Street Art & Urban Creativity | Vol. 11, No. 7, 2025 | ISSN 2183-9956
Street Art & Urban Creativity. Scientific Journal
https://doi.org/10.62161/sauc.v11.6023

SLM PARA LA INTERACCIÓN DE BASES DE DATOS DE LENGUAJE NATURAL CON EL SENSE CITIVERSE DE LA CIUDAD DE CARTAGENA

PALABRAS CLAVE

SLM
texto a sql
Trino
Dahahub
KServe
KubeFlow
Sense Citiverse
Smart City
Espacios de datos

RESUMEN

Los espacios de datos europeos albergan amplios catálogos, pero rara vez convierten los datos en respuestas oportunas. Presentamos un asistente de datos de lenguaje natural autónomo para SENSE Citiverse de Cartagena. Coordina tres SLM (entidades, SQL y experto en dominios) en una pila DataHub, Trino y KServe/Kubeflow. Se evaluaron dos pilas SLM (Compact y Expanded) en seis tareas de calidad del aire y de operaciones portuarias frente a Gemini 2.5-Pro-LLM. La extracción de entidades logró un descubrimiento perfecto de tablas. SQL resolvió 4/6 (Compact) y 5/6 (Expanded); Gemini resolvió 6/6. El diseño SLM preserva la soberanía de los datos y reduce los costes. Este trabajo demuestra una nueva forma de interactuar en Citiverse basada en el lenguaje natural.

RECIBIDO: 05/09/2025
ACEPTADO: XX/ XX / XXXX

1. Introducción

Los espacios de datos de la Unión Europea (UE) prometen un intercambio seguro y soberano de datos entre organismos públicos y actores privados, pero su principal interfaz con los seres humanos sigue siendo el catálogo de conjuntos de datos. Las administraciones publican los conjuntos de datos como valores separados por comas o JSON y luego los catalogan para su descubrimiento. El personal no técnico puede navegar por esos catálogos, pero a menudo no puede convertir las filas y columnas en respuestas. El resultado es paradójico: los datos están disponibles, pero la información es escasa. Trabajos recientes sobre espacios de datos describen claramente esta brecha y abogan por un acceso centrado en las personas a los activos orientados a las máquinas (Inglés-Romero et al., 2025). Si bien los modelos de lenguaje grandes (LLM) ofrecerían una interfaz de lenguaje natural (NL), los modelos de uso general imponen cargas de coste, privacidad y sostenibilidad que también invitan a una dependencia innecesaria de las API externas. En el caso de los datos de las ciudades, los registros sanitarios o la telemetría industrial, esa dependencia entra en conflicto con los requisitos de soberanía de los datos y complica el cumplimiento normativo.

Cartagena es una de las ciudades piloto de la iniciativa SENSE (SENSE Project, 2025c), que desarrolla «Citiversos», espacios digitales interconectados en los que las ciudades europeas planifican, simulan y cocrean servicios públicos. En este contexto municipal, la calidad del aire es un caso de uso temprano y específico en el que Cartagena ha realizado una gran inversión para desarrollar sus planes de sostenibilidad y su zona de bajas emisiones. Cartagena agrega telemetría medioambiental, como los contaminantes atmosféricos por hora (O_3, NO_2, SO_2 y CO). A continuación, los técnicos deben responder a preguntas sencillas pero urgentes: ¿Tenemos valores de O_3 por encima de un umbral en una fecha determinada? ¿Son estables los valores de CO durante el turno de noche?

Un asistente conversacional de texto a SQL cubriría esa laguna. Permite al personal utilizar el lenguaje natural, luego descubre el conjunto de datos correcto, compone el SQL con las comillas adecuadas para los espacios y los diacríticos, lo valida, lo ejecuta y luego explica el resultado.

Para satisfacer esta demanda y evitar al mismo tiempo los inconvenientes inherentes a los LLM, este artículo propone un asistente de datos autónomo basado en modelos de lenguaje pequeños (SLM), que son LLM más pequeños que caben en los dispositivos de consumo. El asistente de datos acepta preguntas en lenguaje natural, descubre los conjuntos de datos pertinentes, compone y ejecuta el SQL y, a continuación, explica los resultados. Se utilizan tres SLM especializados: un SLM de entidades para la extracción de términos, un SLM de SQL para la generación de consultas y un SLM de expertos en el dominio para explicaciones en lenguaje natural, todo ello impulsado por una infraestructura de código abierto de extremo a extremo. DataHub proporciona información de tablas y metadatos para el descubrimiento y el contexto, Trino sirve como motor SQL federado entre archivos y bases de datos, y KServe ejecuta los SLM en un clúster de Kubernetes, proporcionando puntos finales de modelo seguros, observables y autoescalables. Cada componente está bien establecido, se mantiene activamente y es compatible con la implementación local. Los datos nunca salen del entorno controlado.

Hemos preferido los SLM a los LLM monolíticos por cuatro razones: en primer lugar, los SLM cumplen las restricciones de latencia de las herramientas interactivas. Responden en cientos de milisegundos o en unos pocos segundos en una sola GPU o incluso en una CPU potente. En segundo lugar, los SLM reducen los costes operativos y el consumo de energía, lo cual es importante cuando el sistema debe prestar servicio a muchos usuarios o funcionar de forma continua (Belcak et al., 2025). En tercer lugar, los modelos especializados son más fáciles de depurar porque cada uno tiene un contrato limitado: extraer entidades, escribir SQL o explicar resultados. En cuarto lugar, el diseño modular admite una degradación elegante: un único modelo generalista puede desempeñar múltiples funciones en una máquina pequeña, mientras que un clúster más grande puede escalar horizontalmente y asignar un experto diferente a cada función. En el caso de

Cartagena, una menor demanda computacional se traduce directamente en la capacidad de alojar los modelos junto con los datos de la ciudad.

El uso de componentes de código abierto permite no tener dependencias externas y una implementación independiente. DataHub ofrece un servicio de metadatos basado en grafos con puntos finales GraphQL y REST y canalizaciones de ingestión flexibles. Indexa conjuntos de datos, esquemas y linajes, y admite actualizaciones casi en tiempo real para la búsqueda y el descubrimiento. Estas características proporcionan el contexto que los sistemas de texto a SQL necesitan para la vinculación de esquemas y la recuperación de ejemplos. Trino expone una interfaz SQL unificada para archivos con diferentes formatos, bases de datos relacionales y almacenes de objetos a través de un rico ecosistema de conectores. También puede validar consultas con EXPLAIN, lo que respalda la capacidad del asistente para la depuración de consultas. KServe, del ecosistema KubeFlow, permite implementar los SLM como definiciones de recursos personalizados de Kubernetes. Abstrae la estructura del servidor de modelos, la gestión del tráfico y el autoescalado, y admite la densidad multimodelo a través de ModelMesh.

En conjunto, estas herramientas hacen que el asistente sea portátil e independiente. ¿Por qué insistir en la autosuficiencia? La soberanía y la ecología lo exigen. Los proyectos europeos abordan la soberanía de los datos como una restricción de diseño en los espacios de datos. Exigen que el procesamiento se realice bajo control local, con autorización y trazabilidad completas (Inglés-Romero et al., 2025). Un asistente autohospedado respeta esa restricción porque tanto los modelos como los datos permanecen en la empresa. También reduce el consumo de energía al evitar la pesada inferencia remota.

El enfoque propuesto tiene como objetivo un asistente que puedan utilizar personas no expertas sin necesidad de llamar a un ingeniero o a un experto en la materia. Los usuarios escriben frases sencillas en lenguaje natural (NL) y el sistema genera los resultados de una consulta SQL, así como una respuesta en NL a la pregunta utilizando la salida SQL. Este bucle refleja los diseños agenciales que renuncian a la magia de un solo uso en favor de una descomposición fiable y un refinamiento iterativo. La propuesta de valor es clara para los municipios y las empresas. El asistente reduce las barreras al uso de datos, preserva la privacidad y evita la dependencia de un único proveedor. Puede ejecutarse en servidores estándar y aprovecha el software abierto. Sus partes modulares son reemplazables y pueden actualizarse de forma independiente en función de las características específicas de cada caso de uso, lo que puede ser más importante que las ganancias marginales de los modelos más grandes.

1.1. Objetivos

Este documento propone y detalla un asistente de base de datos modular e independiente para el SENSE Citiverse de Cartagena que hace hincapié en las demandas de los espacios de datos europeos mediante el uso de SLM y herramientas de código abierto. El asistente funciona mediante un proceso de varios pasos en el que:

(1) extrae entidades del lenguaje natural; (2) utiliza DataHub para vincular esas entidades a tablas y columnas específicas; (3) solicita a un especialista en SQL que redacte y repare consultas; (4) valida y ejecuta en Trino; y devuelve una explicación clara del dominio.

La arquitectura, basada en DataHub, Trino, KubeFlow y KServe, se ha diseñado teniendo en cuenta los ecosistemas de código abierto SENSE y EU Local Digital Twin Toolbox, con el fin de garantizar que estas soluciones puedan adoptarse en las plataformas abiertas existentes ampliamente adoptadas por el ecosistema de ciudades inteligentes.

