UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU D'ENGINYERIA INFORMÀICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

# Xatbot per a l'Oficina Tècnica de Projectes i Serveis Docents: aplicació de LLM i RAG per al suport al Personal Docent

Autor: Pau Segura Baños

Director: Dr. Eloi Puertas i Prats
Realitzat: Departament de Matemàtiques i Informàtica
Barcelona, 10 de juny de 2025

# Resum

Aquest projecte se centra en el desenvolupament d'un xatbot dedicat al programa RIMDA de l'Oficina Tècnica de Projectes i Serveis Docents de la Universitat de Barcelona. El xatbot es va desenvolupar per oferir una resposta ràpida als dubtes tècnics del professorat de la Universitat a l'hora de presentar projectes d'innovació docent, o presentar-se com a grup d'innovació docent.

El xatbot està construït amb la combinació de la tècnica RAG i un LLM. El xatbot recupera la informació dels documents interns de l'oficina i construeix la seva resposta un cop detecta la consulta de l'usuari.

# Resumen

Este proyecto se basa en el desarrollo de un chatbot dedicado al programa RIMDA de la "*Oficina Tècnica de Projectes i Serveis Docents*" de la *Universitat de Barcelona*. El chatbot se ha desarrollado para ofrecer una respuesta rápida a las dudas técnicas del profesorado de la universidad a la hora de presentar proyectos de innovación docente, o presentarse como un grupo de innovación docente.

El chatbot está construido con la combinación de la técnica RAG y un LLM. El chatbot recupera la información de los documentos internos de la oficina y construye su respuesta una vez detecta la consulta del usuario.

# Abstract

This project is based about the development of a chatbot dedicated on RIMDA program of the "*Oficina Tècnica de Projectes i Serveis Docents*" of "*Universitat de Barcelona*". The chatbot has been developed to provide quick answers to technical doubts of the university teaching staff when presenting teaching innovation projects or presenting as teaching innovation groups.

This chatbot is built using the combination of RAG technique and a LLM. The chatbot retrieves its answers from the documental database and it builds the answer once it detects the user's query.

# Agraïments

M'agradaria expressar els meus agraïments a diverses persones que m'han acompanyat, no només durant el procés de desenvolupament de Treball de Final de Grau, sinó durant tot el transcurs de la carrera.

Per començar, agraeixo profundament al Dr. Eloi Puertas per la seva predisposició a ajudar-me i guiar-me durant el transcurs del projecte. El seu acompanyament m'ha ajudat a saber què fer i per on guiar el treball. També li agraeixo que em presentés aquesta oportunitat amb la beca de col·laboració, ja que ha estat una etapa molt profitosa per mi.

També vull agrair a la meva mare Montse, la meva germana Andrea i el meu pare Josep Maria, per acompanyar-me sempre, sense posar-me pressió i confiant en mi. Sense el seu suport no hauria arribat fins aquí.

Per continuar, vull expressar la meva gratitud als meus amics Oscar, Adrià, David, Jan, Pau i Aïxa i a tots els meus amics del barri, per sempre preocupar-se per mi, quan més ho necessitava i animar-me a tirar endavant quan les coses durant els primers anys de carrera eren més complicades.

Per finalitzar, vull agrair als membres de l'OTPSD de la UB. Els vull agrair la llibertat i confiança que han dipositat en mi. Al Jordi vull agrair-li la complicitat i les converses que hem tingut. A la Sònia vull agrair-li la preocupació que ha tingut per com estava jo personalment en tot moment. A la Mila li vull agrair els bons consells que m'ha donat. A tots els vull agrair les facilitats que han disposat perquè la meva estança entre ells fos còmoda i amena. No només m'emporto un gran record d'aquests mesos de la meva beca, sinó grans converses sobre pedagogia i una dosi gran de coneixement, ganes de continuar aprenent i bones estones.

# Índex

# 1 Introducció

Aquest projecte consisteix en el desenvolupament d'una aplicació web. Aquesta aplicació web és un xatbot per l'Oficina Tècnica de Projectes i Serveis Docents de la Universitat de Barcelona. El xatbot està dissenyat per proporcionar als docents de la universitat respostes ràpides i automàtiques a qüestions tècniques. Per al seu funcionament, s'ha implementat una combinació de tècniques basades en Retrieval-Augmented Generation (RAG)[1] i models de llenguatge gran (LLM)[1], que permeten millorar la precisió i contextualització de les respostes.

Gràcies als coneixements adquirits durant els anys del grau, he aconseguit crear una eina capaç de resoldre el problema inicial i fer una solució pràctica i disponible pels docents.

## 1.1 Context

La integritat d'aquest treball s'ha fet durant la meva beca de col·laboració amb la mencionada Oficina Tècnica de Projectes i Serveis Docents de la Universitat de Barcelona.

La OTPSD és l'encarregada del Programa RIMDA, àrea en la qual jo he centrat la meva aplicació. El Programa de Recerca i Innovació per la Millora de la Docència i l'Aprenentatge (RIMDA) del Vicerectorat de Política Docent desenvolupa les línies estratègiques del Vicerectorat sobre promoció i reconeixement de la millora de la docència i dels aprenentatges mitjançat convocatòries de projectes i grups d'innovació. Ofereix suport al PDI en la seva tasca constant de revisió de l'enfocament docent orientat a l'assoliment per part de l'alumnat d'un aprenentatge profund. Durant els primers dies de la beca, els membres de l'oficina em van fer una ràpida immersió en el seu dia a dia i em van explicar certs problemes en els quals, jo com a enginyer informàtic podria ajudar-los a optimitzar la seva activitat.

Durant els períodes de convocatòria de propostes d'innovació docent, l'OTPSD es veu desbordada per un gran volum de consultes procedents de docents. Aquestes consultes, relacionades amb dubtes tècnics sobre les bases i els requisits de la convocatòria, arriben per diferents canals com trucades telefòniques o correus electrònics. Aquesta demanda obliga els membres de l'oficina a destinar gran part del seu temps a tasques de resolució de dubtes repetitius i de baixa complexitat, amb el qual assumeixen més un rol de gestors i servei d'atenció que no de pedagogs pendents d'ajudar a enfocar correctament les propostes. Això dificulta que puguin dedicar temps i atenció a aquells equips realment interessats a desenvolupar projectes de qualitat, i acaba afectant negativament tant l'eficiència de l'oficina com a l'impacte dels projectes que finalment es presenten.

És per això que vam decidir centrar els meus esforços en el desenvolupament del xatbot, ja que la correcta implementació d'aquesta eina els facilitarà molt la problemàtica exposada en el paràgraf anterior.

## 1.2 Objectius

Els objectius es poden separar en objectius personals i objectius sobre el projecte.

### Objectius generals

1. Aplicar tots els aprenentatges adquirits durant els anys de la carrera
2. Ser capaç de desenvolupar un projecte complet de manera autònoma, assumint totes les responsabilitats des de la definició inicial fins a la seva implementació i avaluació final, passant per la planificació, l'anàlisi i el disseny.
3. Desenvolupar un xatbot que es faci servir durant anys pel professorat de la UB.
4. Aprofundir de forma significativa en els meus coneixements sobre les tecnologies específiques escollides per desenvolupar el xatbot, com ara entorns de desenvolupament, biblioteques i frameworks.
5. Aprendre i familiaritzar-me amb l'ús de RAG i LLM, ja que mai he treballat amb aquestes eines.

### Objectius específics

1. Fer una interfície visual entenedora i fàcil d'utilitzar per a tots els usuaris.
2. Integrar correctament tots els microserveis necessaris per al funcionament complet del xatbot.
3. Desenvolupar un sistema fàcil de mantenir, tenint en compte que els documents que alimenten el model poden canviar i hauran d'actualitzar-se en el futur.
4. Permetre que el xatbot doni respostes contextualitzades i fiables mitjançant la combinació de RAG i LLM, minimitzant el risc de respostes errònies o inventades.
5. Establir un sistema de derivació o escalat cap a personal humà en cas que la consulta superi la capacitat del xatbot.

## 1.3 Estructura de la memòria

La memòria s'estructura seguint una lògica progressiva que facilita la comprensió del projecte i del seu desenvolupament. La redacció parteix d'un enfocament general que es va concretant a mesura que s'aprofundeix en el detall tècnic.

En primer lloc, s'introdueix el context general del projecte, es presenten els conceptes previs necessaris i es revisa l'estat de l'art per situar-lo dins el marc actual del desenvolupament. A continuació, es descriu la planificació inicial, així com l'anàlisi de requisits i els casos d'ús identificats.

Un cop establertes les bases, es presenta el disseny i l'arquitectura general de l'aplicació. A partir del capítol d'implementació s'entra en un nivell de detall més específic, on es descriuen els components individuals, les tecnologies utilitzades i les decisions adoptades durant el desenvolupament.

Finalment, els capítols dedicats a proves i validació justifiquen les decisions tècniques amb resultats, i es tanquen amb una anàlisi de resultats finals i les conclusions.

# 2 Conceptes Previs

En aquesta secció explicaré de manera profunda alguns dels conceptes més importants que seran mencionats durant tota la memòria.

### Large Language Models (LLM):

Els LLM [2] són una classe de models entrenats sobre enormes quantitats de dades per dotar-lo de la capacitat de resoldre multitud de tasques. Els LLM estan entrenats per millorar les capacitats de comprensió del llenguatge natural (CLN) [3] i processament del llenguatge natural (PLN) [3]. Els LLM són un model d'IA generativa [4] dissenyada a comprendre i generar text com un ésser humà. Tenen la capacitat d'inferir des del context, generar respostes coherents, traduir, resumir text i respondre preguntes.

Els LLM funcionen aprofitant tècniques de *deep learning* [5] i grans quantitats de dades textuals. Els LLM consten de diverses capes de xarxes neuronals i transformadors. Gràcies al seu entrenament el model és capaç de predir la següent paraula més probable depenent del context. El model ho aconsegueix atribuint una puntuació de probabilitat a la recurrència de paraules. Aquest procés implica entrenar l'LLM en un corpus massiu de text, amb el qual aprèn semàntica, gramàtica i relacions conceptuals.

### Tokenització i Embeddings

Un pas clau en el funcionament dels LLM és la tokenització del text d'entrada. Aquesta tècnica consisteix a dividir el text en unitats més petites anomenades tokens [6], que poden ser paraules senceres, fragments de paraula o caràcters individuals, segons el model utilitzat. Aquesta fragmentació permet tractar textos complexos de forma més flexible i detallada.

Un cop obtinguts els tokens, aquests es transformen en embeddings [7], vectors numèrics en un espai multidimensional. Cada embedding representa un token i conté informació sobre el seu significat semàntic, la seva funció gramatical i el context en què s'utilitza habitualment.

Aquests vectors s'organitzen en un espai tal que paraules o expressions amb significats similars es troben a prop les unes de les altres. Per exemple, les paraules "rei" i "reina" estaran més a prop en aquest espai que "rei" i "dona", com es pot veure a la figura 1 Això permet que el model pugui "entendre" relacions conceptuals i fer inferències.

Els embeddings constitueixen una mena de mapa semàntic intern per al model, que el guia en el procés de predicció i generació de text. Aquesta representació és essencial perquè el model pugui manejar sinònims i polisèmia.

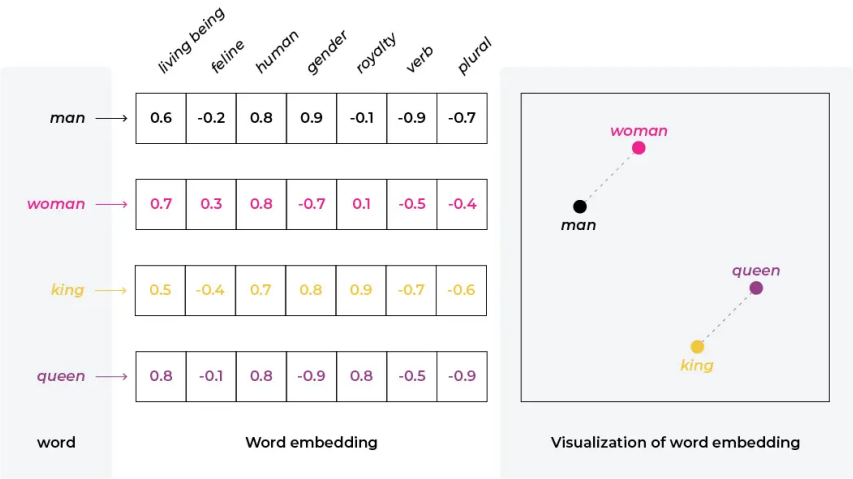


Figura 1: Espai vectorial d'embeddings. Es veu com les paraules "man", "woman", "king" i "queen" són classificades i les seves distàncies en l'espai vectorial

En aquesta imatge es veu com es classifiquen les paraules d'una manera molt simplificada. La classificació real es fa amb milions de paràmetres que ubiquen cadascuna de les paraules en un espai vectorial.

### Prompts i el Prompt Engineering

Els prompts [8] són les instruccions o entrades que es proporcionen a un model per generar una resposta. Un prompt pot ser tan senzill com una pregunta ("Què és la fotosíntesi?") o tan complex com un paràgraf amb indicacions estilístiques específiques. La seva formulació és clau per obtenir resultats útils i rellevants.

La qualitat i precisió de la resposta generada depèn en gran manera de com està formulat el prompt. Aquesta realitat ha donat lloc a una nova disciplina: el prompt engineering, o enginyeria de prompts. Aquesta pràctica consisteix a dissenyar i provar formulacions de prompts per millorar les respostes dels LLM.

Per exemple, un prompt com "Explica'm la fotosíntesi com si fos un conte per a nens" donarà una resposta molt diferent a "Fes-me un resum científic de la fotosíntesi". El model adapta el seu registre, la profunditat i la forma segons les instruccions rebudes.

Els enginyers de prompts sovint utilitzen tècniques com:

- Few-shot learning: oferir exemples dins del prompt perquè el model aprengui la dinàmica de la tasca.
- Chain-of-thought prompting: demanar al model que expliqui els passos del seu raonament.
- In-context learning: incloure context addicional dins del prompt per guiar millor la resposta.

### Retrieval Augmented Generation (RAG)

Els Large Language Models (LLM), tot i la seva capacitat per generar text coherent i contextual, poden ser inconsistents i generar respostes errònies o inexactes. Aquestes limitacions es deuen, en gran part, al fet que els LLM estan entrenats únicament amb dades prèvies, que poden quedar desactualitzades o no incloure informació específica o detallada sobre determinats temes. A més, els LLM no tenen accés directe a fonts de coneixement en temps real, i tampoc "entenen" la informació en el sentit humà, sinó que es basen en patrons estadístics.

Per fer front a aquestes limitacions, ha sorgit la tècnica del Retrieval-Augmented Generation (RAG) [9], una arquitectura híbrida que combina els avantatges del raonament generatiu dels LLM amb la capacitat de consulta d'un sistema de recuperació d'informació.

El funcionament del RAG es divideix en dues fases clau: recuperació i generació.

