
This is the **published version** of the bachelor thesis:

Ferret Font, Oriol. *Disseny i implementació d'un sistema d'anàlisi FinOps mitjançant LLM*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Enginyeria Informàtica)

This version is available at <https://ddd.uab.cat/record/334409>

under the terms of the  license.

Disseny i implementació d'un sistema d'anàlisi FinOps mitjançant LLM

Oriol Ferret Font

Resum— Aquest treball presenta el disseny i desenvolupament d'una Prova de Concepte (PoC) per a un sistema d'anàlisi FinOps basat en intel·ligència artificial generativa completament local. Mitjançant l'orquestració de microserveis amb Docker, FastAPI i un model de llenguatge (LLM) executat via Ollama, la plataforma permet als usuaris consultar mètriques i costos d'infraestructura cloud utilitzant llenguatge natural. L'arquitectura integra tècniques de Generació Augmentada per Recuperació (RAG) per injectar dinàmicament el context de la base de dades, garantint la traducció precisa a sentències SQL. A més, el sistema incorpora un perímetre de seguretat amb control d'accés (RBAC) i filtres lèxics preventius (guarda-rails) que bloquegen operacions destructives. Els resultats certifiquen la viabilitat de generar explicacions analítiques de manera autònoma i ràpida, facilitant la presa de decisions sense comprometre la privacitat ni el control total sobre les dades corporatives.

Paraules clau—Model de Llenguatge gran(LLM), FinOps, RAG, Docker, Ollama, SQL, FastAPI, Streamlit

Abstract— This work presents the design and development of a Proof of Concept (PoC) for a completely local, generative AI-based FinOps analytics system. Through the orchestration of microservices using Docker, FastAPI, and a Large Language Model (LLM) executed via Ollama, the platform enables users to query cloud infrastructure metrics and costs using natural language. The architecture integrates Retrieval-Augmented Generation (RAG) techniques to dynamically inject database context, ensuring accurate translation into SQL queries. Furthermore, the system incorporates a security perimeter featuring Role-Based Access Control (RBAC) and preventive lexical filters (guardrails) that block destructive operations. The results confirm the viability of generating analytical explanations autonomously and rapidly, facilitating decision-making without compromising corporate data privacy and control.

Index Terms— Large Language Model (LLM), FinOps, RAG, Docker, Ollama, SQL, FastAPI, Streamlit

1 INTRODUCCIÓ - CONTEXT DEL TREBALL

En l'actualitat, l'adopció massiva d'infraestructures cloud ha transformat l'agilitat operativa de les empreses, però ha introduït una complexitat sense precedents en el control de costos. El moviment FinOps [1] sorgeix per gestionar aquesta despesa, tot i que encara avui dia l'anàlisi de dades depèn sovint de consultes SQL manuals o de taulers de control rígids en Power BI [2]. Aquesta dependència tecnològica crea una barrera que dificulta la immediatesa en la presa de decisions i limita l'accés a la informació per a perfils no tècnics.

La motivació per desenvolupar aquest projecte neix de l'experiència directa durant la meua incorporació a l'empresa ITnow [3], on la gran densitat de taules i estructures complexes dificulta l'adaptació inicial i el seguiment àgil dels recursos. Aquesta necessitat real ha portat a plantejar l'ús de la intel·ligència artificial generativa com un pont per simplificar la interacció amb les dades, permetent transformar la consulta tècnica en una conversa natural i accessible per a qualsevol perfil professional.

Per abordar aquestes mancances, s'ha optat per la creació d'un Agent FinOps basat en un model de llenguatge

gran (LLM) [4] d'execució local mitjançant Ollama [5]. L'estratègia se centra a garantir la privacitat i el control exclusiu sobre la localització de la informació mitjançant un entorn totalment contenidoritzat amb Docker [6], que permet la interpretació de consultes complexes i la detecció d'anomalies de forma autònoma dins de la xarxa corporativa.

Mitjançant la integració de FastAPI [7] i Streamlit [8], s'ha desenvolupat una eina que prioritza la seguretat i la immediatesa, eliminant la dependència de processos manuals i optimitzant el seguiment dels recursos cloud.

Aquest document presenta de manera detallada tot el procés de conceptualització i desenvolupament del sistema, des de l'anàlisi de les necessitats i l'estat de l'art fins a la implementació final de l'arquitectura. S'hi exposen les decisions tècniques adoptades per a l'orquestració dels microserveis, el disseny dels mecanismes de seguretat perimetral i l'aplicació de tècniques d'optimització contextual. Així mateix, es mostren els resultats obtinguts mitjançant proves empíriques de rendiment i escalabilitat que validen l'assoliment dels objectius plantejats inicialment. Finalment, s'identifiquen les línies de futur que permetran evolucionar la plataforma i s'extreuen les conclusions tècniques i econòmiques derivades de la viabilitat d'aquest projecte.

- E-mail de contacte: oriol.ferret@autonoma.cat, oriolf2003@gmail.com
- Menció realitzada: Tecnologies de la Informació
- Treball tutoritzat per: Miguel Carpio Miranda (DEiC) i Jose Santos Guijarro (FinOps a ITnow)
- Curs 2025/26

2 OBJECTIUS

L'objectiu principal d'aquest treball és el disseny i desenvolupament d'una PoC [9] que consisteix en un sistema conversacional local basat en models de llenguatge gran (LLM) per a l'anàlisi de dades FinOps. Aquesta eina ha de permetre als usuaris de l'entorn FinOps analitzar dades d'infraestructura cloud d'una manera simple i immediata, facilitant tasques com la detecció d'anomalies o la distribució d'objectes cloud mitjançant la interacció en llenguatge natural. Els objectius específics definits per al desenvolupament són els següents.

2.1 ORQUESTRAR L'ENTORN CONTENIDORITZAT

Dissenyar i desplegar una arquitectura de microserveis mitjançant Docker, integrant de manera aïllada el motor de base de dades MySQL [10], el node d'intel·ligència Ollama, l'API centralitzada desenvolupada amb FastAPI i la interfície client en Streamlit. El disseny ha de garantir la privacitat de la informació com a requisit no funcional. En aquest sentit, es prioritza la construcció d'un perímetre de seguretat robust mitjançant un sistema de control d'accés basat en rols (RBAC) [11] gestionat a través de tòquens JWT [11], el qual restringeix l'operativitat segons el perfil d'usuari, així com la implementació de guarda-raïls lèxics per validar preventivament la integritat i la intencionalitat de les sentències SQL abans de la seva execució.

2.2 DESENVOLUPAR LA Lògica DE L'AGENT I RAG [2]

Crear un flux de generació de sentències SQL a partir de consultes de l'usuari en llenguatge natural, utilitzant tècniques avançades de prompt engineering [13] per mitigar les al·lucinacions del model i assegurar respostes estructurades. Així mateix, s'estableix com a objectiu la integració d'una millora en la resposta final mitjançant l'arquitectura de Generació Augmentada per Recuperació (RAG). Aquesta tècnica ha de permetre la generació de resums explicatius i narratius fonamentats exclusivament en dades reals, contrastades i extretes directament de la base de dades de l'organització.

2.3 IMPLEMENTAR LA INTERFÍCIE I VALIDACIÓ

Construir una interfície gràfica d'usuari interactiva i modular mitjançant Streamlit que unifiqui l'experiència d'operació. Aquest component ha de facilitar l'autenticació de l'operador, la introducció de consultes i la visualització dual dels resultats, combinant la resposta narrativa de l'agent amb taules de dades dinàmiques estructurades. Finalment, es durà a terme una validació d'extrem a extrem (end-to-end) exhaustiva del sistema, mitjançant proves automatitzades, per certificar que el flux operatiu complet s'executa amb precisió i compleix amb els requisits tècnics, funcionals i de seguretat establerts a l'inici del projecte.

3 ESTAT DE L'ART

El mercat actual de solucions FinOps està experimentant una transformació cap a l'ús d'agents d'intel·ligència artificial. Per tal de contextualitzar la proposta, s'han revisat les solucions existents per identificar-ne les limitacions i justificar la necessitat del sistema desenvolupat. Actualment, el sector es divideix principalment en plataformes comercials SaaS [14] i eines de visualització interna.

En l'àmbit comercial, destaquen aplicacions com Amnic [15] i FinOpsly [16]. Aquestes eines ofereixen agents conversacionals potents que permeten interaccionar amb les dades de costos cloud mitjançant llenguatge natural. Tot i això, el seu enfocament es basa principalment en el cloud, fet que implica que les dades financeres de l'empresa s'han d'enviar a servidors externs i dependre d'APIs [17] de tercers com les d'OpenAI [18]. Aquesta característica suposa una limitació crítica per a organitzacions que prioritzen la confidencialitat i el confinament de les dades dins de la seva pròpia infraestructura.