El resto del artículo se organiza de la siguiente manera: la sección 2 aborda los antecedentes y el estado actual de la técnica (SoTA); la sección 3 describe la arquitectura y el proceso; la sección 4 incluye una evaluación práctica del sistema con una configuración específica para el caso de uso de las lecturas de la calidad del aire en Cartagena, y la sección 5 concluye el artículo y aborda futuras investigaciones.

2. Antecedentes

2.1. Text-to-sql: conectando a los humanos con las máquinas

Encuestas recientes recogen tres oleadas en text-to-sql: sistemas basados en reglas y en secuencia a secuencia; modelos de lenguaje preentrenados; y, desde 2023, LLM que siguen instrucciones (Hong et al., 2024; Shi et al., 2025). Los LLM destacan por su capacidad de aprender con pocos ejemplos y por un análisis semántico robusto. Hay dos paradigmas de implementación dominantes: la ingeniería de prompts mediante el aprendizaje en contexto y el ajuste de modelos (Hong et al., 2024; Shi et al., 2025). Desde el punto de vista metodológico, los sistemas modernos descomponen la tarea. El preprocesamiento formatea los esquemas con plantillas estructuradas («CREATE TABLE...») y filas de muestra; la vinculación de esquemas reduce las tablas y columnas candidatas antes de la generación (Shi et al., 2025). La inferencia utiliza cadenas de pensamiento y flujos de trabajo de varios pasos que escriben SQL cláusula por cláusula o dividen la indicación en subproblemas. El posprocesamiento introduce la retroalimentación de la base de datos: autodepuración con registros de errores, selección guiada por la ejecución y comprobaciones de coherencia entre modelos (Hong et al., 2024; Shi et al., 2025).

Los estudios de casos empresariales refuerzan esta visión del proceso. Los investigadores de LinkedIn propusieron un sistema multiagente con recuperación de contexto, un agente Query-Writer y un agente Researcher que corrige las alucinaciones de tablas y columnas (Chen et al., 2025). A pesar de los avances, siguen existiendo lagunas. Incluso con indicaciones y agentes avanzados, la precisión de la ejecución puede ser baja en los benchmarks de texto a sql SoTA como Spider 2.0 (Shi et al., 2025), con problemas como esquemas de contexto largo que superan los presupuestos de tokens, lo que reduce la precisión y aumenta la latencia y el coste.

2.2. DataHub: plataforma de datos y gobernanza de datos

DataHub es una plataforma de metadatos de código abierto que unifica el descubrimiento, la gobernanza y la observabilidad de los datos y los activos de IA. Expone varias API, incluida una interfaz GraphQL pública, y ofrece puntos finales OpenAPI para casos de uso comunes. Su servicio de metadatos, también llamado GMS, mantiene el gráfico de entidades y relaciones y atiende las consultas y mutaciones de los clientes. El proyecto hace hincapié en un «modelo de metadatos vivo» que evoluciona mediante herramientas, paneles de control, conjuntos de datos, modelos y ejecuciones de entrenamiento (DataHub Project, 2015a; DataHub Project, 2015b; DataHub Project, 2024; DataHub Project, 2025).

Una arquitectura de ingestión flexible admite los modos push y pull y puede transmitir los cambios para una indexación casi en tiempo real. La implementación de inicio rápido ilustra las partes móviles de la plataforma, incluyendo Kafka, Elasticsearch u OpenSearch, y la interfaz web. El perfil predeterminado se ejecuta localmente, lo que se adapta al desarrollo y a pequeños proyectos piloto. Las implementaciones de producción utilizan los mismos componentes a mayor escala y a menudo adoptan gráficos Helm para la orquestación (DataHub Project, 2015b).

Como se muestra en otros trabajos (Chen et al., 2025), para un asistente de texto a SQL, DataHub puede desempeñar dos funciones. En primer lugar, cataloga conjuntos de datos con atributos enriquecidos, etiquetas, propietarios y linaje, de modo que la extracción de entidades pueda resolverse en tablas y columnas específicas. En segundo lugar, proporciona los esquemas y las claves que un LLM necesita para generar uniones y filtros correctos.

2.3. Trino: motor SQL distribuido

Trino es un motor de consultas SQL distribuido diseñado para el análisis interactivo de datos almacenados en sistemas heterogéneos. Coordina las consultas entre muchos trabajadores, envía los cálculos a los datos a través de conectores y devuelve los resultados con baja latencia. Trino habla ANSI SQL, se integra con herramientas de inteligencia empresarial y opera a escalas que van desde gigabytes hasta exabytes (Trino, 2025a).

El marco de conectores es fundamental para el valor de Trino. Los conectores implementan una interfaz de metadatos que enumera esquemas, tablas y columnas, y una interfaz dividida que transmite datos a los trabajadores. Los conectores oficiales cubren sistemas comunes como Hive, Iceberg, MySQL, PostgreSQL, Kafka y Elasticsearch, y las organizaciones pueden desarrollar conectores personalizados para acceder a almacenes internos (Trino, 2025b).

Trino admite la validación de consultas y la introspección a través de EXPLAIN y EXPLAIN ANALYZE. EXPLAIN imprime el plan distribuido y valida la instrucción sin ejecutarla. EXPLAIN ANALYZE ejecuta la consulta e informa de los costes por operador, como filas, CPU, memoria y red (Trino, 2020; Trino, 2025c).

2.4. KServe: implementación de modelos de IA en KubeFlow

KServe es un sistema nativo de Kubernetes para servir modelos de aprendizaje automático. Comenzó como KFServing dentro del proyecto Kubeflow y más tarde se convirtió en una iniciativa independiente de código abierto. KServe define un recurso personalizado, InferenceService, que describe cómo implementar un servidor de modelos, lanzar nuevas revisiones y enrutar el tráfico. Abstrae el autoescalado, las redes, las comprobaciones de estado y la observabilidad, y estandariza los protocolos de inferencia para el tráfico REST o gRPC (KServe Project, 2025c; KServe Project, 2025d; KServe Project, 2025e).

KServe se integra con el ecosistema más amplio de Kubernetes. Kubernetes gestiona pods, servicios y escalado horizontal para aplicaciones en contenedores, y proporciona una configuración declarativa para operaciones fiables (Kubernetes, 2024). Además, Kubeflow proporciona componentes adicionales para los flujos de trabajo de aprendizaje automático en Kubernetes, incluyendo canalizaciones, cuadernos, un operador de formación y un panel de control central. KServe encaja en ese ecosistema como la capa de servicio para los modelos entrenados (Kubeflow, 2025).

Hay dos características de KServe que resultan especialmente relevantes. En primer lugar, la plataforma ofrece flexibilidad en sus protocolos API. No solo define una API de inferencia independiente del marco, a menudo denominada «protocolo v2», que estandariza los formatos de solicitud y respuesta, sino que también es compatible directamente con el formato API de completaciones de OpenAI (KServe Project, 2025b; KServe Project, 2025c). En segundo lugar, ModelMesh admite un servicio multimodelo de alta densidad. Carga y descarga modelos bajo demanda y enruta las solicitudes con una sobrecarga mínima. Estas características permiten que un único clúster aloje varios puntos finales SLM sin perder capacidad de respuesta ni eficiencia de recursos (KServe Project, 2024).

El plano de control de Kserve gestiona las revisiones de los modelos, el despliegue canario y las pruebas A/B. Los operadores pueden configurar divisiones de tráfico basadas en porcentajes para comparar las versiones de los modelos bajo carga. Dado que los modelos son recursos de Kubernetes, heredan las capacidades de seguridad, supervisión y auditoría a nivel de clúster. Esta adecuación es importante para las implementaciones soberanas, en las que todas las rutas de solicitud y los artefactos deben ser trazables (KServe Project, 2025a).

2.5. Modelos de lenguaje pequeños (SLM)

Los argumentos a favor de los SLM se basan en tres afirmaciones. En primer lugar, las tareas especializadas no requieren la generalidad ni el tamaño completos de los modelos de vanguardia. En segundo lugar, las realidades operativas hacen que los modelos pequeños sean más rápidos de implementar, supervisar y actualizar. En tercer lugar, tanto la economía como la ecología favorecen la reducción del número de parámetros. Tal y como proponen Belcak et al. (2025): un SLM se adapta al hardware común de los consumidores y admite interacciones agenciales con baja latencia.

El diseño agencial también aprovecha las ventajas de los SLM. En nuestro caso, cuando el sistema aísla la extracción de entidades, la vinculación de esquemas, la generación de código y la

explicación, cada subtarea se beneficia de un modelo que puede ajustarse con precisión en un corpus más reducido y entrenarse posteriormente para formatos de salida estrictos. Estos modelos pueden incorporar barreras de protección y decodificación determinista para tablas, JSON o SQL. También exponen los fallos de forma clara, lo que reduce el coste de la depuración y el reentrenamiento (Belcak et al., 2025). El asistente descrito en este artículo sigue esa estrategia. Combina modelos ligeros con herramientas potentes —DataHub para el contexto y Trino para la ejecución—, lo que reduce la carga cognitiva de los modelos y evita tokens innecesarios.

