de retención y agrupación por programa; y recorta intervalos temporales cuando la consulta menciona años específicos. Además, ordena las columnas para producir siempre una tabla legible, priorizando tiempo, institución, programa y métricas principales, y limita el número máximo de columnas visibles para mantener la claridad.

clear_previous_image. Elimina el archivo del gráfico anterior en el directorio de salida. Con ello se asegura que el frontend muestre siempre la visualización más reciente y se eviten confusiones con imágenes previas.

render_chart. Genera el gráfico en formato estático utilizando `Matplotlib` y lo guarda en disco. Incorpora criterios de usabilidad: homogeneiza tasas/porcentajes a razón y formatea el eje en %; aplica separadores de miles a valores absolutos; usa barras horizontales cuando hay muchas categorías o etiquetas largas; y, si la especificación no tiene datos suficientes para graficar, responde solo con la tabla. La imagen se guarda como `static/plots/viz_current.png`.

explain_chart. Produce una explicación breve y legible de la visualización final. Para ello se construye un prompt en línea que instruye al modelo de lenguaje a comportarse como un analista en educación superior en Chile. El modelo recibe la consulta original, el título del gráfico y la tabla en texto, y debe devolver entre cuatro y cinco viñetas con las principales conclusiones, sin incluir código ni repetir la tabla o la imagen. El texto generado se añade al historial de la conversación y, al finalizar, se limpian variables temporales del estado (datos de visualización y especificaciones) para no arrastrar contexto innecesario a turnos posteriores.

8.5.3. Criterios de diseño y robustez

Se priorizan tres principios:

1. Interpretabilidad por defecto: ante ambigüedad, se responde con `table`; nunca se fuerza un gráfico sin soporte en los datos.
2. Consistencia semántica: tasas y porcentajes se homogeneizan antes de graficar; las series temporales se identifican y se organizan en columnas o leyendas claras.
3. Pragmatismo guiado por datos: se activan heurísticas útiles (pivote temporal, Top-N de retención, proporciones de género) solo cuando los datos lo justifican.

8.5.4. Relación con el subgrafo de filtrado

La visualización no vuelve a ejecutar análisis, sino que consume directamente lo producido por el subgrafo de filtrado:

- `analysis_result` (agregaciones, tendencias, comparaciones, retención), o
- `filtered_data` (cuando no hubo plan de análisis explícito).

Si el grafo padre detecta que no hay filas para mostrar, evita esta etapa. Cuando existen datos, `prepare_visual_df` los normaliza y el resto de nodos completan la especificación, el renderizado y la explicación.

8.5.5. Limitaciones y consideraciones

- La selección del tipo de gráfico combina heurísticas y LLM; si el perfil de datos es escaso o ruidoso, se degrada a tabla.
- El renderizado es estático con `Matplotlib`; la interactividad (por ejemplo, `Altair` o `Plotly`) se plantea como trabajo futuro.
- La imagen `viz_current.png` se sobrescribe en cada ejecución; la persistencia por sesión se maneja en la capa de API/Frontend.

En conjunto, el subgrafo entrega un paquete completo para la interfaz: imagen del gráfico, tabla legible y explicación en lenguaje natural, alineado con la intención del usuario y el plan analítico previamente resuelto.

8.5.6. Ejemplos ilustrativos

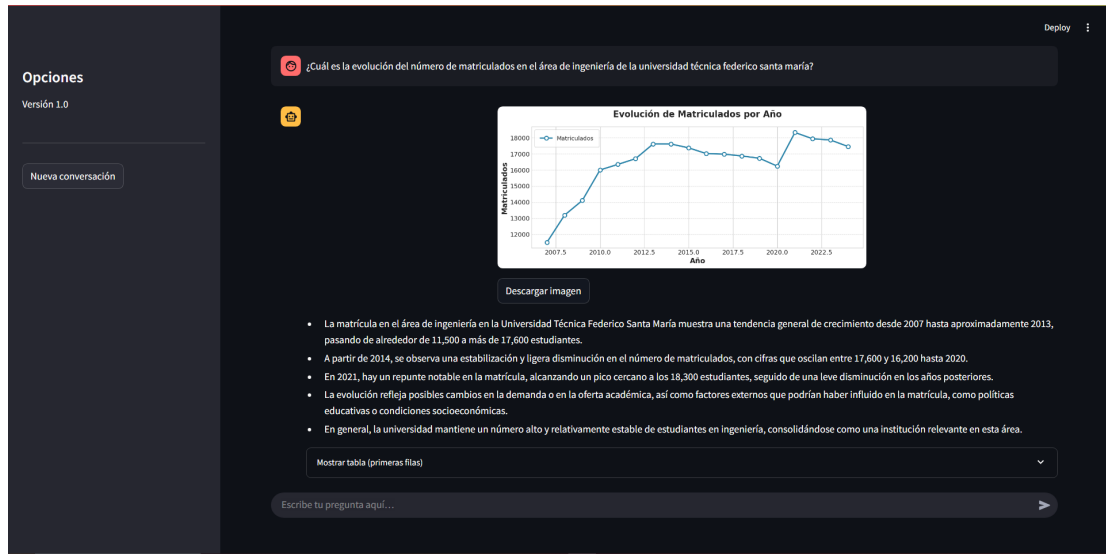
A continuación se muestran evidencias del subgrafo de visualización, desde la decisión del tipo de gráfico hasta el render final, junto con un breve resumen del flujo completo registrado en el log.

Ejemplo 1: Evolución de matrícula en Ingeniería (UTFSM). *Consulta del usuario:* “¿Cuál es la evolución del número de matriculados en el área de ingeniería de la Universidad Técnica Federico Santa María?”

Ejecución registrada.

- `prepare_visual_df`: consolida un *DataFrame* de 18 filas (años) y 2 columnas (`year_aux`, `total_enrolled`); véase evidencia en el log de la Fig. 8.11(b).
- `llm_visualization_type`: el LLM decide `chart_type = line` con `reason = “La consulta pide evolución en el tiempo”`; metadatos: `rows = 18`, `unique{year_aux} = 18` (Fig. 8.11(b)).
- `generate_chart_spec`: construye especificación con eje X = `year_aux`, serie Y = `total_enrolled` y título “Evolución de Matriculados por Año”; ver confirmación en el log (Fig. 8.11(b)).
- `render_chart`: render estático guardado en `static/plots/viz_current.png` y tabla asociada; el resultado visible en el frontend se muestra en la Fig. 8.11(a).
- `explain_chart`: genera 4–5 viñetas interpretativas y las añade al historial; se omite código y duplicación de tabla/imagen (flujo completado).

CAPÍTULO 8 : DESARROLLO



(a) Frontend: gráfico de línea para la evolución de matrículas en Ingeniería (UTFSM).

```
2025-09-24 00:07:46,870 INFO [router_intent] last_query="¿cuál es la evolución del número de matriculados en el área de ingeniería de la universidad técnica federico santa maría?" previous_query="¿cuál es la evolución del número de matriculados en el área de ingeniería de la universidad técnica federico santa maría?"
2025-09-24 00:07:46,878 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 00:07:46,884 INFO router_intent_intent=domain_query
2025-09-24 00:07:46,888 INFO llm select_datasets_yoder: selected_datasets=None
2025-09-24 00:07:46,890 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 00:07:46,922 INFO select_datasets_node selected_datasets=[ 'Benchmarking_enrollment_program' ]
2025-09-24 00:07:46,927 INFO [extract_filters] [reset list call] Intrudido con selected_datasets=[ 'Benchmarking_enrollment_program' ]
2025-09-24 00:07:50,766 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 00:07:50,801 INFO [normalize_and_validate] Database=benchmarking_enrollment_program, raw_filters={'field_knowledge': 'Ingeniería', 'institution_name': 'Universidad Técnica Federico Santa María'}
2025-09-24 00:07:50,804 INFO [normalize_and_validate] Normalized filters for Benchmarking_enrollment_program: {'field_knowledge': 'Tecnología', 'institution_name': 'Universidad Técnica Federico Santa María'}
2025-09-24 00:07:50,805 INFO [extract_filters] Ilimitado filtro 'year_aux' para trend en benchmarking_enrollment_program:2025-09-24 00:07:50,805 INFO [extract_filters] Filtros extraídos: {'Benchmarking_enrollment_program': {'field_knowledge': 'Tecnología', 'institution_name': 'Universidad Técnica Federico Santa María'}}
2025-09-24 00:07:50,807 INFO [extract_filters] Configuración de análisis: {'type': 'trend', 'by': 'year_aux', 'metric': 'total_enrolled'}
2025-09-24 00:07:51,400 INFO [apply_filters] Ejecutando análisis genérico
2025-09-24 00:07:51,406 INFO [apply_filters] 2339 registros en 'Benchmarking_enrollment_program'
2025-09-24 00:07:51,423 INFO [execute_analysis] Ejecutando análisis genérico
2025-09-24 00:07:51,423 INFO [execute_analysis] DataFrame para Benchmarking_enrollment_program: columns=['year', 'total_enrolled', 'female_enrolled', 'male_enrolled', 'total_new_enrolled', 'female_new_enrolled', 'male_new_enrolled', 'institution_class_1', 'institution_name', 'institution_accred', 'region', 'province', 'city', 'campus_name', 'program_name', 'field_knowledge', 'std_program_name', 'global_level', 'program_class_1', 'teaching_mode', 'session_name', 'program_class_type', 'program_sies_code', 'program_sector', 'municipal_school', 'local_service_school', 'upload_date', 'year_aux', 'institution_code', 'std_program_name_normalized', 'program_institution'] files=2339
2025-09-24 00:07:51,534 DEBUG [prepare_visual_df] DataFrame shape: (18, 2)
2025-09-24 00:07:54,512 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 00:07:54,517 DEBUG [llm_visualization_type] chart_type=line, reason=La consulta pide evolución en el tiempo, meta={'rows': 18, 'unique': {'year_aux': 18}}
2025-09-24 00:07:54,525 DEBUG [generate_chart_spec] spec_keys: ['type', 'filters', 'x', 'y', 'x_label', 'y_label', 'title']
2025-09-24 00:07:54,529 DEBUG [clear_previous_image] Removed static/plots/viz_current.png
2025-09-24 00:08:01,893 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 00:08:01,908 INFO [chat_session=202510cc32573] image_urls=table_mover.sql; len=24
```

(b) Backend (log): decisión `chart_type = line` y razón “La consulta pide evolución en el tiempo”. También se observa la normalización de filtros, el recuento de 2,339 filas tras el filtrado y la tabla final de 18 puntos temporales.

Figura 8.11: Evidencia extremo a extremo del subgrafo de visualización: desde filtros y plan analítico hasta decisión del tipo de gráfico y render final.

Fuente: Elaboración propia.