D'altra banda, en l'àmbit intern, empreses com ITnow utilitzen actualment plataformes com Power BI o consultes manuals en SQL per al seguiment de costos. Aquestes solucions, tot i ser sòlides, resulten sovint rígides i obliguen els usuaris a dependre de coneixements tècnics o de taules de control que no permeten una exploració àgil de les dades. A més, la corba d'aprenentatge per a nous perfils és elevada a causa de la gran densitat de taules i estructures complexes que cal dominar.

Després d'analitzar aquest context, s'identifiquen mancances clares que la solució proposada pretén resoldre. Mentre que les aplicacions comercials requereixen connectivitat externa i no garanteixen que el flux de dades es mantingui íntegrament en l'entorn local, les eines internes tradicionals no disposen de la personalització i la interfície natural necessàries per facilitar l'accés a usuaris no tècnics.

El sistema desenvolupat es diferencia per oferir una solució 100% local mitjançant Docker i Ollama. Aquesta arquitectura combina l'autonomia de no dependre d'APIs externes amb un sistema de seguretat basat en guarda-raïls que validen cada consulta SQL generada pel model abans d'executar-la sobre la base de dades. D'aquesta manera, s'aporta un valor afegit en privacitat i seguretat, permetent una interacció molt més intuïtiva amb el model de dades de FinOps.

4 METODOLOGIA

El desenvolupament s'ha estructurat sota el marc de les metodologies àgils [19], combinant l'organització de Scrum [20] amb la gestió visual de Kanban [21]. Aquesta elecció permet un enfocament iteratiu i incremental, idoni per a una prova de concepte on la prioritat és evolucionar el sistema mitjançant petits increments funcionals fins a assolir-ne l'operativitat total. L'adopció d'aquest marc de treball segueix les pràctiques operatives de l'empresa ITnow.

La gestió de les tasques s'articula mitjançant un tauler visual a Jira [22], on cada User Story (US) es controla en temps real. Per garantir la traçabilitat, el flux de treball s'or-

ganitza en els següents estats: Backlog, Planned, In Progress, In Review i Done.

Aquesta metodologia facilita una gestió dinàmica on es poden introduir millores constants en el prompt engineering o en la seguretat dels guarda-raïls a mesura que es detecten noves necessitats durant les proves. El caràcter visual de Kanban permet mantenir el flux de treball entre la base de dades, el backend i el model de llenguatge (LLM) constantment alineat amb els objectius finals del projecte.

5 PLANIFICACIÓ

L'estratègia temporal s'ha establert per garantir el compliment de la data límit d'entrega a finals de juny de 2026. Tot i basar-se en un flux de treball continu, el calendari s'ha segmentat en quatre sprints que s'alineen amb els requisits del TFG i els components definits en l'arquitectura del sistema:

1. Sprint d'Infraestructura (2 de març - 8 d'abril): L'objectiu principal és el desplegament de l'entorn amb Docker Compose. Això inclou la creació de la base de dades finops-mock i el sistema de càrrega de dades sintètiques per garantir la coherència de les proves posteriors.

2. Sprint de Nucli i Intel·ligència (9 d'abril - 12 de maig): En aquest període es desenvolupa el motor del sistema. Es construeix el Backend amb FastAPI, es configura el servei d'Ollama i s'implementa la seguretat mitjançant rols i permisos JWT per protegir les dades sensibles.

3. Sprint d'Interfície i Enriquiment (13 de maig - 16 de juny): Es posa en funcionament el Frontend amb Streamlit per facilitar la interacció de l'usuari. Paral·lelament, s'integra la millora de les respostes mitjançant RAG, que permet al model generar explicacions en llenguatge natural a partir dels resultats SQL obtinguts.

4. Sprint de Validació i Tancament (17 de juny - Entrega): L'últim sprint es dedica a la validació end-to-end de tot el flux, assegurant que l'agent respon correctament als casos d'ús de Kubernetes [23] i anomalies cloud.

Aquesta planificació preveu marges de maniobra per gestionar la complexitat de la generació de SQL per part de l'agent. En cas d'endarreriments, les tasques pendents s'integren com a prioritat absoluta en l'inici del període següent per assegurar la continuïtat del projecte.

La planificació inicial s'ha formalitzat mitjançant un cronograma (veure la figura 1 de l'apèndix A.1), el qual permet una visualització clara de la ruta crítica del projecte. No obstant això, per a la gestió operativa de les tasques i el seguiment dels sprints en temps real, està previst traslladar aquesta planificació a un tauler de Jira, facilitant així una gestió més àgil i alineada amb la metodologia Scrumban [24] adoptada.

6 IMPLEMENTACIÓ I ARQUITECTURA TÈCNICA

El procés de construcció del sistema s'ha basat en l'orquestració de microserveis contenidoritzats per garantir l'aïllament de la informació. L'objectiu ha estat materialitzar una solució que permeti la transició de consultes en llenguatge natural a sentències SQL executables sota un

entorn de privacitat absoluta.

6.1 DISSENY DE LA INFRAESTRUCTURA

L'estratègia de desplegament utilitza Docker Compose per gestionar el cicle de vida de cinc nodes fonamentals que operen en una xarxa virtual aïllada:

- **Node de Persistència (database):** Aquest node allotja una instància de MySQL, configurada com el motor de persistència de dades del projecte. La seva funció és garantir l'emmagatzematge estructurat de la informació de facturació i recursos, permetent l'execució de consultes SQL complexes per part de l'agent. S'ha optat per una organització de dades optimitzada per a la lectura, facilitant que el LLM pugui extreure mètriques i costos de forma eficient.
- **Node d'Intel·ligència (ollama):** Executa el servei d'Ollama, un motor especialitzat en l'execució i exposició de LLMs com a servei en l'àmbit local. Aquesta configuració és clau per a la viabilitat de la PoC, ja que elimina la latència de xarxa associada a serveis externs i suprimeix els riscos de privacitat derivats de l'enviament de metadades financeres a APIs de tercers.
- **Node d'Orquestració (backend):** Basat en FastAPI, actua com a punt d'entrada i gestor de la lògica de negoci, seguretat i comunicació asíncrona amb el LLM.
- **Node d'Interfície d'Usuari (frontend):** Executa l'aplicació Streamlit, contenidoritzada per garantir que l'entorn de presentació és completament portable i connecta exclusivament amb l'API del backend.
- **Node d'Administració (cloudbeaver [25]):** Proveeix una interfície de gestió gràfica per a la base de dades, facilitant la verificació de l'estat de les taules i la càrrega de dades sintètiques durant la fase de desenvolupament.

L'arquitectura dissenyada en la figura 1 segueix els principis de desacoblament que inspiren estàndards emergents com el Model Context Protocol (MCP) [26], on s'estableix una separació estricta entre el model d'intel·ligència i les fonts de dades estructurades, actuant el backend com un proveïdor de context segur i controlat.

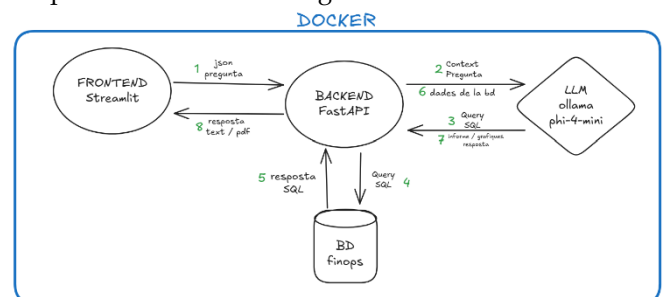


Fig. 1. Diagrama de la infraestructura.

Per validar l'estabilitat d'aquesta infraestructura, s'implementen mecanismes de monitoratge a través del terminal. Mitjançant la comanda `docker ps` (veure figura 2), es ve-

rifica en temps real la salut i l'assignació de ports dels cinc nodes, garantint que la xarxa virtual interna és operativa abans de qualsevol interacció.

```

• ~/documentation$ docker ps
CONTAINER ID   IMAGE
edd8c83d5201   mysql:8.0.35
4405c4e55482   streamlit:latest
8ce637a0600a   dbeaver/cloudbeaver:24.2.0
812d322ba103   ollama/ollama:latest

```

Fig. 2. Execució i orquestració dels cinc contenidors a l'entorn Docker.