Desde el punto de vista de la gobernanza, los SLM mejoran la accesibilidad y la equidad. Su entrenamiento y funcionamiento cuestan mucho menos que los LLM de vanguardia, lo que hace que el autoalojamiento sea viable para pequeñas organizaciones y municipios. Esa democratización es importante en regiones con presupuestos limitados y en proyectos en los que la infraestructura se financia con fondos públicos. También es importante para la sostenibilidad. Los modelos más pequeños también consumen menos energía por solicitud, especialmente cuando se cuantifican (Ukil et al., 2025).

2.6. Procesamiento del lenguaje natural en las administraciones públicas

El procesamiento del lenguaje natural (NLP) es la herramienta de IA generativa más utilizada en la gobernanza local. En sus diferentes formas, ya aporta valor a las administraciones públicas de todo el mundo en áreas como la gobernanza de datos, donde el NLP simplifica la gestión y el procesamiento de datos heterogéneos (Jiang et al., 2023; Yigitcanlar et al., 2024).

Los municipios utilizan principalmente soluciones basadas en el PLN para la gestión de la información y de tareas administrativas, como la automatización de funciones administrativas rutinarias y la gestión de las consultas de los ciudadanos, mejorando la accesibilidad de los servicios públicos mediante chatbots y asistentes virtuales (Yigitcanlar et al., 2024).

Y, aunque el PLN puede mejorar la transparencia al facilitar el acceso a los datos abiertos del gobierno, sigue existiendo una laguna: los asistentes existentes a nivel de la administración pública suelen limitarse a recuperar documentos o enumerar los conjuntos de datos disponibles. No permiten a los usuarios sin conocimientos técnicos consultar por sí mismos el contenido de los datos únicamente a través del lenguaje natural, ya que este no puede transformarse eficazmente, por ejemplo, en consultas SQL programáticas para recuperar los datos pertinentes (Cortés-Cediel et al., 2023).

2.7 SENSE Citiverse y Cartagena

El proyecto SENSE (*Fortalecimiento de las ciudades y mejora del sentido de pertenencia de los barrios*) crea una red de gemelos digitales interconectados que reflejan ciudades reales y se alinean con la iniciativa de Comunidades Inteligentes de la UE (SENSE Project, 2025c). SENSE sigue una arquitectura por capas que comienza con la gestión de dispositivos (IoT y periféricos), pasa por la gestión de datos (plataforma e integración), añade un espacio de datos para el intercambio soberano, habilita un gemelo digital para la simulación y la IA, y culmina en Citiverse para la interacción 3D/AR/VR. El proyecto SENSE describe CitiVerse como un espacio digital compartido en el que las ciudades europeas utilizan herramientas comunes y datos compartidos, especialmente gemelos digitales, para planificar, probar ideas e involucrar a los ciudadanos en la toma de decisiones (SENSE Project, 2025a; SENSE Project, 2025b). Cartagena es una de las dos ciudades piloto (SENSE Project, 2025a; SENSE Project, 2025b), junto con Kiel, Alemania, que, al igual que Cartagena, también es famosa por su puerto. El proyecto piloto de Cartagena se basa explícitamente en datos en tiempo real y pilas interoperables dentro de un espacio de datos soberano y un contexto de gemelos digitales, por lo que una interfaz de base de datos en lenguaje natural podría suponer un gran avance.

3. Propuesta

La figura 1 muestra el sistema autónomo completo. Un orquestador coordina tres funciones LLM: entidades, SQL y experto en el dominio; y dos servicios de datos: DataHub para el descubrimiento de metadatos y Trino para la ejecución federada de SQL. Todos los componentes se ejecutan en clúster en Kubernetes y KServe expone cada función LLM impulsada por un SLM como un InferenceService con autoescalado, métricas y control de revisiones. Esta opción de implementación encarna los principios operativos del proyecto: modularidad (cada capacidad es un servicio reemplazable), soberanía (no se requieren API externas) y robustez (cada paso es observable, validable y recuperable).

El sistema adopta la modularidad sin sacrificar la simplicidad. Con un espacio mínimo, un único SLM ajustado a las instrucciones puede asumir las tres funciones mediante indicaciones específicas para cada función. En un espacio escalado, el Orchestrator vincula cada función a un modelo distinto: un extractor compacto para entidades, un modelo ajustado al código para SQL y un modelo orientado al diálogo para explicaciones. KServe dirige el tráfico hacia estos puntos finales y los escala de forma independiente.

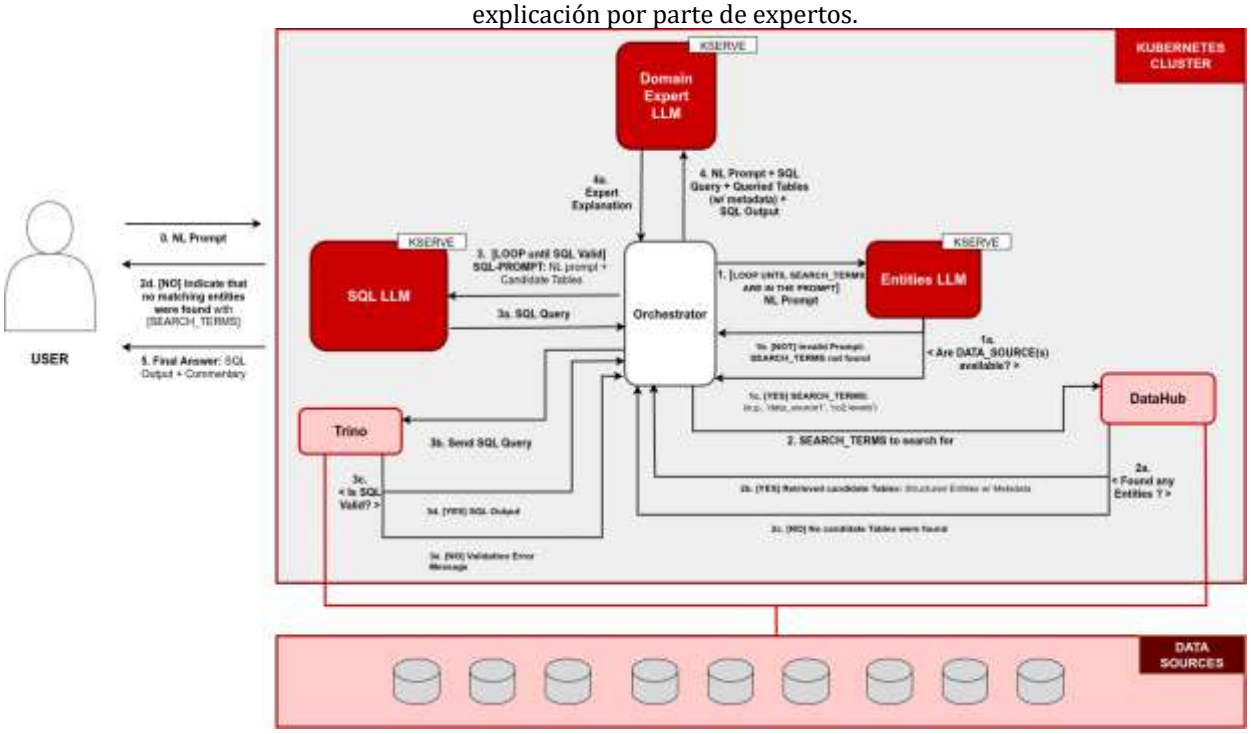
Esta composición permite dar soporte a diferentes casos de uso, ya que algunos escenarios solo necesitan manejar unas pocas tablas con columnas limitadas, en las que basta con un LLM más general, ya que todo el contexto necesario se puede añadir en el prompt, lo que ayuda a que funcione mejor y a que las consultas complejas no sean la norma. Sin embargo, la arquitectura también permite admitir escenarios más complejos en los que no es posible pasar todo el contexto, por lo que el LLM de entidades debe encontrar todos los términos de búsqueda posibles a partir de un conjunto más amplio, el LLM de SQL debe ser más preciso y se necesita un experto en el dominio para revisar las indicaciones y los resultados más complejos. El proceso sigue los siguientes pasos, tal y como se muestra en la figura 1:

1. Recibir y extraer entidades. Un usuario envía una indicación en lenguaje natural. El Orchestrator invoca al LLM de entidades, que emite términos de búsqueda estructurados: fuentes de datos, ubicaciones, métricas, ventanas de tiempo, identificadores de dispositivos. Su contrato está limitado por el diseño: nombrar cosas, no inferir respuestas.
2. Descubrir el contexto. Utilizando esos términos, el Orchestrator consulta DataHub para recuperar tablas candidatas y metadatos, como descripciones de columnas, etc. Si no se encuentran tablas candidatas, el sistema informa al usuario de que su solicitud no es válida.
3. Redactar una indicación específica. El Orchestrator genera una indicación estructurada para el LLM de SQL que incluye la pregunta del usuario textualmente y las tablas candidatas.
    a. Generar y validar SQL. El SQL LLM devuelve una consulta candidata y el Orchestrator la valida en Trino utilizando *EXPLAIN*. Si Trino informa de un error (columna desconocida, conversión no válida o función incorrecta), el Orchestrator devuelve el mensaje al SQL LLM con una directiva de reparación compacta y vuelve a validarlo.
    b. Ejecutar y recopilar resultados. Cuando se compila la consulta, el Orchestrator la ejecuta en Trino.
4. Explicar y devolver. Orchestrator llama al SLM experto en el dominio con la pregunta original, el SQL validado y el resultado obtenido. El modelo produce una explicación concisa y adecuada a la función que indica las suposiciones, las unidades y la cobertura, y luego Orchestrator devuelve al usuario tanto el resultado SQL como la explicación.