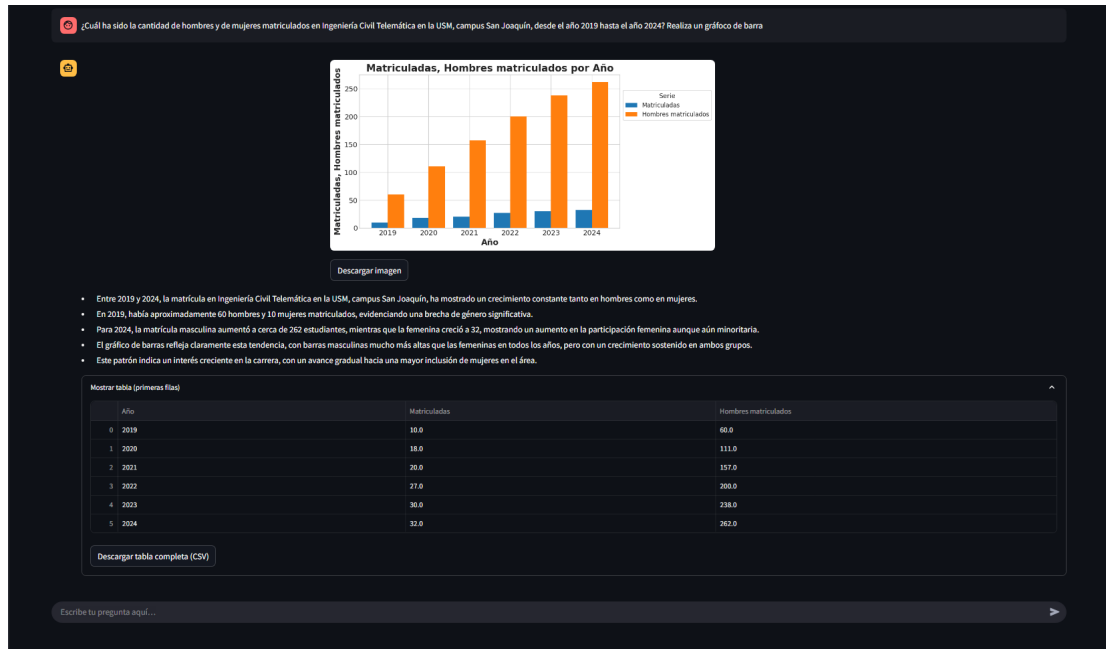
8.5.7. Ejemplo 2: Barras por género en Ingeniería Civil Telemática (USM, San Joaquín, 2019–2024)

Consulta del usuario (con tipo de gráfico explícito): “¿Cuál ha sido la cantidad de hombres y de mujeres matriculados en Ingeniería Civil Telemática en la USM, campus San Joaquín, desde el año 2019 hasta el año 2024? Realiza un gráfico de barra”.

Ejecución registrada (enfoque en visualización).

- `prepare_visual_df`: consolida la salida del análisis en una tabla con la dimensión temporal `year_aux` y métricas de matrícula por género.
- `llm_visualization_type`: detecta que el usuario *pidió explícitamente* un gráfico de barras y fija `chart_type = bar`. En el log se observa la razón reportada por el LLM (“*El usuario pidió explícitamente un gráfico de barras*”) junto con la metadata (`rows` y `unique{year_aux}`).
- `generate_chart_spec`: arma la especificación del gráfico con eje X = `year_aux` y series Y de matrícula (por ejemplo, `female_enrolled`, `male_enrolled` o totales disponibles), además de título y etiquetas legibles.
- `render_chart`: produce el gráfico estático con formato legible (separadores de miles en valores absolutos; disposición horizontal si hay muchas categorías/etiquetas). Junto al gráfico se renderiza una *tabla* en el frontend con las primeras filas, lo que permite verificar los datos exactos usados en la visualización.
- `explain_chart`: genera 4–5 viñetas interpretativas alineadas con la consulta (sin código ni repetición de la tabla/imagen).

CAPÍTULO 8 : DESARROLLO



(a) Frontend: gráfico de barras con la evolución por año y la tabla de apoyo renderizada en la interfaz. La tabla permite contrastar rápidamente los valores exactos visibles en el gráfico.

```

2025-09-24 00:55:11,411 INFO [router_intent] last_query='¿Cuál ha sido la cantidad de hombres y de mujeres matriculados en Ingeniería Civil Telemática en la USM, campus San Joaquín, desde el año 2019 hasta el año 2024? . Realiza un gráfico de barras' previous_query='¿Cuál ha sido la cantidad de hombres y de mujeres matriculados en Ingeniería Civil Telemática en la USM, campus San Joaquín, desde el año 2019 hasta el año 2024? . Realiza un gráfico de barras'
2025-09-24 00:55:15,960 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 00:55:15,911 INFO router_intent intent-domain_query
2025-09-24 00:55:15,916 INFO III select_dataset_code: selected_datasets=None
2025-09-24 00:55:17,046 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 00:55:17,080 INFO select_dataset_code selected_datasets=[ 'Benchmarking_enrollment_program' ]
2025-09-24 00:55:17,083 INFO [extract_filters] (reset list_col) Entregado con selected_datasets=[ 'Benchmarking_enrollment_program' ]
2025-09-24 00:55:19,679 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 00:55:19,684 INFO [normalize_and_validate] Dataset=benchmarking_enrollment_program, raw_filters={'program_name': 'Ingeniería Civil Telemática', 'institution_name': 'UNIVERSIDAD TECNICA FEDERICO SANTA MARIA', 'campus_name': 'SAN JOAQUIN', 'year_aux': {'min': 2019, 'max': 2024}}
2025-09-24 00:55:19,683 INFO [normalize_and_validate] Normalized filters for Benchmarking_enrollment_program: {'program_name': 'Ingeniería Civil Telemática', 'institution_name': 'UNIVERSIDAD TECNICA FEDERICO SANTA MARIA', 'campus_name': 'SAN JOAQUIN', 'year_aux': {'min': 2019, 'max': 2024}}
2025-09-24 00:55:19,683 INFO [extract_filters] Filtros extraídos: {'Benchmarking_enrollment_program': {'program_name': 'Ingeniería Civil Telemática', 'institution_name': 'UNIVERSIDAD TECNICA FEDERICO SANTA MARIA', 'campus_name': 'SAN JOAQUIN', 'year_aux': {'min': 2019, 'max': 2024}}}
2025-09-24 00:55:19,683 INFO [extract_filters] Configuración de análisis: {'type': 'group_and_aggregate', 'group_by': ['year_aux'], 'metrics': {'female_enrolled': 'sum', 'male_enrolled': 'sum'}}
2025-09-24 00:55:19,684 INFO [apply_filters] aplicando filtros
2025-09-24 00:55:22,175 INFO [fuzzy_filter] Match parcial 'SAN JOAQUIN' en 'campus_name'
2025-09-24 00:55:22,181 INFO [apply_filters] 0 registros en 'Benchmarking_enrollment_program'
2025-09-24 00:55:22,180 INFO [execute_analysis] Ejecutando análisis generico
2025-09-24 00:55:22,191 INFO [execute_analysis] DataFrame para Benchmarking_enrollment_program: columns=[ 'year', 'total_enrolled', 'female_enrolled', 'male_enrolled', 'total_new_enrolled', 'female_new_enrolled', 'male_new_enrolled', 'in_situation_class_1', 'institution_name', 'institution_accred', 'region', 'province', 'city', 'campus_name', 'program_name', 'field_knowledge', 'std_program_name', 'global_level', 'program_class_1', 'teaching_mode', 'session_name', 'program_plan_type', 'program_sies_code', 'program_accred', 'municipal_school', 'local_service_school', 'upload_date', 'year_aux', 'institution_code', 'std_program_name_normalized', 'program_institution' ], filas=0
2025-09-24 00:55:22,292 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 00:55:22,290 DEBUG [llm_visualization_type] chart_type=bar, reason=El usuario pidió explícitamente un gráfico de barras para mostrar la cantidad de hombres y mujeres matriculados a lo largo de los años, meta-={'rows': 6, 'unique': ['year_aux': 6]}
2025-09-24 00:55:23,300 DEBUG [generate_chart_spec] spec keys: ['type', 'filters', 'x', 'y', 'x_label', 'y_label', 'title']
2025-09-24 00:55:23,311 DEBUG [clear_previous_image] Remove static/plots/011-current.png
2025-09-24 00:55:20,718 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 00:55:20,706 INFO [chat] 8502194c280d194cc52f573 Image_url=ves table_mujeres expl_len=811
272.17.0.115324 - "POST /chat HTTP/1.1" 200
272.17.0.115780 - "GET /static/plots/sessions/c2065d104cc52f573/v12_1758675320.png HTTP/1.1" 200
272.17.0.115780 - "GET /static/plots/sessions/c2065d104cc52f573/v12_1758675320.png HTTP/1.1" 200
    
```

(b) Backend (log): se observa la extracción/normalización de filtros, la configuración analítica con agregación por year_aux, y la decisión chart_type = bar por solicitud explícita del usuario.

Figura 8.12: Evidencia del flujo de visualización para el caso de barras por género: desde la decisión del LLM (respeto a la petición explícita del usuario) hasta el render final y la tabla asociada en el frontend.

Fuente: Elaboración propia.

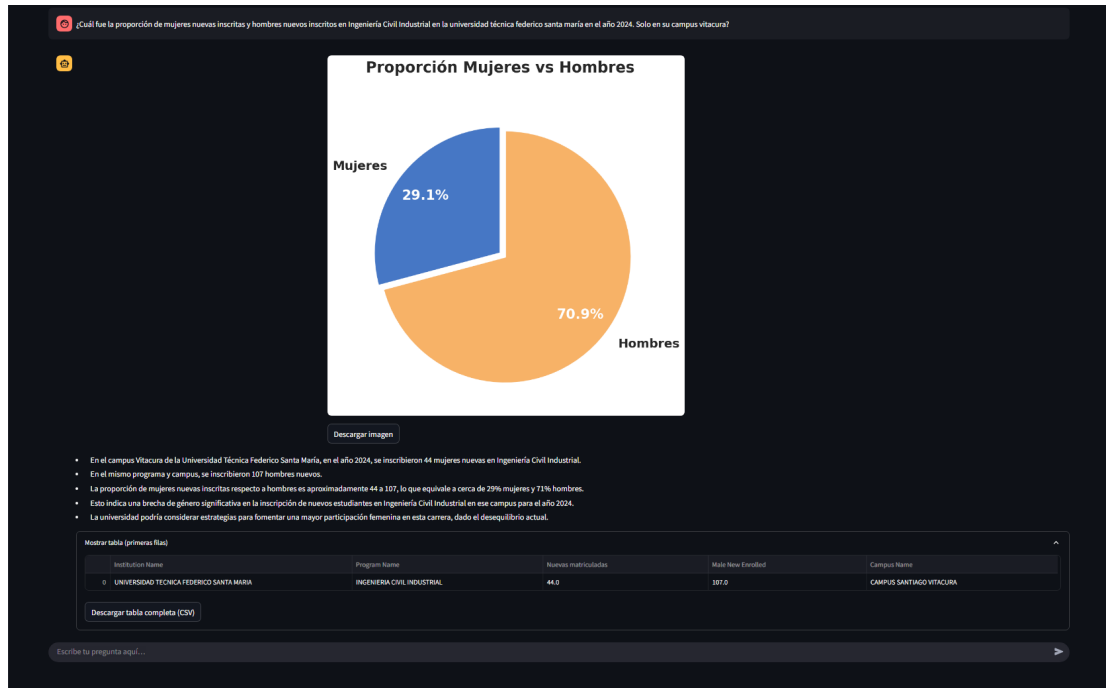
8.5.8. Ejemplo 3: Proporción por género (UTFSM–Vitacura, 2024)

Consulta del usuario: “¿Cuál fue la proporción de mujeres nuevas inscritas y hombres nuevos inscritos en Ingeniería Civil Industrial en la Universidad Técnica Federico Santa María en el año 2024, solo en su campus Vitacura?”

Ejecución registrada (foco en visualización).

- `select_dataset: Benchmarking_enrollment_program.`
- `extract_filters` → filtros normalizados: `{year_aux: 2024, program_name: “Ingeniería Civil Industrial”, institution_name: “Universidad Técnica Federico Santa María”, campus_name: “Vitacura”}`.
- `apply_filters + fuzzy_filter:` coincidencia sobre `campus_name = “Vitacura”`; quedan 2 registros crudos del dataset para ese campus.
- `analysis_config:` `{type: “group_and_aggregate”, metrics: {female_new_enrolled: “sum”, male_new_enrolled: “sum”}}`; tras la agregación, 1 fila (Vitacura 2024).
- `prepare_visual_df:` DataFrame consolidado (1×5) con `female_new_enrolled` y `male_new_enrolled`.
- `llm_visualization_type:` `chart_type = "pie", reason = “La consulta es sobre proporciones de género y se tienen métricas de mujeres y hombres nuevos inscritos”`; metadatos: `rows = 1`.
- `generate_chart_spec:` `labels = [“Mujeres”, “Hombres”], values = [44, 107]`.
- `add_proportion_columns:` porcentajes automáticos (29.1 % vs 70.9 %).
- `render_chart:` imagen `viz_current.png` y tabla asociada en el frontend (una única fila para Vitacura).

