Desenvolupament d'una plataforma de coneixement col·laborativa per unitats empresarials tecnològiques

Arnau Berenguer

4 de febrer de 2025

Resum– Aquest Treball de Fi de Grau es centra en el desenvolupament i desplegament d'una plataforma personalitzada d'estil wiki per l'empresa Eurecat, amb l'objectiu de gestionar i centralitzar el coneixement intern de manera eficient dins de la unitat de Big Data & Data Science Analytics. La plataforma integra funcionalitats tradicionals de wiki amb un sistema de Retrieval-Augmented Generation (RAG) basat en tècniques d'aprenentatge automàtic i de processament del llenguatge natural (NLP), permetent als usuaris interactuar amb la base de dades de coneixement mitjançant una interfície de xat. Les principals contribucions inclouen l'organització eficient del coneixement, l'ús innovador de tecnologies punteres i una gestió integral del cicle de vida del programari, des de la presa de requeriments fins al desplegament en servidors interns. El treball també planteja millores futures, com la creació de pipelines de desplegament i integració contínua, l'optimització del microservei RAG i l'obertura del projecte a la comunitat Open Source.

Paraules clau– Wiki, gestió del coneixement, Retrieval-Augmented Generation, RAG, aprenentatge automàtic, processament del llenguatge natural, NLP, desenvolupament de software.

Abstract– This final degree project focuses on the development and deployment of a customized wiki-style platform for Eurecat, aimed at efficiently managing and centralizing internal knowledge within its Big Data & Data Science Analytics unit. The platform integrates traditional wiki functionalities with a Retrieval-Augmented Generation (RAG) system powered by machine learning and natural language processing (NLP), enabling users to query the knowledge base interactively via a chat interface. Key contributions include streamlined knowledge organization, innovative use of state-of-the-art technologies, and a complete lifecycle software development approach from requirements gathering to deployment on internal servers. Future work outlines enhancements like continuous deployment pipelines, improved RAG microservice architecture, and open-source community involvement.

Keywords– Wiki, knowledge management, Retrieval-Augmented Generation, RAG, machine learning, Natural Language Processing, NLP, software development.

1 Introducció - Context del treball

A l'actualitat, en plena era de la transformació digital, el coneixement s'ha convertit en un actiu estratègic per les organitzacions, especialment aquelles vinculades al sector tecnològic. Les empreses que aconsegueixen gestionar de manera eficient el seu coneixement intern i promoure la col·laboració entre els seus treballadors, obtenen un avantatge competitiu significatiu en mercats que cada cop es troben més dinàmics i globalitzats. No obstant, però, moltes organitzacions s'encallen davant el desafiament d'intentar estructurar i compartir el coneixement de manera efectiva, generant redundàncies, pèrdua d'informació clau i sobretot una ralentització als processos d'innovació i projectes a executar.

• E-mail de contacte: arnauapps@gmail.com
• Menció realitzada: Enginyeria del Software
• Treball tutoritzat per: Elitza Maneva (Departament de Ciències de la Computació)
• Curs 2024/25

En el cas concret *d'Eurecat - Centre Tecnològic de Catalunya*, la unitat de *Big Data & Data Science Analytics (BD&DSA)* també s'enfronta a aquesta problemàtica. La unitat està dividida en diferents línies segons les àrees d'acció, tals com *Applied Machine Learning, Computational Social Science, Big Data Architectures, etc.* Actualment, no existeix cap mecanisme o eina de transferència de coneixement entre les diferents línies i fins i tot a dins d'aquestes. Es fa de forma tradicional i *rudimentària* a través del boca a boca, demanant ajuda als companys o bé consultant els repositoris de projectes anteriors relacionats.

Anteriorment va existir un projecte intern anomenat *AIWorks*. Dins del marc d'aquest projecte s'abarcava l'intent de creació d'una eina estil *wiki* on poder guardar i compartir tot aquest tipus de coneixement. Malauradament però, aquest projecte no va quallar per diversos motius i es va acabar abandonant. No obstant, sempre ha quedat pendent la idea de millorar la transferència de coneixement i d'aquí surt el context d'aquest treball. Així doncs, l'objectiu d'aquest treball és desenvolupar una aplicació que permeti compartir coneixement de forma senzilla i que sigui un punt central d'informació pels treballadors de la unitat, on es pugui trobar tant informació sobre projectes, com propostes, com consells i tutorials de com fer servir algun tipus de tecnologia.

En aquest document es trobarà la secció d'objectius on s'expliquen els objectius del projecte, la secció d'estat de l'art wiki, on es redacta l'estat de l'art en aplicacions *wiki*, la secció d'estat de l'art de sistemes de *Retrieval-Augmented Generation*, on es redacta també l'estat de l'art de sistemes RAG, la secció de metodologia, on s'explica la metodologia seguida durant el desenvolupament del projecte, la secció de requisits, on es descriuen els requisits del projecte i es categoritzen segons la seva prioritat, la secció de resultats, on s'exposen els resultats del projecte i fins on s'ha pogut arribar, i finalment la secció de conclusions, on es descriuen les aportacions que fa aquest projecte, el nivell d'assoliment i les línies futures.

2 Objectius

Els objectius d'aquest projecte són:

- Desenvolupar una aplicació wiki personalitzada que permeti transferir coneixement de forma eficient i centralitzada.
- Desenvolupar un sistema *RAG* a través de tècniques de *machine learning i NLP* per poder realitzar preguntes sobre la informació continguda a l'aplicació
- Integrar el sistema *RAG* a la pròpia aplicació en forma de xat per facilitar-ne la interacció

3 Estat de l'Art en Wiki

Les aplicacions d'estil *wiki* s'han consolidat al llarg del temps com una eina fonamental per la gestió i la col·laboració envers el coneixement dins les organitzacions. Aquestes plataformes permeten la creació, organització i edició col·laborativa de contingut, facilitant que els equips comparteixin informació de manera estructurada i accessible.

A continuació es descriuen algunes de les aplicacions més rellevants d'aquest àmbit amb un anàlisi de les seves principals característiques.

3.1 MediaWiki

MediaWiki [1] és una de les plataformes més conegudes, utilitzada per alimentar grans pàgines web com *Wikipedia*. És un software de codi obert, el què permet la seva implementació en entorns diferents i certa personalització segons les necessitats de cada entitat.

3.1.1 Característiques principals

1. **Codi obert:** Al ser gratuït i de codi obert, ofereix flexibilitat per adaptar-se a les necessitats de les empreses.
2. **Historial de revisions:** Permet als usuaris veure l'historial de canvis a cada pàgina, facilitant la transparència sobre cada modificació.
3. **Extensibilitat:** Compta amb una àmplia gamma d'extensions que permeten afegir noves funcionalitats, des de controls de permisos fins eines avançades d'anàlisi.
4. **Escalabilitat:** Està dissenyada expressament per tractar amb grans volums de dades, per tant és una opció ideal per projectes de gran envergadura i col·laboracions massives.

3.2 Confluence

Confluence [3], desenvolupat per la companyia *Atlassian* (propietària d'altres aplicatius com *Jira, Trello i Bitbucket*) és una plataforma wiki orientada a empreses que busquen millorar la col·laboració i gestió del coneixement dins d'equips distribuits. S'integra especialment bé amb altres eines pròpies com *Jira i Trello* i està enfocada a entorns de desenvolupament de software. És un software de pagament a través de llicències.

3.2.1 Característiques principals

1. **Integració amb Jira:** Destaca per la seva integració nativa amb Jira, permetent als equips de desenvolupament gestionar tant la documentació com els projectes en un únic entorn.
2. **Plantilles personalitzables:** Ofereix una sèrie de plantilles personalitzades per facilitar la creació de contingut, des de la documentació tècnica fins a informes de projectes.
3. **Permisos i control d'accés:** Inclou una gestió avançada de permisos per definir qui pot veure o editar certs continguts, funció essencial en entorns corporatius.