- **Fase de recuperació:** Quan l'usuari introdueix una consulta, el sistema utilitza mecanismes de cerca semàntica per identificar fragments de documents, bases de dades o fonts de coneixement que puguin contenir informació rellevant. Aquesta cerca es fa mitjançant la comparació d'embeddings, tant de la pregunta com dels continguts disponibles. S'utilitza un índex vectorial, com FAISS, per accelerar la recuperació dels documents més semblants semànticament.
- **Fase de generació:** Un cop s'han recuperat els documents rellevants, aquests es concatenen i s'insereixen dins del prompt que es passarà a l'LLM. Aquest nou prompt proporciona al model accés a informació actualitzada i específica, i li permet generar una resposta més precisa i contextualitzada.

Aquesta arquitectura té múltiples avantatges com l'actualització del coneixement del sistema sense tornar a entrenar el model o l'augment de la traçabilitat, ja que permet mostrar a l'usuari d'on extreu la informació.

### Base de dades vectorial:

Les bases de dades vectorials [10] són bases de dades dissenyades per emmagatzemar i gestionar embeddings. Aquest tipus de bases de dades s'utilitzen principalment per millorar la recerca i la recuperació d'informació en aplicacions d'intel·ligència artificial, com la cerca d'imatges similars, cerques semàntiques o el suport a models d'IA generativa. A més, contribueixen a reduir errors i millorar la qualitat i la coherència de les respostes generades pels models d'intel·ligència artificial. Entre els avantatges que ofereixen, destaca la capacitat de realitzar cerques ràpides i precises dins grans volums de dades, la gestió eficient de dades no estructurades i la seva escalabilitat, tolerància a fallades i seguretat pròpies d'una base de dades moderna. Aquestes característiques fan que les bases de dades vectorials siguin una eina clau per al desenvolupament d'aplicacions avançades basades en intel·ligència artificial.

**Base de dades documental:**

Les bases de dades de documents [11] són un tipus de base de dades NoSQL que emmagatzema la informació en documents flexibles, normalment en format JSON o similar. Aquests documents contenen dades en forma de parells clau-valor i poden incloure estructures complexes com arrays o documents incrustats.

Les bases de dades de documents disposen d'un esquema flexible que permet emmagatzemar documents amb camps diferents dins d'una mateixa col·lecció, cosa que facilita l'adaptació als canvis i noves necessitats sense necessitat de definir un esquema fix com a les bases de dades SQL. Les operacions habituals, com consultes i actualitzacions, són més ràpides i senzilles, ja que no calen unions complexes entre taules, la qual cosa millora el rendiment. A més, utilitzen formats com JSON, que són lleugers, llegibles i fàcils d'intercanviar entre diferents aplicacions.

# 3 Estat de l'art

En aquest apartat parlaré dels tipus de xatbots que han existit, d'alguns dels xatbots existents en l'àmbit educatius, així com de les tecnologies que els fan possibles.

## 3.1 Tipus de xatbots

Per començar a parlar de xatbots [12] hem de descriure que és exactament un xatbot. Un xatbot és un programa dissenyat per simular una conversa amb usuaris humans. Tenen diverses utilitats com entreteniment, xatbots d'ajuda, educatius o fins i tot assistents personals virtuals.

A mesura de la implementació de la intel·ligència artificial [13] en el mercat i l'evolució d'aquesta, els xatbots han evolucionat juntament amb la IA, cada cop oferint més possibilitats.

### Xatbots Pre-LLM

Els xatbots previs a la implementació dels LLM eren xatbots basats en regles. [14] Els xatbots reconeixen certes paraules claus en l'*input* de l'usuari, i responen amb respostes predefinides en una base de dades. El coneixement general d'aquests xatbots era molt reduït, però l'específic era molt acurat. Molts xatbots pre-LLM no permetien l'entrada lliure de text, sinó, que mitjançant botons, l'usuari formulava les preguntes ja predefinides. Tot i les seves limitacions, eren útils en entorns molt controlats on les consultes dels usuaris es podien anticipar fàcilment.

Un dels xatbots més coneguts de l'època era ALICE [15], un xatbot conversacional basat en regles. No comptava amb intel·ligència artificial, però sí amb una gran quantitat de patrons i respostes escrites a mà.

Simsimi [16] és un altre exemple destacat de xatbot conversacional preLLM. En aquest cas, i a diferència d'ALICE, Simsimi aprenia de les respostes de tots els usuaris mitjançant xarxes neuronals convolucionals.

### Xatbots LLM

Gràcies a l'evolució de l'aprenentatge automàtic i, en particular, a l'aparició dels Large Language Models (LLM) (vegeu apartat 2), els xatbots han experimentat una transformació radical. Aquests nous sistemes es basen en arquitectures de xarxes neuronals profundes [17], com els transformadors [18], i han estat entrenats amb enormes quantitats de dades textuals. Aquesta formació els permet reconèixer patrons, estructures lingüístiques i significats contextuals, generant respostes més coherents, naturals i adaptades a l'entrada de l'usuari.

Una de les millores més destacades dels xatbots amb LLM és la seva capacitat per gestionar entrada lliure de text i comprendre consultes complexes, fins i tot quan no estan formulades de manera òptima.

Un dels problemes més habituals és que, a vegades, poden generar respostes errònies o inexactes. Això es deu al fet que aquests models no entenen la informació com ho faria una persona, sinó que generen respostes en funció de patrons estadístics apresos durant l'entrenament. A més, també és possible que el xatbot desviï la conversa cap a temes no relacionats amb la consulta inicial de l'usuari, especialment si la petició és ambigua o poc clara. Aquest comportament afecta negativament l'experiència de l'usuari.

Per afrontar aquesta limitació, recentment ha guanyat protagonisme l'enfocament conegut com a RAG (vegeu apartat 2). Així, el model no es limita al coneixement intern après durant l'entrenament, sinó que pot accedir a informació més actualitzada i específica del domini. L'ús de RAG millora la precisió de les respostes, redueix la probabilitat de derivacions irrellevants i ofereix una experiència més fiable i contextualitzada per a l'usuari.

## 3.2 Xatbots en l'àmbit educatiu

### Pounce

Pounce [19] és el xatbot de la universitat de Wisconsin - Milwaukee. Està dissenyat per donar suport als estudiants, especialment als de nou accés, responent preguntes freqüents relacionades amb el procés d'admissió, la gestió dels expedients acadèmics, la facturació i l'ajuda financera. Durant l'ús de Pounce el sistema ofereix possibles suggeriments de preguntes a fer-li, i també mostra les fonts internes d'on treu la informació.

El fet que ens mostri les fonts internes indica que Pounce funciona amb un RAG, i les seves respostes es construeixen gràcies als documents.

### LOLA

LOLA [20] és el xatbot desenvolupat per la Universitat de Múrcia (UMU) amb l'objectiu de guiar i assistir als futurs estudiants de la UMU i de la Universitat Politècnica de Cartagena en tot el procés de preinscripció i matrícula. El projecte ha estat impulsat pel Servei d'Informació Universitària (SIU), en col·laboració amb l'Àrea de Tecnologies de la Informació i Comunicacions Aplicades (ATICA) i l'empresa 1MillionBot.

Aquest sistema d'intel·ligència artificial utilitza la tecnologia *Dialogflow* de Google, que permet al xatbot aprendre a mesura que interactua amb els usuaris, millorant així la seva capacitat de resposta. Lola ha estat capaç de resoldre el 90% de les consultes rebudes, principalment relacionades amb notes d'EBAU, revisions d'exàmens, notes de tall i tràmits de preinscripció.

El seu ús permet automatitzar gran part de l'atenció informativa que anteriorment es gestionava manualment, com ho demostren les més de 28.000 consultes ateses pel SIU l'any anterior. Lola està disponible a través de la web del SIU i de l'aplicació mòbil DURM, amb accés 24/7.

# 4 El projecte

En aquesta secció, explicaré tot el procés de creació del xatbot, des de la planificació inicial fins a la implementació, passant per l'anàlisi dels requisits i funcionalitats, el procés de disseny i presa de decisions en l'àmbit de software.

## 4.1 Planificació

En iniciar el projecte, vaig planificar fer-ho seguint una metodologia Agile Scrum. La idea inicial era estructurar el treball en sprints d'un mes de durada cadascun, amb un total de tres sprints previstos per completar les funcionalitats principals del projecte.

Després de cada un d'aquests sprints, es refinaven les User Stories, amb l'objectiu d'ajustar els requisits.

A cada sprint feia una selecció d'unes 5-6 històries d'usuari de les quals les dividia en tasques més petites.

A continuació presento el diagrama de Gantt, un diagrama que mostra la distribució temporal de les diferents tasques durant el procés de realització del TFG. Les tasques que es veuen en el meu diagrama són èpiques. Aquestes èpiques descriuen en trets generals el que ha sigut tot el procés d'anàlisi, disseny i implementació del xatbot, incloent-hi el deploy i la redacció de la memòria.

Com es pot veure a les figures 2 i 3 observem com la gran majoria de tasques s'han anat succeint en el temps. Això vol dir que la planificació inicial s'ha seguit i no s'ha hagut de treballar en vàries coses a la vegada. Sí que és cert que sí que coincideixen la redacció de la memòria amb altres tasques, però no s'interferien i s'han pogut fer de manera paral·lela i no depenent.

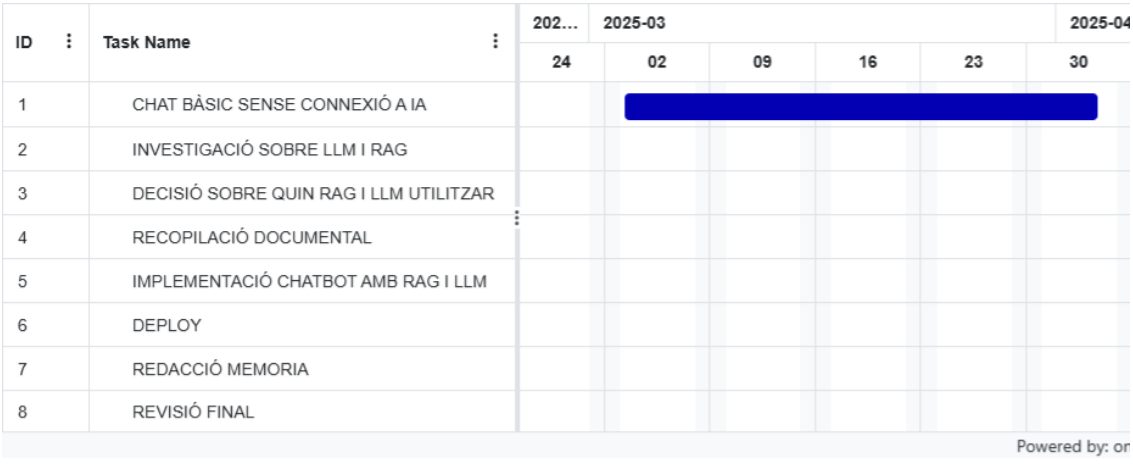


Figura 2: Primera part del diagrama de Gantt. Es poden observar les diferents èpiques en les quals he separat tot el procés de desenvolupament, i es veu una graella temporal que va del març al juny.

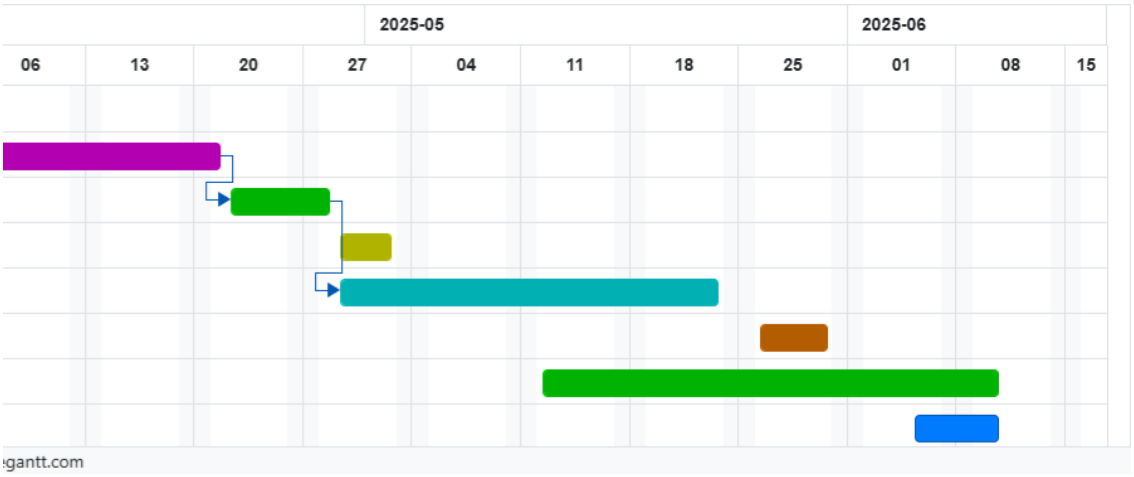


Figura 3: Continuació del diagrama de Gantt començat a la figura 2. En aquesta figura es veuen les tasques escrites a la primera part. En ordre: Investigació sobre LLM i RAG, Decisió sobre quin LLM i RAG utilitzar, Recopilació documental, Implementació de xatbot amb RAG i LLM, Deploy, Redacció de la memòria i Revisió final.

## 4.2 Anàlisis

En aquest apartat descriuré tant els requisits funcionals com els no funcionals i així com un diagrama dels casos d'ús del xatbot.

### 4.2.1 Requisits funcionals

Són les accions que el software ha de ser capaç de realitzar per satisfer les necessitats de l'usuari.

Durant l'especificació dels requisits funcionals es mencionen diverses històries d'usuari. Per veure-les, vegeu l'Annex A.

En ser un desenvolupament enfocat a resoldre una problemàtica molt específica, el disseny de les primeres històries d'usuari i els seus requisits funcionals va ser en cooperació amb els membres de l'oficina. Els membres de l'oficina em van descriure els escenaris davant els quals es trobaven en les èpoques de convocatòries de propostes i jo vaig transformar aquests escenaris en històries d'usuari, que prèviament van ser aprovades pels membres de l'oficina.