CAPÍTULO 8 : DESARROLLO



(a) Frontend: gráfico de torta para la proporción de mujeres y hombres nuevos inscritos en Ingeniería Civil Industrial (UTFSM, campus Vitacura, 2024). La tabla bajo la figura muestra la única fila resultante tras el filtrado por campus.

```

2025-09-24 00:45:15,274 INFO [router_intent] last_query: ¿Cuál fue la proporción de mujeres nuevas inscritas y hombres nuevos inscritos en Ingeniería Civil Industrial en la universidad técnica federico santa maría en el año 2024. Solo en su campus vitacura? previous_query: ¿Cuál fue la proporción de mujeres nuevas inscritas y hombres nuevos inscritos en Ingeniería Civil Industrial en la universidad técnica federico santa maría en el año 2024. Solo en su campus vitacura?
2025-09-24 00:45:16,245 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 00:45:16,249 INFO router_intent intent=domain_query
2025-09-24 00:45:16,252 INFO IN select_dataset_codes: selected_datasets=None
2025-09-24 00:45:18,252 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 00:45:18,257 INFO select_dataset_node selected_datasets=[Benchmarking_enrollment_program]
2025-09-24 00:45:18,266 INFO [extract_filters] [reset_list_col] Entrando con selected_datasets=[Benchmarking_enrollment_program]
2025-09-24 00:45:20,415 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 00:45:20,419 INFO [normalize_and_validate] Dataset=Benchmarking_enrollment_program, raw_filters={'program_name': 'Ingeniería Civil Industrial', 'institution_name': 'Universidad Técnica Federico Santa María', 'year_aux': 2024, 'campus_name': 'Vitacura'}
2025-09-24 00:45:20,423 INFO [normalize_and_validate] Normalized filters for Benchmarking_enrollment_program: {'program_name': 'Ingeniería Civil Industrial', 'institution_name': 'Universidad Técnica Federico Santa María', 'year_aux': 2024, 'campus_name': 'Vitacura'}
2025-09-24 00:45:20,423 INFO [extract_filters] Filtros extraídos: {'Benchmarking_enrollment_program': {'program_name': 'Ingeniería Civil Industrial', 'institution_name': 'Universidad Técnica Federico Santa María', 'year_aux': 2024, 'campus_name': 'Vitacura'}}
2025-09-24 00:45:20,423 INFO [extract_filters] Configuración de análisis: {'type': 'group_and_aggregate', 'group_by': ['program_name', 'institution_name', 'campus_name'], 'metrics': [{'female_new_enrolled': 'sum', 'male_new_enrolled': 'sum'}]}
2025-09-24 00:45:20,425 INFO [apply_filters] Aplicando filtros
2025-09-24 00:45:22,953 INFO [fuzzy_filters] Match parcial 'Vitacura' en 'campus_name'
2025-09-24 00:45:22,955 INFO [apply_filters] 2 registros en 'Benchmarking_enrollment_program'
2025-09-24 00:45:22,965 INFO [secure_analysis] Ejecutando análisis generico
2025-09-24 00:45:22,965 INFO [execute_analysis] DataFrame para Benchmarking_enrollment_program: columns=['year', 'total_enrolled', 'female_enrolled', 'male_enrolled', 'total_new_enrolled', 'female_new_enrolled', 'male_new_enrolled', 'institution_class_1', 'institution_name', 'institution_accred', 'region', 'province', 'city', 'campus_name', 'program_name', 'field_knowledge', 'std_program_name', 'global_level', 'program_class_1', 'teaching_name', 'session_name', 'program_plan_type', 'program_size_code', 'program_access', 'municipal_school', 'local_service_school', 'upload_date', 'year_aux', 'institution_code', 'std_program_name_normalized', 'program_institution'], files=2
2025-09-24 00:45:23,044 DEBUG [prepare_visual_df] DataFrame shape: (1, 5)
2025-09-24 00:45:24,924 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 00:45:24,928 DEBUG [llm_visualization_type] chart_type=pie, reason=La consulta es sobre proporciones de género y se tienen métricas de mujeres y hombres nuevos inscritos, meta={"rows": 1, "unique": {"institution_name": 1, "program_name": 1}}
2025-09-24 00:45:24,944 DEBUG [generate_chart_spec] spec keys: ['type', 'filters', 'labels', 'values', 'title']
2025-09-24 00:45:24,948 DEBUG [clear_previous_image] Removed static/plots/v12_current.png
2025-09-24 00:45:27,938 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 00:45:27,949 INFO [chat] session=c205d194cc52f573 image_urls=table_mdyes expl_len=601
272.17.0.152484 - POST /chat HTTP/1.1 200
272.17.0.152474 - GET /static/plots/sessions/c205d194cc52f573/v12_1758674727.png HTTP/1.1 200
272.17.0.152484 - GET /static/plots/sessions/c205d194cc52f573/v12_1758674727.png HTTP/1.1 200
    
```

(b) Backend (log): filtros con campus_name = Vitacura, agregación sum de female_new_enrolled y male_new_enrolled, y decisión del LLM chart_type = pie por tratarse de proporciones de género.

Figura 8.13: Evidencia del subgrafo de visualización en un caso de proporciones: filtro por campus, selección de pie por el LLM y render del gráfico con su tabla de soporte.

Fuente: Elaboración propia.

8.5.9. Ejemplo 4: Top 10 programas con más graduados (2023)

Consulta del usuario: “Top 10 programas de pregrado con más graduados en 2023”.

Trazabilidad centrada en visualización (resumida).

- Cambio de base de datos a `Benchmarking_stats_graduate_program`.
- `analysis_config`: {type: “group_and_aggregate”, group_by: [“program_name”], metrics: {total_graduates: “sum”}, order_by: {total_graduates: “desc”}, limit: 10}.
- `llm_visualization_type`: `chart_type = bar` (comparación/ranking de categorías).
- `generate_chart_spec`: barras horizontales con `program_name` en el eje Y y `total_graduates` en el eje X.

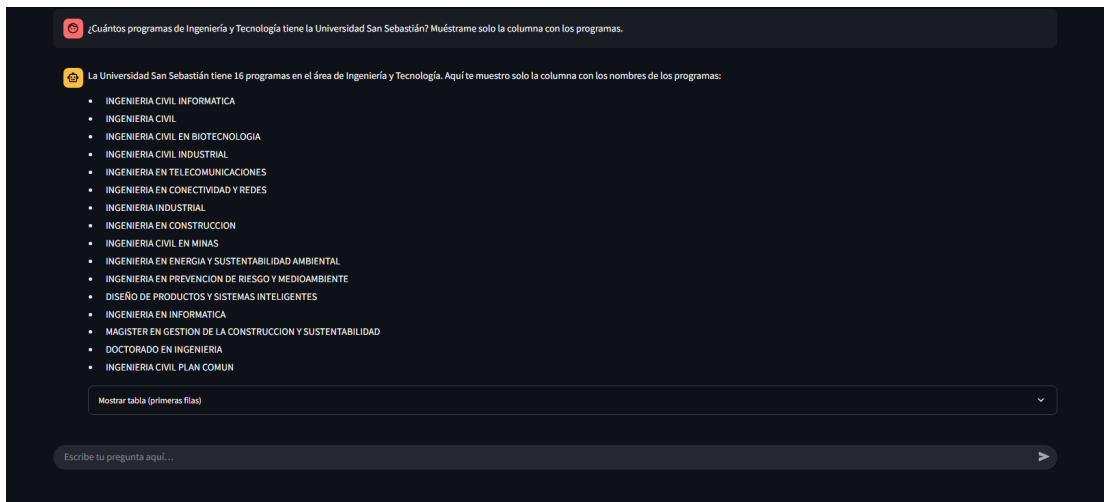
8.5.10. Ejemplo 5: Listado puntual sin gráfico (tabla)

Consulta del usuario: “¿Cuántos programas de Ingeniería y Tecnología tiene la Universidad San Sebastián? Muéstrame solo la columna con los programas.”

Trazabilidad centrada en visualización.

- `select_dataset` → `Benchmarking_programs` (base distinta a los ejemplos previos).
- `extract_filters` → filtros normalizados: `{institution_name: “UNIVERSIDAD SAN SEBASTIÁN”, field_knowledge: [“Ingeniería”, “Tecnología”]}`.
- `apply_filters` → 16 registros válidos.
- `prepare_visual_df` → DataFrame consolidado de tamaño 16×1 (solo la columna `program_name`).
- `llm_visualization_type` → `chart_type = table`, `reason = “La consulta requiere mostrar solo la columna con los programas; es un listado puntual”`.
- `generate_chart_spec` → especificación mínima de tabla; se omiten ejes/series por no aplicar.
- `render_chart` → no se genera imagen; se envía únicamente la tabla (y `clear_previous_image` elimina un PNG previo si existía).

CAPÍTULO 8 : DESARROLLO



(a) *Frontend: respuesta como listado/tabla de los 16 programas; no se renderiza gráfico por tratarse de una consulta puntual.*

```
2025-09-24 01:04:44,570 INFO [router_intent] last_query='¿Cuántos programas de Ingeniería y Tecnología tiene la Universidad San Sebastián? Muéstrame solo la columna con los programas.' previous_query='¿Cuántos programas de Ingeniería y Tecnología tiene la Universidad San Sebastián? Muéstrame solo la columna con los programas.'
2025-09-24 01:04:50,591 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 01:04:50,594 WARNING [router_intent] 2025: intent not found: Expecting ',' delimiter: line 1 column 149 (char 139), resp: [{"dataset": "Benchmarking_programs", "filters": [{"institution_name": "Universidad San Sebastián", "field_knowledge": ["Ingeniería", "Tecnología"]}, {"institution": "Tecnología"}], "columns": ["program_name"]}]. Your favor, confirme si desea que incluya programas que contengan cualquiera de las dos áreas (Ingeniería o Tecnología) o solo aquellos que contengan ambas. También, ¿desea algún filtro adicional, como nivel de programas o año?
2025-09-24 01:04:50,594 INFO [router_intent] intent=domania_query
2025-09-24 01:04:50,597 INFO IN select_dataset_mode: selected_datasets=None
2025-09-24 01:04:51,000 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 01:04:51,004 INFO select_dataset_mode selected_datasets=["Benchmarking_programs"]
2025-09-24 01:04:51,700 INFO [extract_filters] [reset_list_col] Entrando con selected_datasets=["Benchmarking_programs"]
2025-09-24 01:04:51,727 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 01:04:51,772 INFO [normalize_and_validate] Dataset=Benchmarking_programs, raw_filters={"institution_name": "UNIVERSIDAD SAN SEBASTIAN", "field_knowledge": ["Ingeniería", "Tecnología"]}
2025-09-24 01:04:51,778 INFO [normalize_and_validate] Normalized filters for Benchmarking_programs: {"institution_name": "UNIVERSIDAD SAN SEBASTIAN", "field_knowledge": ["Ingeniería", "Tecnología"]}
2025-09-24 01:04:51,775 INFO [extract_filters] Filtros extraídos: {"institution_name": "UNIVERSIDAD SAN SEBASTIAN", "field_knowledge": ["Ingeniería", "Tecnología"]}
2025-09-24 01:04:51,775 INFO [extract_filters] Configuración de análisis: {"type": "distribution", "column": "program_name", "compare_by": "field_knowledge"}
2025-09-24 01:04:51,778 INFO [apply_filters] Aplicando filtros
2025-09-24 01:04:51,923 INFO [apply_filters] 16 registros en 'Benchmarking_programs'
2025-09-24 01:04:51,928 INFO [execute_analysis] Ejecutando análisis generico
2025-09-24 01:04:51,933 INFO [execute_analysis] DataFrame para Benchmarking_programs: columns=["institution_class_1", "institution_code", "institution_name", "program_name", "field_knowledge", "std_program_name", "global_level", "program_class_1", "program_area_code", "order", "institution_group", "std_program_name_normalized", "program_institution"], files=16
2025-09-24 01:04:51,968 DEBUG [prepare_visual_df] DataFrame shape: (16, 1)
2025-09-24 01:04:51,200 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 01:04:51,211 DEBUG [fill_visualization_type] chart_type=table, reason=la consulta requiere mostrar solo la columna con los programas, es un listado puntual, meta={"rows": 16, "unique": ()}
2025-09-24 01:04:51,211 DEBUG [generate_chart_spec] spec keys: ["type", "filters", "title"]
2025-09-24 01:04:51,217 DEBUG [close_previous_image] Remove static/dists/viz_current.png
2025-09-24 01:04:51,269 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 01:04:51,318 INFO [chat] session=2B02194cc22f573 image_url=no table_mdyes expl_len=746
172.17.0.1:54980 - "POST /chat HTTP/1.1" 200
172.17.0.1:54980 - "GET /health HTTP/1.1" 200
```