3.3 Notion

Notion [2] és una eina de productivitat *all-in-one* que combina funcionalitats de *wiki*, gestió de projectes, bases de dades i notes en un únic entorn. El seu enfocament és proporcionar una plataforma flexible on els usuaris puguin personalitzar el seu entorn de treball en funció de les seves necessitats. També es tracta d'un software de pagament per llicències d'ús.

3.3.1 Característiques principals

1. **Versatilitat:** A més de les capacitats de *wiki*, Notion permet als seus usuaris gestionar tasques, crear bases de dades senzilles i realitzar seguiment de projectes, convertint-la en una solució integral de productivitat.
2. **Interfície intuitiva:** La seva interfície intuitiva, polida i fàcil de fer servir permet que els usuaris crein i editin pàgines sense necessitat de coneixements tècnics avançats.
3. **Col·laboració en temps real:** Els equips poden col·laborar en l'edició de pàgines en temps real, amb opcions de comentaris i mencions per facilitar la comunicació.
4. **Plantilles predefinides:** Inclou una àmplia gamma de plantilles preconfigurades que ajuden a organitzar la informació de manera eficient.

3.4 Característiques de l'aplicació

Després d'analitzar les aplicacions *wiki* més conegudes, es va decidir escollir les facetes més important de cadascuna per integrar-les a l'aplicació final:

1. **Interfície intuitiva**: L'aplicació ha de gaudir d'una interfície intuitiva i fàcil de fer servir per l'usuari.
2. **Permisos i controls d'accés**: Es poden definir tres tipus d'usuaris: administrador, normal i no registrat. Els no registrats només podran visualitzar *posts*, els registrats podran crear-ne de qualsevol tipus però només modificar i esborrar els seus i els administradors gaudeixen de tot tipus de permís.
3. **Plantilles predefinides**: Han d'existir un seguit de plantilles que facilitin la creació de diferents tipus de *posts*, com projectes, propostes, articles i eines.

4 Estat de l'Art en RAG

4.1 Introducció als sistemes RAG

Durant els últims anys, la investigació de l'IA ha experimentat un enorme progrés en el desenvolupament de models de llenguatge capaços de generar textos altament coherents i contextualment apropiats. Aquests models, sovintment anomenats Large Language Models (LLM), com GPT-4 o LLama, han demostrat capacitats notables per generar llenguatge similar a llenguatge humà. No obstant, una de les principals limitacions dels LLM és la seva dependència del coneixement present a les seves dades d'entrenament, que pot quedar obsolet o limitat en dominis específics o que no siguin de domini públic. Aquí és on entren en joc els sistemes de recuperació i generació augmentada (RAG per les seves sigles en anglès: Retrieval-Augmented Generation)[17]

Els sistemes RAG tenen com a objectiu millorar el coneixement i la precisió dels LLM mitjançant la integració d'un mecanisme de recuperació extern (retrieval). Permeten que el model obtingui informació rellevant de grans conjunts de dades, bases de dades externes o fins i tot la web en temps real, cosa que augmenta el procés de generació amb coneixement actualitzat i específic del domini. Aquest enfocament híbrid permet una generació de text més precisa, contextual i informativa.

4.2 Arquitectura bàsica dels sistemes RAG

Els sistemes RAG combinen tres components principals:

1. **Indexació**: Aquest component s'encarrega de transfomar els textos que formen part del cos de documents a vectors d'embeddings amb un model embedder i s'indexen a una base de dades vectorial que permeti la cerca de documents amb operacions vectorials.
2. **Retriever**: Aquest component busca i recupera informació relevant d'un corpus extern en funció de la consulta d'entrada.
3. **Generador**: El generador, normalment un LLM, processa la informació recuperada en combinació amb la consulta d'entrada per produir una resposta final i coherent

L'arquitectura general es pot descriure de la següent manera:

- **Indexació**: S'indexen els documents sobre els que es respondran les consultes dels usuaris.
- **Entrada**: L'usuari proporciona una consulta o un missatge al sistema RAG.
- **Fase de recuperació**: El sistema primer envia aquesta consulta a un mecanisme de recuperació que busca documents o fragments rellevants a la base de documents indexada.
- **Creació del prompt**: Es combina la pregunta de l'usuari amb la informació retornada pel retriever juntament amb unes instruccions de com ha de respondre l'LLM, utilitzant les capacitats de resposta d'un LLM entrenat.
- **Fase de generació**: El prompt s'introdueix al generador (LLM), que processa tant la consulta com la informació context recuperada i genera una resposta adeqüada.

Aquest procés de tres components essencials permet que els sistemes RAG funcionin millor que els LLM sols, especialment en tasques que requereixen accés a coneixement actualitzat, especialitzat o privat, com respondre preguntes factuals o proporcionar informació basada en dades externes en temps real.

4.3 Fase de retrieval

La fase de retrieval o recuperació en els sistemes RAG implica tècniques similars als motors de cerca normals, on l'objectiu és trobar documents o *chunks* de documents que siguin més rellevants per la consulta de l'usuari. El principal mètode de recuperació que s'empra als sistemes RAG és la recuperació densa: la recuperació densa utilitza models d'aprenentatge profund per codificar la consulta en un vector dens (embedding) de les mateixes característiques que els documents indexats per després comparar els vectors i recuperar els documents que tenen una similitud més elevada amb el vector de la consulta.

Per trobar quins són els embeddings amb major similitud, es fa servir la similitud del cosinus. Es tracta d'una mètrica molt utilitzada en l'àmbit del processament del llenguatge natural (NLP) que serveix per determinar com de semblants són dos vectors en un espai multidimensional. Aquesta mètrica és especialment útil per comparar textos que han estat convertits a embeddings. La similitud del cosinus calcula el cosinus de l'angle entre dos vectors. Aquesta mètrica té un rang entre -1 i 1, on:

- 1 indica que els dos vectors són exactament iguals (l'angle entre ells és de 0 graus).
- 0 indica que els vectors són ortogonals (l'angle entre ells és de 90 graus), suggerint que no hi ha relació aparent.
- -1 indica que els vectors són diametralment oposats (l'angle és de 180 graus).

Es recuperen llavors els documents amb puntuació més alta i es passen com a entrada a la fase de generació.

4.4 Fase de generació

Un cop que es recuperen els documents rellevants, comença la fase de generació. L'LLM encarregat d'aquesta fase pren tant la consulta com els documents com a entrada. Això permet que el model generi una resposta que integra la informació trobada en els documents recuperats.

Aquest procés de generació generalment implica els següents passos:

1. **Creació del prompt**: El primer pas és crear un prompt ben estructurat que combini la consulta d'entrada de l'usuari amb els documents recuperats. El prompt ha d'estar dissenyat de manera que ajudi a l'LLM a comprendre la tasca, el context i la rellevància del contingut recuperat, es proporciona al model les instruccions que ha de seguir per respondre, sovint s'afegeixen restriccions de longitud i concreció de la resposta, el to desitjat, com ha d'actuar si no és possible respondre, etc. Aquest pas és vital perquè la qualitat i claredat del missatge afecten directament al resultat. En molts casos, es requereix d'una fase de *prompt engineering* per optimitzar el rendiment del model. Per exemple, el prompt pot incloure instruccions específiques pel model sobre com manipular les dades recuperades o enfatitzar certs fets.
2. **Contextualització**: Després de generar el prompt, l'LLM el processa juntament amb la informació recuperada. Ha d'integrar la consulta original de l'usuari amb el context addicional proporcionat pels documents. Això permet a l'LLM refinar la seva comprensió de la tasca i generar una resposta més informada i rellevant.
3. **Síntesis**: Per últim, l'LLM sintetitza la consulta i les dades recuperades per produir un resultat coherent. Aquest procés de síntesis aprofita la informació recuperada per garantir que la resposta estigui basada en fets.