6.2. DESENVOLUPAMENT DEL BACKEND I SEGURETAT

El backend s'ha estructurat com l'orquestrador central de la PoC, implementat amb el framework FastAPI per la seva naturalesa asíncrona i el seu alt rendiment. L'API s'ha dissenyat seguint un esquema RESTful [27] modular, on cada endpoint té una responsabilitat ben definida dins del flux de control, seguretat i execució.

6.2.1. ARQUITECTURA D'API, ROBUSTESA I CONTROL D'ACCÉS

Aquesta capa de seguretat perimetral s'articula mitjançant un model de Control d'Accés Basat en Rols (RBAC) [28], materialitzat a través de rutes específiques com `POST /login/access-token` per a la validació i emissió de credencials, i endpoints d'administració com `POST /users` i `GET /roles`. S'han definit tres taules a la base de dades (`backend_user`, `backend_role` i `backend_user_role`) que permeten una gestió granular dels permisos.

Actualment, el sistema suporta tres perfils: Admin (gestió total), User i Ollama (lectura analítica). Els tòquens JWT generats tenen una vigència de 30 minuts per minimitzar riscos de seguretat en la sessió (veure figura 3). Tot i que en aquest capítol s'ofereix una visió sintètica d'aquests mòduls de control, el llistat exhaustiu i expandit de les rutes i els mètodes HTTP que conformen el catàleg d'operacions CRUD (Create, Read, Update, Delete) per a la governança d'identitats es troba detallat a l'Apèndix A.2.

Per garantir l'alta disponibilitat i la traçabilitat de la infraestructura contenidoritzada, s'ha implementat un sistema de verificació d'estat a través de l'endpoint `GET /health`. Aquest component realitza una validació proactiva de la connectivitat entre el contenidor del backend, la instància de MySQL i el node del LLM. Aquesta pràctica és útil en arquitectures de microserveis per assegurar que el flux de dades és operatiu abans d'iniciar processos complexos de generació de text per part del LLM.

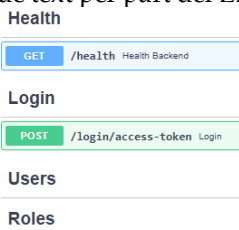


Fig. 3. Interfície Swagger UI [29] mostrant l'estructura dels endpoints destinats al control d'accessos (usuaris i rols) i al monitoratge de salut del sistema.

6.2.2. AÏLLAMENT DE DADES I GUARDA-RAÏLS LÈXICS

El sistema ha estat dissenyat sota un principi de confiança zero (Zero Trust) [30]. La seguretat de l'arquitectura es fonamenta en el fet que el model d'intel·ligència actua com un component cec, sense cap credencial per interactuar directament amb el motor de persistència de MySQL. L'únic punt d'accés autoritzat per materialitzar consultes a la base de dades és l'endpoint intern `POST /get-data`.

Aquest endpoint aprofita el mecanisme d'injecció de dependències de FastAPI (`Depends (allow_sql)`) per garantir que només peticions autenticades i amb els rols adients puguin accedir-hi. Ara bé, per protegir el sistema davant d'al·lucinacions del model o atacs d'injecció SQL, tota consulta generada passa prèviament per un mòdul de validació estricta (guarda-raïls).

Aquesta capa de seguretat preventiva processa la sortida en cru del LLM, en neteja el format de blocs de codi (Markdown) i aplica un filtre basat en expressions regulars. Aquest patró verifica lèxicament que la sentència comenci ineludiblement amb la instrucció de lectura `SELECT`. Qualsevol intent d'executar ordres destructives o de modificació d'estat, com ara `UPDATE`, `DELETE` o `DROP`, és interceptat i fa saltar una excepció lògica (`ValueError`) que bloqueja la petició abans que aquesta arribi al connector de dades. A més, aquest mòdul s'encarrega d'aplicar correccions estructurals automàtiques a la sintaxi (com ara resoldre defectes freqüents en les agrupacions `GROUP BY`). Un cop validada la consulta de manera segura, s'assegura la finalització semàntica de la comanda afegint-hi el punt i coma final (`;`), i el servei central extreu els resultats de MySQL en una llista de diccionaris en format JSON homogeni, preparats per ser consumits pel frontend o pel mòdul d'explicació. El detall de la implementació d'aquest filtre lèxic es troba recollit a l'Apèndix A.3.

6.2.3. SEGURETAT EN EL TRANSPORT I LOCALITZACIÓ

D'acord amb l'objectiu de privacitat absoluta, tota la comunicació entre el backend, els nodes d'interfície i el node d'intel·ligència Ollama es realitza internament dins de la xarxa virtual aïllada de Docker. Això evita l'exposició de les consultes financeres a internet, ja que el processament es manté confinat en l'entorn local, complint amb els requisits de confidencialitat definits en el context d'ITnow.

6.2.4. TRAÇABILITAT DE CONSULTES I VALIDACIÓ D'APIS

Per certificar la integritat i la precisió del flux de dades entre l'orquestrador (backend) i els serveis d'intel·ligència artificial, s'ha utilitzat la interfície interactiva de Swagger UI. Aquesta eina permet validar l'estructura exacta de les comunicacions de manera aïllada i sense necessitat d'interactuar amb l'aplicació client, corroborant que els acords de comunicació (contractes d'API) es compleixen rigorosament a través de tres fases ben definides.

La figura 4 mostra la primera fase d'aquest procés mitjançant l'endpoint `POST /llm/sql`. En aquesta etapa de generació, el cos de la petició només requereix la pregunta en llenguatge natural formulada per l'usuari. Internament, el

codi de l'orquestrador s'encarrega d'injectar-hi automàticament les instruccions estrictes del sistema (system prompts) i el context dinàmic de l'esquema de dades recuperat via RAG. L'èxit d'aquesta validació es demostra en la resposta JSON, on s'obté la sentència SQL traduïda i estructurada, aïllada de qualsevol soroll narratiu.

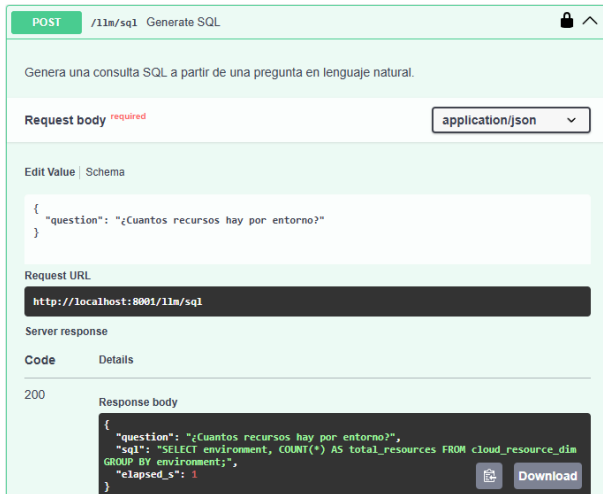


Fig. 4. Validació de l'endpoint `/llm/sql`. S'observa l'estructura del JSON de resposta que conté exclusivament la sentència SQL generada a partir de la pregunta de l'usuari.

Un cop obtinguda la sentència generada pel model, el cicle operatiu continua utilitzant l'endpoint independent `POST /get-data`. Aquest servei actua com el pont d'execució i la barrera de seguretat del sistema: rep la instrucció SQL, hi aplica els filtres lèxics (guarda-raïls) per garantir que es tracti d'una operació de lectura pura (SELECT) i, un cop validada, l'executa contra la base de dades MySQL. El resultat és l'extracció dels registres reals empaquetats en una llista de diccionaris JSON.

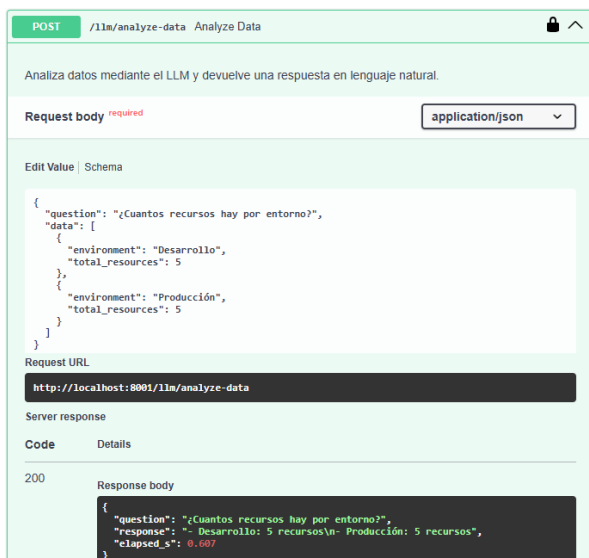


Fig. 5. Validació del servei `/llm/analyze-data`, demostrant la transformació de dades estructurades a una resposta concisa en llenguatge natural.