Els requisits funcionals del xatbot són els següents:

- **Enviament de preguntes via xat**: el sistema ha de permetre a qualsevol usuari (PDI) enviar una pregunta a través d'una interfície de xat. *(Derivat de US1)*
- **Resposta coherent i en temps real**: el sistema ha de respondre de manera coherent, clara i en temps real. *(Derivat de US1)*

- **Base documental interna**: el sistema ha de disposar d'una base de dades documental interna amb els documents de l'OTPSD. *(Derivat de US2, US13, US14, US16, US17, US18)*
- **Generació RAG**: el sistema ha de generar les respostes utilitzant tècniques RAG sobre aquests documents. *(Derivat de US2, US13, US14, US16, US17, US18)*
- **Emmagatzematge d'historial**: el sistema ha de guardar automàticament totes les converses dels usuaris. *(Derivat de US3, US4)*
- **Accés a l'historial**: l'usuari ha de poder accedir a l'historial de converses des del xat. *(Derivat de US3, US4)*
- **Ordenació per data**: les converses s'han de mostrar ordenades per data, mostrant primer les més recents. *(Derivat de US3, US4)*
- **Reobrir converses**: l'usuari ha de poder reobrir una conversa anterior i continuar-la en el punt on es va deixar. *(Derivat de US3, US4)*
- **Manteniment de context**: el sistema ha de poder mantenir el context d'una conversa en curs i usar-lo per respondre preguntes que no facin referència explícita al tema. *(Derivat de US13, US16, US17, US18)*
- **Canvi de context**: l'usuari ha de poder canviar de context dins la mateixa conversa especificant el nou tema. *(Derivat de US13, US16, US17, US18)*
- **Adaptació al tipus de projecte**: el sistema ha de detectar i adaptar les respostes en funció del tipus de projecte (PMID, RSU, Digitalització, GID, etc.). *(Derivat de US13, US16, US17, US18)*
- **Preguntes sobre formularis**: el sistema ha de permetre que el PDI pregunti per camps específics del formulari. *(Derivat de US7)*
- **Explicacions de camps**: el sistema ha de proporcionar una explicació detallada del camp en qüestió. *(Derivat de US7)*
- **Identificació de límits del sistema**: el sistema ha d'identificar quan una pregunta supera el seu abast o requereix valoració humana. *(Derivat de US9)*
- **Recomanació de derivació**: en aquests casos, ha de recomanar contactar amb l'oficina. *(Derivat de US9)*
- **Respostes en català**: el xat ha de contestar en català, ja que és l'idioma institucional de la universitat.

### 4.2.2 Requisits no funcionals

són les característiques no relacionades amb les funcions del sistema, però igualment són importants per la correcta experiència d'ús.

Els requisits no funcionals són els següents:

- **Rendiment**: el sistema ha de proporcionar respostes amb un temps de resposta ràpid i adequat per garantir una experiència d'usuari fluida i sense interrupcions.
- **Disponibilitat**: el servei ha d'estar operatiu les 24 hores del dia.
- **Escalabilitat**: el xat ha de ser capaç de créixer en nombre d'usuaris i adaptar-se a les noves funcionalitats.
- **Usabilitat**: la interfície ha de fàcil d'utilitzar i intuïtiva per a tota mena d'usuaris.
- **Sostenibilitat**: el codi ha de ser fàcil de mantenir i els documents han de ser fàcilment actualitzables. Els treballadors de l'oficina han de ser capaços d'introduir o modificar documents en cas de canvi de bases i/o normatives.

### 4.2.3 Casos d'ús

Com es pot veure a les figures 4 i 5, aquests són els casos d'ús del xatbot.

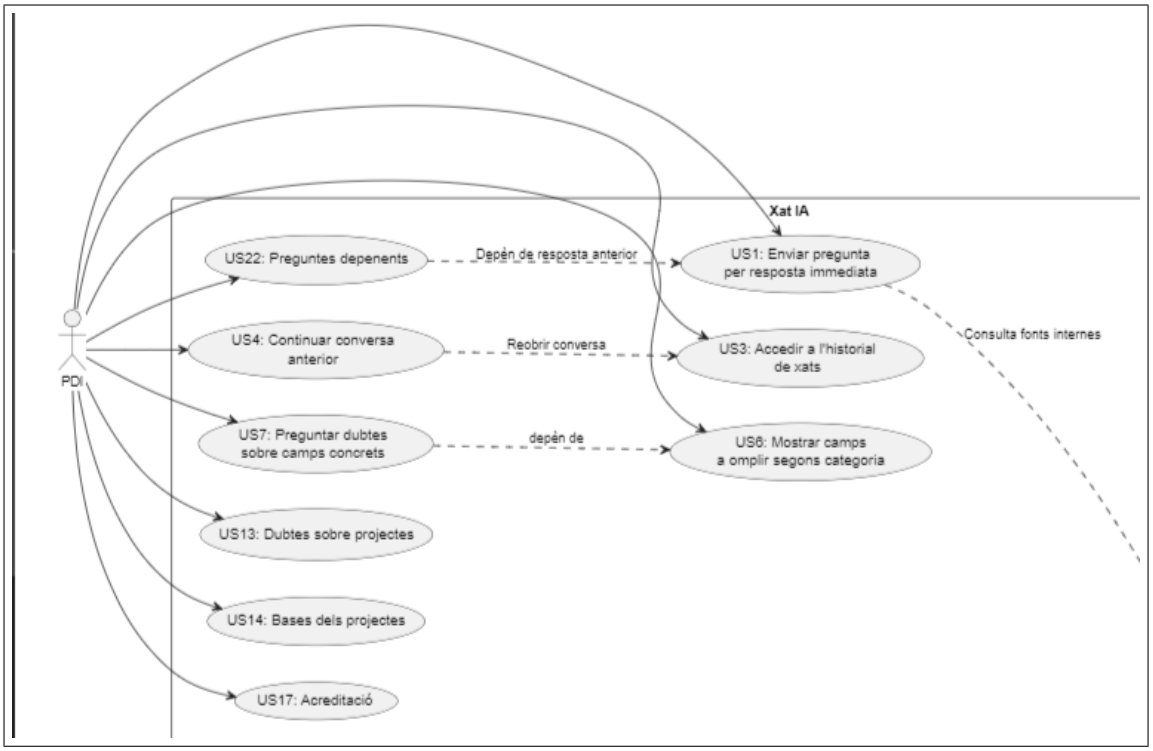


Figura 4: Primera part de diagrama de casos d'ús. L'actor principal és el PDI i pot fer les històries US1, 3,4,6,7,13,14,17 i 22. A la segona part es veurà la continuació de la US1

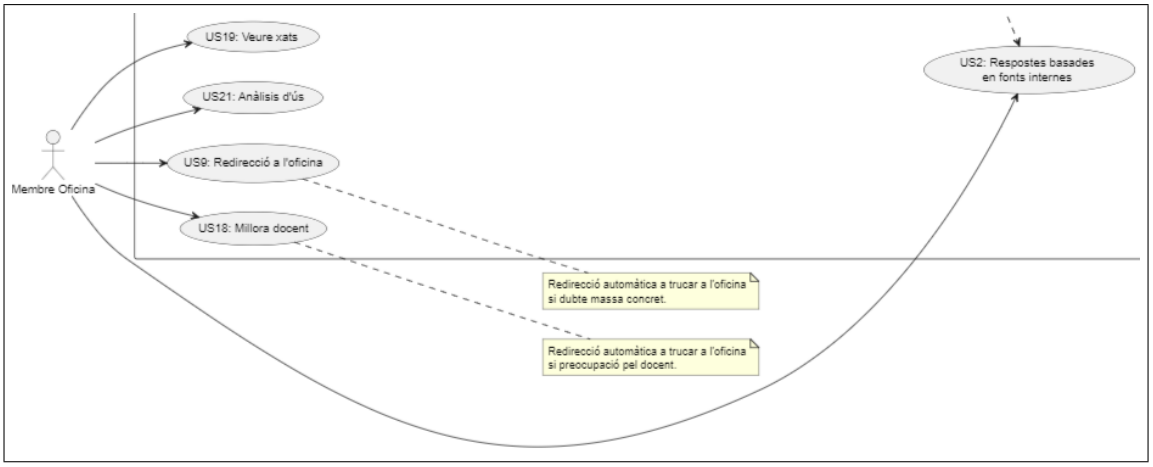


Figura 5: Segona part del diagrama de casos d'ús: Es veuen les històries que pot fer l'actor "Membre de l'oficina", que són les US19, 21, 9 i 18

Els casos d'ús tenen dos actors principals:

- **PDI**: són el Personal Docent Investigador, és a dir els professors i professores de la Universitat de Barcelona, que poden fer propostes de projectes d'innovació.

- **Membres de l'oficina**: són els membres de la OTPSD. Són els encarregats de les gestions de les convocatòries i acreditacions.

Els PDI són els principals actors del xatbot. És una aplicació dissenyada per a ells i són qui tenen més accions i casos d'ús assignats. Els PDI han de capaços de resoldre els seus dubtes amb el xatbot. El xatbot ha de poder respondre qüestions tècniques sobre les convocatòries dels grups i projectes d'innovació i l'acreditació com a docent. La construcció de les respostes del xat es basa segons una base de dades documental, on hi ha recollides les bases i normatives de les convocatòries. També pot sol·licitar ajuda sobre la compleció dels formularis a la web per tal d'optar a aquestes convocatòries. El xat també ha de ser capaç de detectar automàticament el context sobre el qual està parlant l'usuari, així com de detectar canvis d'aquest context.

D'altra banda, els membres de l'oficina, qui tindran un rol d'administradors i podran veure els xats dels altres usuaris. També podran veure un rànquing de les preguntes més repetides, ordenades pel document que fan referència.

## 4.3 Disseny

En aquest apartat descriuré el disseny de l'aplicació, que inclou l'arquitectura general, les tecnologies utilitzades en la creació del software, el flux de funcionament i les decisions de disseny.

### 4.3.1 Arquitectura

El xatbot es basa en una arquitectura modular de tipus *client-servidor*, amb diversos microserveis especialitzats que treballen de manera coordinada. El sistema està format per:

**Frontend:** és la part amb la qual els usuaris interactuen per fer funcionar l'aplicació. Està fet amb Vue.js.

**Backend:** El llenguatge de programació amb el que està fet és *Python*. Per la creació de l'API he implementat FastAPI, un framework especialitzat en el ràpid desenvolupament d'API REST. És la connexió entre el frontend i els diferents microserveis. Bàsicament, el backend està dissenyat per gestionar de manera eficient les peticions dels usuaris a través de endpoints.

**Base de dades documental MongoDB:** MongoDB s'utilitza per emmagatzemar dades de manera persistent. Les dades que es desen són les converses de tots els usuaris, els documents per formar la base de coneixements interns i també una col·lecció per mantenir un registre de les consultes dels usuaris per document.

**Base de dades vectorial Qdrant:** s'utilitza per indexar i recuperar documents rellevants a partir de vectors embeddings. És essencial per al sistema RAG (Retrieval-Augmented Generation).

**OpenAI:** es fa servir l'API de models per detectar el context a partir de prompts.

Per fer-la servir necessitem un compte d'OpenAI, amb l'API Key i crèdits per poder fer les peticions a l'API.

Quan vaig començar el projecte, tenia molt clar quines opcions de backend i frontend volia fer servir. Ràpidament, vaig optar per triar dos entorns amb els quals ja estava familiaritzat, com Vue i FastAPI amb Python. Era important fer-ho d'aquesta manera, ja que el repte era aprendre les noves tècniques com el RAG i les bases de dades vectorials, o la integració del LLM amb Langchain (Vegeu apartat 4.4.5).

**Quina base de dades documental utilitzar?**

Per la base de dades documental vaig optar per `MongoDB`. Vaig plantejar-me altres alternatives com CouchDB, però vaig decidir-me per MongoDB per la meva experiència utilitzant-ho, i les fàcils opcions per desplegar-ho remotament, com MongoDB Atlas.

**Quina base de dades vectorials utilitzar?**

Per triar quina base de dades vectorial utilitzar, vaig dur a terme una recerca exhaustiva, ja que era la primera vegada que treballava amb aquest tipus de tecnologia i necessitava documentar-me adequadament abans de prendre una decisió.

Després d'analitzar diverses opcions, em vaig centrar especialment en dues: FAISS i Qdrant.

FAISS està més enfocat en la cerca eficient per similitud de vectors, però no és una base de dades a l'ús, ja que no té persistència. Per altra banda, Qdrant sí que té aquesta persistència, tot i que la cerca vectorial és menys eficient. Qdrant, a més, té una integració senzilla mitjançant la seva imatge docker, i possibilitat de tenir un clúster remot per poder fer el desplegament en producció.

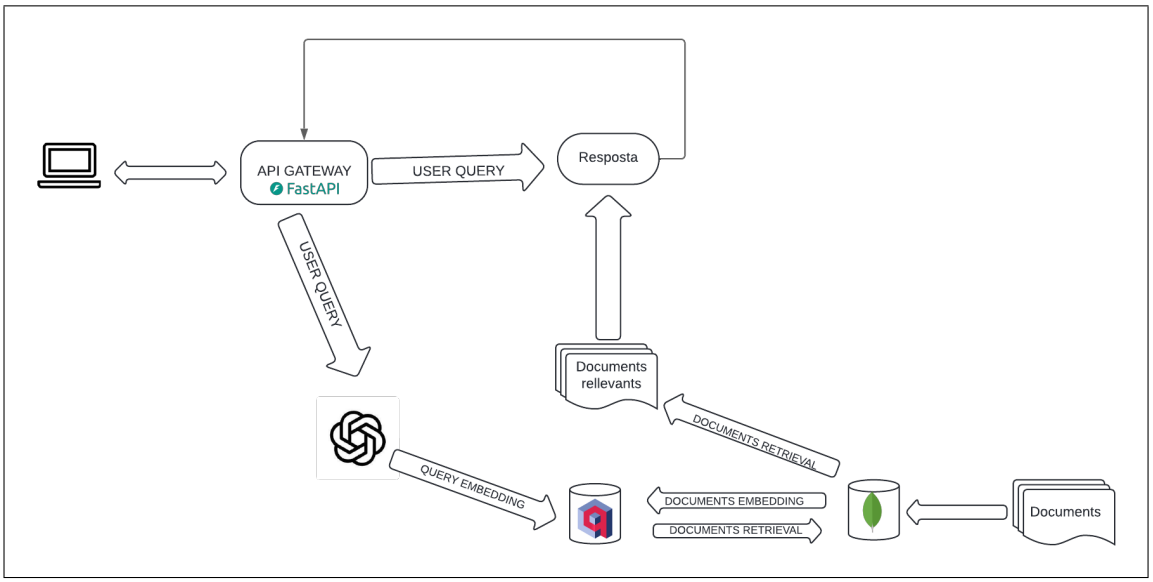


Figura 6: Diagrama d'arquitectura de software. Aquesta imatge descriu l'estructura interna i els microserveis que munten l'aplicació. Podem veure com tot comença per l'entrada de l'usuari via frontend, i podem seguir tota la seqüència de passes.

El sistema es basa en una arquitectura client-servidor. El frontend és el servei que permet la interacció amb l'usuari. FastAPI és el servei que connecta els altres microserveis. MongoDB s'usa per emmagatzemar documents i converses i Qdrant com a base de dades vectorials i per la recuperació eficient. L'API d'OpenAI és la que permet generar els embeddings i les respostes a les queries dels usuaris ajuntant el context dels documents.

### 4.3.2 Disseny de les Bases de Dades

Pel correcte funcionament del xatbot s'han implementat dues bases de dades.

**Qdrant**: Qdrant és una base de dades de tipus vectorial (vegeu apartat 2). Aquesta base de dades guarda els embeddings dels documents. Quan es guarda cada un dels embeddings, es guarden amb la mateixa id que es guarden a MongoDB.

**MongoDB**: La importància de MongoDB en el sistema és clau. És l'encarregada de guardar els xats i els documents sobre els quals es basa el RAG.