4.5 Avantatges dels sistemes RAG

Els sistemes RAG ofereixen vàris avantatges sbre els models purament generatius, entre ells:

- **Precisió factual**: Un dels majors desafiaments dels LLM és que poden 'al·lucinar' informació, és a dir, generar respostes que sonen plausibles però incorrectes en els fets. Els sistemes RAG mitiguen aquesta problemàtica basant les seves respostes en informació real recuperada externament.
- **Coneixement específic del domini**: Els sistemes RAG es poden ajustar per buscar bases de dades o corpus específics del domini. Per exemple, un sistema RAG emprat en una aplicació mèdica podría recuperar dades de revistes mèdiques o bases de dades de medicaments, proporcionant informació altament precisa i especialitzada.
- **Accés a informació actualitzada**: Els LLM tradicionals estan limitats pel coneixement que van adquirir durant l'entrenament, el què significa que no poden accedir a la informació que s'ha produit després d'aquest. Pel contrari, els RAG poden recuperar informació en temps real de fonts externes, cosa que els fa molt útils per tasques com respondre preguntes recents, actualitzacions de notícies o suport al client en temps real.

4.6 Principals limitacions dels sistemes RAG

Tot i els seus avantatges, els sistemes RAG no estan exempts de desafiaments:

- **Qualitat de la recuperació**: La qualitat de la resposta final depèn en gran mesura de la qualitat dels documents recuperats. Si es recuperen documents irrellevants o de baixa qualitat, la resposta generada també patirà en precisió i rellevància. Millorar l'eficàcia dels mecanismes de recuperació és un àrea clau d'investigació en curs.
- **Integració del contingut recuperat**: Integrar eficaçment el contingut recuperat el el procés de generació no és una tasca fàcil. A vegades l'LLM pot tenir dificultats per sintetitzar correctament la informació de múltiples fonts o tractar informació contradictòria en els documents recuperats.

- **Complexitat del prompt engineering**: Dissenyar un prompt eficaç no sempre és senzill. El procés de prompt engineering sovint requereix una experimentació significativa per obtenir el resultat desitjat. Aquesta complexitat augmenta quan es canvia de model, ja que els prompts que funcionen bé per un model, podria ser que no funcionéssin bé en un altre.

5 METODOLOGIA

5.1 Organització

Per organitzar el desenvolupament d'aquest treball, s'ha seguit una metodologia *Agile* adaptada al context d'un Treball de Fi de Grau, en tant que només s'ha adoptat els conceptes *Scrum* de *sprint, epics i tasques*. A la següent taula es poden veure com queden els *sprints* repartits:

Sprint 1	Sprint 2	Sprint 3
Presa de requeriments	Definició dels models de la BBDD	Millores a la interfície d'usuari
Anàlisi d'eines similars	Definició i creació d'endpoints	Implementació de la interfície del xat RAG
Selecció de *l'stack* tecnològic	Desenvolupament de les vistes dels endpoints	Dockerització de l'aplicació
	Desenvolupament del microservei RAG	Desplegament al servidor
	Addició de reactivitat amb framework frontend	

TAULA 1: TAULA PLANIFICACIÓ DE SPRINTS

Per al repartiment de les tasques s'ha utilitzat l'eina organitzativa de metodologia *Kanban*, *Trello*, d'Atlassian (vegeu Figura 1).

5.2 Control de versions i política de branques

S'ha fet servir *Git* com a tecnologia de control de versions, degut a que és l'estàndard a l'indústria del Software. Com a política de branques s'ha fet servir la següent

1. **Main**: Branca final on només arriben els canvis finals testejats de la branca *develop*.
2. **Develop**: Branca de desenvolupament principal. Es fusionaran els canvis a la branca *main* un cop acabats i testejats.

6 REQUISITS

Els requisits del sistema recollits al llarg de la fase de presa de requeriments de l'*sprint 1* es poden veure a la Taula 3. Aquests requisits s'han classificat també segons la metodologia MoSCoW [18], la qual classifica requisits segons la seva prioritat dins el projecte. Es classifiquen de la següent forma:

- **(M)ust have**: obligatori pel bon funcionament del sistema.
- **(S)hould have**: afegeix valor a l'aplicació i hauria d'existir al sistema.
- **(C)ould have**: afegeix valor a l'aplicació però no afecta al funcionament la seva absència.
- **(W)on't have**: no és necessari, però si es gaudeix de temps suficient i no implica molt esforç, es podria afegir.

Must	Should	Could	Won't
RF-00	RF-06	RF-14	RF-15
RF-01	RF-09		
RF-02	RF-10		
RF-03	RF-13		
RF-04	RF-16		
RF-05			
RF-07			
RF-08			
RF-11			
RF-12			
RF-17			
RF-18			

TAULA 2: TAULA DE REQUISITS MOSCOW

7 RESULTATS

En aquesta secció es descriuen els resultats finals del desenvolupament del projecte, ordenats per els tres *sprints* esmentats.

7.1 Sprint 1

El primer *sprint* va abarcar la presa de requeriments amb els meus superiors:

Identificador	Nom	Descripció
RF-00	Tecnologia	S'ha de fer servir el framework de Ruby on Rails, Laravel, Django o Phoenix
RF-01	Creació d'articles	S'ha de poder crear i penjar articles
RF-02	Eliminació d'articles	S'ha de poder eliminar articles anteriorment penjats
RF-03	Edició d'articles	S'ha de poder editar articles anteriorment penjats
RF-04	Visualització d'articles	S'ha de poder visualitzar articles anteriorment penjats
RF-05	Registre d'usuaris	S'ha de poder registrar usuaris i fer log in

Identificador	Nom	Descripció
RF-06	Usuari administrador	El rol d'usuari administrador pot fer-ho totes les accions
RF-07	Usuari registrat	L'usuari registrat pot visualitzar, crear i editar els seus articles
RF-08	Usuari no registrat	L'usuari no registrat només pot visualitzar articles
RF-09	Markdown	Els articles s'escriuran en format Markdown per facilitar l'escriptura
RF-10	Markdown preview	A mesura que s'escrigui un document en MD, s'ha d'anar visualitzant el resultat final
RF-11	Funcionalitat de cerca	Ha d'existir un cercador que permeti buscar els articles segons les paraules buscades
RF-12	Editor d'articles	S'ha de poder editar articles ja fets a través d'un editor Markdown
RF-13	Categories	Els articles han d'estar ordenats per categories per facilitar la seva cerca
RF-14	Àmbit	Els articles han d'estar classificats pels àmbits d'actuació d'Eurecat per facilitar la seva cerca
RF-15	Endpoint per crear articles	S'haurà de proporcionar un endpoint al que penjar fitxers markdown i els camps necessaris per crear articles remotament
RF-16	Barra de navegació	Ha d'existir una barra de navegació que permeti escollir la categoria que es vol veure
RF-17	RAG	Existirà un microservei RAG per poder fer consultes sobre els documents de la wiki
RF-18	Xat	Hi haurà d'haver un xat amb el que interactuar amb el microservei RAG

TAULA 3: TAULA PRELIMINAR DE REQUISITS

També va abarcar l'anàlisi de l'estat de l'art, i finalment la selecció de l'stack tecnològic a fer servir, que va ser el següent:

- **Aplicació**: Ruby on Rails [6], per la seva rapidesa de desenvolupament i bona experiència de desenvolupament.
- **Estilatge**: TailwindCSS [7], un framework estàndard de CSS avui en dia que facilita molt l'estilatge i té una comunitat de llibreries de components oberts molt gran.
- **Reactivitat**: Stimulus [8], un framework de JavaScript desenvolupat per l'equip de Rails dissenyat per afegir reactivitat al front-end i facilitar-ne la seva integració.