Finalment, la figura 5 documenta la tercera i última fase a través de l'endpoint `POST /llm/analyze-data`. Les dades d'entrada d'aquesta petició inclouen tant la pregunta original com el conjunt de dades tabulars obtingudes en el pas anterior. Novament, el backend hi adjunta automàticament la directriu analítica que assigna el rol d'expert FinOps a l'agent. Aquesta captura serveix com a evidència tècnica de la capacitat del sistema per processar matrius de dades denses de manera aïllada, abstraure'n els valors numèrics i retornar una explicació narrativa llegible per a l'usuari final.

6.3. DISSENY DE DADES I PERSISTÈNCIA

La capa de persistència opera sobre una instància de MySQL amb un esquema de sis taules (vegeu la figura 6), dividit en dues àrees funcionals: seguretat i anàlisi.

L'àrea de seguretat implementa el control d'accés (RBAC) mitjançant tres entitats (`backend_user`, `backend_role` i `backend_user_role`) vinculades amb relacions de molts a molts i una integritat referencial estricta. D'altra banda, el nucli analític es dissenya seguint les pràctiques de modelatge dimensional d'un magatzem de dades (Data Warehouse) en estrella (star schema). El pilar fonamental és la dimensió mestra desnormalitzada `cloud_resource_dim`, que centralitza els atributs descriptius dels actius cloud. Aquesta s'enllaça mitjançant claus foranes optimitzades amb la taula de fets `cloud_resource_cost_fact` (registre de costos diaris) i la dimensió temporal `date_dim`.

Aquesta estructura dimensional és una decisió estratègica per a l'entorn FinOps. Tot i requerir operacions d'enllaç relacional (JOIN), atorgar el protagonisme a la dimensió de recursos estandaritza les relacions perifèriques. Aquest disseny facilita que el model de llenguatge interpreti l'esquema de manera intuïtiva, permetent-li generar sentències SQL d'anàlisi de costos ràpides i lliures d'al·lucinacions estructurals.

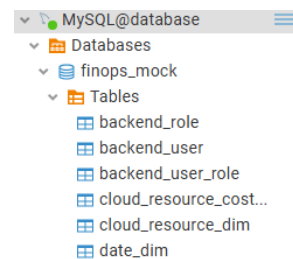


Fig. 6. Vista de les taules dins de la base de dades `finops_mock`.

6.4. MOTOR D'INTEL·LIGÈNCIA I PROMPT ENGINEERING

El motor de raonament utilitza el model local Phi-4-mini [31], seleccionat pel seu equilibri òptim entre estabilitat en codi estructurat i eficiència computacional. Per maximitzar la precisió, l'arquitectura s'estructura en dues fases aïllades: generació sintàctica (traducció a SQL) assistida per RAG, i enriquiment semàntic dels resultats.

La primera fase optimitza el context mitjançant un RAG basat en indexació i cerca per paraules clau. S'ha descartat l'ús de bases de dades vectorials per evitar complexitats innecessàries, ja que la recuperació lèxica és suficient. Aquest

mòdul recupera exclusivament les definicions de les taules coincidents, injectant un esquema dinàmic que evita saturar la finestra de context. Per prevenir al·lucinacions, s'introdueix una lògica d'associació en la petició: es pauta el model per mapejar conceptes naturals (com "equip" o "tècnica de sistemes") cap a les columnes tècniques corresponents, i es defineix que cercar "actius" implica un filtratge d'estat. A més, se li restringeix generar exclusivament codi MySQL 8.0 en una sola línia, bloquejant qualsevol explicació narrativa.

Un cop l'orquestrador valida i executa la consulta de forma segura, s'activa la segona fase: la interpretació. Aquesta fase no utilitza RAG, sinó que pauta l'agent com un expert FinOps. Abans d'injectar els registres, el sistema serialitza automàticament els tipus de dades complexos (com formats Decimal per als costos o objectes datetime per a les dates) cap a una càrrega útil en JSON. Per garantir l'exactitud empírica, s'apliquen límits de context rigorosos: es restringeix explícitament la invenció d'informació inexistents al JSON proporcionat i se li condiciona a emetre un avís si les estructures són buides. Aquest cicle conclou transformant matrius de dades denses en respostes clares i llistades en llenguatge natural per a l'usuari final.

6.5. INTERFÍCIE D'USUARI I VISUALITZACIÓ (STREAMLIT)

L'aplicació client, desenvolupada amb el framework Streamlit, actua com l'únic punt d'interacció directa amb l'usuari final. El disseny de la interfície s'ha estructurat de manera modular per oferir una experiència fluida i dual, dividint l'espai de treball de manera que hi convisquin l'explicació textual de l'agent analític i la presentació estructurada de les dades. Aquest enfocament minimitza la complexitat d'ús i permet que qualsevol perfil operatiu realitzi exploracions sobre el catàleg de dades sense requerir coneixements avançats de sintaxi SQL.

La interfície s'organitza mitjançant un panell lateral de control dedicat a la configuració bàsica i l'autenticació de l'operador, el qual dona accés directe a l'àrea principal de treball un cop validat el perfil de seguretat. Dins de l'espai central, l'aplicació implementa un camp de consulta on l'usuari introdueix la seva petició en llenguatge natural. El sistema processa la petició a través de l'orquestrador central i mostra visualment la interpretació textual generada pel model local, acompanyada d'un bloc dinàmic on s'exposen els registres en format de taula interactiva.

Aquesta disposició permet verificar visualment les dades base que fonamenten la conclusió de l'agent, optimitzant la traçabilitat de la informació i facilitant la detecció primerenca d'anomalies en l'inventari de recursos cloud (veure figura 7). Tot i que en aquest apartat es descriu l'entorn de treball analític de la interfície, les captures de pantalla corresponents al panell de control de seguretat i a la pantalla d'inici s'il·lustren a l'Apèndix A.4.

7 RESULTATS

En la fase actual de desenvolupament de la Prova de Concepte (PoC), els resultats es materialitzen en l'operativitat completa d'una plataforma d'anàlisi FinOps end-to-

end. El sistema ha demostrat empíricament la capacitat d'interpretar peticions no estructurades, coordinar la generació de codi amb el model local d'Ollama i extreure informació financera de manera autònoma, segura i integrable en una interfície gràfica d'usuari.



Fig. 7. Vista principal de la interfície d'usuari (Streamlit), on es mostra la interacció en llenguatge natural i la visualització dual de l'explicació textual juntament amb la taula estructurada de dades obtingudes.

7.1. PROCESSAMENT I ORQUESTACIÓ DE CONSULTES

El resultat tecnològic principal és l'èxit en la gestió del cicle de vida de la dada sota demanda. A través de l'arquitectura modular del backend, s'ha assolit una integració on el model de llenguatge opera eficaçment sota les directrius del sistema de doble prompt explicat anteriorment. El sistema aconsegueix traduir intencions de negoci complexes a consultes SQL de dialecte MySQL, executar-les de forma aïllada i rebre'n els resultats sense comprometre l'aïllament de la capa de persistència. Finalment, l'orquestrador completa el flux generant una explicació textual que sintetitza els descobriments analítics amb precisió.

7.2. ESTRUCTURACIÓ I NORMALITZACIÓ DE RESPOSTES

Un assoliment destacable de l'arquitectura és la normalització de les sortides del sistema. Independentment de la complexitat o naturalesa de la pregunta inicial, l'API consolida la informació en càrregues útils (payloads) JSON estandarditzades. Aquesta encapsulació assegura que tant la instrucció SQL validada, com els registres obtinguts de la base de dades i l'explicació textual de l'agent, es transmetin de forma estructurada cap a la capa de presentació de Streamlit, tal com s'ha il·lustrat prèviament a la figura 7. Aquesta integració completa permet que el client web interpreti i mostri els resultats de forma immediata, eliminant de manera definitiva la distància tecnològica entre la consulta tècnica relacional i la visualització conceptual per a l'usuari final.

7.3. DISCUSSIÓ DE RESULTATS I CONTEXTUALITZACIÓ TECNOLÒGICA

L'anàlisi dels resultats obtinguts demostra que el sistema desenvolupat supera les barreres d'accessibilitat pre-

sents en les solucions d'intel·ligència de negoci tradicionals. Si comparem aquesta prova de concepte amb realitats anàlogues del sector, com ara els taulers rígids de Power BI esmentats en l'estat de l'art, la interacció basada en llenguatge natural redueix dràsticament la corba d'aprenentatge per a perfils financers no tècnics.