Hi ha dos tipus de documents en la base de dades:

- **Chats**: Cada document representa una sessió de conversa amb un usuari. Conté informació com l'identificador del xat i l'identificador de sessió, un array de les interaccions entre l'usuari i la resposta produïda per la IA. També es guarda l'última pregunta de l'usuari al xat, l'últim context i l'última data d'interacció.
- **Documents interns**: Cada àmbit crea una col·lecció diferent, però l'estructura dels documents de cada una de les col·leccions és igual. Són els textos base sobre els quals es construeix la resposta. Cada document conté el text original en format pregunta-resposta i un identificador únic que coincideix amb els embeddings de Qdrant. Aquests documents són textos provinents d'una base de coneixement prèvia.
- **Registre de consultes**: És una col·lecció per mantenir un registre de consultes dels usuaris. És important perquè els membres de l'oficina puguin tenir una constància de les consultes dels usuaris.

## 4.4 Implementació

En aquest apartat explicaré com he implementat tècnicament tot el plantejament previ descrit als apartats d'anàlisis i disseny.

### 4.4.1 Metodologia

Abans d'explicar la implementació i el desenvolupament del xatbot, és necessari descriure com s'ha organitzat el procés de desenvolupament.

Tal com s'ha explicat a l'apartat 4.1 s'ha implementat una metodologia agile. Cadascuna de les històries es "trencava" en tasques més petites.

Les tasques s'agrupaven segons si eren *backend*, *frontend* i *testing*. Primer desenvolupava les tasques de backend. Un cop desenvolupades, continuava amb el testing, i per acabar connectava el frontend amb el backend. Una història d'usuari no es donava per acabada fins a acabar totes les tasques.

Per mantenir un control de versions he utilitzat `Github`. El repositori remot està organitzat de la següent manera:

- Branca **main**: és la branca on van els canvis una vegada s'ha acabat un sprint.
- Branca **dev**: cada cop que acabava una tasca, els canvis es portaven a aquesta branca, i la següent tasca es creava a partir de l'últim estat de *dev*.
- Branca **USX-Nom de la tasca**: com el nom de la branca indica, es creava una branca per cada una de les tasques.

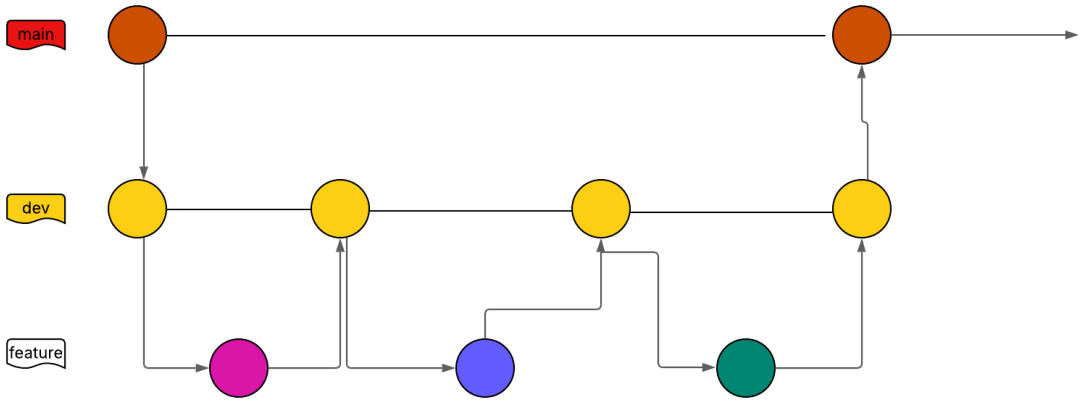


Figura 7: Estructura de branques de Github. Cadascun dels punts que surten de dev cap a feature són tasques diferents. Encara que la branca feature no existeixi en el meu repositori, és una manera de representar el flux de desenvolupament amb les diferents tasques.

### 4.4.2 Preparació de la base de coneixement

Previ a la posada en marxa de l'aplicació, s'ha d'executar l'script *rag.py*. Aquest script navega pel directori data/documents, on estan continguts els documents en format .txt endreçats per diferents carpetes. Els fitxers .txt que s'utilitzen com a base de coneixement per a l'aplicació segueixen un format estructurat per facilitar el seu processament automàtic. Cada fitxer conté parelles de preguntes i respostes, on la pregunta sempre comença amb la paraula clau PREGUNTA: seguida del text de la pregunta, i la resposta es marca amb la paraula clau RESPOSTA: seguida del text corresponent 8, extret dels documents interns de la OTPSD. L'script processa cada un dels arxius: agafa les preguntes i respostes i les desa a MongoDB. També

fa els embeddings de cada una de les preguntes i les desa Qdrant. Cadascuna de les carpetes del directori de documents és una col·lecció diferent a Qdrant i a MongoDB. Aquest script es llança automàticament i no depèn de l'acció de l'usuari.

```
PREGUNTA:
Quins beneficis aporta la participació a una proposta de digitalització

RESPOSTA:
El responsable del projecte tindrà un reconeixement de 60 hores al seu PDA.
Els membres de l'equip que desenvolupa el projecte tindran un reconeixement de 20 hores al seu PDA
```

Figura 8: Exemple d'arxiu .txt. La imatge descriu l'estructura dels arxius txt, que són la base de coneixements interna. Tots els arxius txt tenen la mateixa estructura, per aconseguir un ràpid processament quan es llença l'script.

### 4.4.3 Flux d'execució del sistema RAG

Una vegada s'ha preparat la base de coneixement per utilitzar la tècnica RAG, l'aplicació està preparada per al seu ús.

El flux de l'aplicació comença amb l'entrada d'una consulta mitjançant la interfície gràfica del xat. Una vegada l'endpoint rep aquesta petició, si és un nou xat, el sistema, mitjançant un prompt que li passem a l'LLM, decideix el context sobre el qual està parlant l'usuari. Una vegada el context està decidit, aquest s'utilitza per decidir a quina col·lecció de Qdrant s'ha de fer la cerca. Es fa la cerca a la col·lecció indicada, i si troba un resultat coincident, s'extreu la resposta del document desat a MongoDB. Si no es troba una coincidència en la col·lecció, l'LLM s'informarà l'usuari que el xat no ha entès la petició. Tant la pregunta com la resposta es desen a la col·lecció de Chats de MongoDB de manera automàtica.

Si l'usuari decideix seguir amb una conversa, i no començar una de nova, el sistema ha de decidir si mantenir el context o canviar-lo. Això ho fa mitjançant un prompt especialitzat que ens dona la resposta.

En els següents diagrames es mostra el flux del procés RAG, en cas que es recupera un document amb èxit.

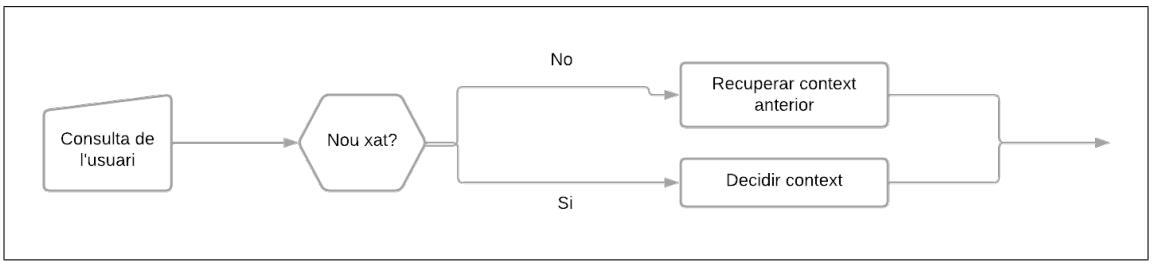


Figura 9: Flux del procés RAG. En aquesta imatge es descriu el flux pel qual passa l'aplicació d'ençà que l'usuari introdueix la seva consulta fins que rep una resposta. Comença determinant si és un nou xat o és una continuació d'un xat anterior. Si és una continuació, recupera el context anterior, i si no ho és decideix un nou context.

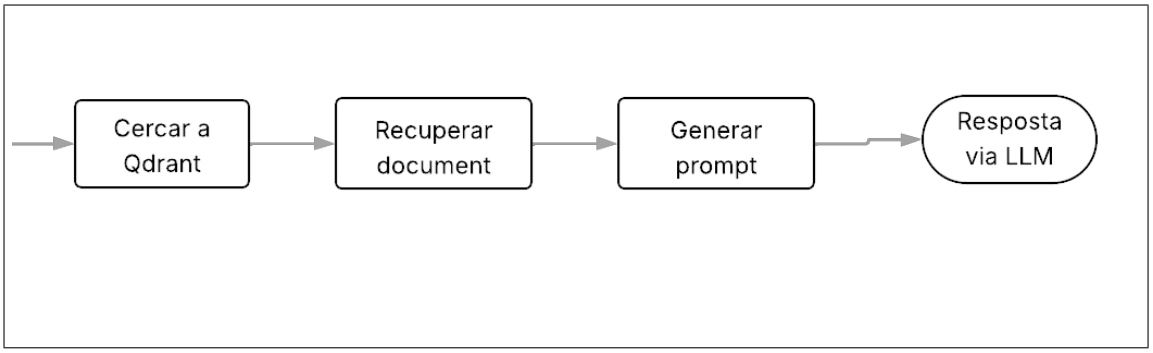


Figura 10: Continuació del flux del procés RAG. Un cop s'ha decidit el context, es busca el document més semblant a la query. Quan s'ha trobat el flux segueix quan es recupera el document de la base de dades documental. Aquesta informació rellevant s'utilitza per construir el prompt per obtenir la resposta desitjada

### 4.4.4 Connexió entre microserveis

Com ja s'ha introduït a l'apartat 4.3.1, el sistema està format per diversos microserveis que interactuen entre si. Aquesta secció descriu com interactuen i com s'han implementat.

Quan es preparen els documents base mitjançant l'script *rag.py* (Vegeu apartat 4.4.2) també es defineix la connexió entre Qdrant i MongoDB. A Qdrant es desen els embeddings de les preguntes i a MongoDB es desen tant les preguntes com les respostes en format text. Per garantir una correspondència, s'assigna la mateixa ID tant al document de la base de dades documental com a la base de dades vectorial, com podem observar a la taula 1. Això permet que, quan es fa una cerca vectorial, el backend pugui recuperar el document consultant el mateix ID.

| | **Qdrant** | **MongoDB** |
|---|---|---|
| **ID** | `b1567077-e38f-471b-941f-0962dfc21189` | `b1567077-e38f-471b-941f-0962dfc21189` |
| **Pregunta** | — | Què és un grup d'innovació? |
| **Resposta** | — | Un grup d'innovació és... |

Taula 1: Taula per comprovar la correlació de les dades entre les diferents bases de dades. A la taula es veu que malgrat no tenir els mateixos camps una base de dades i una altra, les ID sí que coincideixen per garantir

Un cop obtinguda la informació rellevant dels documents de MongoDB, es mostra la resposta per pantalla.

Tant la consulta de l'usuari com el parell pregunta-resposta del model es desen automàticament a MongoDB. Això permet mantenir un historial de converses, tant per l'usuari final com per als administradors.

### 4.4.5 Integració d'OpenAI

En la meva arquitectura, l'ús d'OpenAI és clau en dues etapes del procés.

**Generació d'embeddings**: s'utilitza el model *text-embedding-ada-002* d'OpenAI per convertir tant el text dels documents com les consultes de l'usuari en vectors numèrics.

**Ús del model per a presa de decisions i generació de respostes**: per connectar el backend FastAPI amb OpenAI, s'utilitza Langchain.

Langchain [21] és un framework de python dissenyat per facilitar la creació d'aplicacions basades en LLM. Un dels seus grans beneficis a l'hora de desenvolupar aplicacions basades en intel·ligència artificial és que permet orquestrar fluxos de treball amb diferents agents.

Amb la classe *PromptTemplate* de Langchain, creem els prompts a enviar al model *gpt-4* d'OpenAI. Els dos prompts que he fet servir son els següents:

- **Classificació de context:**

  "Llegeix el text següent i indica a quin d'aquests camps pertany: tenint en compte que digi vol dir projectes de digitalització, rsu de responsabilitat social universitària, pmid de projectes de millora i innovació docent, grups parlen de grups d'innovació docent i millora docent individual engloba tot allo que sigui millorar com a persona docent, és a dir, no com a projectes de millora si no com millorar com a professor

  Text:question

  Resposta només amb el nom d'un sol camp.

  Si no correspon a cap camp, retorna 'cap camp'."

- **Manteniment o canvi de context:**

  "Llegeix el text següent i indica a quin d'aquests camps pertany: tenint en compte que digi vol dir projectes de digitalització, rsu de responsabilitat social universitària i inclou temes d'inclusió sostenibilitat igualtat..., pmid de projectes de millora i innovació docent i grups parlen de grups d'innovació docent, , millora docent individual engloba tot allo que sigui millorar com a persona docent, és a dir, no com a projectes de millora si no com millorar com a professor tingues en compte el context anterior i la ultima pregunta i si la question no menciona cap dels camps, la teva resposta ha de ser el context passat

  Context: context

  Pregunta anterior: ultimapregunta

  Text: question

  Resposta només amb el nom d'un sol camp."

### 4.4.6 Implementació d'endpoints

Com ja s'ha mencionat prèviament, el backend està desenvolupat amb FastAPI. L'aplicació té 5 endpoints principals:

| Endpoint | Què fa? |
|---|---|
| `POST /chats/rag` | Endpoint principal del RAG. Detecta el context, comprova dependències, fa cerca semàntica a Qdrant, recupera informació de MongoDB, genera la resposta amb OpenAI i desa la conversa. |
| `GET /chats/me` | Retorna l'historial de converses de la sessió activa per poder continuar consultes anteriors. |
| `GET /chats/{chat_id}` | Retorna un xat específic per ID, per seleccionar-lo des de l'historial i continuar la conversa. |
| `POST /login` | Permet l'autenticació dels usuaris administradors. |
| `GET /ranking` | Endreça el registre de consultes segons document |

Taula 2: Endpoints i la seva funcionalitat. En aquesta taula s'observen els 5 principals endpoints que fan funcionar tota la lògica de l'aplicació

L'endpoint principal i el més important és l'endpoint de POST /chats/rag. En aquest endpoint, com es pot veure a la taula 2, és on té lloc tota la seqüència explicada a les figures 9 i 10. L'endpoint /chats/me, també és important, ja que forma part dels elements visuals de la interfície, i es crida automàticament quan s'obre aquesta. El `/chats/{chat_id}` és cridat quan triem una conversa antiga per tornar-la a llegir o continuar consultant. Aquests endpoints són importants pel correcte funcionament del xatbot de cara a la interacció amb l'usuari PDI.

Per als membres de l'oficina hi ha dos més endpoints. El /login, com el seu nom indica, serveix per iniciar sessió com a administradors. És un endpoint exclusiu per als membres de l'oficina, ja que només ells coneixen la contrasenya per iniciar sessió a l'únic compte que hi ha creat. El GET /ranking, és per una interfície exclusiva dels administradors, on es pot veure el registre ordenat de consultes de tots els usuaris.