Pel sistema *RAG*, es va escollir Python com a llenguatge degut a la seva omnipresència al món de l'aprenentatge automàtic i la ciència de dades i el gran nombre de llibreries que existeixen. Les llibreries que es van fer servir són les següents:

- **Transformers** [9]: Es tracta de la llibreria de HuggingFace per interactuar amb tot tipus de models basats en l'arquitectura del *Transformer*. Es farà servir per poder interactuar amb l'LLM.
- **LangChain** [10]: Es tracta d'una llibreria que actua de *wrapper* per sobre de moltes altres llibreries i facilita la interacció entre BBDDs vectorials, *embedders* i LLMs.
- **ChromaDB** [11]: Es tracta d'un client per interactuar amb la base de dades Chroma, una base de dades vectorial on es guardaran els documents de text de la wiki.
- **FastAPI** [12]: Es tracta d'una llibreria per desenvolupar APIs en Python de manera molt ràpida i és la que s'usa *de facto* a la unitat BD&DSA per exposar APIs de models.

7.2 Sprint 2

Un cop acabat el primer sprint, es va començar el segon, que abarcava el desenvolupament en si de l'aplicació wiki i del sistema de *Retrieval-Augmented Generation*.

7.2.1 Wiki

El primer que es va fer va ser definir els models de la base de dades que s'utilitzarien a l'aplicació per poder tenir una bona idea de com estructurar l'aplicació a partir d'ells. Els models escollits van ser els següents:

1. Usuari (vegeu Taula 4)
2. Article (vegeu Taula 5)

3. Categoria (vegeu Taula 6)
4. Projecte (vegeu Taula 7)
5. Proposta (vegeu Taula 8)
6. Eina (vegeu Taula 9)

Un cop definits, es va procedir a realitzar el que es coneix com a *CRUD (Create-Update-Read-Delete)* d'un model, és a dir implementar els models com a recursos HTTP i la seva lògica de funcionament. A grans trets, seguint el patró Model-Vista-Controlador, que és en el què es basa *Rails* això va implicar definir-ne els models i els seus atributs i relacions, implementar el controlador amb la lògica corresponent a cada mètode i definir les vistes *[index, view, edit]* de cada model. (Vegeu la referència [19] per a un exemple d'un controlador)

Posteriorment, per poder desenvolupar la cerca per qualsevol ítem dels anteriors, el què es va fer va ser crear un controlador que contingués només un mètode encarregat de realitzar *queries SQL* per trobar els resultats que continguin en algun dels seus camps el contingut de la cerca i retornar-los. [20]

A l'hora d'implementar l'autenticació dels usuaris, es va fer servir la popular gemma *Devise* [4], un estàndard avui en dia per fer autenticació d'usuari amb Rails.

Seguidament, va caldre desenvolupar un sistema de validacions per impedir que un usuari sense els permisos adeqüats realitzi certes accions (p.e. esborrar un article del qual no n'és l'autor). Es va fer servir la gemma *Pundit* [5], que permet implementar polítiques d'accés als diferents recursos de l'aplicació. [21]

Finalment, un cop acabada la part més relacionada amb el *backend*, es va començar amb el sistema de renderitzat dels articles, projectes, propostes i eines *(frontend)*.
Quan es crea un article/projecte/proposta/eina, es fa amb l'eina *Action Text* de Rails, que permet afegir elements Markdown i proporciona un editor interactiu directament. Un cop escrit el text, els camps es tradueixen automàticament a HTML pur per poder-los renderitzar, a través d'una funció que s'executa abans d'inserir el text a la base de dades. Aquesta funció agafa els atributs del model i ho formateja en un fitxer Markdown i s'envia a la API del RAG per a poder-lo indexar. [22]

També es va implementar una pàgina en que es mostra el perfil de cada usuari, on es pot observar els permisos que aquest té, i els articles, projectes, propostes i eines que ha publicat. [23]

Per últim, es va implementar reactivitat amb *Stimulus* per poder animar el plegament i desplegament de la barra de navegació. Stimulus proporciona una forma molt senzilla d'enllaçar el controlador amb un element HTML i poder-lo manipular des del fitxer *JavaScript*. En aquest cas només va caldre canviar les classes que es renderitzaven amb l'element en funció de si es clicava un botó o no, afegint o traient la classe *hidden*. [24]

7.2.2 RAG

El primer que es va fer a l'hora de desenvolupar el sistema *RAG* va ser definir-ne l'arquitectura.
L'arquitectura bàsica de *Retrieval-Augmented Generation* que es va definir és senzilla i només gaudeix de tres simples mòduls essencials:

1. Indexació
2. Retrieval
3. Generation

Es poden afegir més mòduls entremig d'aquests en un futur, com la traducció de preguntes, reformulació de preguntes, reformulació del context, gestor de memòria, enrutament de continguts, etc, però pel marc d'aquest Treball de Fi de Grau es va decidir implementar l'arquitectura més simple i essencial, i un cop acabat el treball, expandir l'arquitectura amb l'addició de més mòduls.

Primer de tot es va haver de decidir quin model *d'embeddings* i quin LLM es volia fer servir per al projecte. Es va decidir fer servir *l'embedder'all-MiniLM-L6-v2'*[13] perquè és un dels *embedders* més utilitzats arreu del món en l'actualitat degut als seus bons resultats i que a més a més és un model lleuger (aproximadament 500MB). D'altra banda, es va decidir fer servir l'LLM *'stablelm-zephyr-3b'* [14] perquè també es tracta d'un model lleuger de només 3 bilions de paràmetres i entrenat per rebre instruccions.

El mòdul d'indexació fa servir ChromaDB perquè és una base de dades vectorial que funciona de manera local a través de *Sqlite3* els *wrappers* de *LangChain* faciliten molt el seu desplegament. Aquest mòdul funciona de la següent manera:

1. Es rep una petició a *l'endpoint* **/index** amb un document de text adjunt al cos de la petició.
2. El document es divideix en diversos *chunks* mitjançant un *text splitter*.
3. Es generen els *embeddings* de cada *chunk* i s'indexen per separat a la base de dades.

El funcionament del mòdul de retrieval és molt senzill:

- Rep una petició a *l'endpoint* **/chat** amb el paràmetre *query*.
- Transforma la *query* en un vector *d'embeddings* i realitza una cerca per similitud de cosinus a tots els documents de la base de dades.
- Recupera els *k* documents que hagin obtingut una puntuació de similitud més alta.

El mòdul de generació també és molt senzill, en tant que l'únic que fa és omplir el següent *prompt* i passar-li a l'LLM per obtenir-ne la resposta:

You are a highly knowledgeable and detail-oriented assistant specializing in answering questions based on specific context documents. Your expertise includes office bureaucracy and a deep understanding of the Big Data

& Data Science Analytics team at Eurecat, a prominent research center.
The provided documents contain critical information and are categorized as follows:

- **Articles: Insightful write-ups on technologies and methodologies beneficial to the team.**
- **Projects: Detailed information about past projects undertaken by the team.**
- **Proposals: Comprehensive outlines of various project proposals developed by the team.**
- **Tools: Descriptions of tools designed and produced by the team.**

Your task is to carefully analyze the provided documents and respond to the following question based exclusively on the relevant content within these documents. Avoid making assumptions or drawing conclusions beyond the given information.
Here are the relevant documents: {documents}
Please answer the following question with clear and accurate detail: {pregunta}

El codi del microservei RAG es pot trobar a: [26]

7.3 Sprint 3

Finalment, es va realitzar el tercer *sprint*, que consistia en realitzar millores a la interfície d'usuari, implementar la interfície per xatejar amb el RAG i la dockerització i desplegament de l'aplicació.