(b) *Backend (log): chart_type = table, rows = 16, DataFrame shape = (16, 1); dataset Benchmarking_programs y filtros por institución y áreas.*

Figura 8.15: Caso sin gráfico: el módulo de visualización degrada a table cuando el usuario pide “solo la columna” (listado puntual).

Fuente: Elaboración propia.

8.5.11. Ejemplo 6: Top 10 universidades con mejor empleabilidad a 1 año en Enfermería (2021)

Consulta del usuario: “Promedio de empleabilidad a 1 año por universidad (sin institutos), en enfermería en 2021. Dame un top 10.”

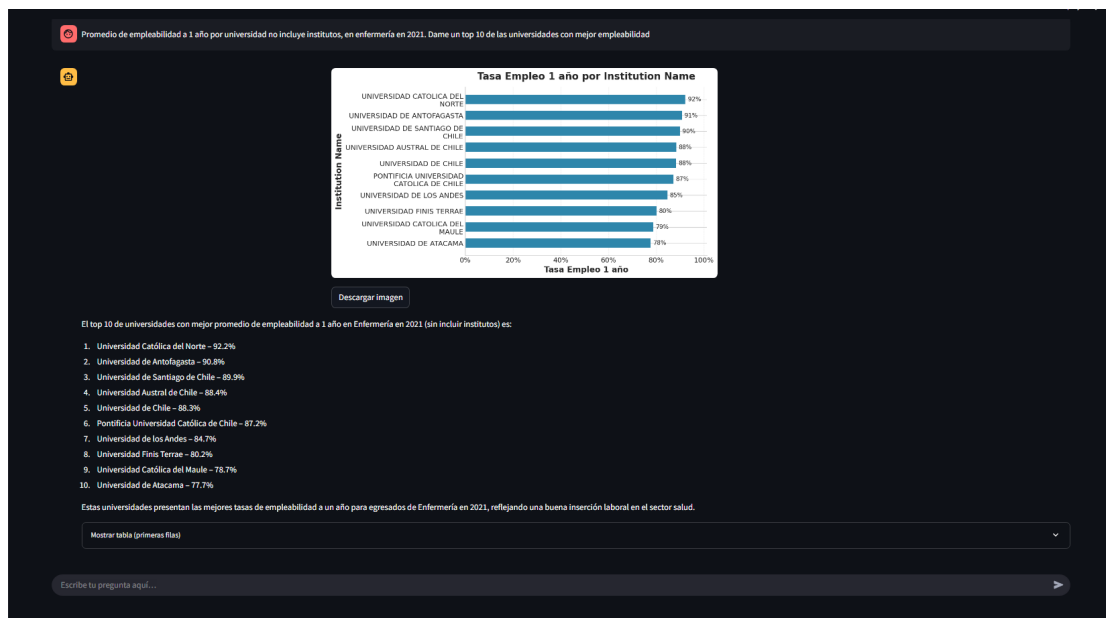
Trazabilidad (concisa).

- Dataset: Benchmarking_national_employment.
- Filtros: year = 2021, program_gen_name_normalized = “ENFERMERIA”,

CAPÍTULO 8 : DESARROLLO

institution_type = “Universidades”.

- Análisis: group_by = institution_name, metric = emp_rate_1yr (mean), order_by = desc, limit = 10.
- Tipo de gráfico: bar (decisión del LLM por tratarse de un top-10 comparativo).
- Render: barras horizontales con % en etiquetas; tabla de respaldo disponible en el frontend.



(a) Frontend: gráfico de barras (top-10) de emp_rate_1yr en Enfermería 2021 (solo universidades).

```
2025-09-24 01:13:57,385 INFO [router:interact] last_query='Promedio de empleabilidad a 1 año por universidad no incluye institutos, en enfermería en 2021. Dame un top 10 de las universidades con mejor empleabilidad' previous_query='Promedio de empleabilidad a 1 año por universidad no incluye institutos, en enfermería en 2021. Dame un top 10 de las universidades con mejor empleabilidad'
2025-09-24 01:13:59,430 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 01:13:59,436 INFO router:interact:dataset_name=query
2025-09-24 01:13:59,440 INFO IN select_dataset_node: selected_datasets=None
2025-09-24 01:14:00,550 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 01:14:00,560 INFO select_dataset_node selected_datasets=['Benchmarking_national_employment']
2025-09-24 01:14:00,568 INFO [extract:filters] (reset list col) Entrando con selected_datasets=['Benchmarking_national_employment']
2025-09-24 01:14:02,630 INFO [normalize_and_validate] Dataset=Benchmarking_national_employment, raw filters={'year': 2021, 'institution_type': 'Universidades', 'program_gen_name_normalized': 'ENFERMERIA'}
2025-09-24 01:14:02,638 INFO [normalize_and_validate] Normalized filters for Benchmarking_national_employment: {'year': 2021, 'institution_type': 'Universidades', 'program_gen_name_normalized': 'ENFERMERIA'}
2025-09-24 01:14:02,639 INFO [extract:filters] Filtrros extraídos: {'Benchmarking_national_employment': {'year': 2021, 'institution_type': 'Universidades', 'program_gen_name_normalized': 'ENFERMERIA'}}
2025-09-24 01:14:02,639 INFO [extract:filters] Configuración de análisis: {'type': 'group_and_aggregate', 'group_by': ['institution_name'], 'metrics': ['emp_rate_1yr': 'mean'], 'order_by': ['emp_rate_1yr': 'desc'], 'limit': 10}
2025-09-24 01:14:02,641 INFO [apply:filters] Aplicando filtros
2025-09-24 01:14:02,741 INFO [apply:filters] 42 registros en 'Benchmarking_national_employment'
2025-09-24 01:14:02,751 INFO [execute:analysis] Ejecutando análisis generico
2025-09-24 01:14:02,753 INFO [execute:analysis] DataFrame para Benchmarking_national_employment: columns=['year', 'institution_type', 'institution_name', 'area', 'accred_status', 'program_gen_name', 'program_name', 'cont_rate', 'retent_index_rate_1yr', 'program_duration_sem', 'emp_rate_1yr', 'emp_rate_2yr', 'income_brackets', 'annual_tuition', 'institution_name_normalized', 'program_gen_name_normalized', 'program_institution'], files=41
2025-09-24 01:14:02,800 DEBUG [prepare_visual:cf] DataFrame shape: (10, 2)
2025-09-24 01:14:05,575 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 01:14:05,579 DEBUG [lim_visualization_type] chart_type=bar, reason-La consulta pide un top-10 de universidades con mejor empleabilidad, que es una comparación de categorías, meta={"rows": 10, "unique": [{"institution_name": 10}
2025-09-24 01:14:05,595 DEBUG [generate_chart_spec] spec keys: ['type', 'filters', 'x', 'y', 'x_label', 'y_label', 'title']
2025-09-24 01:14:09,824 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 01:14:09,823 INFO [chat] session=28019kcc21f73 image_url=yes table_md=yes expl_len=651
172.17.0.1:50234 - "POST /chat HTTP/1.1" 200
172.17.0.1:50234 - "GET /health HTTP/1.1" 200
```

(b) Log (extracto): confirma dataset, filtros y chart_type = bar.

Figura 8.16: Caso de empleabilidad: comparación categórica (top-10) → barras, con tabla visible para verificación.

Fuente: Elaboración propia.

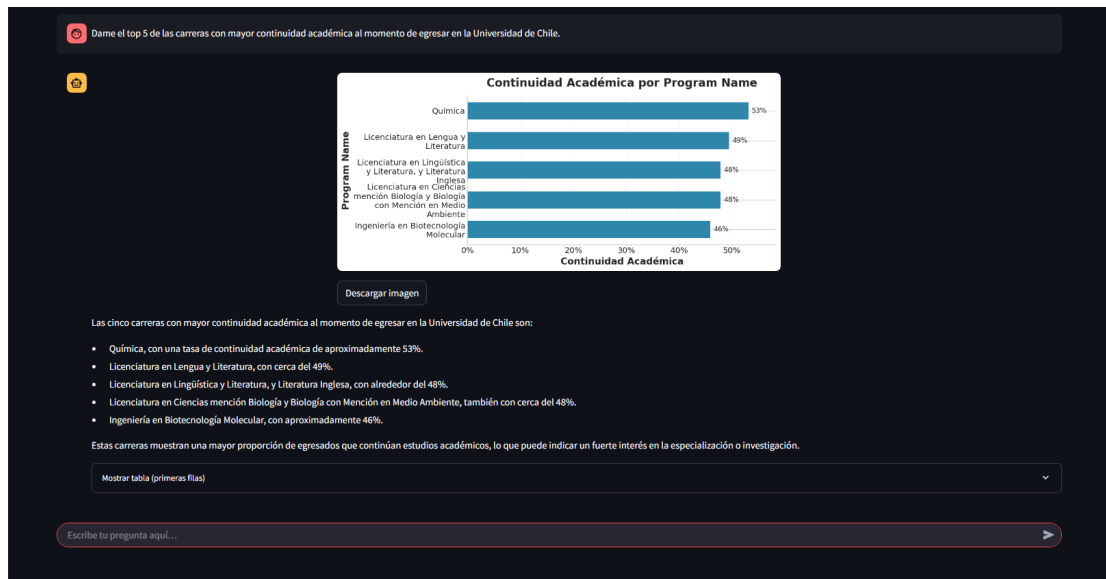
8.5.12. Ejemplo 7: Top-5 de continuidad académica (barra)

Consulta del usuario: “Dame el top 5 de las carreras con mayor continuidad académica al momento de egresar en la Universidad de Chile.”

Trazabilidad centrada en visualización (breve).

- `select_dataset` → `Benchmarking_national_employment`.
- `extract_filters` → `{institution_name: “Universidad de Chile”}`.
- `analysis_config` → `{type: “group_and_aggregate”, group_by: [“program_name”], metrics: {cont_rate: “max”}, order_by: {cont_rate: “desc”}, limit: 5}`.
- `prepare_visual_df` → tabla 5×2 (`program_name`, `cont_rate`).
- `llm_visualization_type` → `chart_type = bar` (detección de *top-5/comparación de categorías*).
- `generate_chart_spec` → eje `X=program_name`, serie `Y=cont_rate`; formateo en %.
- `render_chart` → barras horizontales con etiquetas en porcentaje y tabla asociada.