### 4.4.7 Persistència de sessió

El xatbot ha estat desenvolupat com un sistema sense registre per als PDI. La manera de mantenir la sessió i tenir un control de les converses és mitjançant les `SessionID` lligades al *localStorage* del navegador.

Quan un usuari accedeix a l'aplicació per primera vegada, si no hi ha cap `SessionID` existent, es genera un identificador únic i s'assigna a l'usuari, emmagatzemant-lo localment al navegador. Aquest identificador s'envia com a part de cada petició cap al backend, cosa que permet identificar a quina sessió pertany cada interacció.

Cada `SessionID` està associada a una col·lecció de preguntes i respostes desades a MongoDB. Això permet mantenir l'historial de cada conversa i assegurar que, si

l'usuari tanca la pestanya o recarrega la pàgina, pot reprendre la conversa on l'havia deixada.

Un dels inconvenients d'aquest mètode de persistència és que, si l'usuari inicia sessió amb un navegador diferent o esborrar les dades emmagatzemades al navegador, es perd l'accés a les converses guardades. A més, aquest sistema també presenta dificultats per als administradors a l'hora de fer un seguiment de les converses per usuari, ja que no és possible relacionar de manera directa un número de sessió amb una persona física. Aquesta implementació ha estat discutida en profunditat a l'apartat 4.4.10.

### 4.4.8 Implementació de FastAPI

Un dels avantatges de l'ús de FastAPI és la seva possibilitat de documentar el endpoints i la interfície de prova d'aquests, sense la necessitat de crear un frontend adaptat. Aquesta possibilitat és gràcies a OpenAPI. OpenAPI [22] és un estàndard obert per descriure en format JSON o YAML el comportament HTTP de les APIs. Facilita la comprensió de les funcionalitats que ofereix una API.

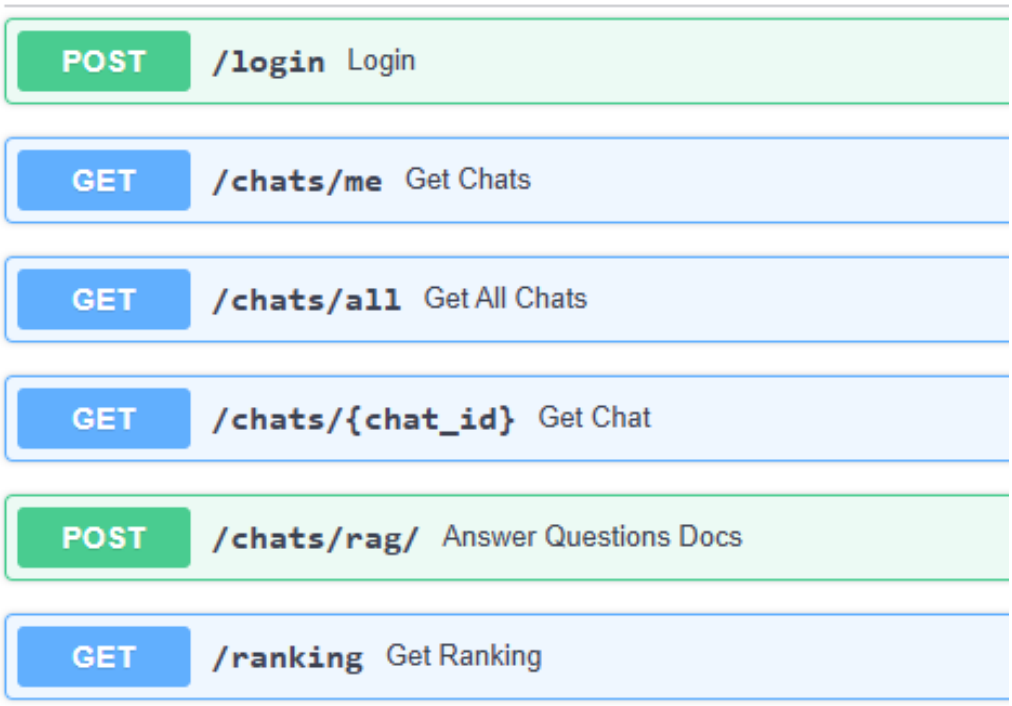


Figura 11: Documentació dels endpoints. Aquesta llista es pot veure a `/docs`

A la Figura 11 es mostra la interfície automàticament generada per FastAPI a l'endpoint `/docs`. Aquesta interfíci es construeix a partir de l'especificació OpenAPI generada de forma automàtica a mesura que es defineixen els endpoints amb FastAPI. Aquesta documentació permet als desenvolupadors:

- Visualitzar tots els endpoints disponibles, agrupats per rutes i mètodes HTTP.
- Consultar els paràmetres d'entrada i els codis de resposta.
- Provar les peticions directament des del navegador, sense necessitat de crear eines addicionals o un frontend específic.

### 4.4.9 Implementació del Frontend

El desenvolupament del frontend s'ha fet mitjançant Vue.js, JavaScript, Bootsrap i CSS. A nivell de frontend, és una aplicació bastant senzilla, ja que només compta amb dues vistes.

Les dues vistes estan clarament diferenciades, ja que la vista principal està dedicada als usuaris PDIs, i l'altra vista és pels administradors.

**Vista principal**

Aquesta vista (Vegeu Figura 12) està dividida en dues parts diferenciades, amb els seus elements.

El lateral de la vista, està format per un *header* que conté el títol i el botó de crear un nou xat. A sota del header, es veuen els xats anteriors de l'usuari. Aquests elements són interactuables i es cliquen es mostra el contingut del respectiu xat. Al footer del lateral es veu la imatge institucional de la `Universitat de Barcelona` i el del programa RIMDA, part del OTPSD.

Al costat del lateral esquerre tenim el *main content*. Si és la primera vegada que l'usuari accedeix, trobarà un missatge de benvinguda i explicatiu de què és el xat i com funciona. Un cop l'usuari escriu al xat el missatge de benvinguda desapareix i es veuen les bombolles de conversa entre l'usuari i el bot 13. Al header del *main content*, veiem la id del xat.

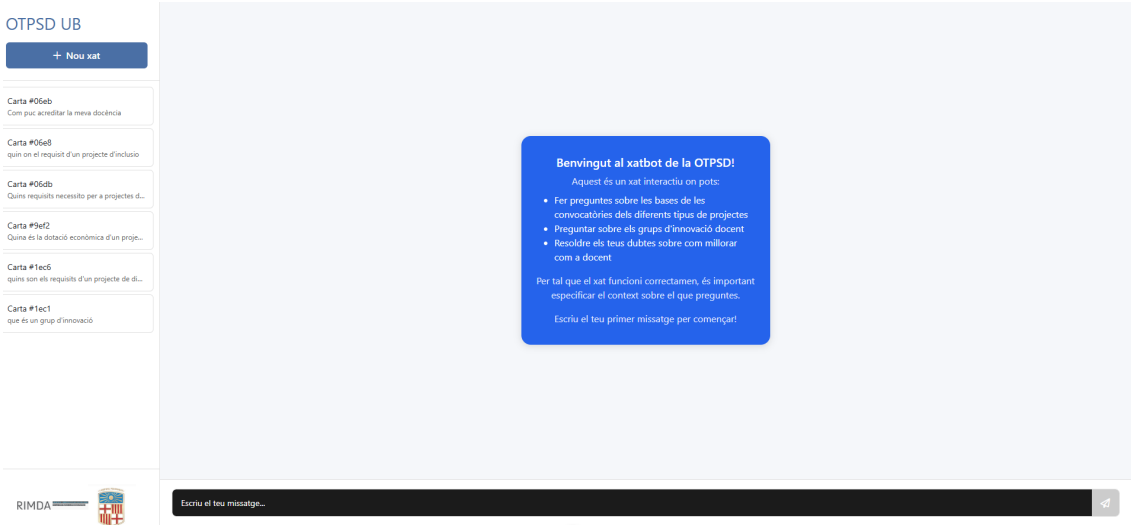


Figura 12: Vista principal orientada als PDI. Podem observar el lateral amb una llista de xats de l'usuari, i el missatge de benvinguda.

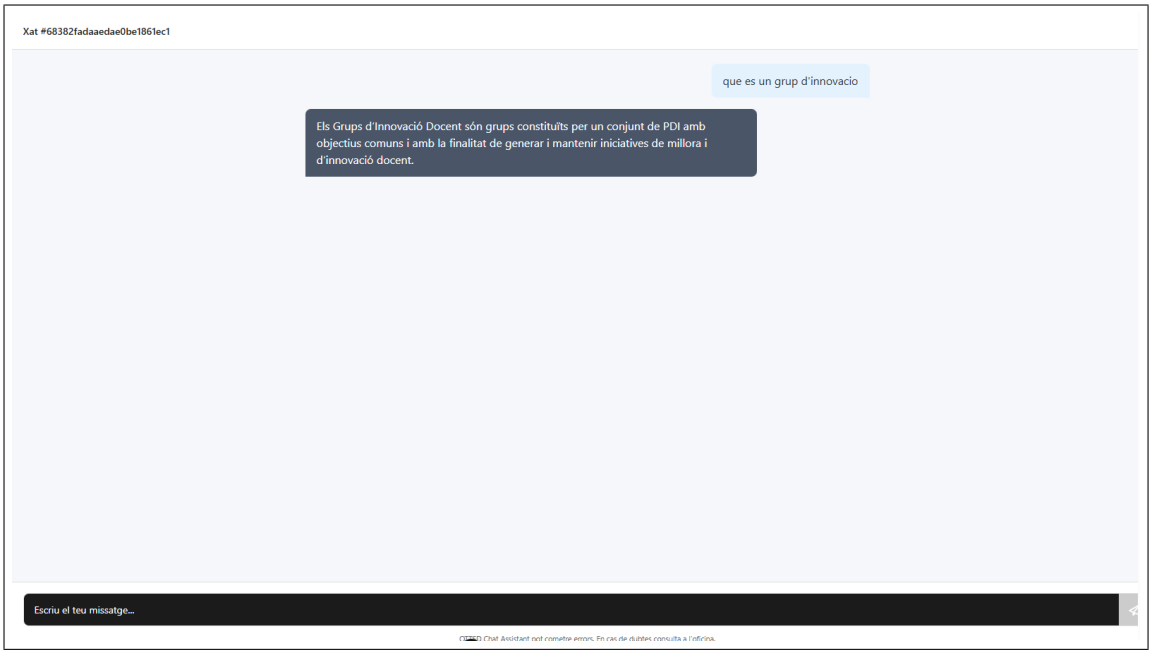


Figura 13: Main content quan l'usuari ha introduït una consulta. Com es pot veure el missatge de benvinguda desapareix. També es pot observar el placeholder per introduir les consultes

### Vista d'administradors

Els administradors tenen una vista diferent i exclusiva per ells, per mantenir un registre i control de l'ús de l'aplicació i les consultes dels altres usuaris.

Per començar, aquesta vista requereix d'un token per poder-hi accedir. Aquest token s'aconsegueix iniciant sessió amb les credencials que només saben els membres de l'oficina.

Una vegada s'ha iniciat la sessió correctament, es veu una llista formada per *cards* de bootstrap. Aquestes cards permeten interacció i si cliquem sobre elles, s'expandeixen i mostren les consultes dels usuaris classificades per document (Vegeu Figura 14).

En aquesta vista, al header, trobem dos botons. Un per tancar la sessió, i un altre per obrir un cercador i cercar un xat per id i consultar quina conversa ha tingut l'usuari en qüestió. No només s'obre un cercador sinó també una llista de tots els xats amb la seva id (Vegeu Figura 15.


Figura 14: En aquesta imatge es veu la vista dels administradors. En el header podem veure els dos botons i el títol de la vista. En el cos de la vista, veiem les cards. La segona card està desplegada i es veu una consulta feta per algun usuari.


Figura 15: Aquest modal de bootstrap és el que apareix quan es clica el botó de "Cercar xat". Es veu la llista completa on es veuen totes les ids dels xats dels usuaris. Quan es clica en una d'aquestes ids, s'obre la vista del xat.

El disseny s'ha desenvolupat seguint els principis d'usabilitat orientats a l'usuari [23]. És una vista simple i clara per l'usuari, la qual segueix un motiu cromàtic amb els colors de la universitat. Els botons i títols són clars i estan acompanyats d'icones com la d'enviar missatge o la de crear un nou xat. El disseny està orientat a les tasques de l'usuari, amb un missatge inicial que explica de manera clara que fa i com funciona el xatbot. A més en tot moment l'usuari sap si el sistema està funcionant o no, ja que a la pantalla del xat he implementat una icona que indica que la resposta està carregant. També en la vista d'administració, en fer el login, es mostren *toasts* de bootstrap indicant si el login ha estat correcte o fallit.

### 4.4.10 Decisions tècniques d'implementació

Un dels principals dubtes durant el procés de desenvolupament ha sigut com fer la cerca i l'ordenació dels documents del Qdrant.

### S'ha d'especificar el context de la consulta?

En un principi, quan s'introduïa una pregunta al xat, el sistema buscava per totes les col·leccions fins a trobar una similitud. Tot i que això era funcional, en casos on la pregunta podia ser ambigua, ja que estava present en diverses col·leccions, el xatbot no retornava la resposta esperada. Per solucionar aquest problema, vaig provar d'especificar un context per buscar directament a la col·lecció referent. Però per basar-me en evidències, vaig fer diferents proves temporals i de resposta davant les mateixes preguntes. En l'àmbit temporal, el resultat va ser molt millor en mitjana de temps per les múltiples col·leccions i la resposta no sempre coincidia en els dos casos. Per fer aquestes proves, vaig modificar temporalment l'endpoint per poder decidir a quina col·lecció buscar. En aquest escenari, es van obtenir millors resultats amb grans diferències. Vaig fer dos escenaris diferents:

- L'usuari introduïa la mateixa pregunta dues vegades, una especificant el context via endpoint (cas A), i l'altra sense(cas B). Repetint diverses vegades la prova, vaig obtenir els següents temps: **3,59 s** en el cas A i **8,56 s** en el cas B, la resposta del Qdrant va ser la mateixa, però tenint en compte que un dels requisits no funcionals és el rendiment, és evident que en aquest escenari la millor decisió és especificar el context.

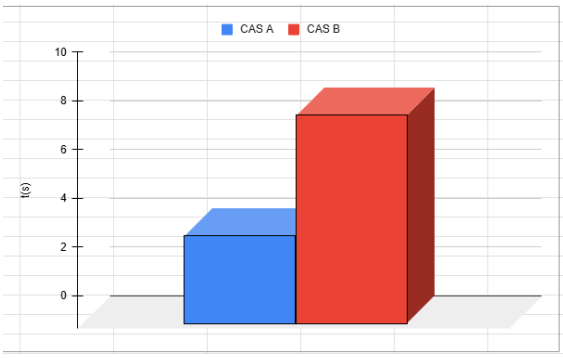


Figura 16: Gràfica temporal on es veu la diferència entre els dos casos d'especificació de context: via endpoint o

- L'usuari introduïa una pregunta ambigua, com per exemple "Quins són els requisits" però amb intenció de saber els requisits sobre les propostes de Responsabilitat Social. En el cas A, especificant el context, el temps de resposta va ser de **1,95 s**, mentre que en el cas B, sense especificar-lo, la resposta va ser de **2,21 s**. No només el temps de resposta va ser més lent sinó que en el cas B, la resposta va ser incorrecta, ja que no va contestar sobre el context que l'usuari volia.