7.3.1 Xat

Primer de tot es va implementar un nou controlador amb un únic mètode que rep com a cos de la petició de tipus POST un missatge, el reenvia a l'API del RAG i retorna el què l'API li ha retornat, és a dir la resposta a la pregunta. Després es va implementar una vista amb una caixa de missatges i un formulari que accepta missatges com a input, que crida al mètode del controlador anterior i mostra a la caixa el missatge i la resposta del RAG.

7.3.2 Millores de la interfície gràfica

Inicialment, s'havia desenvolupat l'aplicació de forma senzilla i orientada a ser plenament funcional, sense dedicar molta atenció a l'aspecte físic de la web. Un cop es va tenir l'aplicació funcionant, va ser hora de canviar els estils *CSS* a través de *TailwindCSS*[7] per donar-li una estètica moderna i agradable a la vista.
S'ha seguit un estil que busca imitar la simplicitat dels components de *shadcn/ui*[16], però com que no es troben disponibles per a Ruby on Rails, s'han hagut d'implementar manualment. A les figures 2, 3, 4, 5, 6 es pot observar la *UI* actual.

7.3.3 Dockerització

Amb l'objectiu de simplificar el desplegament de l'aplicació, es va utilitzar Docker[15], una eina que permet empaquetar aplicacions i les seves dependències en contenidors lleugers, portables i reproduïbles.
Docker és una plataforma de virtualització a nivell de sistema operatiu que utilitza contenidors, els quals són entorns aïllats que contenen tot el necessari per executar una aplicació, incloent-hi el sistema operatiu base, les biblioteques requerides i el codi de l'aplicació. A diferència de de les màquines virtuals tradicionals, els contenidors comparteixen el *kernel* del sistema operatiu de l'amfitrió, fet que els fa significativament més lleugers i ràpids d'iniciar i per tant una gran eina per desplegar aplicacions.
La dockerització de l'aplicació va implicar la creació d'un fitxer *Dockerfile* [25], on es va especificar:

- **Imatge base**: Una versió lleugera del sistema operatiu amb les eines necessàries, en aquest cas Debian amb Ruby preinstal·lat.
- **Dependències**: Totes les biblioteques i paquets que l'aplicació requereix.
- **Codi font**: El propi codi font de l'aplicació necessari per executar-se.
- **Ordres de configuració i execució**: Instruccions i algunes variables d'entorn per configurar i executar l'aplicació dins el contenidor.

Un cop construït el contenidor mitjançant l'ordre *'docker build -t rb-wiki .'*, l'aplicació ja va estar llesta per desplegar de manera consistent en qualsevol entorn on Docker estigui instal·lat.

7.3.4 Desplegament

El desplegament de l'aplicació es va realitzar a un servidor intern de l'empresa, accessible només a través d'una VPN per tal de que només els treballadors d'Eurecat puguin accedir-hi. El fet de que no sigui accessible des d'internet va fer que no fos necessària la presència d'un *reverse proxy* com *NGINX o Apache* que redirigeixin les peticions entrants al servidor de l'aplicació. De la mateixa manera, a l'estar exposat només a connexions de confiança, no va fer falta servir l'aplicació mitjançant el protocol HTTPS.

8 CONCLUSIONS

Aquest Treball de Final de Grau ha consistit en el disseny, desenvolupament i implementació d'una aplicació estil wiki personalitzada per l'empresa Eurecat amb l'objectiu de gestionar i centralitzar el coneixement intern de manera més eficient. Aquest projecte ha abarcat des de la conceptualització inicial fins el desplegament final als servidors interns, cobrint totes les fases del cicle de vida d'un software. A continuació s'exposen les principals aportacions, els aspectes no tractats i les possibles millores futures:

8.1 Aportacions

1. **Organització centralitzada del coneixement**: L'aplicació permet emmagatzemar i gestionar diversos tipus

de contingut, incloent articles, projectes, propostes i documentació sobre eines internes, facilitant l'accés i la reutilització d'informació clau per l'empresa. S'han assolit a dia d'avui 17 dels 18 requisits funcionals declarats a l'inici del projecte.

2. **Integració d'un sistema RAG**: S'ha desenvolupat també un microservei de Retrieval-Augmented Generation que permet als usuaris interactuar amb els documents de manera àgil per resoldre dubtes específics, fent ús de les tecnologies més punteres actualment i tècniques de NLP avançades.

3. **Procés integral del projecte**: Des de la definició dels requisits i disseny d'arquitectura, fins al desenvolupament, proves i desplegament, el projecte ha estat executat de forma autònoma i alineada amb les necessitats identificades per l'empresa Eurecat.

8.2 Elements no assolits

1. **Endpoint per penjar articles de forma externa**: Al RF-15 (Taula 3), s'especifica que un dels requisits és que existeixi un endpoint que permeti la pujada de fitxers Markdown que continguin articles per facilitar la seva creació sense necessitar obrir la interfície gràfica web de l'aplicació. A dia d'avui no s'ha pogut assolir per falta de temps i per prioritzar altres requisits, ja que aquest s'ha considerat *won't have*. Tot i així, s'espera que en un futur s'acabi d'implementar.

8.3 Línies futures

Tot i que el projecte en si és bastant complet i funcional, cap projecte és perfecte i sempre hi ha apartats on es pot innovar o millorar.

1. **Automatització del manteniment**: Actualment, el manteniment i les actualitzacions estan pensats per fer-se de forma manual, és a dir, descarregar els canvis del repositori manualment a través d'SSH i reiniciar el servei amb els nous canvis. Seria molt convenient poder dissenyar pipelines d'integració i entrega contínua (CI/CD) per automatitzar els desplegaments quan es facin canvis al codi.

2. **Refactorització del microservei RAG**: Tot i que el microservei RAG funciona correctament, actualment funciona a través de la llibreria *LangChain* [10], que tot i que sigui una llibreria puntera i actual, està subjecta a molts canvis regulars i és més útil per fer proves de concepte que no pas per sistemes orientats a producció real. Seria interessant poder implementar des de zero tota la pipeline de RAG, intentant tenir el menor nombre de dependències per poder tenir control i configuració total sobre el seu funcionament. Seria molt clau poder afegir el component de memòria de la conversa per tenir en compte missatges anteriors, ja que dins dels marges del TFG no s'ha considerat. També seria interesant afegir les noves tècniques punteres de Retrieval o Generation que vagin apareixent al llarg del temps.

3. **Manteniment continuat**: Aquest projecte es tracta d'una solució personalitzada per Eurecat, i la idea és que segueixi més enllà del marc d'aquest treball, i que per tant es segueixi donant suport a l'hora d'arreglar *bugs* i s'afegeixin funcionalitats a mesura que s'en necessitin de noves.

4. **Open Source**: Tot i que aquest projecte és completament adaptat a Eurecat, segueix essent una eina bastant genèrica que podria arribar a fer-se servir per altres empreses. Si Eurecat en donés permís, seria interessant obrir-la a la comunitat Open Source perquè més gent hi pugui col·laborar i millorar-la de forma continuada.

Agraïments

M'agradaria donar les gràcies primerament al meu amic i tutor a l'empresa Joan Albert Erráez Castelltort i al meu cap Cirus Iniesta Carreras per proporcionar-me la idea de realitzar aquest projecte i la seva confiança, els requeriments necessàris i l'ajuda i suport que m'han donat. També vull agraïr la col·laboració de la meva tutora Elitza Maneva, que m'ha brindat ajuda i bones idees per millorar el projecte en tot moment. Finalment, agraïr a la Maria Font López els seus consells, ànims i suport durant el transcurs del projecte.