La eficiencia ecológica es consecuencia de la especialización y el intercambio. El sistema prefiere SLM pequeños y cuantificados que satisfagan las necesidades de formato y de latencia sin exceso de capacidad. ModelMesh puede mejorar la utilización al cargar modelos solo cuando llega tráfico. Dado que la semántica reside en DataHub y los cálculos pesados se ejecutan en Trino, los

modelos siguen siendo pequeños y sin estado, lo que reduce el consumo de memoria y energía, al tiempo que se conserva la capacidad para las tareas de destino. El resultado es un asistente de bajo consumo que responde rápidamente en hardware modesto a través de fuentes de datos heterogéneas, gracias a Trino y DataHub.

Figura 1. Arquitectura del sistema y canalización para el descubrimiento de datos, la consulta y la explicación por parte de expertos.



Fuente. Elaboración propia, 2025.

Por último, la arquitectura se escala y se adapta. Como se ha indicado, el Orchestrator no tiene estado y puede ejecutar múltiples réplicas detrás de un servicio, por lo que el rendimiento se escala linealmente. El autoescalado de KServe aumenta o reduce los pods de modelos según la demanda. Dado que el Orchestrator filtra las tablas a las pocas que importan, el recuento de tokens se mantiene bajo y la latencia permanece estable a medida que crecen los catálogos. El enfoque en la modularidad también implica que las mejoras en la arquitectura podrían aprovecharse en distintos casos de uso. Por ejemplo, se podría añadir fácilmente un actor LLM traductor para facilitar la interacción con el sistema a los usuarios que no hablan inglés, y también se incluiría un modelo predictivo para predecir datos futuros a partir de los datos actuales, pero más adelante se hablará más sobre los nuevos módulos.

3.1. Aplicación front-end: SENSE NL2SQL

El valor del asistente depende de la rapidez con la que el usuario pueda convertir una pregunta en una respuesta defendible. La interfaz SENSE NL2SQL (Figura 2) pone en funcionamiento el proceso de la Figura 1 mostrando cada paso (detección de entidades, composición SQL, ejecución y explicación) en una única vista comprensible. La interfaz privilegia la claridad sobre la configurabilidad, de modo que los técnicos puedan permanecer en el contexto de su tarea y repetirla rápidamente. Mantiene el cálculo en el clúster y solo muestra la información mínima necesaria para la toma de decisiones, lo cual se ajusta a los objetivos de soberanía y observabilidad de la arquitectura.

3.1.1. Diseño y flujo principal

La interfaz centra el cuadro de consulta de lenguaje natural (NL) y lo combina con dos acciones explícitas: *Sugerir* y *Ejecutar solicitud*. Un chip de estado confirma la conexión con el motor de ejecución («Conectado a Trino • Activo»). A la izquierda, *las solicitudes recientes* y *las acciones rápidas* facilitan la recuperación y los traspasos. El usuario puede volver fácilmente a una solicitud anterior y a su resultado. Esta disposición reduce la carga cognitiva: los usuarios escriben una pregunta sencilla, ven el estado del sistema de un vistazo y vuelven a ejecutar análisis comunes sin tener que navegar fuera de la página. A continuación, la pantalla despliega el análisis en cuatro paneles: *SQL generado, entidades detectadas, estado de la consulta* y *contexto del esquema.* Juntos hacen visible y auditable el razonamiento del modelo.

3.1.2. Asignación transparente de NL→SQL.

El panel *SQL generado* muestra la instrucción exacta que se va a ejecutar. La lista *de entidades detectadas* refleja el resultado del modelo de entidades («aire», «calidad», «datos», «julio»), lo que ayuda a los usuarios a verificar que el sistema ha basado su intención en el catálogo. El bloque *Contexto del esquema* resume la tabla y las columnas clave para que un usuario no experto pueda comprobar las unidades y los significados antes de confiar en un resultado. *El estado de la consulta* informa de la latencia y el resultado (por ejemplo, «Consulta ejecutada correctamente • 116 ms»), lo que establece las expectativas sobre la interactividad y los modos de fallo.

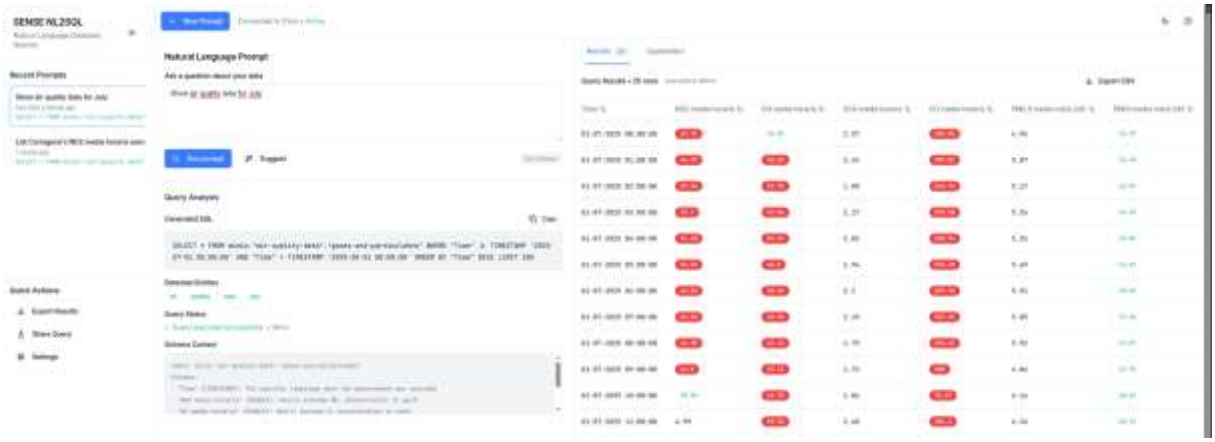
3.1.3. Resultados legibles

La región de resultados presenta dos pestañas: Resultados y Explicación. Resultados muestra una cuadrícula ordenable con el resultado SQL. En este caso, mediciones con marca de tiempo y nombres de columnas en español con diacríticos, que coinciden con el catálogo de origen. Esta fidelidad tranquiliza a los usuarios que comprueban habitualmente los números con los paneles de control operativos. Explicación alberga el resultado del experto en el dominio para resúmenes en lenguaje sencillo, respondiendo a la pregunta del usuario. A continuación, Exportar CSV permite exportar fácilmente la salida SQL a un archivo CSV para su posterior procesamiento. La opción Compartir consulta promueve la reproducibilidad al difundir la pregunta original y su SQL compilado.

3.1.4. Divulgación progresiva y confianza

La interfaz practica la divulgación progresiva. Muestra primero la respuesta y revela el razonamiento justo debajo. Los usuarios que solo necesitan una comprobación rápida pueden detenerse en la tabla y en la respuesta. Los usuarios que deben auditar pueden inspeccionar las entidades detectadas, el SQL y el contexto del esquema. Esto refleja el flujo de trabajo de los agentes de la figura 1, al tiempo que mantiene la superficie reducida. El resultado es una herramienta que permite tanto una rápida percepción de la situación como la elaboración de informes defendibles.

Figura 2. Interacción completa para una solicitud sencilla. La interfaz de usuario muestra la consulta SQL, las entidades detectadas, una insignia de tiempo de ejecución, así como el esquema y el contexto de la tabla. La vista previa del resultado se encuentra a la izquierda y se puede alternar junto a la ventana de explicación.



Fuente. Elaboración propia, 2025.

4. Evaluación

Esta sección especifica la evaluación integral de la arquitectura y del proceso propuestos para impulsar el asistente implementado en el contexto de los casos de uso dentro del SENSE Citiverse de Cartagena. El protocolo mide las tres capacidades alineadas con el proceso explicado en la Figura 1: (1) resolución de entidades frente a DataHub (descubrimiento de tablas y metadatos), (2) generación de SQL frente a Trino (ejecución y validación), y (3) explicación consciente del dominio (interpretación del lenguaje natural) para dos configuraciones del modelo SLM y un LLM SoTA de referencia, Gemini 2.5 PRO. Estas capacidades se evalúan en función de la recuperación de las tablas correctas, la corrección y la optimización de la consulta SQL obtenida para cada tarea, y la calidad de la respuesta.