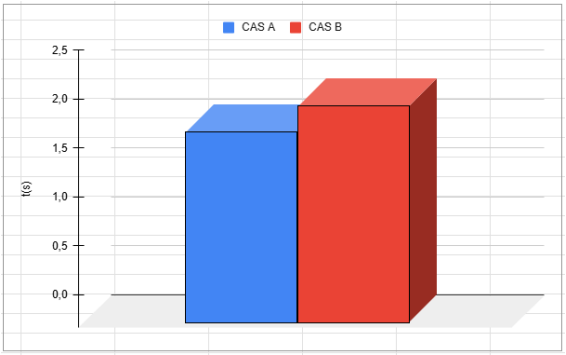


Figura 17: Gràfica temporal on es veu la diferència temporal entre els dos casoss

Un cop decidit que és primordial especificar el context, com s'ha de fer a nivell Frontend?

**L'usuari tria un context manualment o mitjançant la consulta el sistema el detecta?**

Un dels principals problemes va ser haver de decidir sobre com aquest context seria passat al backend perquè pogués fer la cerca eficient. Es plantejaven dues opcions: l'usuari la tria manualment mitjançant botons al frontend o escrivint la consulta el sistema detecta sobre quin tema està parlant. Vaig analitzar els dos escenaris comparant avantatges i inconvenients.

**Tria manual:**

| **Avantatges** | **Inconvenients** |
|---|---|
| L'usuari tria explícitament el context sobre el qual està parlant | Pot ser que l'usuari classifiqui malament sobre quin tema vol parlar. |
| El sistema no s'equivoca en la classificació | Poca sensació de llibertat |
| A nivell de consultes al LLM, és gratis | Dificultat davant de fer un canvi de context |

**Detecció automàtica:**

| **Avantatges** | **Inconvenients** |
|---|---|
| L'usuari escriu lliurement | L'usuari ha d'especificar via text o donar entendre el context perquè el sistema el classifiqui automàticament. |
| Escalable sense haver d'editar codi del frontend | Més prompts i embeddings |
| Facilitat per fer un canvi de context | |

Tot i que la decisió ha sigut complicada, al final he optat perquè el sistema faci les deteccions automàtiques. L'experiència d'usuari és més còmoda i amigable si és una interfície de xat clàssic, sense botons, ja que també fa més sensació de llibertat i menys control.

Una vegada decidit que és el sistema qui ha de classificar i decidir el context sobre el qual està preguntant l'usuari, sorgeix un altre dubte. Com fem la classificació?

**Fer la classificació via prompt o via embeddings?**

Durant el desenvolupament, també vaig haver de decidir de quina manera es podria classificar el context. Es van plantejar dues opcions:

**Classificació via embeddings:** Una opció interessant de les col·leccions de Qdrant són les metadades. Una de les metadades útils són les descripcions, que permeten afegir descripcions per descriure el contingut de les col·leccions. Es poden generar embeddings d'aquestes descripcions i comparar-les amb l'entrada de l'usuari. El problema que vaig detectar després de fer proves, és que la descripció havia de contenir molts termes perquè hi hagués similitud d'embeddings. Un exemple dels embeddings que havia plantejat per fer la classificació és el següent:

"digi": "Projectes de digitalització i transformació digital en l'àmbit universitari."

**Classificació via prompt**: L'altra opció, i la que he implementat, és abans de contestar la pregunta, passar-la amb un prompt al LLM perquè aquest la classifiqui correctament. Testejant aquesta opció ha obtingut un 100% d'èxit, sempre que a l'input de l'usuari es mencioni alguna cosa relacionada amb la col·lecció a explorar. Després de cada entrada de l'usuari, es torna a comprovar el context per si aquest ha canviat.

**Persistència de Sessió**: Per acabar, l'última decisió a prendre és com es mantenen les sessions de l'usuari. En el meu pensament inicial, era poder connectar-me al servidor de credencials de la UB i que els docents poguessin iniciar sessió amb les seves credencials. Per culpa de la impossibilitat d'implementar aquest servei, he optat perquè es creï una session id i es desi a session storage. Aquest session id també es desarà als documents de xats associats a l'usuari, per poder recuperar-los quan l'usuari entri al xat. Una de les possibilitats plantejades pels membres de l'oficina, va ser que a la primera interacció amb el xat, l'usuari indiqués el seu nom i cognom i la facultat on exerceix, però aquesta opció la vaig descartar per evitar problemes amb la LOPD i fer-ho més segur a nivell de seguretat i privacitat.

# 5 Proves i experiments

## 5.1 Tria de model LLM

Per triar el model a utilitzar vaig plantejar dues opcions: utilitzar un model obert i gratuït com `DeepSeek` o un model privat com `OpenAI`.

Per començar, vaig fer proves localment amb DeepSeek. Gràcies a Ollama, que permet fer servir models LLM en la terminal local, vaig poder provar dos diferents models de DeepSeek amb diferents quantitats de paràmetres: DeepSeek, amb 7 bilions de paràmetres, i DeepSeek amb 8. Aquests models compten amb el gran avantatge, que poden funcionar sense connexió a internet, i de manera gratuïta, però depenen molt de la memòria RAM de l'equip on s'executen. El temps de resposta era bastant lent, i la resposta no era sempre la correcta.

En ser DeepSeek un model bastant nou, no tenia gaire suport per les consultes en català.

D'altra banda, les proves amb OpenAI van anar amb un menor temps de resposta, tot i la connexió a l'API remota. El suport amb l'idioma desitjat era òptim i malgrat haver de recórrer a la compra de crèdits per garantir el funcionament, era la millor opció pels requisits no funcionals del xatbot.

A més a més, l'API d'OpenAI compta amb moltes possibilitats com la creació dels embeddings, o el Fine-Tuning per potenciar les possibles funcionalitats de l'aplicació.

Un dels inconvenients d'usar OpenAI és que cada petició té un cost en crèdits. Aquest cost no és molt elevat, sobretot a l'hora de fer els embeddings, però sí que requeriria anar fent recàrregues d'aquests crèdits. Per fer la prova inicial a l'abril es van carregar 5$ de pressupost per poder fer les consultes.

A *04/06/2025* aquesta és la quantitat gastada, com es pot veure a la figura 18

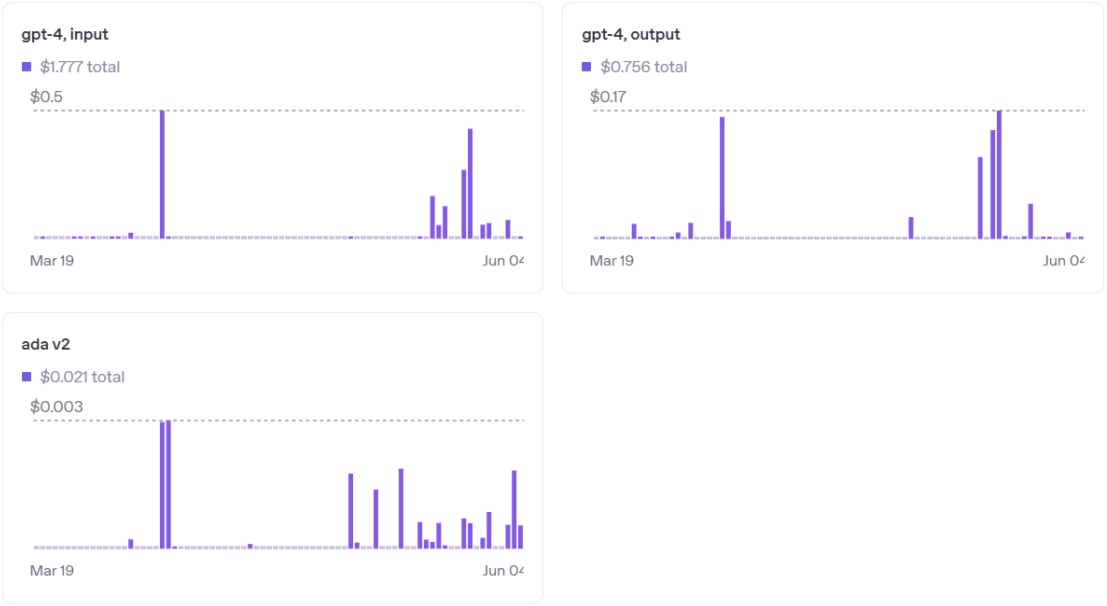


Figura 18: Ús dels crèdits a OpenAi

Com s'observa a la imatge, el cost més elevat és quan li passem prompts al model, ja que aquests prompts estan presents durant tot el procés. Els embeddings tenen un cost molt reduït, tot i que es fan a cada consulta. Dels 5$ inicials, encara tenim uns 2,5$ per continuar fent consultes, així que el sistema es força eficient respecte a l'ús de crèdits.

Tot i haver de comptar amb una despesa recurrent, aquesta no ha de ser molt elevada, ja que amb una inversió mínima s'ha pogut donar cobertura a consultes i proves durant diversos mesos.

## 5.2 Prova d'ús

En aquesta secció explicaré les proves amb usuaris reals que he dut a terme per comprovar el funcionament del software i prendre decisions respecte al desenvolupament.

### 5.2.1 A/B Testing

Per començar la prova d'ús amb els membres de l'oficina es va fer una prova A/B Testing del frontend. Una prova A/B és una prova en la qual es comparen dues o més versions d'una mateixa pàgina web. Tenen l'objectiu de determinar quina és la variant més eficaç. En aquest cas la prova A/B no va ser sobre l'eficiència de dues versions de la mateixa web, sinó sobre dues versions de frontend per valorar-les estèticament.

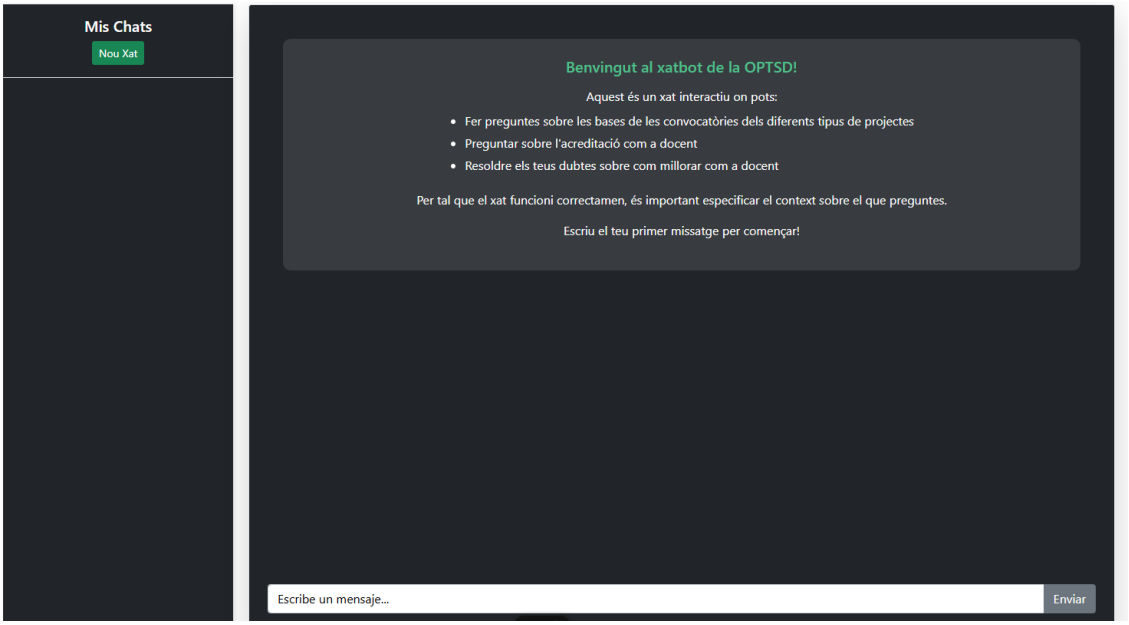


Figura 19: Versió antiga del frontend. Va ser la versió inicial i que es va mantenir fins a haver desenvolupat totes les funcionalitats del backend. Un cop desenvolupada la versió definitiva es van comparar per triar-ne una.

Com es pot veure a la figura 19 era una versió més inicial i menys institucional. Destaca per ser més fosca, que la versió definitiva explicada a l'apartat 4.4.9. Els membres de l'oficina amb els qui vaig fer la prova, ràpidament van triar l'altra versió, advocant per la seva estètica més institucional i clara i seguint els colors i estils de la UB. En l'àmbit funcional no hi havia cap diferència, ja que estaven connectades al mateix backend.

### 5.2.2 Prova de l'aplicació

Durant la fase de prova de l'aplicació, diversos membres de l'oficina van participar com a usuaris PDI simulats. La primera part de la prova es va realitzar sense cap mena de guia prèvia, amb l'objectiu de valorar si la interfície era intuïtiva i fàcil d'utilitzar sense instruccions. El resultat va ser positiu, ja que van saber utilitzar-la fàcilment.

La segona part d'aquesta va ser una demostració guiada per part meva amb l'objectiu d'explicar com funcionava la vista d'administració, què és la que més utilitzaran durant el seu dia a dia.

Durant la prova, es va detectar que les preguntes desades amb un interrogant a la base de dades, es detectaven malament, i no s'obtenia la resposta adequada. El que vaig fer per arreglar aquest problema, va ser eliminar tots els interrogants de la base de dades, i normalitzar l'entrada de l'usuari, eliminant també els interrogants amb *strip*. En conseqüència, els embeddings de la pregunta no es fan amb l'interrogant, malgrat que l'usuari l'introdueixi, i així hi ha més similitud entre el document desat a la base de dades i la consulta.

### 5.2.3 Comentaris de millora

Durant el transcurs de la prova els membres de l'oficina em van fer diversos comentaris de millora i canvi sobre el funcionament del xat.

Durant la prova d'ús, un cop es trobava el fragment amb la informació més adient, aquest es passava a l'LLM perquè el reformulés amb un llenguatge més natural. Tot i això, els participants van recomanar que el xat respongués directament amb el fragment original del document. Argumentaven que, en alguns casos, la generació de text podia introduir errors, o bé oferir respostes massa genèriques. Com que l'aplicació està pensada per resoldre dubtes tècnics del professorat, és més important que la resposta sigui correcta i precisa, encara que el llenguatge pugui ser una mica tècnic.

Una altra característica que van recomanar eliminar va ser l'opció de poder preguntar-li al xat de fer un resum o un aclariment de la resposta anterior. El motiu del suggeriment va ser el mateix, el risc de generació de respostes incorrectes per part del model de llenguatge.