Referències

[1] MediaWiki, https://www.mediawiki.org/wiki/MediaWiki

[2] Notion, https://www.notion.so/

[3] Confluence, https://www.atlassian.com/es/ software/confluence

[4] Devise, https://github.com/heartcombo/devise

[5] Pundit, https://github.com/varvet/pundit

[6] Rails, https://rubyonrails.org/

[7] TailwindCSS, https://tailwindcss.com/

[8] Stimulus, https://stimulus.hotwired.dev/

[9] Transformers, https://huggingface.co/docs/ transformers/index

[10] LangChain, https://www.langchain.com/

[11] ChromaDB, https://www.trychroma.com/

[12] FastAPI, https://fastapi.tiangolo.com/

[13] all-MiniLM-L6-v2, https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[14] Stablelm-zephyr-3b, https://huggingface.co/stabilityai/stablelm-zephyr-3b

[15] Docker, https://docs.docker.com/

[16] ShadCN, https://ui.shadcn.com/

[17] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kuttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. ArXiv, abs/2005.11401.

[18] Mètode MoSCoW, https://www.agilebusiness.org/dsdm-project-framework/moscow-prioririsation.html

[19] Control·lador d'articles, https://github.com/NoOPeEKS/rb-wiki/blob/develop/app/controllers/articles_controller.rb

[20] Control·lador de cerca, https://github.com/NoOPeEKS/rb-wiki/blob/develop/app/controllers/search_controller.rb

[21] Política d'accés, https://github.com/NoOPeEKS/rb-wiki/blob/develop/app/policies/article_policy.rb

[22] Model d'article, https://github.com/NoOPeEKS/rb-wiki/blob/develop/app/models/article.rb

[23] Vista del perfil, https://github.com/NoOPeEKS/rb-wiki/blob/develop/app/views/users/show.html.erb

[24] Control·lador Stimulus, https://github.com/NoOPeEKS/rb-wiki/blob/develop/app/javascript/controllers/sidebar_controller.js

[25] Dockerfile, https://github.com/NoOPeEKS/rb-wiki/blob/develop/Dockerfile

[26] Repositori del projecte, https://github.com/NoOPeEKS/rag-microservice

APÈNDIX

A.1 Taules

Usuari	
Atribut	**Tipus**
nom	string
email	string
admin	bool
articles	relació
projecte	relació
proposta	relació
eina	relació

TAULA 4: TAULA ATRIBUTS MODEL USUARI

Article	
Atribut	**Tipus**
títol	string
autor	relació
categoria	relació
cos	text

TAULA 5: TAULA ATRIBUTS MODEL ARTICLE

Categoria	
Atribut	**Tipus**
nom	string

TAULA 6: TAULA ATRIBUTS MODEL CATEGORIA

Projecte	
Atribut	**Tipus**
títol	string
tipus	string
call	string
domini	string
data inici	data
data finalització	data
clients	string
unitats involucrades	string
persones involucrades	string
pressupost	integer
pressupost eurecat	integer
link pressupost	string
link repositori	string
link deliverables	string
fonts de dades	string
paraules clau	string
descripció	text
resum resultats	text

TAULA 7: TAULA ATRIBUTS MODEL PROJECTE

Proposta	
Atribut	**Tipus**
títol	string
tipus	string
call	string
domini	string
data final	data
duració	integer
clients	string
unitats involucrades	string
persones involucrades	string
pressupost	integer
pressupost eurecat	integer
link pressupost	string
link document	string
paraules clau	string
descripció	text

TAULA 8: TAULA ATRIBUTS MODEL PROPOSTA

Eina	
Atribut	**Tipus**
títol	string
tipus	string
data	data
unitats involucrades	string
link repositori	string
link document	string
tecnologies	string
paraules clau	string
ús	text