A diferència de les solucions comercials SaaS actuals (com Amnic o FinOpsly), que depenen de l'enviament de dades a models externs (com OpenAI), els resultats d'aquesta PoC certifiquen que és possible obtenir respostes analítiques d'alta qualitat mantenint el confinament total de la informació íntegrament dins del perímetre corporatiu.

Des de la perspectiva de l'enginyeria de programari, s'ha justificat que la construcció d'una arquitectura d'aquesta complexitat és plenament assolible comptant amb els recursos i coneixements propis d'un enginyer informàtic. L'èxit d'aquest desenvolupament recau en l'orquestració eficient de tecnologies de codi obert madures (FastAPI, Docker, MySQL) i la democratització dels models de llenguatge oberts (Phi-4-mini). La integració d'aquests components mitjançant patrons de disseny de microserveis ha permès abstraure la complexitat dels models fundacionals, requerint exclusivament capacitat computacional local, un esforç de disseny lògic i un domini de les tècniques de seguretat (RBAC i filtres lèxics o guarda-raïls).

7.4. AVALUACIÓ DE RENDIMENT I IMPACTE DEL RAG

Per quantificar empíricament les millores de disseny implementades a la prova de concepte, s'ha dissenyat un entorn de proves centrat en l'avaluació de la latència, l'ús de memòria contextual i la precisió del model generatiu Phi-4-mini durant la traducció de llenguatge natural a SQL. Totes les mètriques s'han extret sobre una estació de treball equipada amb un processador AMD Ryzen 7 5700G de 8 nuclis i 16 fils d'execució, 32 GB de memòria RAM i una targeta gràfica NVIDIA RTX 3060 de 12 GB de VRAM, amb Windows 11 com a sistema operatiu base i utilitzant el subsistema WSL 2 per executar Ubuntu 26.04 LTS. El detall exhaustiu de la configuració del model, el conjunt de dades avaluat i l'algorisme dels escenaris d'estrès es troba documentat a l'Apèndix A.5.

Aquest entorn permet establir una línia base de rendiment màxim. No obstant això, el desplegament a producció dins d'ITnow s'efectuarà sobre equips portàtils corporatius de l'estàndard Lenovo ThinkPad, equipats amb processadors mòbils de la sèrie AMD Ryzen 7 sense acceleració gràfica dedicada. En conseqüència, l'anàlisi de les execucions basades exclusivament en processament per unitat central reflecteix de forma directa les limitacions del maquinari objectiu.

7.4.1. EFICIÈNCIA COMPUTACIONAL I COMPORTAMENT DE LA MEMÒRIA CAU

L'avaluació de l'impacte base demostra que el mòdul de recuperació resol l'optimització del context d'entrada. L'orquestrador d'injecció dinàmica redueix la càrrega del text

a menys de la meitat respecte a l'esquema estàndard complet, passant de 2.581 a 1.236 tòquens (veure figura 8). Aquesta reducció manté una estabilitat rigorosa en la sortida, generant de manera constant entre 22 i 30 tòquens. Aquesta mètrica confirma l'eficàcia de les instruccions del sistema per forçar l'agent a emetre exclusivament codi executable, evitant respostes narratives.

A nivell de latència, els resultats evidencien el contrast de rendiment computacional i la gestió de la memòria cau.

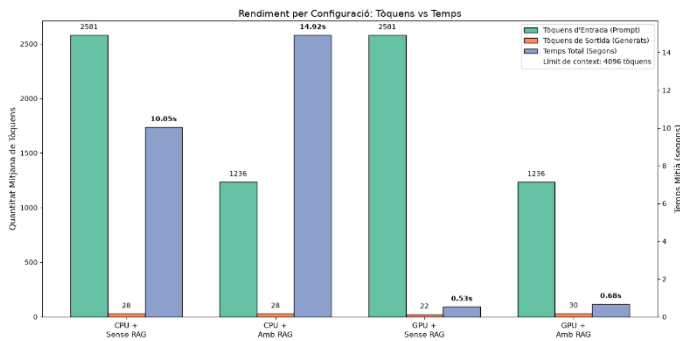


Fig. 8. Comparativa de rendiment base i latència de la generació SQL.

En execució assistida per GPU, l'arquitectura d'injecció estàtica (Sense RAG) registra un temps de 0,53 segons, mentre que la implementació Amb RAG requereix 0,68 segons. En els escenaris restringits exclusivament a CPU, la latència estructural incrementa dràsticament, marcant 10,05 segons sense el filtre actiu i 14,92 segons utilitzant el RAG.

Aquest increment temporal en les execucions dinàmiques (una variació de 0,15 segons en GPU i de 4,87 segons en CPU) respon de forma directa al tractament de la memòria cau del motor generatiu. Una injecció estàtica de l'esquema complet permet al model fixar la informació a la memòria un cop superada la càrrega inicial, accelerant les peticions consecutives. En canvi, el sistema dinàmic altera el text d'entrada en cada petició per carregar exclusivament les taules demanades per l'usuari. Aquest comportament impedeix la reutilització completa de la memòria cau i força l'avaluació de nous tòquens. Aquest cost computacional afegit és estrictament obligatori per garantir l'exactitud semàntica i evitar al·lucinacions estructurals derivades de l'excés de context.

7.4.2. AVALUACIÓ D'ESCALABILITAT I LÍMITS D'ARQUITECTURA

Per validar la viabilitat de la solució davant del creixement de la base de dades corporativa, s'ha executat una prova d'estrès incrementant l'inventari artificialment des de 5 fins a 50 taules. Per alinear-se amb els recursos limitats del maquinari objectiu d'ITnow, s'ha fixat de manera estricta un límit de memòria contextual de 6.500 tòquens.

Les dades de la figura 9 il·lustren el comportament insostenible del model d'injecció estàtica. Sense el filtratge previ, el consum de memòria presenta un creixement completament lineal, escalant des dels 2.769 tòquens (amb només 5 taules) fins a col·lapsar l'arquitectura. La saturació progressiva del context degrada el mecanisme d'atenció

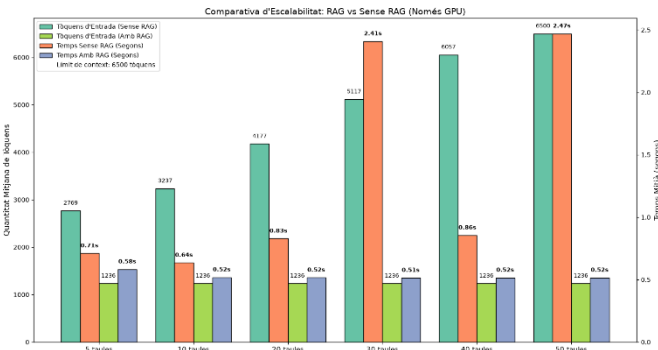


Fig. 9. Avaluació d'escalabilitat del model davant l'increment de volum de l'esquema.

del model i provoca temps de resposta impredecibles. La latència arriba a triplicar-se fins als 2,41 segons a la càrrega de 30 taules a causa de la confusió estructural de l'agent. El sistema assoleix el seu límit d'arquitectura exacte en arribar a les 50 taules: la injecció completa requereix 6.997 tòquens, xifra que desborda el paràmetre de 6.500 permèspels motor. Aquest desbordament causa el truncament de la informació d'entrada i un col·lapse absolut de la generació amb una taxa d'error funcional del 100%.

D'altra banda, l'arquitectura dinàmica assistida pel mòdul de recuperació exhibeix una estabilitat lineal perfecta. L'algorisme de cerca injecta exclusivament les definicions sol·licitades. Independentment de la presència de 5 o 50 taules al catàleg, el consum de tòquens d'entrada es manté ancorat a la franja dels 1.236 tòquens, garantint temps d'inferència estables d'aproximadament mig segon.

Aquests resultats confirmen empíricament que l'escalabilitat del motor analític no requereix inversions incrementals en capacitat computacional per processar estructures de dades gegantines. El disseny modular implementat garanteix un consum operatiu sostingut independentment de l'expansió del catàleg de l'empresa.

8 PROVES I VALIDACIÓ

Per justificar la fiabilitat, la precisió i la robustesa de la solució tecnològica, s'ha dissenyat i executat un protocol de validació exhaustiu. Aquest protocol avalua tant l'exactitud semàntica del motor d'intel·ligència com la immunitat de l'arquitectura davant de vulnerabilitats de seguretat.