CAPÍTULO 8 : DESARROLLO



(a) Frontend: gráfico de barras (horizontal) del Top 5 de continuidad académica por programa en la U. de Chile; el eje Y muestra cont_rate con formato %.

```
2025-09-24 01:32:53,804 INFO [router:Intent] list_query: Dame el top 5 de las carreras con mayor continuidad academica al momento de egresar en la Universidad de Chile. previous_query= Dame el top 5 de las carreras con mayor continuidad academica al momento de egresar en la Universidad de Chile.
2025-09-24 01:32:56,311 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 01:32:56,315 INFO [router:Intent] 2506 Intent mal formado: Expecting ',' delimiter: line 1 column 183 (char 182), resp=({"datasets":["Benchmarking_national_employment"],"filters":{"institution_name":"Universidad de Chile"},"metrics":{"cont_rate"},"group_by":["program_name"],"sort_by":"cont_rate","sort_order":"desc","limit":5} \n\npor favor confirma o indica si deseas filtrar por año o algún otro criterio para realizar el análisis.
2025-09-24 01:32:56,319 INFO IN select_dataset_node: selected_datasets=None
2025-09-24 01:32:57,933 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 01:32:57,938 INFO select_dataset_node: selected_datasets=[ 'Benchmarking_national_employment' ]
2025-09-24 01:32:57,946 INFO [extract_filters] (reset list col) entrando con selected_datasets=[ 'Benchmarking_national_employment' ]
2025-09-24 01:32:59,990 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 01:32:59,999 INFO [normalize_and_validate] Dataset=Benchmarking_national_employment, row_filters={'institution_name': 'Universidad de Chile'}
2025-09-24 01:33:00,001 INFO [normalize_and_validate] Normalized filters for Benchmarking_national_employment: {'institution_name': 'Universidad de Chile'}
2025-09-24 01:33:00,002 INFO [extract_filters] filtros extraídos: {'Benchmarking_national_employment': {'institution_name': 'Universidad de Chile'}}
2025-09-24 01:33:00,003 INFO [extract_filters] Configuración de análisis: {'type': 'group_and_aggregate', 'group_by': ['program_name'], 'metrics': {'cont_rate': 'max'}, 'order_by': ('cont_rate', 'desc'), 'limit': 5}
2025-09-24 01:33:00,008 INFO [apply_filters] aplicando filtros
2025-09-24 01:33:00,166 INFO [apply_filters] 267 registros en 'Benchmarking_national_employment'
2025-09-24 01:33:00,189 INFO [execute_analysis] ejecutando análisis generico
2025-09-24 01:33:00,193 INFO [execute_analysis] dataframe para Benchmarking_national_employment: columns=[ 'year', 'institution_type', 'institution_name', 'area', 'accred_status', 'program_gen_name', 'program_name', 'cont_rate', 'retenti
on_rate_1yr', 'program_duration_sens', 'emp_rate_1yr', 'emp_rate_2yr', 'income_brackets', 'annual_tuition', 'institution_name_normalized', 'program_gen_name_normalized', 'program_institution' ], filas=267
2025-09-24 01:33:01,236 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
2025-09-24 01:33:01,259 INFO [prepare_visual_of] Dataframe shape: (5, 2)
2025-09-24 01:33:01,265 INFO [prepare_visual_of] chart_type=bar, reason=La consulta pide un top 5 de carreras con mayor continuidad académica, lo que implica comparar categorías y el usuario pidió un gráfico de barras, meta={"rows": 5, "unique": ["program_name": 5]}
2025-09-24 01:33:01,268 INFO [generate_chart_spec] spec keys: [ 'type', 'filters', 'x', 'y', 'x_label', 'y_label', 'title' ]
2025-09-24 01:33:01,268 INFO [clear_previous_image] Removed static/plots/viz_current.png
2025-09-24 01:33:01,095 INFO HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
```

(b) Backend (log): selección de dataset, filtros por institución, configuración de group_and_aggregate con order_by: desc y limit: 5, y decisión chart_type = bar.

Figura 8.17: Evidencia del pipeline de visualización para un top-N: el LLM sugiere bar, se construye la chart_spec y se renderiza el gráfico junto a la tabla.

Fuente: Elaboración propia.

8.6. Grafo padre

El *grafo padre* (u *orquestrador*) estructura el ciclo de atención de cada turno conversacional. Su función es coordinar, en un flujo controlado, la detección de intención, la preparación de datos (selección de conjuntos y filtrado) y la generación de salidas visuales y tabulares. La implementación se basa en `StateGraph` de `LangGraph`, que permite definir nodos (unidades de trabajo) y transiciones condicionales sobre un estado compartido.

8.6.1. Estado orquestado

El grafo opera sobre un estado explícito (`OrchestratorState`) que encapsula las variables de trabajo de cada turno:

- `intent` (`greeting`, `domain_query`, `out_of_domain`): intención semántica del último mensaje.
- `selected_datasets`: lista de conjuntos de datos pertinentes a la consulta.
- Artefactos de análisis y visualización: `filters`, `analysis_config`, `analysis_result`, `visualization_df`, `chart_type`, `chart_spec`, entre otros.

Este diseño hace explícitas las dependencias entre pasos y facilita decisiones de control basadas en la presencia efectiva de datos.

8.6.2. Nodos y responsabilidades

El grafo se compone de tres subprocesos funcionales y un cierre:

1. **Router** (`router`): clasifica la intención, valida que la consulta pertenece al dominio y determina los *datasets* candidatos.
2. **Filtro y análisis** (`filter`): a partir de la consulta y de los esquemas disponibles, extrae filtros estructurados (p. ej., institución, carrera, años) y produce un resultado de análisis cuando hay evidencia suficiente en los datos.
3. **Visualización** (`viz`): sintetiza la salida en forma de gráfico y/o tabla, acompañada de una explicación breve.

4. **Limpieza** (`end_cleanup`): elimina del estado las claves temporales para evitar arrastre de resultados entre turnos.

8.6.3. Lógica de control

El flujo se gobierna mediante dos decisiones condicionales (implementadas en `next_after_router` y `next_after_filter`):

- **Transición tras router**: si la intención es saludo o fuera de dominio, o si no se seleccionan *datasets*, el grafo finaliza en `end_cleanup`. En caso contrario, avanza a `filter`.
- **Transición tras filter**: solo continúa a `viz` cuando existen *filas visualizables*; de lo contrario, termina en `end_cleanup`.

Consideramos *filas visualizables* aquellas presentes en `analysis_result` o `filtered_data` con longitud mayor que cero; esta verificación se realiza en `_has_visualizable_rows`.

El recorrido típico puede resumirse así:

$$\text{START} \rightarrow \text{router} \Rightarrow \begin{cases} \text{end_cleanup} & \text{si no hay intención/datasets válidos} \\ \text{filter} \Rightarrow \begin{cases} \text{end_cleanup} & \text{si no hay filas visualizables} \\ \text{viz} \rightarrow \text{end_cleanup} & \text{si hay datos suficientes} \end{cases} \end{cases}$$

8.6.4. Justificación del diseño

El patrón orquestado aporta:

- **Separación de responsabilidades**: cada nodo resuelve una tarea cohesionada (intención, filtros/análisis, visualización).
- **Eficiencia y robustez**: se interrumpe temprano en escenarios no informativos, evitando cómputo innecesario.
- **Higiene de estado**: el cierre sistemático (`end_cleanup`) previene efectos residuales entre turnos y simplifica el razonamiento del flujo.

En conjunto, el grafo padre ofrece trazabilidad clara de decisiones y es extensible: permite incorporar nuevos nodos o refinar condiciones sin alterar el contrato externo.

8.7. API Backend

La API se implementó con *FastAPI* y actúa como puente entre el frontend y el grafo conversacional de análisis. Su responsabilidad es recibir el historial del chat, invocar el motor (`parent_graph`) y devolver una respuesta compacta lista para renderizar (gráfico, explicación y tabla). Además, sirve archivos estáticos (imágenes) y gestiona su ciclo de vida.

8.7.1. Objetivo y alcance

El backend expone un contrato simple que desacopla la interfaz de usuario del motor de análisis:

- **Entrada:** historial de mensajes en formato `[{role, content}, ...]`.
- **Salida:** objeto JSON con `image_url`, `explanation`, `table_md` y `session_id`.

Con ello, toda la lógica de extracción de filtros, aplicación y generación de visualizaciones queda encapsulada en el grafo; la API solo orquesta y normaliza.

8.7.2. Endpoints y contrato

La API publica dos rutas:

GET /health Respuesta de *liveness* para monitoreo (`{"status": ".ok"}`).

POST /chat Punto principal de inferencia.

Solicitud a /chat (enviada por el frontend):

```
[  
  {"role": "assistant", "content": "...mensaje inicial..."},  
  {"role": "user",      "content": "...pregunta..."},
```

```
...  
]
```

Respuesta de /chat:

```
{  
  "image_url": "/static/plots/sessions/<id>/viz_<ts>.png",  
  "explanation": "bullets/explicación en texto plano",  
  "table_md": "| Col | ... |\n| --- | --- |\n...",  
  "session_id": "<id>"  
}
```

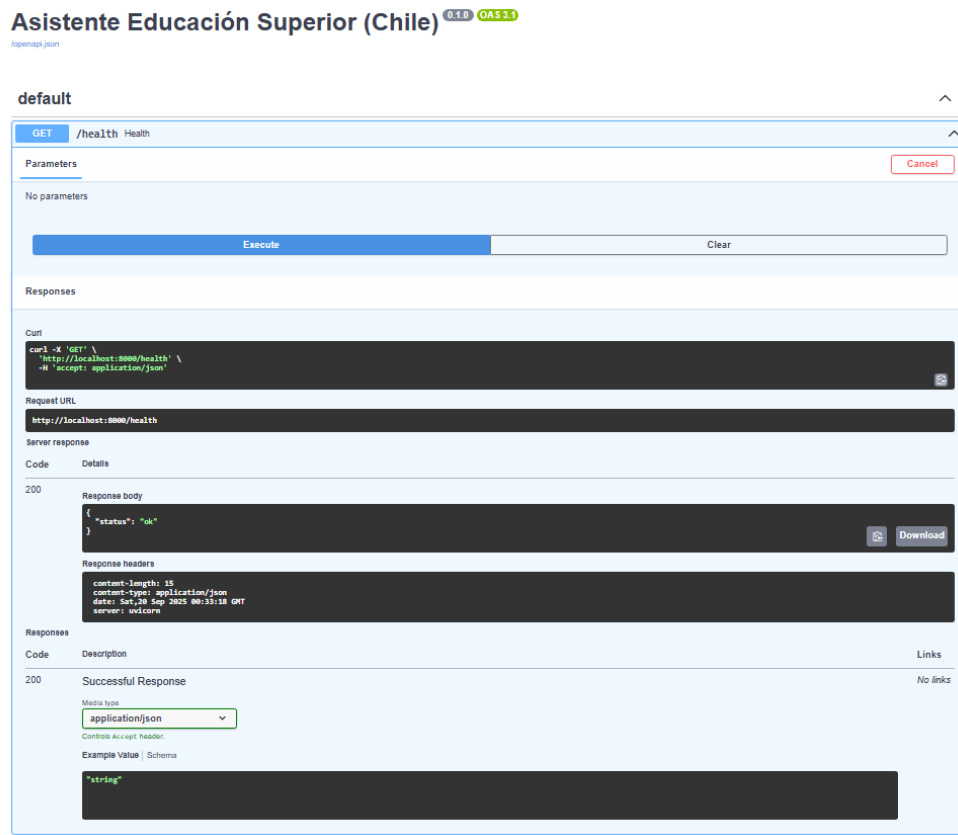


Figura 8.18: Prueba del endpoint GET /health en la documentación interactiva de FastAPI. Se observa la respuesta {"status": ".ok"} con código 200.