TAULA 9: TAULA ATRIBUTS MODEL EINA

A.2 Figures

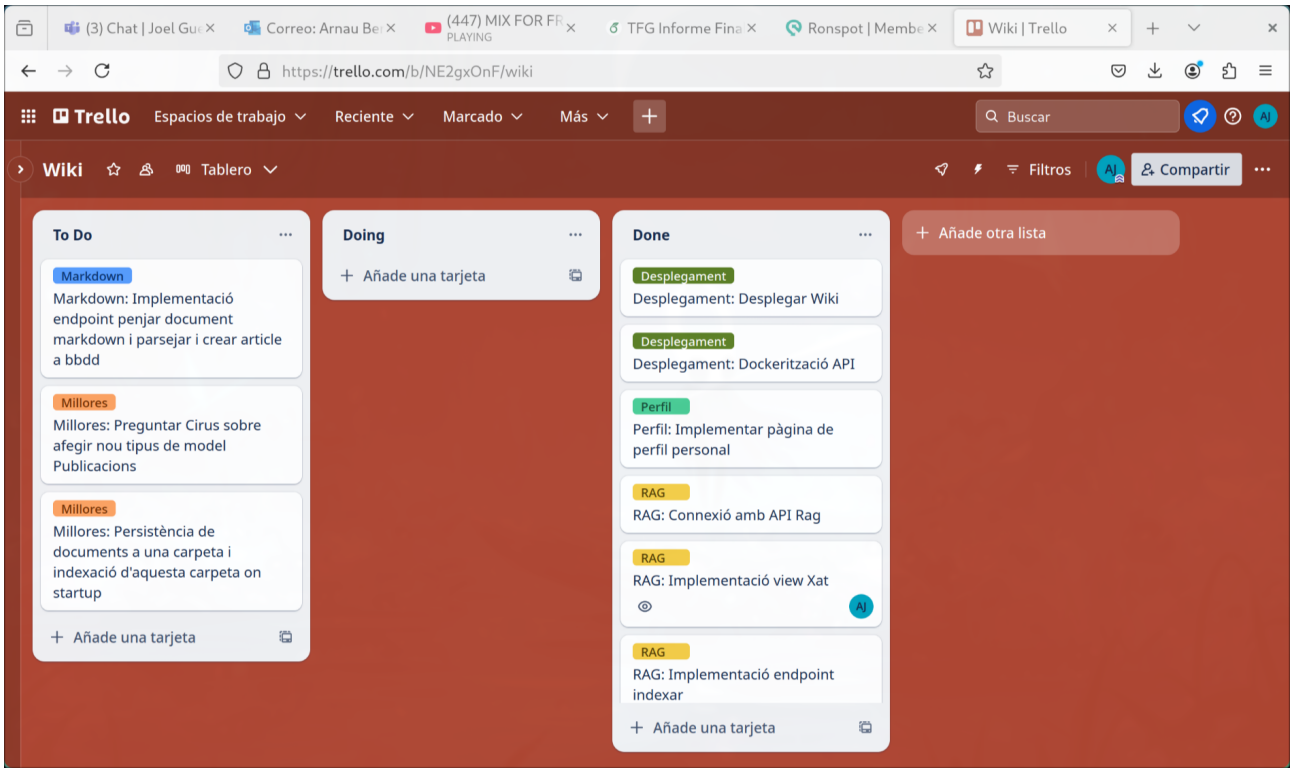

Fig. 1: VISTA DE LES TASQUES TRELLO

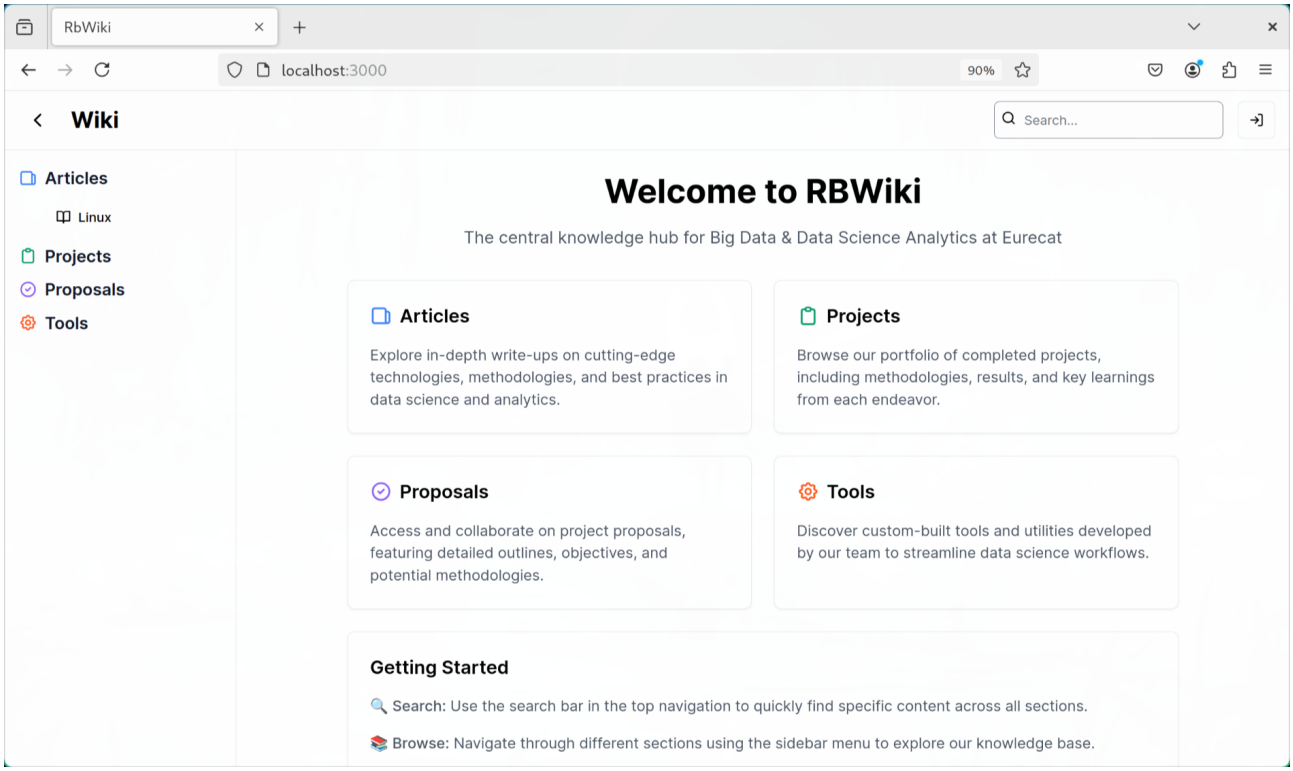

Fig. 2: PÀGINA INICIAL DE LA WIKI

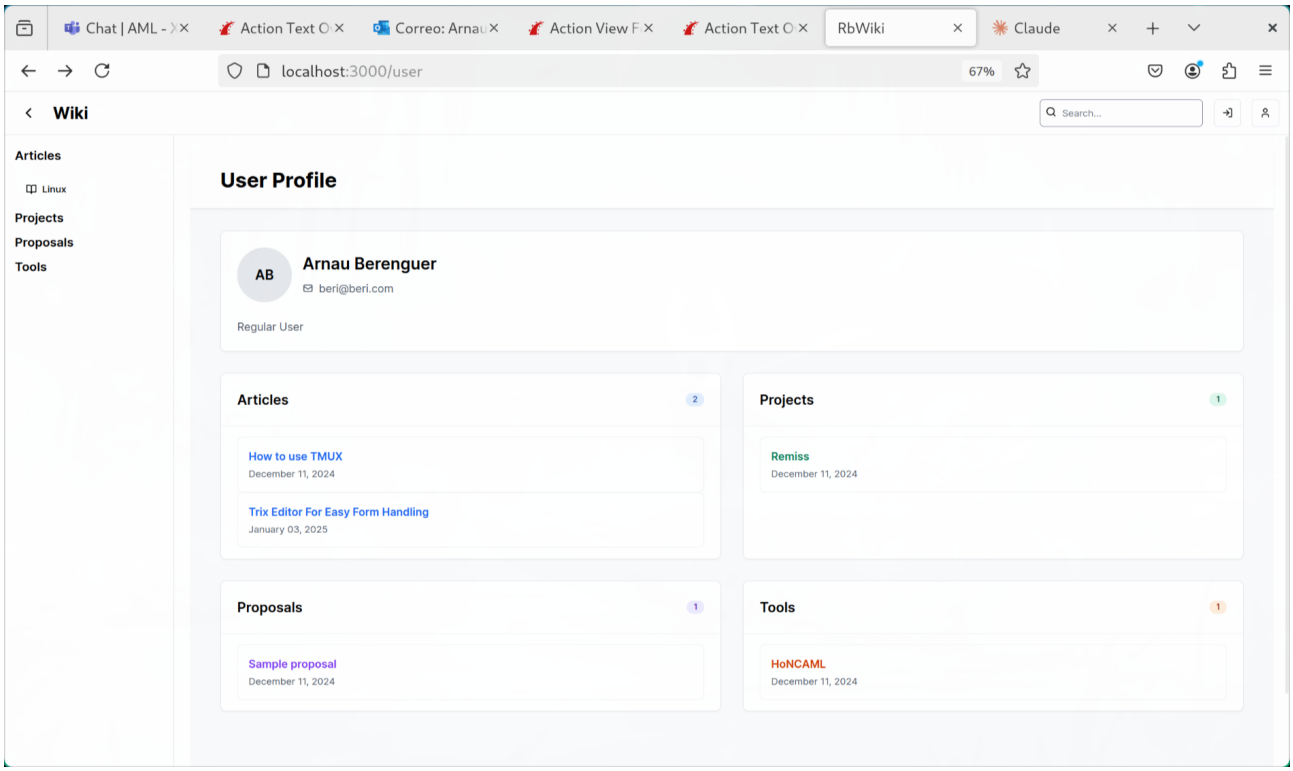