4.1. Banco de pruebas

Tabla 1. Configuración del hardware

Recursos	Dispositivo
GPU	1× NVIDIA L40s-1-gpu (nube)
VRAM de la GPU	45 GiB de RAM
CPU	13 vCores
RAM de la CPU	80 GiB

Fuente. Elaboración propia, 2025

Tabla 2. Configuración de modelos para la pila Compact-SLM

Función	ID del modelo HuggingFace
Entidades LLM	google/gemma-2-2b-it (Google, 2025a)
SQL LLM	Qwen/Qwen2.5-Coder-3B-Instruct Equipo Qwen. (2025a)
Domain Expert LLM	meta-llama/Llama-3.2-3B-Instruct Meta AI. (2024a)

Fuente. Elaboración propia, 2025.

Tabla 3. Configuración de modelos para la pila SLM ampliada

Función	HuggingFace Id. del modelo
Entidades LLM	google/gemma-3-4b-it (Google, 2025b)
SQL LLM	Qwen/Qwen2.5-Coder-7B-Instruct Equipo Qwen. (2025b)
Domain Expert LLM	meta-llama/Llama-3.1-8B-Instruct Meta AI. (2024b)

Fuente. Elaboración propia, 2025.

Debido a la falta de disponibilidad de hardware de consumo en este momento, las pruebas comparativas se han realizado en una GPU NVIDIA L40. En cuanto al SLM, hemos optado por una configuración de modelo dual para resaltar mejor las ventajas e inconvenientes de cada modelo. En primer lugar, tenemos la pila Compact-SLM (≤ 3000 millones de parámetros), que prioriza una menor latencia y una menor huella de memoria para cumplir con las restricciones interactivas y el hardware de consumo de gama baja, así como la compatibilidad con la computación periférica. Por otro lado, la pila SLM ampliada (≤ 8.000 millones de parámetros) da prioridad a la ventana de contexto adicional que ofrecen los LLM más grandes para permitir una composición SQL más robusta mediante indicaciones más grandes y estructuradas, lo que tiene como contrapartida requisitos de hardware más pronunciados. Comparamos ambas pilas con Gemini 2.5 Pro, utilizado como un único modelo generalista que desempeña todas las funciones (entidades, SQL, experto en el dominio). Las indicaciones serán las mismas para el LLM de entidades y el LLM de experto en el dominio, pero no para el LLM de SQL, ya que este permite indicaciones más grandes con más información relevante para SQL y ejemplos genéricos.

4.2. Descripción del caso de uso

Esta evaluación utiliza dos tablas con una mezcla de datos reales y sintéticos: calidad del aire por hora y llegadas de cruceros (tablas 4-5). La tabla de calidad del aire contiene un registro por hora y conserva los nombres de las columnas en español, con espacios y signos diacríticos, mientras que las llegadas de cruceros tienen nombres de columnas en inglés. Ambas tablas tienen descripciones accesibles a través de Datahub, en inglés. Los identificadores, especialmente en español, enfatizan deliberadamente la vinculación de esquemas mediante DataHub y la citación estricta de identificadores en SQL. Los contaminantes son promedios por hora, aunque PM2,5/PM10 son promedios móviles de 24 horas. La tabla de llegadas codifica los intervalos de atraque, así como la información de los cruceros con marcas de tiempo de inicio y fin, además de los atributos de capacidad. Juntos, ambos conjuntos de datos crean un entorno compacto pero realista para la resolución de entidades, la generación de SQL y la interpretación de dominios.

Tabla 4. Tabla de calidad del aire de Cartagena: minio. «air-quality-data». «gases-and-particulates»

Columna	Descripción	Tipo de datos SQL
«Time»	La marca de tiempo específica (fecha y hora) en la que se registró la medición de la calidad del aire.	TIMESTAMP
«NO2 media horaria»	Concentración media horaria de dióxido de nitrógeno (NO_2), un indicador clave de la contaminación provocada por el tráfico y la combustión. Se mide en $\mu g/m^3$.	DOBLE
«O3 media horaria»	Concentración media horaria de ozono troposférico (O_3), un contaminante secundario que se forma a partir de otras emisiones en presencia de la luz solar. Se mide en $\mu g/m^3$.	DOBLE
«SO2 media horaria»	Concentración media horaria de dióxido de azufre (SO_2), emitido principalmente por la quema de combustibles fósiles. Se mide en $\mu g/m^3$.	DOBLE

«CO media horaria»	Concentración media horaria de monóxido de carbono (CO), un gas resultante de la combustión incompleta de combustibles que contienen carbono. Se mide en mg/m^3.	DOBLE
«PM2,5 media móvil 24 h»	Concentración media móvil en 24 horas de partículas finas con un diámetro de 2,5 micrómetros o menos. Estas partículas pueden penetrar profundamente en los pulmones. Se mide en $\mu g/m^3$.	DOBLE
«PM10 media móvil 24h»	La concentración media móvil en 24 horas de partículas inhalables con un diámetro de 10 micrómetros o menos. Se mide en $\mu g/m^3$.	DOBLE

Fuente. Elaboración propia, 2025.

Tabla 5. Tabla de llegadas de cruceros a Cartagena: minio. «cruiseship_data». «arrivals_july»

Columna	Descripción	Tipo de datos SQL
«arrival_id»	Identificador numérico único asignado a cada llegada individual de un crucero.	INTEGER
«cruise_ship _name»	El nombre oficial y registrado del crucero.	VARCHAR
«ship_size_ category»	Clasificación del barco en función de su tamaño o capacidad de pasajeros (por ejemplo, pequeño, mediano, grande).	VARCHAR
«passenger_ capacity»	El número máximo de pasajeros que el crucero está certificado para transportar.	INTEGER
«crew_ a bordo»	El número total de tripulantes que trabajan en el barco para ese viaje específico.	ENTERO
«total_ a bordo»	El número total de personas a bordo del barco, que representa la suma de pasajeros y tripulación.	ENTERO
«arrival_dat etime»	La fecha y hora exactas en que el barco atracó oficialmente en el puerto.	TIMESTAMP
«departure_ datetime»	La fecha y hora exactas en que el barco zarpó del puerto.	TIMESTAMP
«berth_dura tion_hours»	El tiempo total, medido en horas, que el barco permaneció atracado en su amarre.	DOUBLE

Fuente. Elaboración propia, 2025.

Seis casos de uso ejercitan progresivamente al asistente con datos sintéticamente ampliados de Cartagena de julio de 2025: consultas sencillas y complejas sobre la calidad del aire, sobre la llegada de cruceros y sobre dos uniones temporales. La secuencia valida la resolución de entidades, la generación de SQL compatible con Trino y las explicaciones de dominio, desde búsquedas de umbrales hasta la detección de horas consecutivas alineadas con las ventanas de atraque.

4.2.1. Caso de uso 1 (UC-1): calidad del aire

- Solicitud NL: «¿Tenemos alguna lectura horaria de ozono superior o igual a 60 el 01-07-2025 y a qué horas?».

UC-1 confirma la recuperación básica: una consulta de ozono (O_3) con umbral en un solo día. Valida el descubrimiento de tablas con metadatos, la conversión de marcas de tiempo y las ventanas semiabiertas *sargables* SQL en «tiempo».

4.2.2. Caso de uso 2 (UC-2): calidad del aire

- Solicitud NL: «Para el 02-07-2025, enumere las horas en las que los valores de SO_2 y NO_2 son superiores a la media diaria».

UC-2 aumenta la complejidad al calcular las líneas de base del mismo día (medias diarias) y filtrar las horas en las que el SO_2 y el NO_2 superan esas medias. Comprueba la agregación, la proyección de deltas y si se aplica o no el orden por hora.

4.2.3. Caso de uso 3 (UC-3): llegadas de cruceros

- Solicitud NL: «Muestre todas las llegadas y salidas de cruceros programadas para el 06-07-2025, con el nombre del barco y las horas de estancia en el muelle».

UC-3 verifica el calendario de llegadas en una sola fecha con una proyección y un orden mínimos para comprobar que la segunda tabla también se recupera y procesa correctamente.

4.2.4. Caso de uso 4 (UC-4): llegadas de cruceros

- Solicitud de NL: «Entre el 20-07-2025 a las 14:30 y el 22-07-2025 a las 12:00, enumere los tres cruceros atracados más importantes por número total de pasajeros a bordo, con las horas de llegada y salida».

UC-4 amplía la ventana y clasifica los barcos por total_onboard, verificando que el SQL SLM utilice la columna proporcionada o un recálculo seguro. Estas tareas confirman el análisis correcto de «arrival_datetime» / «departure_datetime», el orden y la limitación.

4.2.5. Caso de uso 5 (UC-5): unión temporal para llegadas de cruceros y calidad del aire

- Solicitud de NL: «Para cada llegada de crucero el 13-07-2025, muestra los valores de SO2 por hora que se encuentran dentro de su llegada y salida».