## 5.3 Testing d'endpoints i prompts

Per provar el comportament dels endpoints van ser testejats amb *pytest*. Amb aquests tests, he provat l'endpoint de login, i també els prompts per mantenir el context. Amb aquestes proves també s'ha pogut decidir en algun dels dubtes sobre la implementació, com si decidir el context mitjançant prompts o embeddings, com s'ha explicat a l'apartat 4.4.10.

## 5.4 Testing del Qdrant

Per acabar amb l'apartat de proves, vaig dur a terme proves de la recuperació documental amb Qdrant. Quan volem fer la cerca, hi ha un atribut anomenat score_threshold què és el que determina el llindar de similitud entre l'entrada i els vectors desats a la base de dades. Aquest factor és un atribut clau, ja que condiciona la resposta del sistema.

Aquest valor no ha de ser ni massa estricte (0.9 o 1) ni massa baix (¡ 0.5), ja que si el valor és alt, la cerca no obtindrà cap resultat, i si és massa baix, la resposta pot ser una no desitjada.

Després de diverses proves, el valor més eficient que s'ha trobat ha estat **0.7** sobre 1, ja que ofereix un bon equilibri. Aquest valor permet que les entrades no siguin tal com estan desades a la base de dades, però sí que filtra aquelles que no tenen relació a veure amb la consulta inicial.

# 6 Resultats

En aquest apartat, descriuré els resultats de l'aplicació, on veure'ls i com ha quedat la implementació final.

## 6.1 Diagrama de casos d'ús final

Des de l'inici de la construcció de l'aplicació, fins al final d'aquesta, hi ha hagut diversos canvis en refinaments d'històries d'usuari, o en canvis de peticions de part de la OTPSD.

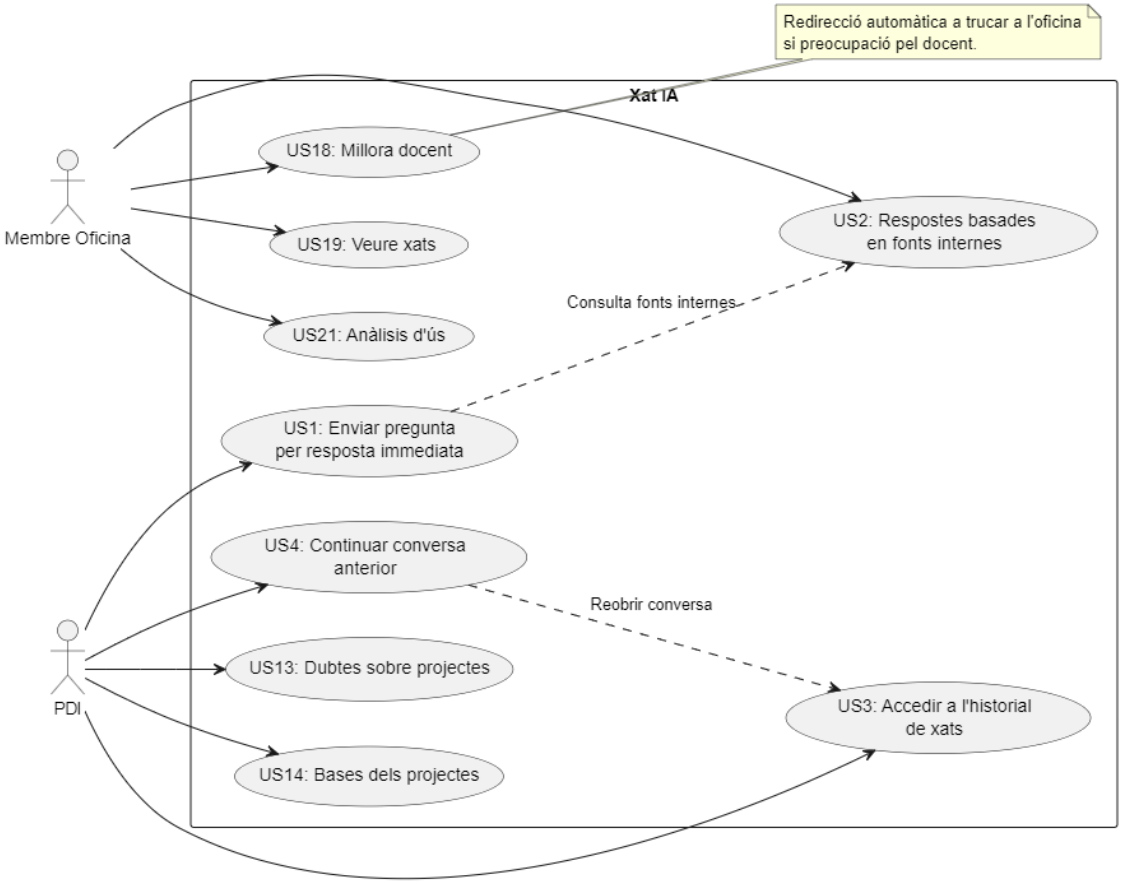


Figura 20: Diagrama de casos d'ús final

En comparació amb el diagrama inicial a la secció 4.2.3, es pot observar que algunes històries d'usuari no estan presents al diagrama final.

- Les històries USS7 i US6 s'han eliminat, ja que el formulari per introduir les dades del projecte ha canviat durant el procés de desenvolupament del xat i ara és més complet.

- La US9 no s'ha desenvolupat, ja que la US18, és molt similar
- La US22 s'ha eliminat seguint els consells de la prova d'ús explicada a l'apartat 5.2.

## 6.2 Publicació de l'aplicació

La implementació del projecte es pot consultar a `https://rimda-chatbot.vercel.app/`.

El desplegament s'ha fet mitjançant 4 diferents entorns remots. Per fer el deploy de FastAPI i tota la lògica de l'aplicació s'ha usat *Render*, una opció de *cloud* gratuïta. Té un inconvenient, ja que en la seva versió de franc, si hi ha un període de temps sense peticions, quan detecta una nova, s'ha de tornar a "encendre" el backend, i la primera petició pot trigar uns 50 segons a ser resposta. Pel deploy de MongoDB i Qdrant s'han utilitzat els seus propis serveis gratuïts, MongoDB Atlas i Qdrant Cloud. Finalment, el servei de frontend s'ha desplegat en *Vercel*.

Com a estudiant de la Universitat, disposo de crèdits anuals d'`Azure` que permeten fer desplegaments al núvol. Tot i això, ja que es tracta d'una aplicació que ha de ser accessible i funcional a llarg termini, s'ha optat per utilitzar opcions gratuïtes i sostenibles en el temps.

## 6.3 Rendiment i comportament del sistema

Tot i que no s'han pogut fer proves amb un gran volum d'usuaris simultanis, sí que s'han pogut analitzar les respostes que dona el xatbot. En aquestes proves, s'ha vist que aproximadament el 80% de les respostes són correctes i útils. A més, en converses on és important mantenir el context, el sistema respon bé i manté el fil de la conversa sense problemes.

# 7 Conclusions

Després del desenvolupament i les proves d'ús, es pot arribar a la conclusió que el projecte ha complert els seus objectius principals. Ha demostrat ser una eina útil i pràctica. A continuació es detallen els objectius aconseguits i les possibles àrees de memòria i de treball futur.

## 7.1 Objectius aconseguits

- **Aplicació dels aprenentatges adquirits**: Durant el desenvolupament del projecte, he aplicat els màxims coneixements adquirits i relacionats amb les assignatures de la carrera. Per exemple, a l'hora de dissenyar les vistes, he tingut en compte el que he après a *Factors Humans*, i en la construcció d'una arquitectura distribuïda, el que vam fer a *Bases de Dades Avançades* i *Software Distribuït*.

- **Capacitat de desenvolupament d'un projecte complet de manera autònoma**: Considero que he sigut capaç de desenvolupar un projecte des de zero, el qual compleix els requisits descrits pels membres de l'oficina. Ha sigut la primera vegada que feia un projecte complet d'aquesta envergadura individualment, ja que sempre he treballat en grup o en projectes individuals petits, i crec que el resultat ha sigut l'esperat.

- **Aprendre sobre RAG i LLM**: El desenvolupament del projecte m'ha permès aprendre des de zero sobre les tècniques RAG (Retrieval-Augmented Generation) i els models LLM (Large Language Models). Inicialment, tenia poc coneixement sobre aquestes tecnologies, però després de tot el procés d'investigació i implementació, considero que he assolit una bona comprensió del seu funcionament i aplicació pràctica.

- **Fer una interfície visual entenedora**: L'aplicació compta amb una interfície d'usuari dissenyada tenint en compte els principis d'experiència de l'usuari. Els usuaris que l'han provat, s'han sabut moure per ella sense ajuda, fet que demostra la seva usabilitat i el bon disseny en termes de simplicitat i claredat.

- **Respostes contextualitzades**: La combinació de la tècnica RAG i els models LLM ha permès complir l'objectiu principal funcional de l'aplicació: donar respostes contextualitzades i fiables, basades en la documentació proporcionada, reduint al mínim el risc que el xatbot generi informació incorrecta o inventada.

## 7.2 Àrees de millora i treball futur

- **Interfície responsiva**: L'aplicació ha estat pensada per ser utilitzada en ordinadors. No s'ha optimitzat per dispositius mòbils i és un aspecte de millora en un futur.

- **Respostes amb llenguatge natural**: Al principi, la idea era passar els fragments trobats als documents a l'LLM perquè generés respostes amb un llenguatge més natural i fàcil d'entendre. Però finalment es va descartar perquè hi havia el risc que el model inventés o modifiqués informació incorrecta, cosa que no és acceptable quan es tracta de temes tècnics. Per això, es va decidir mostrar directament els fragments més rellevants del document, per assegurar que la informació fos fiable.

  De cara al futur, m'agradaria poder implementar aquesta opció de respostes en llenguatge natural si es troba la manera de crear un prompt que eviti que el model inventi dades. Així es podria combinar la facilitat d'entendre les respostes amb la seguretat que la informació és correcta.

- **IA reflexiva**: De cara al futur, als membres de l'oficina els agradaria que el xatbot no només respongués consultes tècniques, sinó que també plantegés preguntes de caràcter reflexiu als docents. L'objectiu seria ajudar-los a qüestionar-se el perquè de les coses, promovent una mirada més crítica i profunditat en els projectes sobre la docència. Així, l'eina deixaria de centrar-se només en la quantitat de projectes i apostaria més per la seva qualitat, oferint també una possibilitat de millora com a docents al PDI.

- **Afegir funcionalitats**: Tot i que s'han eliminat algunes User Stories perquè no eren estrictament necessàries, i perquè ja existeixen altres eines per a aquestes consultes, seria interessant incorporar-les al xatbot en el futur per oferir una eina més completa i centralitzada.

- **Actualització documental**: Un dels objectius era implementar un sistema que permetés actualitzar fàcilment els documents que formen la base de dades sobre la qual es generen les respostes. No s'ha aconseguit aquesta funcionalitat, i actualment, perquè els membres de l'oficina puguin actualitzar els documents, han de modificar directament el codi font, cosa que requereix coneixements tècnics.

- **Velocitat de resposta**: Com he explicat a l'apartat 6.2, el deploy s'ha fet de manera gratuïta. Això ha permès abaratir costos, però en conseqüència, es perd velocitat. Només es perd velocitat si el backend ha patit molt temps sense rebre peticions, ja que s'ha de tornar a activar. Una vegada està iniciat, la velocitat és adient. Per millorar, si es comptés amb un pressupost, es podria fer el desplegament en millors plataformes.

En resum, el projecte ha aconseguit complir els objectius definits a la definició del projecte.

A nivell personal, estic molt satisfet i orgullós amb el resultat obtingut, i de l'opinió dels membres de l'oficina amb el producte final.

A més les àrees a millorar, indiquen que el projecte es pot continuar i continuar-lo millorant en el temps.

# Referències


[1] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024.

[2] Sophie Hundertmark. Llm-chatbots — an introduction to the new world of bots. `https://sophiehundertmark.medium.com/llm-chatbots-an-introduction-to-the-new-world-of-bots-485db17da7b2`, 2024. Accés el 23 de maig de 2025.

[3] Adrián Ramírez Fernández, Laura Pamela Solís Bastos, Esteban Martínez Porras, and José André Díaz González. Uso de procesamiento de lenguaje natural para procesar respuestas abiertas de una encuesta de opinión pública. *Revista Latinoamericana de metodología de la investigación social*, (29):51–67, 2025.

[4] IBM Research. What is generative ai? `https://research.ibm.com/blog/what-is-generative-AI`, 2023. Accés el 23 de maig de 2025.

[5] Charu C. Aggarwal. *Deep Learning: Principles and Training Algorithms*. Springer International Publishing AG, Cham, second edition edition, 2023.

[6] Nanobaly. Tokenización: clave para la ia generativa y los llm. `https://es.innovatiana.com/post/tokens-for-generative-ai`, 2025. Accés el 23 de maig de 2025.

[7] Pablo Huet. Embeddings: Qué son y cómo transforman datos en información. `https://openwebinars.net/blog/embeddings/`, 2024. Accés el 23 de maig de 2025.

[8] Erik Herman. *Optimizing Prompt Engineering for Generative AI*. Mercury Learning and Information, Boston, MA, first edition. edition, 2025.

[9] IBM Research. What is retrieval-augmented generation (rag)? `https://research.ibm.com/blog/retrieval-augmented-generation-RAG`, 2023. Accés el 23 de maig de 2025.

[10] Amazon Web Services. ¿qué es una base de datos vectorial? `https://aws.amazon.com/es/what-is/vector-databases/`, 2025. Accés el 23 de maig de 2025.

[11] MongoDB, Inc. Bases de datos de documentos: conceptos básicos. `https://www.mongodb.com/es/resources/basics/databases/document-databases`, 2025. Accés el 23 de maig de 2025.

[12] Eduard Babulak. *Chatbots*. IntechOpen, 2023.

[13] Ramon. Lopez de Mantaras Badia and Pedro Meseguer Gonzalez. *Inteligencia artificial*. ·Que sabemos de? ; 87. Editorial CSIC, Madrid, 2017.

[14] Moin AI. Rule-based chatbots: Examples & possible applications. https://www.moin.ai/en/chatbot-wiki/rule-based-chatbots#toc-3, 2025. Accés el 23 de maig de 2025.

[15] Robert Ciesla. *The Current Era of Chatbots*, pages 53–54. Springer Nature Switzerland, Cham, 1st ed. 2024 edition, 2024.

[16] Robert Ciesla. *The Book of Chatbots : From ELIZA to ChatGPT*, pages 60–61. Springer Nature Switzerland, Cham, 1st ed. 2024 edition, 2024.

[17] Fernando Izaurieta and Carlos Saavedra. Redes neuronales artificiales. *Departamento de Física, Universidad de Concepción Chile*, 2000.

[18] Tong Xiao and Jingbo Zhu. Introduction to transformers: an nlp perspective, 2023.

[19] University of Wisconsin-Milwaukee. Ask pounce! https://uwm.edu/finances/finances/ask-pounce/, 2025. Accés el 24 de maig de 2025.