8.1. AUTOMATITZACIÓ DE PROVES I VERIFICACIÓ DE COMPONENTS

D'acord amb les pràctiques estàndard de l'enginyeria de programari, el nucli de la validació es fonamenta en una bateria de proves automatitzades amb una alta cobertura de codi. S'han desenvolupat conjunts de proves unitàries i d'integració que verifiquen el comportament esperat de cadascuna de les funcions i endpoints crítics del sistema. Aquest conjunt de proves garanteix que la lògica de generació de tòquens, la recuperació de context mitjançant RAG, el filtratge de comandes i l'encaminament intern de dades funcionin correctament en diversos escenaris, actuant com una xarxa de seguretat per identificar regressions

durant el cicle de desenvolupament iteratiu.

8.2. VALIDACIÓ FUNCIONAL DE L'AGENT FINOPS

Paral·lelament a les proves de codi, s'ha avaluat empíricament la capacitat del model Phi-4-mini per operar de forma robusta sobre l'esquema en estrella (star schema) de la base de dades corporativa. La validació s'ha basat en l'enviament de consultes de complexitat variable a través de la interfície d'orquestració:

- Consultes Descriptives (Nivell 1): Proves destinades a la recuperació de dades estructurals bàsiques i llistats simples. El model ha traduït eficaçment el llenguatge natural a projeccions senzilles de selecció, demostrant que pot associar els termes de negoci amb les columnes correctes de l'inventari.
- Consultes d'Agregació Complexa (Nivell 2): Proves que exigeixen operacions de filtratge múltiple i agrupació de dades (ús de clàusules GROUP BY, WHERE i COUNT). S'ha verificat la capacitat de l'agent per segmentar els recursos segons l'entorn de treball lògic, la regió de desplegament o el proveïdor de serveis.
- Consultes Relacionals (Nivell 3): Proves orientades a validar l'encreuament de dades entre múltiples taules (JOIN). S'ha avaluat l'habilitat del model per extreure els costos agregats (cloud_resource_cost_fact) associats als atributs físics de l'inventari i segmentats per franges temporals (date_dim), resolent amb èxit escenaris avançats d'auditoria FinOps sense requerir la intervenció d'un enginyer de dades.

8.3. VALIDACIÓ DE LA SEGURETAT I DEFENSA EN PROFUNDITAT

L'avaluació de l'arquitectura de seguretat s'ha dividit en dos escenaris d'estrès per certificar el compliment dels requisits de privacitat i protecció corporativa:

1. Auditoria del Control d'Accés (JWT): S'ha sotmès el sistema a proves d'accés no autoritzat sobre l'endpoint intern de comunicació amb la base de dades. En ometre el toquen de sessió o en subministrar credencials manipulades o caducades, el protocol RBAC ha interceptat la sol·licitud correctament, retornant els codis d'estat HTTP 401 corresponents i bloquejant qualsevol interacció amb el motor de dades.
2. Auditoria dels Guarda-raïls Lèxics: Per validar l'eficàcia del filtre preventiu, s'ha simulat un vector d'atac injectant peticions de manipulació d'estat i dades en llenguatge natural. L'algorisme de filtratge ha reconegut correctament les ordres no permeses (com ara UPDATE, DELETE o DROP), impedit l'execució de qualsevol sentència que no correspongués estrictament a una operació de lectura pura (SELECT) i generant les excepcions lògiques esperades abans de la connexió. Aquesta bateria de

proves certifica la resiliència del sistema davant de possibles al·lucinacions estructurals del model generatiu o manipulacions malicioses.

9 FUTURES MILLORES

Assolits els objectius de la prova de concepte (PoC), s'identifiquen diverses línies estratègiques per evolucionar el sistema cap a un entorn de producció plenament operatiu. En l'àmbit de la integració i la seguretat, el pas més immediat és la connexió directa a les rèpliques de dades reals d'ITnow. Aquesta transició s'acompanyarà d'una capa de Defensa en Profunditat mitjançant la creació d'un usuari MySQL estrictament de lectura (READONLY), bloquejant de manera nativa qualsevol ordre de modificació davant d'una hipotètica fallada dels filtres lògics del backend.

A nivell de manteniment i rendiment, es proposa el descobriment dinàmic de l'esquema mitjançant un mòdul d'extracció automàtica de metadades. Això permetrà actualitzar l'índex del RAG de manera autònoma per gestionar canvis estructurals a la base de dades sense intervenció en el codi. Paral·lelament, es planteja la implementació d'una catxé semàntica amb una base de dades en memòria (com Redis) per emmagatzemar i resoldre consultes recurrents de manera gairebé instantània, minimitzant així la càrrega computacional del model LLM.

Finalment, per millorar l'experiència d'usuari, el sistema hauria de transicionar d'un processament de peticions aïllades cap a la incorporació de memòria conversacional. Dotar el model d'un context de sessió continuat permetria a l'operador realitzar consultes de seguiment complexes de manera natural, completant l'evolució cap a un autèntic agent analític avançat.

10 LIMITACIONS I CONSIDERACIONS ECONÒMIQUES

La principal limitació d'aquesta arquitectura completament aïllada és la seva estricta dependència del maquinari amfitrió. Per garantir temps d'inferència òptims en producció, es recomana l'ús d'acceleració gràfica (GPU). En el cas d'operar exclusivament amb CPU, es requereix una arquitectura mínima de vuit nuclis físics per evitar la degradació del rendiment; una limitació que en aquesta PoC s'ha mitigat parcialment gràcies a l'ús d'un model lleuger (Phi-4-mini) i a la injecció eficient del context (RAG).

Des d'una perspectiva financera, aquesta solució de codi obert transforma el paradigma comercial. Al no dependre d'APIs externes ni de llicències SaaS de tercers, el cost operatiu recurrent (OpEx) s'elimina completament. L'esforç econòmic es concentra de manera exclusiva en les despeses de capital inicials (CapEx) corresponents a l'enginyeria, consolidant una eina de des, finançament sostenible i pressupostàriament previsible per a ITnow.

11 CONCLUSIÓ

La realització d'aquest Treball de Final de Grau demostra de manera empírica la viabilitat tècnica de dissenyar i desenvolupar una prova de concepte (PoC)

analítica basada en un sistema conversacional completament local. Mitjançant la implementació d'una arquitectura modular, s'ha comprovat que és possible mitigar la rigidesa dels sistemes tradicionals d'intel·ligència de negoci i apropar l'accés a dades d'infraestructura cloud a perfils corporatius no tècnics. L'aportació principal d'aquest treball destaca per oferir una solució que unifica l'accessibilitat operativa i la confidencialitat: s'ha assolit un flux de treball que opera sota un control absolut de la dada, preservant la privacitat de l'organització sense renunciar a les capacitats de raonament dels models fundacionals moderns.

Pel que fa al disseny de la infraestructura, s'ha complert amb èxit la consolidació d'una xarxa virtualitzada mitjançant Docker, on coexisteixen els nodes de persistència, intel·ligència i orquestració de manera aïllada. El confinament total de la informació s'ha garantit gràcies a la integració d'un control d'accés basat en rols (RBAC) gestionat a través de tòquens JWT. Aquest disseny blindat el motor de base de dades, ja que el node d'intel·ligència actua de manera cega respecte als connectors físics, delegant a l'API del backend la funció de filtre preventiu, capaç d'interceptar qualsevol ordre destructiva i autoritzar únicament instruccions de lectura.

En relació amb la lògica de l'agent, s'ha complert la fita de traduir intencions de negoci a codi SQL executable sense incórrer en al·lucinacions semàntiques estructurals, utilitzant una arquitectura seqüencial de dues fases. L'objectiu d'escalabilitat i precisió s'ha assolit plenament amb la integració d'un sistema de Generació Augmentada per Recuperació (RAG), el qual injecta dinàmicament l'esquema de dades rellevant. Aquesta estratègia, combinada amb instruccions de sistema (System Prompts) estrictes, assegura que el model no només extreu la informació de l'inventari i dels costos de manera precisa, sinó que postprocessa els resultats financers en format JSON per oferir explicacions textuais fonamentades i comprensibles per a l'operador humà.

Finalment, la validació end-to-end s'ha materialitzat en una interfície gràfica a Streamlit, unificant autenticació i visualització en un únic espai de treball. En definitiva, aquest treball avala el valor de l'enginyeria de programari aplicada a la IA dins d'ITnow, oferint una alternativa previsible, lliure de costos operatius recurrents i preparada per escalar orgànicament com a eina corporativa.