El UC-5 alinea las mediciones horarias de la calidad del aire con la ventana de atraque de cada barco el 13-07-2025, devolviendo el valor de SO_2 para cada hora dentro de [arrival_datetime, departure_datetime). Verifica los predicados de intervalo correctos, la cita del identificador con espacios/diacríticos y la proyección/agrupación mínima por llegada, sin agregaciones adicionales.

4.2.6. Caso de uso 6 (UC-6): unión temporal para llegadas de cruceros y calidad del aire

- Solicitud de NL: «Identifique los valores de SO_2 superiores a la media de julio de 2025 durante ≥ 5 horas consecutivas. A continuación, enumere los cruceros atracados durante ese periodo de tiempo, si los hay, así como los valores medios de SO_2 durante su atraque y la media de julio de 2025».

El UC-6 detecta primero ≥ 5 horas consecutivas en las que el SO_2 supera la media mensual de julio de 2025, utilizando una detección robusta basada en marcas de tiempo por hora. A continuación, cruza esos intervalos de SO_2 alto con la [fecha_hora_llegada, fecha_hora_salida) de cada barco para enumerar los cruceros atracados y calcular el SO_2 medio durante su estancia en el muelle, validando las referencias mensuales, la lógica de intervalos y los límites de tiempo precisos.

4.3. Referencia

4.3.1. Comprobaciones de validez y etiquetado

En esta subsección se define la rúbrica utilizada para puntuar cada función y el sistema global en las tablas de referencia que se presentan a continuación.

4.3.1.1. Entidades LLM (descubrimiento de tablas)

- Óptimo (✓) si los términos de búsqueda emitidos son suficientes para recuperar la tabla o tablas exactas requeridas por la solicitud de NL (una sola tabla para UC-1/2/3/4; ambas tablas para UC-5/6). Los términos deben ser discriminatorios y estar alineados con el vocabulario del esquema (incluidos los diacríticos). Incorrecto (x) en caso contrario. La recuperación de las dos tablas cuando solo se requiere una no se considerará incorrecta.

4.3.1.2. SQL LLM (generación de consultas).

- Incorrecto (x): SQL no se ejecuta en Trino, hace referencia a identificadores inexistentes, interpreta erróneamente las ventanas de tiempo o devuelve filas incorrectas.
- Correcto (✓*): se ejecuta y devuelve el resultado correcto, aunque puede que no sea óptimo desde el punto de vista computacional.
- Óptimo (✓): se ejecuta, devuelve el resultado correcto y es lo más eficiente posible desde el punto de vista computacional.
- Nota: Si una indicación es ambigua y admite múltiples interpretaciones, siempre que la consulta sea correcta y devuelva un resultado compatible, será válida. Si la indicación incluye un orden no explícito, no se considera subóptimo no incluir ORDER BY, aunque mejore la usabilidad. Lo mismo ocurre con las columnas no solicitadas.

4.3.1.3. Domain Expert LLM (calidad de la respuesta en lenguaje natural).

- Incorrecto (x): No produce una respuesta correcta a la indicación NL del usuario.
- Correcto (✓*): Responde correctamente a la indicación NL del usuario, basándose fielmente en el resultado SQL y en las descripciones de sus columnas. Datos numéricos reproducidos con un redondeo de ≤ 1 decimal.
- Óptimo (✓): Todas las comprobaciones ✓* más el contexto del dominio y los conocimientos para técnicos, como la interpretación cuando sea aplicable y el significado general.

Tabla 6. Notación y descripciones

Notación	Descripción
✓	Cumple plenamente todos los criterios de corrección.
✓*	Correcto, pero el SQL no es óptimo O la explicación del experto en la materia no ofrece información técnica.
✓**	Correcto, pero el SQL no es óptimo Y la explicación del experto en la materia no ofrece información técnica.
x	No cumple algunos criterios de corrección.
-	No se puede completar debido a un error anterior

Fuente. Elaboración propia, 2025.

4.3.2. Resultados

Tabla 7. Evaluación de casos de uso para la pila Compact-SLM

Caso de uso	Entidades LLM	SQL LLM	Experto en el dominio LLM	Válido Salida
UC-1	✓	✓	✓*	✓*
UC-2	✓	✓	x	x
UC-3	✓	✓	✓*	✓*
UC-4	✓	✓	x	x
UC-5	✓	x	-	-
UC-6	✓	x	-	-

Fuente. Elaboración propia, 2025.

Tabla 8. Evaluación de casos de uso para la pila SLM ampliada

Casos de uso	Entidades LLM	SQL LLM	Experto en el dominio LLM	Válido Salida
UC-1	✓	✓*	✓*	✓**
UC-2	✓	✓*	x	x
UC-3	✓	✓	✓*	✓*
UC-4	✓	✓*	✓*	✓**
UC-5	✓	✓*	✓*	✓**
UC-6	✓	x	-	-

Fuente. Elaboración propia, 2025.

Tabla 9. Evaluación de casos de uso para Gemini 2.5 PRO

Caso de uso	Entidades LLM	SQL LLM	LM de experto en el dominio	Válido Salida
UC-1	✓	✓	✓	✓
UC-2	✓	✓	✓	✓
UC-3	✓	✓	✓	✓
UC-4	✓	✓	✓	✓
UC-5	✓	✓	✓	✓
UC-6	✓	✓	✓	✓

Fuente. Elaboración propia, 2025.

4.4. Discusión

En los seis casos de uso y en todas las configuraciones, el modelo de lenguaje pequeño (SLM) de Entities mostró de forma consistente términos discriminativos que se resolvieron en las tablas correctas en DataHub. google/gemma-2-2b-it funcionó lo suficientemente bien como para ser indistinguible de su homólogo Expanded, google/gemma-3-4b-it y Gemini 2.5 PRO, por lo que parece que esta pequeña tarea se maneja perfectamente con un SLM compacto a pesar de la mezcla de español e inglés en las columnas y descripciones.

En cuanto a la generación de consultas SQL, empiezan a surgir diferencias. La función SQL tuvo éxito en 4 de las 6 tareas con la pila Compact-SLM (UC-1/2/3/4) y en 5 de las 6 con la pila Expanded-SLM (UC-1/2/3/4/5). La referencia Gemini 2.5 PRO completó las 6 tareas.

Al comparar las consultas de una sola tabla, encontramos que el Compact-SLM SQL LLM impulsado por Qwen/Qwen2.5-Coder-3B-Instruct maneja la carga lo suficientemente bien con consultas óptimas y problemas menores, como la falta de una cláusula ORDER BY en UC-2 y UC-4. Su hermano mayor, Qwen/Qwen2.5-Coder-7B-Instruct, también genera consultas correctas, pero tienden a presentar expresiones más complejas (por ejemplo, agregaciones innecesarias), lo que da como resultado un ✓*, además de una falta de orden. Por su parte, Gemini genera consultas óptimas, con órdenes adicionales no solicitadas, que mejoran la usabilidad.

En el caso de varias tablas, Qwen/Qwen2.5-Coder-3B-Instruct no puede generar consultas precisas, ya que falla en UC-5 y UC-6. Para UC-5, genera una consulta compilable que devuelve el crucero correcto y «so2 media horaria», sin embargo, toma la hora de la tabla de llegadas en lugar de la hora de medición real de la tabla de calidad del aire. En este caso, Expanded-SLM Stack SQL LLM lo resuelve y genera las horas correctas, aunque utiliza un filtro de día no sargable, cuando un rango semiabierto sería más económico, como lo hace Gemini para optimizar aún más la consulta. Sin embargo, en UC-6, ni siquiera Qwen/Qwen2.5-Coder-7B-Instruct puede generar la consulta correcta, ya que su predicción presenta un uso incorrecto del alias GROUP BY que provoca un error de compilación, al igual que ocurre con la configuración Compact. Gemini 2.5 PRO utiliza correctamente el patrón de islas y huecos (desplazamientos ROW_NUMBER() + COUNT(*)≥5) para encontrar series realmente consecutivas con altos niveles de SO_2 y, a continuación, realiza una unión por solapamiento adecuada a los cruceros atracados y calcula los promedios del período de atraque, al tiempo que genera un plan computacionalmente óptimo bajo la rúbrica.

Un aspecto clave de nuestra arquitectura aquí es la falta de corrección de una consulta no compilable con EXPLAIN y un aviso posterior. Ninguna de las dos configuraciones fue capaz de corregir el aviso para compilar la consulta SQL UC-6 fallida, lo que puede indicar que su uso en SLM todavía no es tan útil como en LLM, aunque no hay ningún inconveniente en volver a intentarlo una vez que sabemos que la consulta no se compila.