Fig. 3: VISTA DEL PERFIL D'UN USUARI

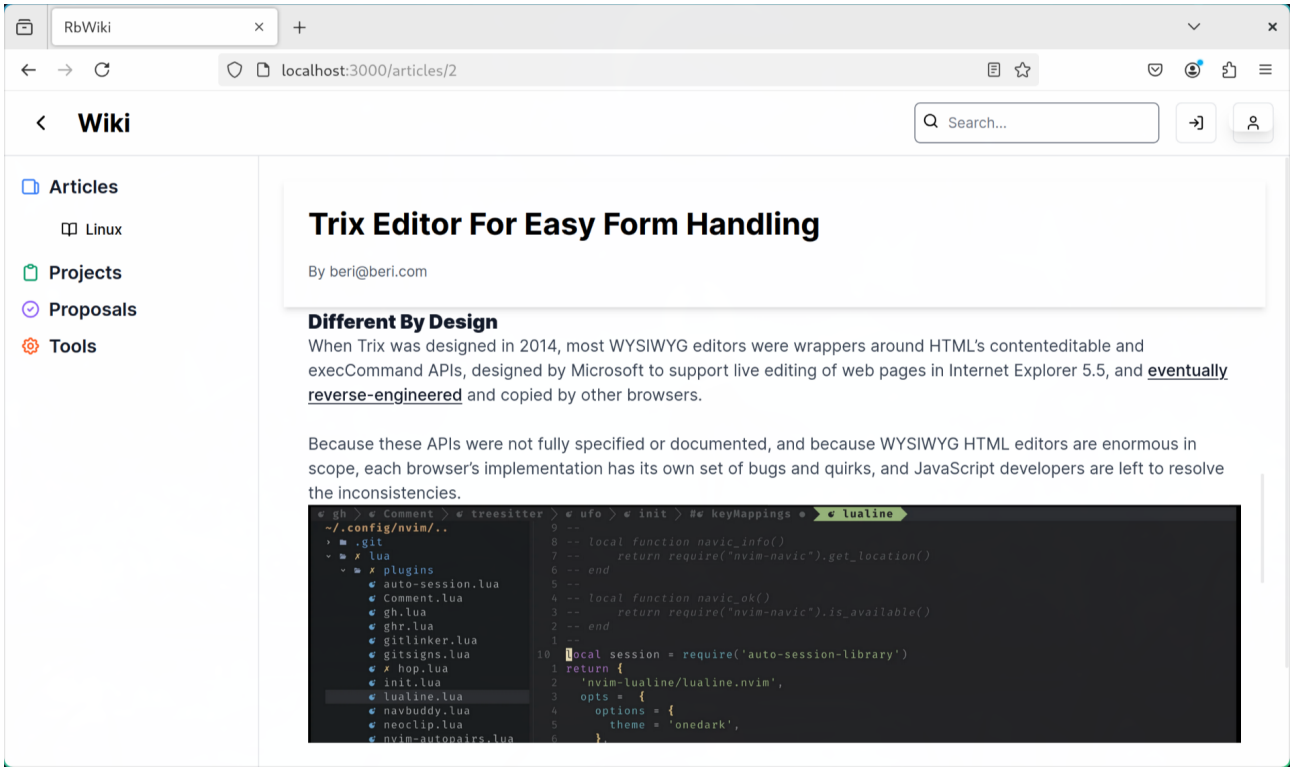

Fig. 4: VISTA D'UN ARTICLE DE LA WIKI

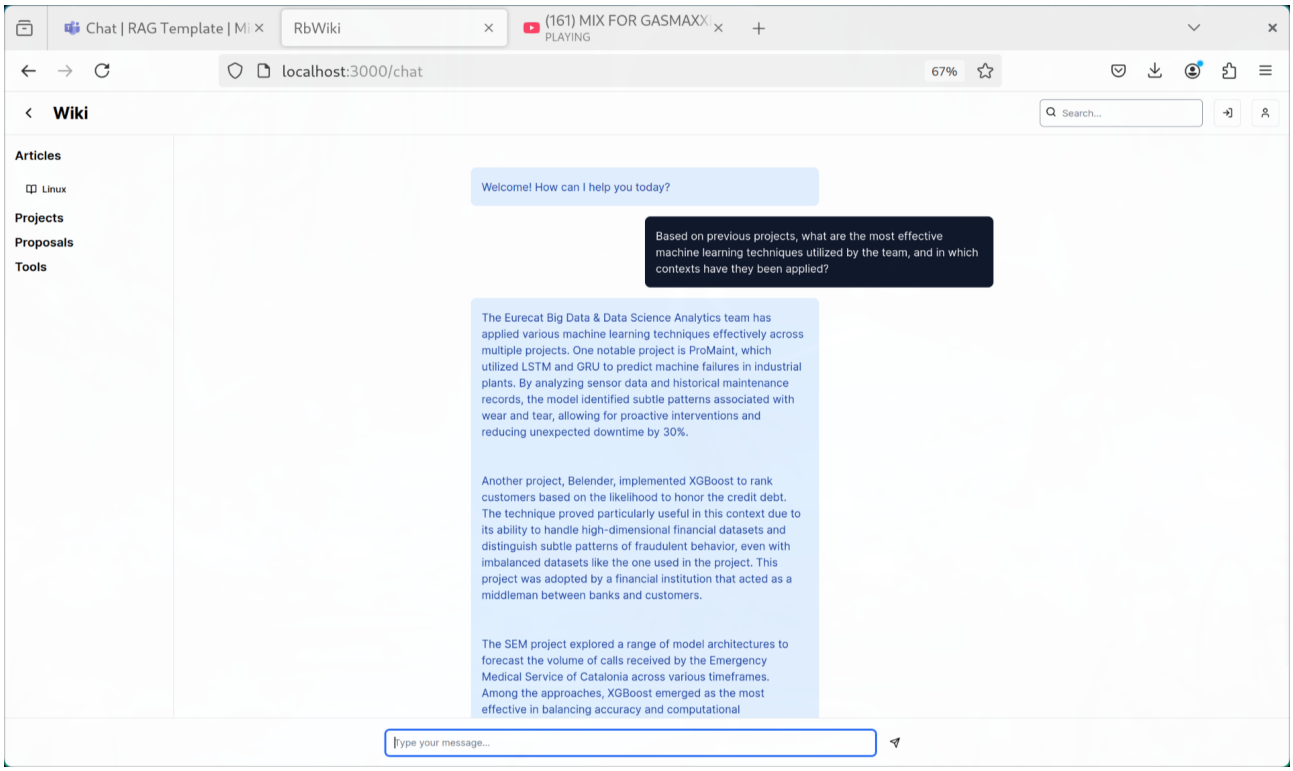

Fig. 5: VISTA DEL XAT ENTRE USUARI I RAG

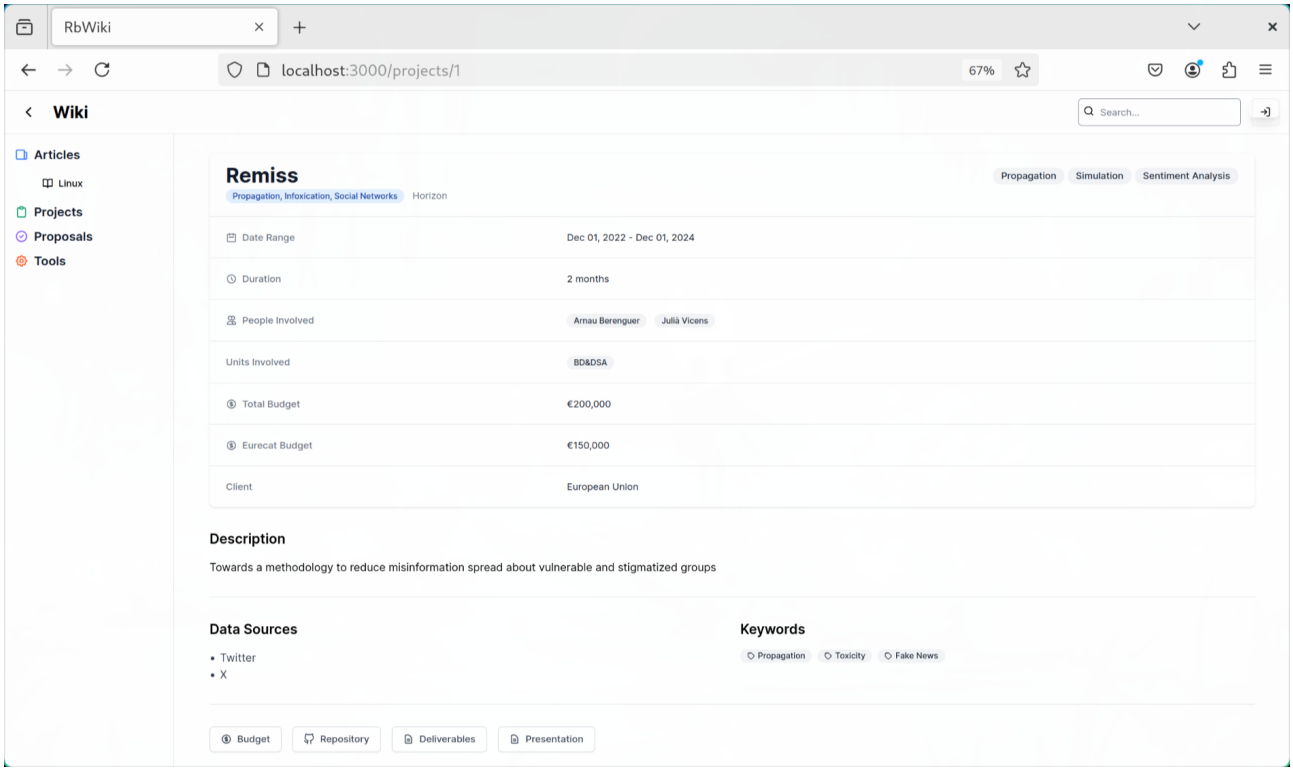

Fig. 6: VISTA D'UN PROJECTE DE LA WIKI