Fuente: Elaboración propia.

```
C:\Users\ignac>curl -s -X POST "http://localhost:8000/chat" -H "Content-Type: application/json" -H "X-Session-ID: demo123" -d
[{"role": "assistant", "content": "Hola, soy tu asistente de datos."}, {"role": "user", "content": "Cual es la cantidad de hombres y de mujeres que se matricula a ingenieria civil industrial en la usm, desde el año 2019 al 2024, realiza un grafico de barras"}]
{"image_url": "/static/plots/sessions/demo123/viz_1758329377.png", "explanation": "- Entre 2019 y 2024, la cantidad de hombres matriculados en Ingeniería Civil Industrial en la USM es consistentemente mayor que la de mujeres.\n- La matrícula masculina muestra una tendencia ligeramente creciente, alcanzando su punto máximo en 2023 con aproximadamente 1449 estudiantes.\n- La matrícula femenina también presenta un aumento, aunque más moderado, con un pico en 2022 con cerca de 744 estudiantes.\n- En general, la brecha de género se mantiene estable a lo largo de los años, con aproximadamente el doble de hombres matriculados en comparación con mujeres.\n- El gráfico de barras refleja claramente esta diferencia, con barras masculinas significativamente más altas que las femeninas en cada año analizado.", "table_md": "| Año | Matriculadas | Hombres matriculados |\n| --- | --- | --- |\n| 2019 | 621.0 | 1272.0 |\n| 2020 | 634.0 | 1222.0 |\n| 2021 | 728.0 | 1383.0 |\n| 2022 | 744.0 | 1445.0 |\n| 2023 | 716.0 | 1449.0 |\n| 2024 | 702.0 | 1408.0 |", "session_id": "demo123"}
C:\Users\ignac>
```

Figura 8.19: Prueba del endpoint POST /chat con curl: respuesta 200 y JSON con image_url, explanation, table_md y session_id.

Fuente: Elaboración propia.

Comando usado (curl):

```
curl -s -X POST "http://localhost:8000/chat" \
  -H "Content-Type: application/json" \
  -H "X-Session-ID: demo123" \
  -d '[{"role": "assistant", "content": "Hola, soy tu asistente de datos."},
    {"role": "user", "content": "Cual es la cantidad de hombres y de mujeres \
que se matricula a ingenieria civil industrial en la usm, desde el año 2019 \
al 2024, realiza un grafico de barras"}]'
```

8.7.3. Documentación interactiva y prueba rápida

La API expone documentación interactiva en /docs (Swagger UI), desde donde se pueden ejecutar GET /health y POST /chat sin salir del navegador.

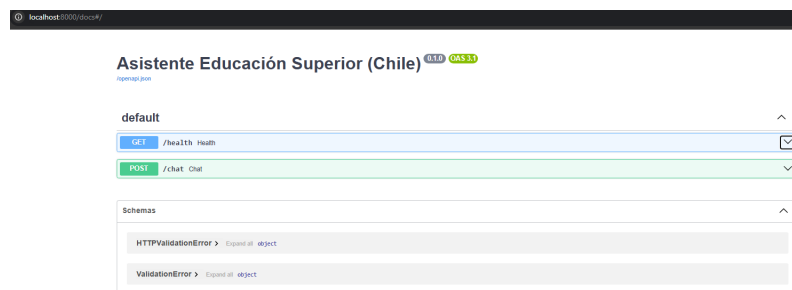


Figura 8.20: Portada de /docs generada por FastAPI con los endpoints /health y /chat.

Fuente: Elaboración propia.

8.7.4. Gestión de sesión e imágenes

Cada conversación se asocia a un `session_id` (del encabezado `X-Session-ID` o derivado de IP, agente y hora). Las visualizaciones se producen inicialmente como `/static/plots/viz_current.png`. Para evitar colisiones entre conversaciones y mejorar el almacenamiento en caché del navegador, la API:

1. **Promueve** el archivo global a una ruta de sesión con nombre versionado:
`/static/plots/sessions/<session_id>/viz_<timestamp>.png`.
2. Mantiene un **límite por sesión** (`MAX_FILES_PER_SESSION`) y elimina los más antiguos.
3. Ejecuta un proceso asíncrono (*janitor*) que **purga** imágenes con TTL vencido y limpia carpetas vacías.

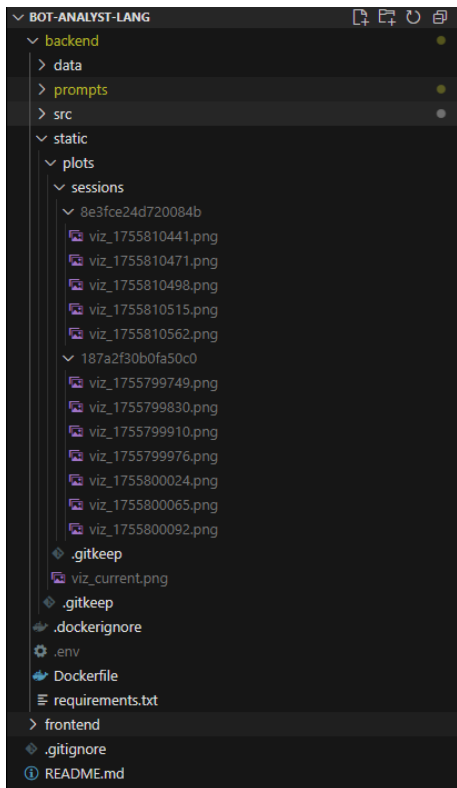


Figura 8.21: Evidencia de gestión de sesión e imágenes: creación de archivos `viz_<ts>.png` por sesión en `static/plots/sessions/<session_id>/` y archivo global `viz_current.png`.

Fuente: Elaboración propia.

8.7.5. Caché HTTP de estáticos

Para balancear frescura y desempeño se agregan encabezados de caché:

- Imágenes versionadas de sesión: `Cache-Control: public, max-age=31536000, immutable.`
- Recursos no versionados (p.ej., `viz_current.png`): `Cache-Control: public, max-age=300.`

Si la respuesta del grafo contiene una URL relativa a `/static`, la API la deja lista para el frontend y, cuando procede, agrega un parámetro de *cache busting*.

8.7.6. Normalización de la salida del grafo

El grafo produce mensajes en Markdown. La API:

1. **Extrae** la primera URL de imagen del Markdown.
2. **Detecta** tablas en sintaxis GitHub (ya sea como bloque `“ text` o líneas con `|`) y conserva el bloque como `table_md`. Si no vino una tabla explícita, intenta **reconstruirla** desde `state` (por ejemplo, `analysis_result`) serializándola a Markdown.
3. **Limpia** la explicación eliminando bloques de código para entregar texto plano amigable.

8.7.7. Invocación del motor conversacional

El historial recibido se convierte a objetos de *LangChain* (`HumanMessage`, `AIMessage`, `SystemMessage`) y se arma el estado mínimo:

- `messages`: historial completo,
- `last_query`: último mensaje del usuario,
- `session_id`: identificador de la conversación.

Luego se invoca `parent_graph.invoke(state, config)` con un `thread_id` igual a la sesión, para que `LangGraph` pueda **guardar checkpoints** y encaminar correctamente el flujo.

8.7.8. CORS, estáticos y robustez

Se habilita **CORS** abierto para permitir el consumo desde el cliente de Streamlit y se sirve `/static` con *FastAPI StaticFiles*. El backend captura *payloads* inválidos y devuelve mensajes de error seguros que el frontend muestra como avisos (sin exponer trazas internas).

8.7.9. Flujo resumido

1. El frontend envía el **historial** a `/chat`.
2. La API construye el **estado** e invoca el **grafo**.
3. Se **extraen** del Markdown la imagen (y se promueve a carpeta de sesión), la **explicación** y la **tabla**.
4. Se responde con **JSON** listo para renderizar. Un proceso asíncrono mantiene **limpia** la carpeta de imágenes por TTL.

8.7.10. Consideraciones actuales

- La API no ejecuta análisis por sí misma; delega el cómputo al grafo.
- Las tablas se entregan en Markdown; si el grafo no deja datos estructurados en el estado, la reconstrucción puede no ser posible en ciertos casos.
- El esquema de caché asume imágenes *inmutables* por nombre; los cambios requieren un nuevo archivo versionado.

8.8. Interfaz de Usuario (Frontend)

La interfaz de usuario se desarrolló en *Streamlit* y permite conversar con el asistente, ver resultados en forma de gráficos generados cuando el LLM determina que corresponde graficar e inspeccionar tablas con opción de descarga. Al igual que las tablas, los gráficos se presentan centrados y ofrecen un botón para descargar la imagen (por ejemplo, en formato PNG). El objetivo es que el usuario pueda preguntar en lenguaje natural y obtener una respuesta clara sin conocer detalles técnicos del sistema.

8.8.1. Objetivo y alcance

El frontend cumple tres funciones: (i) recoger la consulta del usuario y mantener el historial de la conversación; (ii) mostrar las salidas del asistente (gráficos, tablas y explicación); y (iii) ofrecer acciones simples como abrir una nueva conversación o descargar datos. La lógica de análisis y generación de contenidos se ejecuta en el backend; aquí solo se orquesta la interacción y la presentación.

8.8.2. Componentes principales de la interfaz

Entrada de chat. En la parte inferior se dispone de un campo de texto (`st.chat_input`) donde el usuario escribe su pregunta. Al enviar, el mensaje se agrega al historial local y se realiza una llamada HTTP al endpoint `/chat` del backend.

Historial de conversación. Cada mensaje se guarda en `st.session_state["history"]` con su `role` (`user/assistant`) y `content`. Al iniciar, se muestra un mensaje de bienvenida con instrucciones y un ejemplo, lo que ayuda a orientar la primera consulta.

Renderizado de respuestas. Las respuestas del asistente pueden incluir:

- **Imágenes:** si llega una URL de gráfico (`image_url`), se muestra centrada y con un botón de descarga. Si la ruta es relativa (`/static/...`), el frontend la convierte a URL absoluta para su visualización.

- **Tablas:** si el asistente envía una tabla en formato Markdown (estilo GitHub), el frontend la detecta mediante una expresión regular, la convierte a `DataFrame` y la presenta dentro de un *expander* para no saturar la vista, con opción de descarga en CSV.
- **Mensajes informativos:** cuando el backend pide aclaraciones o comunica advertencias/errores, se muestran como avisos (`st.info`, `st.warning`, `st.error`) para facilitar la corrección de la consulta.
- **Explicación:** cualquier texto explicativo se imprime como Markdown debajo del gráfico y/o la tabla.

Barra lateral. Incluye un botón “Nueva conversación” que limpia el historial y reinicia el estado de la sesión, además de la versión de la aplicación.

8.8.3. Gestión de estado y sesión

El frontend mantiene variables en `st.session_state`:

- `history`: lista de mensajes intercambiados.
- `processing/pending_response`: controlan el ciclo de envío y espera de respuesta.
- `session_id`: identificador de sesión devuelto por el backend para asociar gráficos y resultados a una misma conversación.
- `error`: último error mostrado al usuario.

Este estado evita perder el contexto al recargar la página y permite continuar la conversación sin interrupciones.

8.8.4. Flujo de interacción con el backend

Cada vez que el usuario envía un mensaje:

1. Se agrega al `history` local.
2. Se envía una petición POST a `BACKEND_URL /chat` con todo el historial (formato: lista de `{role, content}`).
3. El backend responde con un objeto JSON que puede contener: `image_url`, `explanation`, `table_md` y `session_id`.
4. El frontend compone un único mensaje de salida combinando (si existen) la imagen, la explicación y la tabla, y lo agrega al historial para su visualización por parte del usuario.