La función de experto en el dominio fue la principal fuente de resultados no válidos en todo el sistema para los SLM: Compact proporcionó explicaciones fundamentadas en 2/6 tareas (✓* en UC-1/3), Expanded en 4/6 (✓* en UC-1/3/4/5), mientras que Gemini produjo **6/6** explicaciones contextuales correctas. En el caso de UC-2, ambas configuraciones de SLM malinterpretaron la salida SQL y mezclaron medias diarias con valores horarios debido al nombre de las columnas y a la falta de una columna específica con la media diaria en la salida. Esto podría solucionarse en el futuro con más reglas que obliguen a mostrar siempre valores discriminatorios, como las medias. En el caso de UC-4, el experto en dominios compactos leyó mal la columna de la hora de llegada e indicó incorrectamente que todos los barcos llegaron el mismo día, aunque dos llegaron el 22 y uno el 21. Aunque se trata de un error menor, confunde al usuario final. Expanded no cometió ese error y, para UC-1/3/4/5, resumió y reformuló correctamente la información valiosa en un lenguaje natural. Sin embargo, ninguna de las dos configuraciones fue capaz de aportar información por sí sola con el contexto proporcionado (información general sobre las funciones y las descripciones de las columnas), a diferencia de Gemini, que intentó establecer correlaciones y contextualizar los resultados para usuarios finales no expertos.

Al final, la configuración Compact-SLM demostró ser válida para el descubrimiento de datos y las consultas sencillas, pero carecía de explicaciones más complejas. La pila Expanded-SLM podía ofrecer consultas y explicaciones más sólidas, pero aún no puede procesar las consultas más complejas al nivel de los modelos de vanguardia. Como asistente de descubrimiento, ambas configuraciones, y especialmente la Expanded, funcionan bien en la actualidad: pueden encontrar tablas relevantes, componer y validar SQL y resumir los resultados rápidamente. Sin embargo, al igual que ocurre con las consultas LLM que superan los límites, no se puede confiar ciegamente en los resultados. No obstante, puede ser mucho más rápido utilizar el asistente para obtener la información deseada y luego verificarla que tener que buscarla desde cero. Por ello, puede ser una ventaja para los usuarios no técnicos que desean una forma más fácil de interactuar con la base de datos y puede ser muy útil para el personal técnico como punto de referencia.

4.4.1. Implicaciones prácticas, limitaciones del estudio y riesgos de implementación

Nuestro asistente basado en SLM ofrece un potencial significativo, listo para usar, para las administraciones públicas, como se ha visto con el SENSE Citiverse para la ciudad de Cartagena. Aborda directamente el reto de hacer que los vastos conjuntos de datos no solo sean accesibles, sino también inspeccionables por personal no técnico mediante el lenguaje natural. A su vez, se podría reducir la carga de trabajo de todos los empleados, lo que permitiría al personal centrarse en tareas más complejas y de mayor valor, objetivo que debería ser del sistema de IA general, especialmente en las aplicaciones de servicio público.

Sin embargo, tanto el estado actual de la solución como el propio estudio presentan limitaciones. El estudio se basa en casos de uso y en datos específicos del dominio, que no pueden cubrir por completo todos los escenarios de consultas espontáneas de los usuarios. Para generalizar la usabilidad de la solución, se debería realizar un estudio a gran escala entre los usuarios para recabar opiniones humanas sobre métricas cruciales como la confianza, la facilidad de uso, la corrección general y la satisfacción de los usuarios, así como introducir nuevos dominios en los que implementar el asistente.

Por último, la implementación de un sistema de este tipo conlleva varios riesgos, entre los que destacan la privacidad y la seguridad de los datos. Aunque un modelo autohospedado permite una soberanía de datos clara y un mayor control de la seguridad en el acceso, el sistema sigue requiriendo protocolos claros y políticas de acceso de los usuarios, especialmente si se incluye la ingesta de datos como funcionalidad en el futuro. Por lo tanto, las pruebas de seguridad proactivas, o red-teaming, son esenciales para identificar estas y otras vulnerabilidades. También existe el riesgo de sesgo algorítmico y de equidad, ya que los modelos de IA pueden amplificar los sesgos presentes en sus datos de entrenamiento, lo que podría dar lugar a resultados de servicio desiguales (Cortés-Cediel et al., 2023). Además, una implementación exitosa requeriría cierto grado de preparación organizativa. Para mitigar los riesgos anteriores y aprovechar al máximo el sistema, el personal debería recibir formación tanto sobre cómo utilizar nuestra aplicación front-end como sobre técnicas de solicitud para garantizar una utilización óptima. Además, se debería contar con personal técnico para mantener el sistema e integrarlo con los sistemas existentes.

5. Conclusiones

Los espacios de datos europeos tienen como objetivo ampliar el acceso a los datos públicos e industriales y ahora llegan a las operaciones municipales a gran escala. Sin embargo, los usuarios aún tienen que navegar por estos recursos con precisión. El SENSE Citiverse de Cartagena ejemplifica esta brecha: los técnicos deben evaluar la telemetría de la calidad del aire e incluso correlacionarla con los datos operativos del puerto y de la ciudad, pero redactar consultas SQL específicas para analizar fuentes de datos heterogéneas puede llevar mucho tiempo, especialmente para usuarios sin conocimientos técnicos. Las interfaces de lenguaje natural (NL) pueden reducir esta barrera si generan consultas fieles y devuelven explicaciones fundamentadas.

Este trabajo aborda esa necesidad con un sistema compacto e interno que convierte NL a SQL y responde en NL, al tiempo que preserva la soberanía de los datos y el control de los costes. Con tres modelos de lenguaje pequeños (SLM) que cooperan entre sí y desempeñan las funciones de los modelos de lenguaje grandes (LLM): 1) LLM de entidades; 2) LLM de SQL y 3) LLM de expertos en el dominio, este enfoque promueve el acceso centrado en el ser humano, al tiempo que respeta las restricciones de soberanía, coste y sostenibilidad que a menudo impiden la dependencia de LLM remotos.

La arquitectura combinó los SLM con potentes herramientas de código abierto: DataHub para el descubrimiento de datos y metadatos, Trino para la ejecución de SQL y KServe para un servicio fiable y de autoescalado. Esta separación de funciones redujo la longitud de las indicaciones, limitó los presupuestos de tokens y permitió observar y recuperar los fallos. El diseño también se alineó con las pruebas recientes de que las canalizaciones multiagente mejoran la robustez del texto a SQL mediante la descomposición y la retroalimentación de la base de datos.

El proceso es sencillo: en primer lugar, el LLM de entidades extrae términos estructurados de la solicitud en lenguaje natural del usuario. En segundo lugar, el orquestador consulta DataHub para vincular esos términos con tablas y columnas específicas. En tercer lugar, el LLM de SQL compone una consulta que ejecuta Trino; en caso de error, el orquestador vuelve a intentarlo con la retroalimentación del compilador y luego ejecuta la consulta reparada. Y, por último, el LLM experto en el dominio lee la pregunta, el SQL validado, el conjunto de resultados y su contexto SQL, y produce una explicación concisa y consciente del dominio. KServe aloja cada función como un punto final autoescalado con control de revisiones y métricas.

Para la evaluación, se implementaron dos pilas SLM: Compact (≤ 3.000 millones de parámetros) y Expanded (≤ 8.000 millones de parámetros), lo que reveló fortalezas y limitaciones claras. Independientemente de la pila, el LLM de entidades emitió de forma consistente términos discriminativos que se resolvieron en las tablas correctas, a pesar de que estas tablas y metadatos mezclaban español e inglés. Las capacidades de generación de SQL de la pila Compact destacaron en las consultas de una sola tabla, mientras que la pila Expanded añadió compatibilidad con las consultas entre tablas; sin embargo, ambas se quedaron cortas en el caso más complejo de una unión temporal, en el que la referencia Gemini 2.5 Pro tuvo éxito. El SLM Domain Expert produjo interpretaciones fieles pero escasas en los casos más sencillos y perdió terreno en los resultados complejos, en particular en el caso de la pila Compact, mientras que la Expanded pudo explicar eficazmente el resultado SQL incluso en escenarios más complejos.

En resumen, una pila compacta y soberana puede ofrecer una conversión rápida y útil de texto a SQL para la telemetría y las operaciones municipales. El sistema reduce las barreras para los no expertos, se escala horizontalmente y preserva la privacidad por diseño. Su modularidad invita a realizar actualizaciones incrementales, al tiempo que contiene el riesgo operativo.

5.1. Trabajo futuro

Este artículo valida la arquitectura propuesta mediante dos configuraciones específicas de modelos en hardware en la nube. Sin embargo, el trabajo futuro evaluará el rendimiento en múltiples configuraciones de modelos y hardware para establecer puntos de referencia claros para su adopción en el mundo real, en particular en hardware de consumo y de nivel empresarial. El objetivo de esta investigación futura será cuantificar las compensaciones relacionadas con el hardware, por lo que se proponen tres niveles distintos de configuraciones de modelos: compacto (≤ 4 GB VRAM), equilibrado (8-16 GB) y ampliado (≥ 24 GB).