AGRAÏMENTS

Vull expressar el meu agraïment a totes les persones que m'han donat suport al llarg d'aquest treball. Especialment, al meu tutor universitari, Miguel Carpio, i a tot l'equip d'ITnow, per haver-me ajudat sempre amb una orientació professional.

ÚS DE LA IA GENERATIVA

L'ús de la intel·ligència artificial generativa en aquest treball es divideix en dues parts clarament diferenciades: la IA com a component central de la solució tecnològica i la IA com a eina d'assistència en el desenvolupament del projecte.

Pel que fa a la solució tecnològica, la intel·ligència artificial constitueix el motor principal de la PoC mitjançant l'ús de LLM locals gestionats amb Ollama. Aquesta tecnologia permet traduir les consultes dels usuaris en llenguatge natural a codi SQL de forma segura, utilitzant tècniques de prompt engineering i RAG per garantir que les respostes sobre costos i objectes cloud siguin precises.

D'altra banda, la IA generativa s'ha usat activament com a eina de suport durant el procés de realització del projecte. Ha servit d'assistència tècnica en el desenvolupament del codi per al backend i el frontend, així com de suport en la redacció d'aquesta memòria, facilitant una comunicació més clara de les decisions tècniques adoptades.

BIBLIOGRAFIA

- [1] Microsoft, "Informació general sobre FinOps," Microsoft Learn, 2024. [En línia]. Disponible a: <https://learn.microsoft.com/es-es/cloud-computing/finops/overview> [Últim accés: 2-abril-2026]
- [2] Microsoft, "Power BI: Visualització de dades interactiva," Microsoft Power Platform, 2026. [En línia]. Disponible a: <https://www.microsoft.com/es-es/power-platform/products/power-bi> [Últim accés: 4-abril-2026]
- [3] ITNow, "Coneix-nos: Qui som i què fem," ITNow, 2026. [En línia]. Disponible a: <https://www.itnow.es/es/conocenos> [Últim accés: 6-abril-2026]
- [4] Amazon Web Services, "¿Qué son los modelos de lenguaje de gran tamaño (LLM)?," AWS Research, 2026. [En línia]. Disponible a: <https://aws.amazon.com/es/what-is/large-language-model/> [Últim accés: 7-abril-2026]
- [5] Ollama, "Ollama is now available as an official Docker image," Ollama Blog, 2024. [En línia]. Disponible a: <https://ollama.com/blog/ollama-is-now-available-as-an-official-docker-image> [Últim accés: 1-abril-2026]
- [6] Red Hat, "¿Qué es Docker?," Red Hat Topics, 2026. [En línia]. Disponible a: <https://www.redhat.com/es/topics/containers/what-is-docker> [Últim accés: 3-abril-2026]
- [7] Raiola Networks, "¿Qué es FastAPI de Python?," Raiola Networks Ayuda, 2026. [En línia]. Disponible a: <https://raiola-networks.com/ayuda/que-es-fast-api-python/> [Últim accés: 4-abril-2026]
- [8] Streamlit, "The fastest way to build and share data apps," Streamlit, 2026. [En línia]. Disponible a: <https://streamlit.io/> [Últim accés: 2-abril-2026]
- [9] Asana, "Guía de la prueba de concepto (PoC)," Asana Resources, 2024. [En línia]. Disponible a: <https://asana.com/es/resources/proof-of-concept> [Últim accés: 2-abril-2026]
- [10] Amazon Web Services, "¿Qué es MySQL?," AWS, 2026. [En línia]. Disponible a: <https://aws.amazon.com/es/rds/mysql/what-is-mysql/> [Últim accés: 7-abril-2026]
- [11] JWT.io, "Introduction to JSON Web Tokens," JWT.io, 2024. [En línia]. Disponible a: <https://www.jwt.io/introduction#when-to-use-json-web-tokens> [Últim accés: 2-abril-2026]
- [12] IBM, "What is Agentic RAG?," IBM Think, 2024. [En línia]. Disponible a: <https://www.ibm.com/es-es/think/topics/agentic-rag> [Últim accés: 3-abril-2026]
- [13] IBM, "¿Qué es el prompt engineering?," IBM Think, 2026. [En línia]. Disponible a: <https://www.ibm.com/es-es/think/topics/prompt-engineering> [Últim accés: 7-abril-2026]
- [14] Microsoft Azure, "¿Qué es SaaS? Software como servicio," Diccionario de informática en la nube, 2026. [En línia]. Disponible a: <https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-saas> [Últim accés: 7-abril-2026]
- [15] Amnic, "SaaS FinOps Infrastructure Management," Amnic, 2026. [En línia]. Disponible a: <https://amnic.com/> [Últim accés: 18-abril-2026]
- [16] FinOpsly, "Budgeting and Forecasting for Cloud Costs," FinOpsly, 2026. [En línia]. Disponible a: <https://www.finopsly.com/product/budgeting-and-forecasting> [Últim accés: 18-abril-2026]
- [17] Amazon Web Services, "¿Qué es una API? - Explicación de interfaces de programación de aplicaciones," AWS, 2026. [En línia]. Disponible a: <https://aws.amazon.com/what-is/api/> [Últim accés: 5-abril-2026]
- [18] OpenAI, "Acerca de OpenAI," OpenAI, 2026. [En línia]. Disponible a: <https://openai.com/es-419/about/> [Últim accés: 4-abril-2026]
- [19] Atlassian, "¿Qué es la metodología Agile?," Atlassian Agile Guide, 2024. [En línia]. Disponible a: <https://www.atlassian.com/es/agile> [Últim accés: 1-abril-2026]
- [20] Atlassian, "¿Qué es Scrum?," Atlassian Agile Coach, 2026. [En línia]. Disponible a: <https://www.atlassian.com/es/agile/scrum> [Últim accés: 1-abril-2026]
- [21] Atlassian, "Guía de los tableros Kanban," Atlassian Agile Guide, 2024. [En línia]. Disponible a: <https://www.atlassian.com/es/agile/kanban/boards> [Últim accés: 4-abril-2026]
- [22] Atlassian, "Software de seguimiento de proyectos y tareas | Jira," Atlassian, 2026. [En línia]. Disponible a: <https://www.atlassian.com/es/software/jira> [Últim accés: 21-maig-2026]
- [23] Kubernetes, "¿Qué es Kubernetes?," Kubernetes Documentation, 2026. [En línia]. Disponible a: <https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/> [Últim accés: 15-maig-2026]
- [24] Atlassian, "Scrumban: domina dos metodologías ágiles," Atlassian Agile Coach, 2026. [En línia]. Disponible a: <https://www.atlassian.com/es/agile/project-management/scrumban> [Últim accés: 15-maig-2026]
- [25] DBEaver, "CloudBeaver Community deployment from docker image," GitHub, 2026. [En línia]. Disponible a: <https://github.com/dbbeaver/cloudbeaver/wiki/CloudBeaver-Community-deployment-from-docker-image> [Últim accés: 12-maig-2026]
- [26] Model Context Protocol, "Introduction: What is the Model Context Protocol?," MCP Documentation, 2026. [En línia]. Disponible a: <https://modelcontextprotocol.io/docs/getting-started/intro> [Últim accés: 7-abril-2026]
- [27] IBM, "¿Qué es una API de REST?," IBM Topics, 2026. [En línia]. Disponible a: <https://www.ibm.com/es-es/topics/rest-apis> [Últim accés: 19-maig-2026]
- [28] IBM, "¿Qué es el control de acceso basado en roles (RBAC)?," IBM Think, 2026. [En línia]. Disponible a: <https://www.ibm.com/es-es/think/topics/rbac> [Últim accés: 12-maig-2026]
- [29] SmartBear, "Swagger UI," Swagger.io, 2026. [En línia]. Disponible a: <https://swagger.io/tools/swagger-ui/> [Últim accés: 23-maig-2026]
- [30] Cloudflare, "¿Qué es la seguridad Zero Trust?," Cloudflare Centro de aprendizaje, 2026. [En línia]. Disponible a: <https://www.cloudflare.com/es-es/learning/security/glossary/what-is-zero-trust/> [Últim accés: 23-maig-2026]
- [31] Ollama, "Phi 4-mini," Ollama Library, 2026. [En línia]. Disponible a: <https://ollama.com/library/phi4-mini> [Últim accés: 20-maig-2026]

APÈNDIX

A1. PLANIFICACIÓ DEL DESENVOLUPAMENT

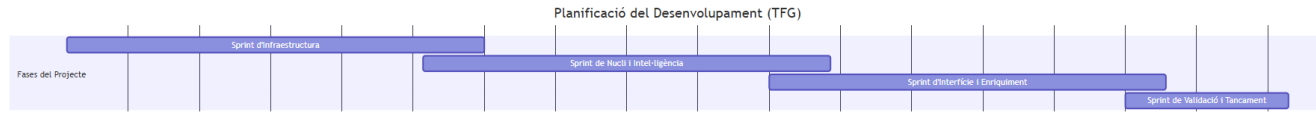


Fig. 1: Diagrama de Gantt dividit en els quatre sprints.