Este diseño mantiene el frontend simple y permite que la decisión sobre qué mostrar (gráfico, tabla o texto) recaiga en el backend.

8.8.5. Diseño visual

Se aplican estilos CSS ligeros para mejorar la lectura:

- **Área de entrada** con mayor ancho y tipografía más grande.
- **Mensaje de bienvenida** en un bloque destacado que guía la primera interacción.
- **Tablas** con encabezado resaltado y separación de filas para lectura rápida.
- **Imágenes** con ajuste al ancho disponible y centradas.

El resultado es una interfaz limpia, con jerarquía visual clara y sin distracciones.



Figura 8.22: Pantalla inicial del frontend. Se observa la barra lateral con la opción Nueva conversación, el título de la aplicación y el mensaje de bienvenida con un ejemplo de consulta.

Fuente: Elaboración propia.

8.8.6. Manejo de tablas e imágenes

Tablas Markdown. El asistente suele devolver tablas en sintaxis Markdown. El frontend:

1. Detecta el bloque de tabla con una expresión regular.
2. Convierte el bloque a `pandas.DataFrame`.
3. Muestra las primeras filas y ofrece descarga del CSV completo.

Esto permite inspeccionar resultados de forma interactiva y, si se desea, continuar el análisis fuera de la aplicación.

Imágenes y descargas. Si la respuesta incluye un gráfico, se muestra y se prepara un botón de “Descargar imagen”. Para evitar descargas repetidas, el frontend usa caché de *Streamlit* sobre la URL.

8.8.7. Experiencia de usuario

La experiencia se centra en tres ideas: (i) **preguntar y ver**, sin configurar nada; (ii) **explicaciones breves**, junto a cada salida; y (iii) **control simple**, con un botón para reiniciar la conversación y opciones de descarga directa. Con esto, la herramienta resulta útil tanto para exploración rápida como para generar insumos (gráficos y CSV) que se pueden incorporar en informes.

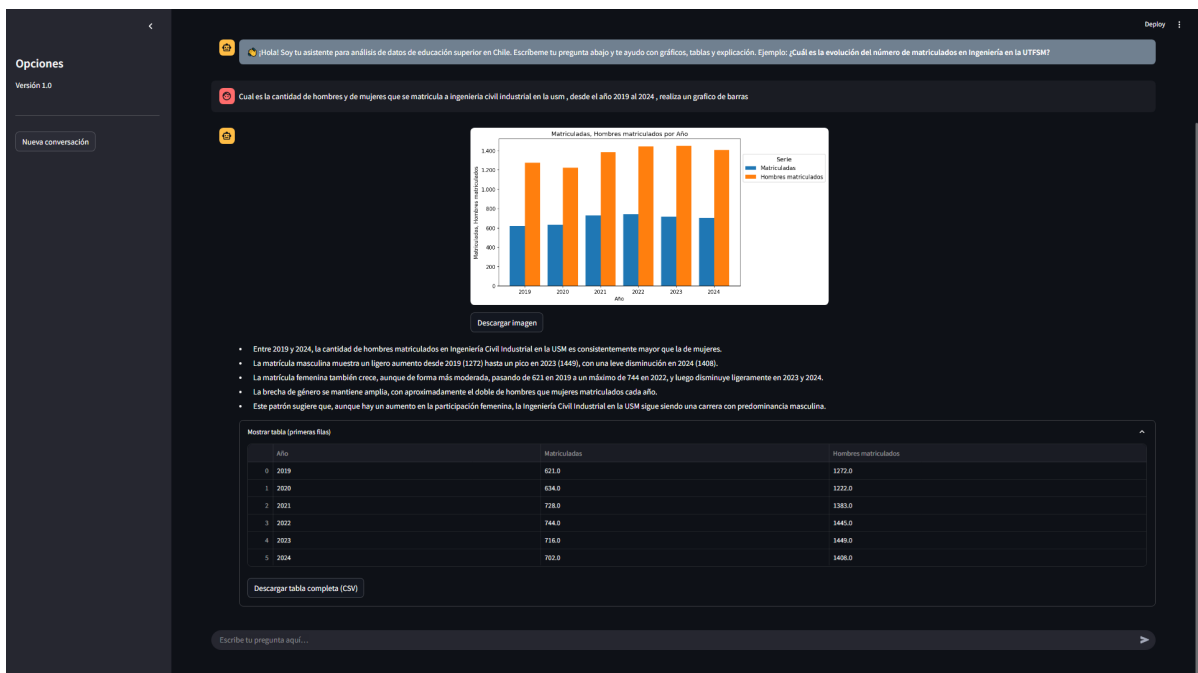


Figura 8.23: Flujo de respuesta en el frontend: tras la consulta se muestra el gráfico centrado con botón de descarga, una explicación en viñetas y la tabla de resultados (con opción de descargar CSV).

Fuente: Elaboración propia.

8.8.8. Ejemplo ilustrativo de interacción

A continuación, se muestra un ejemplo completo de interacción: desde la consulta inicial hasta la presentación del gráfico, la explicación y la tabla de resultados, con sus opciones de descarga. Este caso ilustra cómo los distintos componentes del frontend se integran en una única respuesta y cómo el usuario puede exportar fácilmente los resultados.



Figura 8.24: Respuesta del asistente mostrando un gráfico de barras centrado y el botón Descargar imagen. Esta opción permite exportar la visualización para informes o análisis adicionales. La barra lateral puede ocultarse con el control superior para disponer del área completa del chat.

Fuente: Elaboración propia.

CAPÍTULO 8 : DESARROLLO

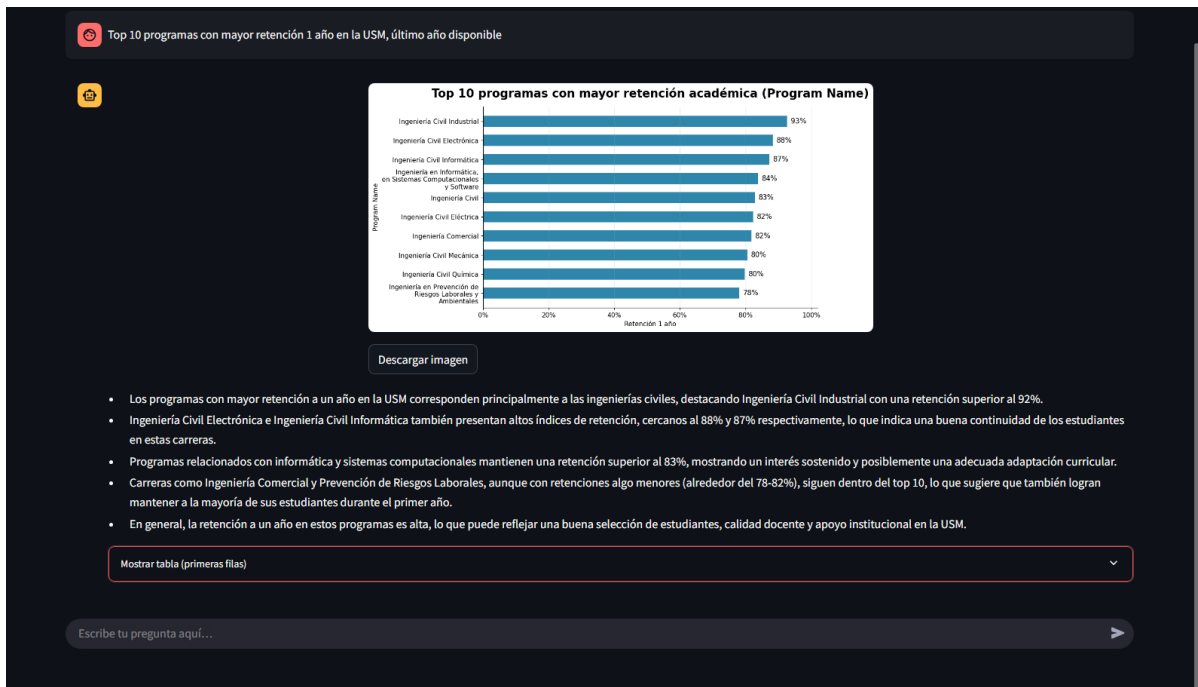


Figura 8.25: Explicación breve en viñetas bajo el gráfico. Este acompañamiento textual sintetiza hallazgos y orienta la lectura del resultado sin requerir conocimientos técnicos.

Fuente: Elaboración propia.

Mostrar tabla (primeras filas)

Program Name	Retención 1 año
0 Ingeniería Civil Industrial	0.926751592
1 Ingeniería Civil Electrónica	0.883116883
2 Ingeniería Civil Informática	0.872727273
3 Ingeniería en Informática, en Sistemas Computacionales y Software	0.838709677
4 Ingeniería Civil	0.828375286
5 Ingeniería Civil Eléctrica	0.823076923
6 Ingeniería Comercial	0.818181818
7 Ingeniería Civil Mecánica	0.804878049
8 Ingeniería Civil Química	0.797979798
9 Ingeniería en Prevención de Riesgos Laborales y Ambientales	0.78125

Descargar tabla completa (CSV)

Figura 8.26: Tabla interactiva presentada dentro de un expander (“Mostrar tabla (primeras filas)”) para no saturar la vista, con la opción de Descargar tabla completa (CSV). Esto permite inspeccionar los datos de manera rápida y exportarlos para su uso en otras herramientas.

Fuente: Elaboración propia.

En conjunto, este ejemplo muestra la integración de los elementos principales del frontend:

campo de consulta, historial de conversación, gráfico con descarga, explicación en viñetas y tabla interactiva con exportación a CSV. De esta forma, la herramienta entrega al usuario un flujo claro y autosuficiente para explorar los datos, manteniendo una interfaz limpia y centrada en las acciones más útiles (expandir tabla, descargar imagen y datos, y alternar la visibilidad de la barra lateral para maximizar el área de trabajo).

8.8.9. Validación y pruebas

Para comprobar la robustez del sistema se realizaron pruebas de validación manual. El procedimiento consistió en:

1. Formular consultas en lenguaje natural al chatbot y observar las tablas generadas.
2. Cargar los mismos conjuntos de datos en Jupyter Notebook y aplicar directamente los filtros con Pandas.
3. Comparar ambos resultados, verificando la consistencia en métricas, conteos y registros devueltos.

Este método permitió identificar posibles inconsistencias en el filtrado y asegurar que el subgrafo de análisis respeta la semántica de las consultas. Los cuadernos de prueba utilizados incluyen:

- `enrollment_program.ipynb`
- `graduate_programs.ipynb`
- `national_employment.ipynb`
- `Benchmarking_programs.ipynb`

```

¿Cuál es la tasa de empleo a dos años para las carreras de Salud de la Pontificia Universidad Católica de Chile en el año 2024?

mask_inst = df["institution_name_UP"].str.contains(norm("Pontificia Universidad Católica de Chile"), na=False)
mask_area = df["area"].astype(str).map(norm).str.contains(norm("Salud"), na=False)
mask_year = df["year"] == 2024

salud_puc_2024 = df.loc[mask_inst & mask_area & mask_year].dropna(subset=["emp_rate_2yr"]).copy()

out = (
    salud_puc_2024.groupby(["year", "program_name"], as_index=False)["emp_rate_2yr"]
    .mean()
    .sort_values("emp_rate_2yr", ascending=False)
)

# Pasamos a porcentaje
out["emp_rate_2yr"] = (out["emp_rate_2yr"] * 100).round(2)

print(out.to_string(index=False))
[ ]
...
year    program_name  emp_rate_2yr
2024    Química y Farmacia      98.62
2024          Enfermería      97.10
2024          Odontología      89.32
2024          Kinesiología      89.16
2024    Fonoaudiología      84.27
2024    Nutrición y Dietética    78.07
2024          Medicina      73.32
    
```

Figura 8.27: Validación manual con Pandas: cálculo de la tasa de empleabilidad a dos años en programas de Salud de la PUC (2024).