Para cada configuración, se medirá el rendimiento en función de métricas críticas, como el tiempo de inferencia (latencia) de cada SLM individual, la latencia total de extremo a extremo desde el envío de la solicitud hasta la respuesta final y la tasa de generación de tokens (tokens/segundo). El objetivo general es identificar configuraciones que puedan ofrecer una respuesta fiable en menos de 10 segundos, garantizando una experiencia de usuario fluida e interactiva.

Los experimentos compararán el ajuste fino ligero con el ajuste fino solo por prompt para las funciones de SQL y para el experto en el dominio. También se tendrán en cuenta casos de uso ampliados para validar aún más la arquitectura propuesta en diferentes escenarios, con pruebas específicas por SLM y de extremo a extremo.

Una nueva canalización de ingestión con LLM enriquecerá los metadatos de las nuevas tablas. El SLM de experto en el dominio generará ~200 tokens, descripciones específicas del dominio por columna a partir de filas de muestra y de convenciones de nomenclatura. A continuación, el Orchestrator realizará la ingestión de datos a través de Trino y cargará las descripciones de los campos a través de DataHub. Además, como se ha presentado, un actor traductor de español permitirá que las indicaciones en español-inglés aprovechen mejor los SLM cuando la indicación esté en español, lo cual es clave para la adopción de Cartagena. Con los resultados ampliados del experimento, así como con las nuevas funcionalidades, se pueden sentar las bases para futuros estudios de usuarios con el fin de obtener comentarios humanos críticos.

La hoja de ruta actual tiene como objetivo evaluar la arquitectura actual, ampliar su funcionalidad y especializar sus conocimientos en las funciones específicas de SENSE Citiverse de Cartagena, al tiempo que se mantiene el marco modular y abierto para futuros proyectos. Si tiene éxito, este asistente puede servir como plantilla compatible con la UE para interfaces NL en todos los espacios de datos, que cumple con la Ley de IA y las nuevas regulaciones relacionadas con la IA en Europa para que este tipo de modelos sean adecuados para la economía digital europea, con un enfoque especial en la transformación digital para ciudades inteligentes. Este modelo estaría disponible en el EDIC Marketplace para que se pudiera utilizar como código abierto junto con los gemelos digitales locales europeos, compatible con la caja de herramientas LDT de la UE y en entornos de pruebas como Citcom.AI TEF Sandbox.

Referencias

Basant, A., Khairnar, A., Paithankar, A., Khattar, A., Renduchintala, A., Malte, A., Bercovich, A., Hazare, A., Rico, A., Ficek, A., Kondratenko, A., Shaposhnikov, A., Bukharin, A., Taghibakhshi, A., Barton, A., Mahabaleshwarkar, A. S., Shen, A., Tao, A., Guan, A., Shors, A. (…) Chen, Z. (2025). NVIDIA Nemotron Nano 2: An accurate and efficient hybrid mamba-transformer reasoning model. *ArXiv*, *V4*. https://arxiv.org/abs/2508.14444

Belcak, P., Heinrich, G., Diao, S., Fu, Y., Dong, X., Muralidharan, S., Lin, Y.-C., & Molchanov, P. (2025). Small language models are the future of agentic AI. *ArXiv*, *v2*. https://arxiv.org/abs/2506.02153

Chen, A., Bundele, M., Ahlawat, G., Stetz, P., Wang, Z., Fei, Q., Jung, D., Chu, A., Jayaraman, B., Panth, A., Arora, Y., Jain, S., Varma, R., Ilin, A., Melnychuk, I., Chueh, C., Sil, J., & Wang, X. (2025). Text-to-SQL for enterprise data analytics. *Workshop on Agentic AI for Enterprise at KDD '25*, Toronto, ON, Canada. https://arxiv.org/abs/2507.14372

Cortés-Cediel, M. E., Segura-Tinoco, A., Cantador, I., & Rodríguez Bolívar, M. P. (2023). Trends and challenges of e-government chatbots: Advances in exploring open government data and citizen participation content. *Government Information Quarterly*, *40*(4), 101877. https://doi.org/10.1016/j.giq.2023.101877

DataHub Project. (2015a). *DataHub GraphQL API*. DataHub. https://docs.datahub.com/docs/api/graphql/overview

DataHub Project. (2015b). *DataHub quickstart guide*. DataHub. https://docs.datahub.com/docs/quickstart/

DataHub Project. (2024). *DataHub APIs and SDKs overview*. DataHub. https://docs.datahub.com/docs/api/datahub-apis

DataHub Project. (2025). *DataHub | Modern data catalog & metadata platform*. DataHub. https://datahub.com/

Google. (2025a). *Gemma-2-2b-it* [Large language model]. Hugging Face. https://huggingface.co/google/gemma-2-2b-it

Google. (2025b). *Gemma-3-4b-it* [Large language model]. Hugging Face. https://huggingface.co/google/gemma-3-4b-it

Hong, Z., Yuan, Z., Zhang, Q., Chen, H., Dong, J., Huang, F., & Huang, X. (2024). A next-generation survey of LLM-based text-to-SQL database interfaces [Preprint]. *arXiv*. V8. https://arxiv.org/abs/2406.08426

Inglés-Romero, J. F., Ferri, M., & Jara, A. J. (2025). Exploring human usability challenges in dataspaces. En J. Theissen-Lipp, P. Colpaert, A. Pomp, E. Curry & S. Decke (Eds.) *The Third International Workshop on Semantics in Dataspaces, ESWC 2025*, Portorož, Slovenia.

Jiang, Y., Pang, P. C.-I., Wong, D., & Kan, H. Y. (2023). Natural language processing adoption in governments and future research directions: A systematic review. *Applied Sciences*, *13*(22), 12346. https://doi.org/10.3390/app132212346

KServe Project. (2024). *modelmesh: Distributed model serving framework* (Version 0.12.0) [Computer software]. GitHub. https://github.com/kserve/modelmesh

KServe Project. (2025a). *Control plane*. KServe Documentation. https://kserve.github.io/website/docs/concepts/architecture/control-plane

KServe Project. (2025b). *Deploy your first GenAI service*. KServe Documentation. https://kserve.github.io/website/docs/getting-started/genai-first-isvc

KServe Project. (2025c). *Deploy your first predictive AI service*. KServe Documentation. https://kserve.github.io/website/docs/getting-started/predictive-first-isvc

KServe Project. (2025d). *KServe*. Retrieved October 3, 2025, from https://kserve.github.io/website/

KServe Project. (2025e). *kserve: Standardized distributed generative and predictive AI inference platform for scalable, multi-framework deployment on Kubernetes* (Version 0.15.2) [Computer software]. GitHub. https://github.com/kserve/kserve

Kubeflow. (2025, 31 de julio). *Introduction: A brief introduction to KServe*. Kubeflow Documentation. https://www.kubeflow.org/docs/components/kserve/introduction/

Kubernetes. (2024, 20 de abril). *Kubernetes documentation*. https://kubernetes.io/docs/home/

Meta AI. (2024a). *Llama-3.1-8B-Instruct* [Large language model]. Hugging Face. https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

Meta AI. (2024b). *Llama-3.2-3B-Instruct* [Large language model]. Hugging Face. https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct

Qwen Team. (2025a). *Qwen2.5-Coder-3B-Instruct* [Large language model]. Hugging Face. https://huggingface.co/Qwen/Qwen2.5-Coder-3B-Instruct

Qwen Team. (2025b). *Qwen2.5-Coder-7B-Instruct* [Large language model]. Hugging Face. https://huggingface.co/Qwen/Qwen2.5-Coder-7B-Instruct

SENSE Project. (2025a, 19 de junio). *CitiVerse*. Senseverse. https://senseverse.eu/citiverse/

SENSE Project. (2025b, 13 de junio). *Pilots*. Senseverse. https://senseverse.eu/pilots/

SENSE Project. (2025c, 5 de junio). *Senseverse | Building smart, connected citizens with SENSE*. Senseverse. https://senseverse.eu/

Shi, L., Tang, Z., Zhang, N., Zhang, X., & Yang, Z. (2025). A survey on employing large language models for text-to-SQL tasks. *ACM Computing Surveys, 58*(2), Article 54. https://doi.org/10.1145/3737873

Trino. (2020). *EXPLAIN*. Trino Documentation. https://trino.io/docs/current/sql/explain.html

Trino. (2025a). *Concepts*. Trino Documentation. https://trino.io/docs/current/overview/concepts.html

Trino. (2025b). *Connectors*. Trino Documentation. https://trino.io/docs/current/connector.html

Trino. (2025c). *EXPLAIN ANALYZE*. Trino Documentation. https://trino.io/docs/current/sql/explain-analyze.html

Ukil, A., Jara, A., Gama, J., & Bellatreche, L. (2025). Buck the trend: Make LLMs specific and reduce the cost of intelligence. *28th European Conference on Artificial Intelligence*, Bologna, Italy.

Yigitcanlar, T., David, A., Li, W., Fookes, C., Bibri, S. E., & Ye, X. (2024). Unlocking artificial intelligence adoption in local governments: Best practice lessons from real-world implementations. *Smart Cities, 7*(4), 1576–1625. https://doi.org/10.3390/smartcities7040064