A2. CATÀLEG EXHAUSTIU D'ENDPOINTS DE L'API (USUARIS I ROLS)

La documentació interactiva a Swagger UI detalla l'esquema de rutes complet per a la governança d'identitats sota el model RBAC. Com es mostra a la figura 2, els mòduls d'usuaris (Users) i rols (Role) s'han implementat seguint un disseny RESTful simètric de cinc rutes basat en operacions CRUD (Create, Read, Update, Delete).

Aquest catàleg inclou rutes generals (GET i POST) per a la indexació i creació de registres, així com rutes parametritzades per identificador únic (GET, PUT i DELETE) per a la consulta específica, modificació d'estat i esborrat d'entitats, actuant com l'evidència tècnica d'una gestió de permisos completa i totalment acoblada a la verificació de tòquens JWT.

Health	
GET	/health Health Backend
Login	
POST	/login/access-token Login
Users	
GET	/users Get Users
POST	/users Create User
GET	/users/{user_id} Get User By ID
PUT	/users/{user_id} Update User
DELETE	/users/{user_id} Delete User
PUT	/users/{user_id}/deactivate Deactivate User
PUT	/users/{user_id}/activate Activate User
Roles	
GET	/roles Get Roles
POST	/roles Create Role
GET	/roles/{role_id} Get Role By ID
PUT	/roles/{role_id} Update Role
DELETE	/roles/{role_id} Delete Role

Fig. 2. Vista expandida de la documentació a Swagger UI on es detallen els mètodes HTTP i les rutes RESTful que conformen el cicle de vida operatiu de la gestió d'usuaris i rols sota el model RBAC.

A3. IMPLEMENTACIÓ DEL FILTRE LÈXIC (GUARDARAÏLS)

La seguretat preventiva de l'arquitectura es fonamenta en la intercepció i validació de les respostes generades pel model d'intel·ligència abans de la seva execució a la base de dades. En el següent fragment de codi (corresponent al mòdul guardrails.py), es detalla la implementació lògica d'aquest tallafocs intern (veure figura 3).

A més d'implementar el patró de cerca basat en expressions regulars per forçar exclusivament instruccions SELECT, l'algorisme inclou una fase de sanejament (sanitization) fonamental: identifica i elimina dinàmicament les etiquetes de format Markdown (``) que els models de llenguatge gran (LLMs) acostumen a incloure per defecte en la generació de codi.

```
backend > finops_backend > services > fin > guardrails.py
1 import re
2
3
4 _SELECT_START_PATTERN = re.compile(r"^\s*select\b", flags=re.IGNORECASE)
5
6
7 def validate_question(question: str) -> str:
8     question = question.strip()
9     if not question:
10         raise ValueError("La pregunta no puede estar vacia.")
11     return question
12
13
14 def validate_sql(raw: str) -> str:
15     sql = raw.strip()
16     if sql.startswith("```"):
17         lines = sql.splitlines()
18         sql = "\n".join(line for line in lines if not line.startswith("```"))
19         sql = sql.strip()
20
21     if not _SELECT_START_PATTERN.match(sql):
22         raise ValueError(f"La respuesta del modelo no es un SELECT valido:\n{raw}")
23
24     return sql.rstrip().rstrip(";") + ";"
```

Fig. 3. Codi del mòdul Guardarails.py.

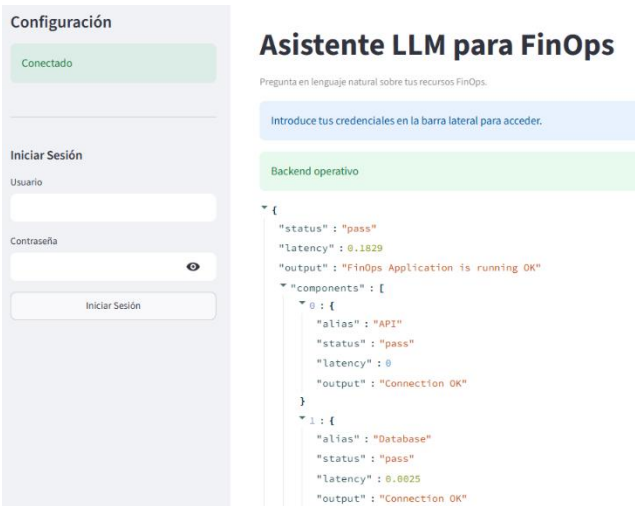


Fig. 4. Detall de la interfície d'usuari amb accés restringit, requerint la validació de credencials a la barra lateral abans d'exposar l'assistent de llenguatge natural.

A4. ESTATS D'AUTENTICACIÓ I CONTROL EN LA INTERFÍCIE D'USUARI

Per garantir que l'accés a l'agent conversacional es realitza sota un marc estrictament protegit, l'aplicació gràfica gestiona de manera asíncrona l'estat de la sessió de l'usuari. Com es pot observar a la figura 4, el sistema manté un estat bloquejat per defecte que impedeix qualsevol interacció amb el bloc d'intel·ligència artificial, instant l'operador a introduir les seves credencials en el panell lateral de l'entorn de treball.

Un cop completat el formulari d'inici de sessió de forma reeixida, la interfície actualitza dinàmicament el seu estat per confirmar la connexió activa amb l'orquestrador central, habilitant el camp de consulta analítica i mostrant el perfil d'usuari elidat per al flux de treball (veure figura 5).



Fig. 5. Vista de la barra lateral de control un cop completat el procés d'autenticació amb èxit, confirmant el perfil d'operador actiu dins de la plataforma.

A5. METODOLOGIA DE LES PROVES DE RENDIMENT I BENCHMARKS

Per tal de quantificar l'impacte arquitectònic del sistema de recuperació d'informació i avaluar el rendiment de la plataforma presentat anteriorment, es va dissenyar una instrumentació de proves basada en scripts d'avaluació automatitzats. Totes les execucions es van dur a terme interactuant amb el node local d'Ollama mitjançant els paràmetres fixats a l'arxiu d'entorn corporatiu. Específicament, es va utilitzar el model d'inferència phi4-mini, limitant la finestra de context a un màxim de 6.500 tòquens i restringint la predicció de sortida a 512 tòquens. Addicionalment, l'entorn es va configurar amb un temps d'espera màxim de 240 segons i es va forçar l'ús de 8 fils de processament per garantir l'estabilitat i la reproductibilitat de les mètriques extretes.

Pel que fa a les dades de prova, la validació semàntica qualitativa del sistema es va basar en una bateria completa de vint consultes en llenguatge natural. Tot i això, per a l'extracció de les mètriques gràfiques de latència i escalabilitat computacional, l'algorisme d'avaluació es va programar per iterar exclusivament sobre un subconjunt format per les cinc primeres consultes. Aquesta reducció del volum d'entrada es va aplicar de forma programàtica per evitar la saturació tèrmica de la màquina local durant els bucles d'estrès, assegurant així mesures de temps fidedignes. Les cinc consultes avaluades, mantingudes en l'idioma original d'operació de l'agent, van ser les següents:

1. ¿Cuántos recursos hay por entorno?
2. ¿Cuántos recursos activos hay por cuenta?
3. ¿Qué recursos pertenecen al proveedor Micro-

soft Azure?

4. ¿Cuál es el nombre y el equipo responsable del recurso 'bucket-logs-01'?
5. ¿Cuál es el coste acumulado por proveedor?

La captura de dades es va orquestrar mitjançant dos scripts de desenvolupament específics (`benchmark_rag.py` i `benchmark_tables.py`) que registren automàticament el consum de tòquens d'entrada i sortida, així com els temps de cicle intern. El primer script, dissenyat per a l'avaluació de l'eficiència computacional, itera la bateria de preguntes forçant l'alternança entre els motors d'execució de la CPU i la GPU, tant aplicant la injecció completa com el filtratge dinàmic del context. D'altra banda, la prova d'escalabilitat s'executa a través d'un segon programa que modifica dinàmicament l'inventari injectant des de cinc fins a cinquanta taules simulades (dummy tables). Aquesta estratègia d'estrès estructural permet perfilar el comportament asimptòtic del sistema analític i determinar els límits de desbordament del model d'una manera totalment aïllada i empírica.