Fuente: Elaboración propia.

- La tasa de empleo a dos años para las carreras de Salud en la Pontificia Universidad Católica de Chile en 2024 varía entre aproximadamente 73% y 99%.
- La carrera con la tasa de empleo más alta es Química y Farmacia, con un 98.6%, seguida por Enfermería con un 97.1%.
- Medicina presenta la tasa de empleo más baja dentro de las carreras de salud, con un 73.3%.
- Otras carreras como Odontología (89.3%), Kinesiología (89.2%) y Fonoaudiología (84.3%) muestran tasas de empleo intermedias.
- En general, las carreras de salud en esta universidad presentan una alta inserción laboral a dos años, reflejando una buena empleabilidad en el sector.

Mostrar tabla (primeras filas)

Program Name	Tasa Empleo 2 años
0 Enfermería	0.971014493
1 Fonoaudiología	0.842696629
2 Kinesiología	0.891562605
3 Medicina	0.733133639
4 Nutrición y Dietética	0.780701754
5 Odontología	0.893203883
6 Química y Farmacia	0.986206897

Descargar tabla completa (CSV)

Figura 8.28: Respuesta generada por el chatbot para la misma consulta, mostrando tabla y explicación en lenguaje natural.

Fuente: Elaboración propia.

En la figura 8.27 se aprecia el resultado obtenido al aplicar manualmente la consulta con Pandas. La figura 8.28 muestra la salida del chatbot frente a la misma pregunta en lenguaje natural. Se observa que ambos resultados coinciden en los valores y que el sistema complementa la tabla con una explicación textual, validando la consistencia del filtrado y de la etapa de visualización.

Cabe destacar que este no fue un caso aislado: se realizaron múltiples pruebas con diferentes consultas y datasets, confirmando que el comportamiento se mantiene estable. A modo ilustrativo, este ejemplo se presenta como caso de referencia.

9. Conclusiones

Este trabajo implementó un asistente conversacional para el análisis de datos de educación superior en Chile. La solución integra, de manera coherente, una capa de diálogo con un flujo de selección de datos, filtrado, análisis y visualización; además, expone una API simple para el frontend y gestiona imágenes por sesión de forma ordenada. El resultado es un prototipo capaz de transformar preguntas en lenguaje natural en respuestas útiles y verificables, reduciendo la fricción de uso para perfiles no técnicos.

Los objetivos planteados se cumplieron:

- El sistema identifica la intención del usuario y descarta consultas fuera de dominio, evitando cómputo innecesario.
- Selecciona los conjuntos de datos pertinentes y aplica filtros consistentes (institución, programa, años, entre otros).
- Entrega resultados claros en forma de gráficos y tablas, acompañados de una explicación breve y accionable.
- Mantiene trazabilidad básica de lo mostrado y un desempeño estable en escenarios de uso normal.

Desde el punto de vista de la arquitectura, el diseño modular (enrutamiento de intención, filtrado/análisis y visualización) favoreció la separación de responsabilidades y la extensibilidad. La API del backend estandarizó la interacción con el frontend y permitió un manejo robusto de recursos estáticos por sesión, mejorando la experiencia de usuario y la reproducibilidad. En conjunto, estas decisiones facilitan el mantenimiento, la auditoría y la incorporación de mejoras incrementales sin romper el contrato de interacción.

El prototipo muestra que es posible guiar la conversación hacia respuestas útiles mediante un flujo predecible y de baja complejidad para el usuario. Aunque el sistema depende de la calidad y normalización de las fuentes, y hoy se centra en consultas sobre un único conjunto por turno, su diseño deja abierta la evolución hacia escenarios más complejos sin comprometer la simplicidad de uso.

9.1. Trabajo Futuro

En la siguiente etapa, el principal foco será habilitar cruces entre bases de datos. Para ello se incorporará un nodo especializado en uniones (*joins*) que detecte cuándo la consulta requiere combinar fuentes, proponga la unión adecuada según claves y periodo, y aplique los filtros en un contexto multi-dataset. Con este componente, el asistente podrá responder preguntas comparativas en una sola interacción (por ejemplo, contrastar matrícula con empleabilidad), entregando un resultado unificado listo para análisis y visualización.

Asimismo, se proyecta ampliar el alcance hacia nuevas fuentes de datos. Una línea prioritaria será la conexión directa con bases de datos institucionales o *data warehouses*, lo que permitirá consultar la información en tiempo real y a mayor escala, evitando la necesidad de cargar todos los *dataframes* en memoria. De esta forma, el sistema podrá integrarse a entornos productivos más complejos y responder a un espectro más amplio de consultas con mayor eficiencia y escalabilidad.

En conjunto, estas proyecciones apuntan a transformar el prototipo actual en una herramienta más robusta, escalable y versátil. Su desarrollo futuro no solo ampliará el alcance de las consultas posibles, sino que también sentará las bases para integrar el asistente en procesos de análisis institucional de mayor envergadura, demostrando su potencial como apoyo en la toma de decisiones estratégicas en educación superior.

Referencias

- [1] Mahmoud Abdin and et al. Phi-3 technical report: A highly capable language model. *arXiv preprint*, 2024. URL: <https://arxiv.org/abs/2404.14219>, arXiv:2404.14219.
- [2] Joshua Achiam and et al. Gpt-4 technical report. *arXiv preprint*, 2023. URL: <https://arxiv.org/abs/2303.08774>, arXiv:2303.08774.
- [3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv preprint*, 2020. URL: <https://arxiv.org/abs/2005.14165>, arXiv:2005.14165.
- [4] I-Chun Chern, Steffi Chen, Shiqi Chen, and et al. Factool: A unified framework for factuality evaluation of llms across tasks. *arXiv preprint*, 2023. URL: <https://arxiv.org/abs/2307.13528>, arXiv:2307.13528.
- [5] Peerawat Chomphooyod, Lattapon Jeerapradit, Atiwong Suchato, and Proadpran Punyabukkana. Multi-agentic AI for automatic course syllabus generation using langgraph. In *Proceedings of the 10th International STEM Education Conference (iSTEM-Ed)*, Bangkok, Thailand, 2025. IEEE. doi:10.1109/iSTEM-Ed65612.2025.11129303.
- [6] Fiscalía Nacional Económica (FNE). Minuta de lanzamiento del estudio de mercado sobre educación superior (em09-2024), 2024. URL: https://www.fne.gob.cl/wp-content/uploads/2024/01/Minuta_Lanzamiento_EM09_firmada.pdf.
- [7] Gobierno de Chile. Matrícula en educación superior. Portal Datos Abiertos del Estado, 2024. URL: <https://datos.gob.cl/dataset/matricula-en-educacion-superior>.

REFERENCIAS

- [8] Gobierno de Chile. Titulados en educación superior. Portal Datos Abiertos del Estado, 2024. URL: <https://datos.gob.cl/dataset/titulados-en-educacion-superior>.
- [9] Anthony Grattafiori and et al. The llama 3 herd of models. *arXiv preprint*, 2024. URL: <https://arxiv.org/abs/2407.21783>, arXiv:2407.21783.
- [10] Albert Q. Jiang and et al. Mixtral of experts. *arXiv preprint*, 2024. URL: <https://arxiv.org/abs/2401.04088>, arXiv:2401.04088.
- [11] Georgios Katsogiannis-Meimarakis and Georgia Koutrika. Natural language interfaces for databases with deep learning: A survey. *The VLDB Journal*, 2023. doi:10.1007/s00778-022-00776-8.
- [12] Georgios Katsogiannis-Meimarakis and Georgia Koutrika. A survey on deep learning approaches for text-to-sql systems. *The VLDB Journal*, 2023. doi:10.1007/s00778-022-00776-8.
- [13] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint*, 2021. URL: <https://arxiv.org/abs/2107.13586>, arXiv:2107.13586.
- [14] Paula Maddigan and Teo Susnjak. Chat2vis: Generating data visualisations via natural language. *arXiv preprint*, 2023. URL: <https://arxiv.org/abs/2302.02094>, arXiv:2302.02094.
- [15] Ministerio de Educación de Chile. Educación superior creció 3,3% alcanzando 1,4 millones de estudiantes. Comunicado web del Ministerio de Educación, 2025. URL: <https://www.mineduc.cl/educacion-superior-crecio-33-alcanzando-14-millones-de-estudiantes/>.
- [16] OECD. Education at a glance 2024: Chile — country note, 2024. URL: https://www.oecd.org/content/dam/oecd/en/publications/reports/2024/09/education-at-a-glance-2024-country-notes_532eb29d/chile_9fb8bbc6/c860c0e3-en.pdf.

REFERENCIAS

- [17] Bowen Qin, Binyuan Hui, Lihan Wang, and et al. A survey on text-to-sql parsing: Concepts, methods, and future directions. *arXiv preprint*, 2022. URL: <https://arxiv.org/abs/2208.13629>, arXiv:2208.13629.
- [18] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, and et al. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL: <https://arxiv.org/abs/2302.04761>, arXiv:2302.04761.
- [19] Servicio de Información de Educación Superior (SIES) and Ministerio de Educación de Chile. Metodología de los datos publicados en el buscador de empleabilidad e ingresos 2019–2020, 2019. URL: https://www.mifuturo.cl/wp-content/uploads/2019/12/Metodologia-Buscador-Empleabilidad-e-Ingresos_2019-2020.pdf.
- [20] Leixian Shen, Enya Shen, Yuyu Luo, and et al. Natural language interfaces for data visualization: State of the art. *IEEE Transactions on Visualization and Computer Graphics*, 2022. doi:10.1109/TVCG.2022.3148007.
- [21] Subsecretaría de Educación Superior, Chile. Sies / mifuturo: Matrícula, titulados y empleabilidad (datos e informes), 2024. URL: <https://www.mifuturo.cl>.
- [22] Chenglong Wang, Kedar Tatwawadi, Marc Brockschmidt, and et al. Robust text-to-sql generation with execution-guided decoding. *arXiv preprint*, 2018. URL: <https://arxiv.org/abs/1807.03100>, arXiv:1807.03100.
- [23] Q. Wu, G. Bansal, J. Zhang, et al. Automated data visualization from natural language via large language models. *arXiv preprint*, 2024. URL: <https://arxiv.org/abs/2404.17136>, arXiv:2404.17136.
- [24] Qingyun Wu, Gagan Bansal, Jieyu Zhang, and et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint*, 2023. URL: <https://arxiv.org/abs/2308.08155>, arXiv:2308.08155.
- [25] Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint*, 2024. URL: <https://arxiv.org/abs/2402.13116>, arXiv:2402.13116.

REFERENCIAS

- [26] Subsecretaría de Educación Superior y SIES. Servicio de información de la educación superior (sies) — objetivos, productos y fuentes, 2023. URL: https://www.redindices.org/attachments/article/196/Chile_PPT%20SIES%20GASTO%20%2806112023%29.pdf.
- [27] Shunyu Yao, Jeffrey Zhao, Dian Yu, and et al. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL: <https://arxiv.org/abs/2210.03629>, arXiv:2210.03629.
- [28] Tao Yu, Rui Zhang, Michihiro Yasunaga, and et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018. URL: <https://arxiv.org/abs/1809.08887>.
- [29] Zhengxuan Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. *arXiv preprint*, 2021. URL: <https://arxiv.org/abs/2102.09690>, arXiv:2102.09690.
- [30] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint*, 2017. URL: <https://arxiv.org/abs/1709.00103>, arXiv:1709.00103.