[20] Universidad de Murcia. La universidad de murcia presenta a lola, un asistente de inteligencia artificial para ayudar a los nuevos alumnos. https://www.um.es/web/sala-prensa/-/la-universidad-de-murcia-presenta-a-lola-un-asistente-de-inteligencia-artifici 2025. Accés el 24 de maig de 2025.

[21] LangChain. Why langchain? https://python.langchain.com/docs/concepts/why_langchain/, 2025. Accés el 29 de maig de 2025.

[22] OpenAPI Initiative. What is openapi? https://www.openapis.org/what-is-openapi, 2022. Accés el 29 de maig de 2025.

[23] Klaus Kaasgaard. *Software design and usability : talks with Bonnie Nardi, Jakob Nielsen, David Smith...* Copenhagen Bussiness School Press, Copenhagen, 2000.

# 8 Annexos

## 8.1 ANNEX A: Històries d'usuari

Durant el procés de desenvolupament del xatbot, les històries d'usuari han patit un refinament. Inicialment, les primeres històries d'usuari eren una traducció directa dels requisits descrits pels membres de l'oficina, però a mesura que passava el temps van sorgir la necessitat de modificar històries, crear-ne noves i eliminar-ne algunes.

Un cop vaig decidir les eines amb les quals anava a fer el xatbot, algunes històries es van haver d'adaptar a less possibilitats que oferien aquestes eines.

La proposta inicial va ser aquesta:

**US1: Enviar pregunta per rebre resposta immediata**

**Descripció**: Com a PDI, vull poder enviar una pregunta a través del xat, per poder obtenir una resposta immediata.

**Criteris d'acceptació:**

- El PDI entra a l'aplicació web.
- El PDI escriu al xat.
- El sistema reconeix la pregunta i comença a processar-la de forma immediata.
- La IA analitza la pregunta i respon de forma clara en temps real.
- La resposta és coherent amb la pregunta plantejada.
- Si la IA no entén la pregunta o no té informació suficient, respon sol·licitant un aclariment.

**Story points:** 2

**US2: Respostes basades en fonts internes**

**Descripció**: Com a membre de l'oficina, vull que les respostes estiguin basades en fonts internes, per poder garantir respostes amb coneixement.

**Criteris d'acceptació:**

- El xat té una base de dades documental amb els documents interns.
- El xat respon amb la informació extreta dels documents via RAG.

**Story points:** 8

**US3: Accedir a l'historial de xats**

**Descripció:** Com a PDI, vull poder accedir a l'historial de converses per poder revisar respostes anteriors.

**Criteris d'acceptació:**

- El PDI pot accedir a l'historial des del xat.
- Les converses es guarden automàticament.
- L'historial mostra les converses ordenades per data (la més recent a dalt).

**Story points:** 3

**US4: Continuar conversa anterior**

**Descripció:** Com a PDI, vull poder accedir a l'historial de converses per poder seguir amb una conversa anterior.

**Criteris d'acceptació:**

- El PDI pot accedir a l'historial.
- Les converses es guarden automàticament.
- Quan el PDI selecciona una conversa anterior, el xat es reobre en el mateix punt on es va deixar.

**Story points:** 3

**US5: Classificació de propostes**

**Descripció** Com a PDI, vull que la IA m'ajudi a classificar una proposta per poder omplir el formulari corresponent.

**Criteris d'acceptació**:

- El PDI pot descriure la seva proposta mitjançant un text lliure.
- La IA analitza el contingut de la proposta i suggereix una de les tres categories:
  - Projectes de Millora I Innovació Docent
  - Projectes de Digitalització
  - Projectes de Responsabilitat Social Universitària
- La IA justifica la seva classificació amb una breu explicació basada en criteris establerts

**Story points**: 3

**US6: Mostrar camps a omplir segons categoria**

**Descripció:** Com a PDI, vull que la IA em mostri quins camps he d'omplir segons la categoria per poder fer la proposta.

**Criteris d'acceptació**:

- El PDI pot seleccionar manualment la categoria de la seva proposta o deixar que la IA la classifiqui automàticament
- Un cop determinada la categoria, la IA mostra la llista de camps necessaris per completar la proposta

**Story points**: 2

**US7: Preguntar dubtes sobre camps concrets del formulari**

**Descripció:** Com a PDI, vull que la IA em permeti escriure sobre quin camp tinc dubtes per poder obtenir una resposta específica.

**Criteris d'acceptació**:

- El PDI pot escriure lliurement quin camp del formulari vol consultar
- La IA identifica el camp esmentat i proporciona una explicació detallada
- La IA pot proporcionar exemples pràctics de respostes adequades per aquell camp
- Després de la resposta de la IA, el PDI pot demanar més detalls o exemples concrets
- Si la IA no pot resoldre el dubte, recomana contactar amb l'oficina

**Story points** : 3

**US8: Propostes de millora**

**Descripció:** Com a membre de l'oficina, vull que la IA pugui fer propostes de millora a les propostes per poder rebre millors propostes.

**Criteris d'acceptació**:

- Quan un PDI envia una proposta, la IA l'analitza i detecta possibles mancances o aspectes millorables.
- La IA avalua claredat, coherència, alineació amb els objectius i justificació del projecte.

- La IA està entrenada amb exemples de propostes aprovades prèviament i ben valorades per l'oficina.
- La IA identifica patrons de bones pràctiques en aquests projectes i els utilitza per fer recomanacions més precises.
- La IA pot proporcionar un resum amb punts forts i febles de la proposta.

**Story points**: 8

**US9: Redirecció a l'oficina**

**Descripció:** Com a membre de l'oficina, vull que el xat recomani trucar a l'oficina, per poder resoldre dubtes molt concrets.

**Criteris d'acceptació**:

- La IA identifica preguntes que requereixen coneixement expert o una valoració humana.
- Si la IA detecta que la pregunta és massa específica, respon amb un missatge tipus: *Aquesta qüestió requereix una valoració més detallada. Et recomanem contactar amb l'oficina.*
- Si el PDI insisteix a obtenir una resposta de la IA, el sistema reitera la recomanació de contactar amb l'oficina.

**Story points**: 3

**US10: Representació gràfica de dades**

**Descripció:** Com a membre de l'oficina, vull veure una representació gràfica de dades dels projectes, per poder identificar tendències i la big picture dels projectes.

**Criteris d'acceptació**:

- Es mostrarà un dashboard amb gràfics clars i visuals.
- Tipus de dades representades:
  - Tipus de projecte
  - Temàtiques principals
  - Facultats implicades
  - Professors participants
  - Duració dels projectes
  - Estat dels projectes
- Només els membres de l'oficina poden veure aquestes dades.

**Story points**: 8

**US11: Repetició de paraules clau**

**Descripció:** Com a membre de l'oficina, vull veure quines paraules clau es repeteixen més, per poder detectar tendències i enfocaments recurrents.

**Criteris d'acceptació**:

- La IA analitza els textos de les propostes aprovades i extreu les paraules més recurrents.
- Núvol de paraules: mostra les paraules amb diferents mides segons la seva recurrència.
- Només els membres de l'oficina poden veure aquestes dades.

**Story points**: 5

**US12: Comparació amb memòria**

**Descripció:** Com a membre de l'oficina, vull comparar els objectius inicials amb la memòria final, per poder veure si s'ha desviat del pla inicial.

**Criteris d'acceptació**:

- La IA analitza els documents de la proposta inicial i la memòria final.
- Detecta els objectius principals del projecte en la proposta inicial.
- Analitza si aquests objectius es mantenen o es modifiquen a la memòria final.
- Si certes paraules clau dels objectius inicials no apareixen a la memòria final, es marca com a possible desviació.
- Només els membres de l'oficina poden veure aquestes dades.

**Story points**: 5

**Refinació d'històries**: com ja he mencionat anteriorment, durant el desenvolupament del software, vaig haver de redissenyar, eliminar i crear noves històries.

Les històries eliminades van ser les següents:

- US5: Aquesta història ha sigut eliminada per prioritzar el correcte funcionament del xat i preferència a altres històries.
- US8: Aquesta història ha sigut eliminada, ja que no coincidia amb els objectius inicials de la OTPSD. Els objectius eren fer un xatbot capaç de respondre a dubtes tècnics, no un xat capaç de donar idees per a projectes, ja que no es potenciaria la capacitat de pensament crític dels docents.

- US10, 11 i 12: Aquestes històries, es van descartar per prioritzar el correcte funcionament del xatbot.

Algunes històries es van redefinir i augmentar per admetre més escenaris com per exemple la US2, que es va transformar en quatre històries d'usuari diferents (US13, 16, 17 i 18). A part de modificar-ne d'existents, vaig afegir noves històries, sempre consultant als membres de l'oficina. Durant el desenvolupament del xat, van sorgir necessitats no contemplades que també es van transformar en noves històries.

**Històries d'usuari post refinament**

**US13: Dubtes sobre Projectes**

**Descripció:** Com a PDI, vull resoldre els meus dubtes sobre els projectes d'innovació per poder fer les propostes d'innovació docent.

**Criteris d'acceptació**:

- A la pantalla principal del xat l'usuari veu un missatge de benvinguda
- L'usuari escriu la seva pregunta especificant sobre quin tipus de proposta vol resoldre els dubtes (pmid, digitalització o rsu)
- Internament, el RAG tindrà documents sobre els diferents tipus de propostes
- L'usuari podrà canviar el context de la conversa, sense necessitat de començar un altre xat, especificant el nou context en la seva pregunta.

**Story points**: 5

**US14: Bases dels projectes**:

**Descripció:** Com a membre de l'oficina, vull que el xatbot consulti específicament les bases de les convocatòries, per tal que pugui respondre amb precisió les preguntes relacionades.

**Criteris d'acceptació**:

- Les bases oficials de la convocatòria estan incorporades dins el sistema RAG.
- El xatbot pot respondre preguntes sobre requisits, terminis, criteris d'avaluació i documentació requerida.
- Les respostes inclouen fragments textuals extrets directament de les bases.
- Si es fa una pregunta que no està coberta explícitament a les bases, el sistema ho indica clarament.

**Story points**: 5

**US16: Dubtes sobre grups**

**Descripció:** Com a PDI, vull resoldre els meus dubtes sobre els projectes d'innovació per poder presentar-me a les convocatòries.

**Criteris d'acceptació**:

- A la pantalla principal del xat, l'usuari veu un missatge de benvinguda.
- L'usuari escriu la seva pregunta especificant sobre quin tipus de proposta vol resoldre els dubtes (grups).
- Internament, el RAG tindrà documents sobre GID (grups d'innovació docent).
- L'usuari podrà canviar el context de la conversa, sense necessitat de començar un altre xat, especificant el nou context en la seva pregunta.

**Story points**: 2

**US17: Acreditació**

**Descripció:** Com a PDI, vull poder resoldre els meus dubtes sobre el procés d'acreditació com a docent per poder presentar-me al procés.

**Criteris d'acceptació**:

- L'usuari pot preguntar lliurement sobre els requisits i passos del procés d'acreditació.
- La IA està entrenada amb documents interns que regulen el procés d'acreditació dels docents.
- La IA proporciona respostes clares sobre terminis, criteris i documentació necessària.
- Si la pregunta no pot ser resolta per la IA, es recomana contactar amb l'oficina per obtenir suport.

**Story points**: 3

**US18: Millora docent**

**Descripció:** Com a membre de l'oficina, vull que si l'usuari està preocupat pel seu estat com a docent, el xat redirigeixi a trucar-nos per poder fer-li un assessorament personalitzat.

**Criteris d'acceptació**:

- Davant de preguntes com çom puc millorar com a docent", la IA respon amb un missatge recomanant contactar amb l'oficina.

- La resposta de la IA inclou informació clara de contacte i disponibilitat de l'oficina.

**Story points**: 2

**US19: Veure xats**

**Descripció:** Com a membre de l'oficina, vull poder veure què s'ha parlat via xat, per poder tenir un context necessari en cas de trucada dels docents.

**Criteris d'acceptació**:

- Els membres de l'oficina comptaran amb un compte d'administrador creat pel programador
- Han d'estar autenticats per poder veure l'opció de cercar
- Per autenticar-se han d'entrar al domini corresponent, que només sabran ells.
- Amb un buscador es poden buscar xats per id.

**Story points**: 5

**US21: Anàlisis d'ús**

**Descripció:** Com a membre de l'oficina, vull veure informes endreçats per nombre de qüestions fetes al xatbot per poder estar al dia de quins són els temes més preguntats.

**Prerequisits:** S'ha de mantenir un registre de cadascuna de les preguntes dels usuaris

**Criteris d'acceptació**:

- L'aplicació manté un registre de quins documents s'han consultat
- Els administradors poden veure aquest registre+
- El rànquing s'ordena per nombre de referències a cada document, i es mostra una llista on es veuen quants accessos hi ha hagut a cada un dels documents i les preguntes que fan els usuaris.

**Story points**: 3

**US22: Preguntes dependents**

**Descripció:** Com a PDI, vull poder fer una pregunta aclaridora al xat, per poder resoldre els meus dubtes sobre la resposta donada.

**Criteris d'acceptació**:

- L'usuari rep una resposta
- A l'usuari li entren dubtes sobre la resposta i vol fer preguntes de l'estil: "No ho he entès", "Em pots donar un exemple"?
- L'LLM analitza si aquesta és una pregunta depenent
- Si ho és, no busca a cap document intern, sinó que li passa l'historial de la conversa al LLM
- L'LLM respon segons aquell historial i la nova pregunta

**Story points**: 5

Després de la prova d'ús, es va decidir fer un nou refinament, on es van eliminar les històries US6,US7, US9, US17 i la US22.

## 8.2 ANNEX B: Desplegament Local

L'aplicació està publicada remotament, però si es vol executar localment a continuació es descriuen les passes a realitzar:

1. **Requisits previs**
   - Docker desktop instal·lat
2. **Variables d'entorn**

   En l'arrel backend, s'ha de crear un arxiu .env i posar les següents variables:
   - SECRET_KEY (per bcrypt del token)
   - ALGORITHM (bcrypt del token)
   - MONGO_DB=TFG
   - MONGO_COLLECTION=Chats
   - MONGO_REGISTRE_COLLECTION=Registre
   - MONGO_URI_DOCKER=mongo
   - QDRANT_URI_DOCKER=qdrant
   - DATA_PATH_DOCKER=data
   - USUARI=your-user
   - CONTRASENYA=your-password
   - OPENAI_API_KEY=your-api-key

En l'arrel frontend, s'ha de deixar l'arxiu .env tal com està. **3. Desplegament amb docker compose**

Una vegada es descomprimeix el zip del codi, a la consola s'ha de fer `docker compose up -d --build`.

Això construirà els contenidors dels diferents serveis del backend. El procés pot ser llarg, ja que s'han d'instal·lar varies dependències de Python.

**4. Frontend**

Per arrencar el frontend s'ha de fer la comanda `npm install` i una vegada s'ha completat, fer `npm run dev`

Quan els serveis estigui corrent ho faran a aquestes urls:

Frontend: localhost:5173/

FastAPI: localhost:8000/

Qdrant: localhost:6333/