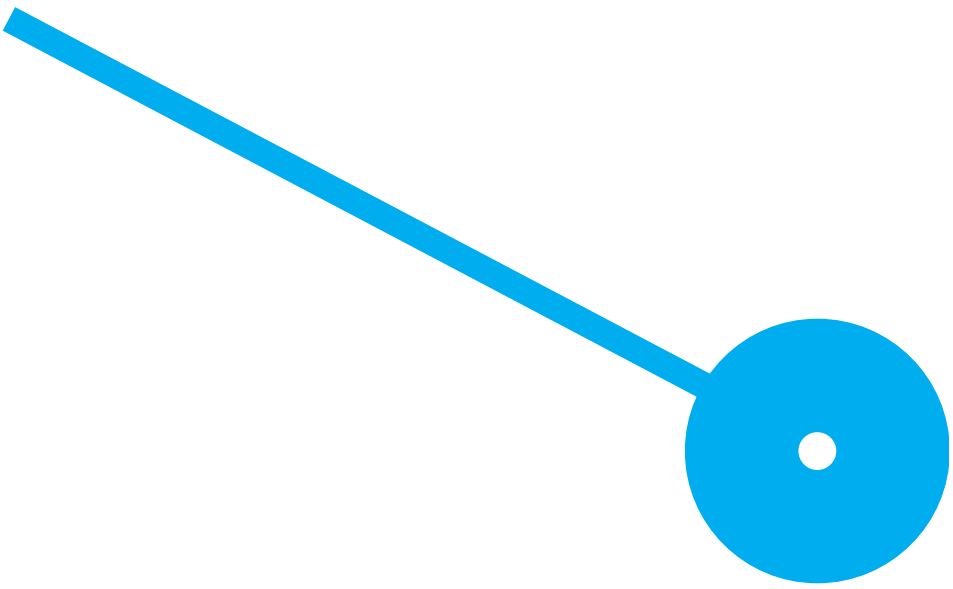
ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

M MESTRADO
ENGENHARIA INFORMÁTICA

Desenvolvimento de um Sistema para o Moodle com Integração de Modelos de Linguagem de Grande Escala para Apoio à Atividade Letiva

Ricardo Rodrigues da Silva

10/2025



ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

MESTRADO
ENGENHARIA INFORMÁTICA

Desenvolvimento de um Sistema para o Moodle com Integração de Modelos de Linguagem de Grande Escala para Apoio à Atividade Letiva

Ricardo Rodrigues da Silva

8190313

Orientador(es)

Prof. Doutor Bruno Moisés Teixeira de Oliveira

Prof. Doutor Óscar António Maia de Oliveira

Dissertação apresentado para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática pela Escola Superior de Tecnologia e Gestão do Instituto Politécnico do Porto.

10/2025

Declaração de integridade

Eu, **Ricardo Rodrigues da Silva**, estudante nº **8190313**, do Mestrado **Engenharia Informática** da Escola Superior de Tecnologia e Gestão do Instituto Politécnico do Porto, declaro que não fiz plágio nem auto-plágio, pelo que o trabalho intitulado "**Desenvolvimento de um Sistema para o Moodle com Integração de Modelos de Linguagem de Grande Escala para Apoio à Atividade Letiva** " é original e da minha autoria, não tendo sido usado previamente para qualquer outro fim. Mais declaro que todas as fontes usadas estão citadas, no texto e na bibliografia final, segundo as regras de referenciação adotadas na instituição.

Agradecimentos

Gostaria de expressar o meu sincero agradecimento aos meus orientadores, Prof. Doutor Bruno Moisés Teixeira de Oliveira e Prof. Doutor Óscar António Maia de Oliveira, pela orientação científica, disponibilidade e incentivo ao longo do desenvolvimento deste trabalho. Estendo também o meu agradecimento aos docentes e colegas do Mestrado em Engenharia Informática da Escola Superior de Tecnologia e Gestão, pela partilha de conhecimento e experiências enriquecedoras. Um reconhecimento especial à minha família e amigos, pelo apoio incondicional, compreensão e motivação durante esta etapa exigente.

A todos, o meu muito obrigado.

Abstract

This dissertation presents the development of a Moodle integrated system leveraging Large Language Models (LLMs) to support teaching through intelligent automation of pedagogical tasks. The proposed solution combines the Model Context Protocol (MCP) for client–server orchestration, a Retrieval Augmented Generation (RAG) pipeline for semantic retrieval, the Qdrant vector database for efficient indexing, and Google Gemini models (dynamically switchable by cost–performance profile) for natural language generation. The system connects to Moodle via web services to discover courses and automatically download PDFs, processes documents through text extraction and chunking, generates embeddings with sentence transformers, and delivers high impact educational features: context aware summaries, quiz generation with difficulty validation, flashcards, reading recommendations, and optional AI generated videos (Gemini Veo). A Flask based web interface provides authentication, conversation management, model selection, and real time invocation of MCP tools.Methodologically, an iterative development approach with continuous functional validation and structured logging was adopted. Evaluation focuses on three dimensions: (i) RAG performance (latency and retrieval relevance), (ii) pedagogical quality of generated content (clarity, source alignment, difficulty appropriateness), and (iii) robustness of MCP–Moodle integration (call reliability and synchronisation). Results indicate efficiency gains in preparing learning materials, tangible support for differentiated instruction, and reduced teacher workload on repetitive tasks. The main contributions are: a modular, reusable architecture for LMS contexts; an education oriented MCP toolset; and an RAG workflow optimised for academic PDFs. Limitations are discussed—document quality dependency, inference cost, and privacy concerns—along with future work, including direct export to Moodle formats, spaced repetition algorithms for flashcards, and effectiveness studies with real cohorts. Overall, the work demonstrates the feasibility of integrating LLMs into the Moodle ecosystem to enhance evidence based, content driven teaching practices.

Keywords: Moodle; MCP; RAG; Qdrant; Gemini; LLM; education; quizzes; summaries; generative AI.

Resumo

Esta dissertação apresenta o desenvolvimento de um sistema para o Moodle com integração de Modelos de Linguagem de Grande Escala (LLMs) destinado a apoiar a atividade letiva através de automação inteligente de tarefas pedagógicas. A solução proposta combina o protocolo Model Context Protocol (**MCP**) para orquestração cliente-servidor, um pipeline RAG (Retrieval-Augmented Generation) para recuperação semântica de conteúdos, a base de dados vetorial Qdrant para indexação eficiente, e modelos Gemini (comutáveis dinamicamente consoante custo e desempenho) para geração de linguagem natural. O sistema integra-se com o Moodle via web services para descoberta de cursos e obtenção automática de PDFs, processa documentos com extração de texto e segmentação, cria embeddings com sentence-transformers e disponibiliza funcionalidades de alto impacto educativo: geração de resumos contextuais, criação de questionários com validação de dificuldade, flashcards, recomendações de leitura e, opcionalmente, geração de vídeos com IA (Gemini Veo). A interface web, desenvolvida em Flask, oferece autenticação, gestão de conversas, seleção de modelos e invocação de ferramentas **MCP** em tempo real. Metodologicamente, adotou-se uma abordagem iterativa com validação funcional contínua e logging estruturado. A avaliação incide em três dimensões: (i) desempenho do RAG (tempo de resposta e relevância de recuperação), (ii) qualidade pedagógica do conteúdo gerado (clareza, alinhamento com fontes, adequação de dificuldade), e (iii) robustez da integração MCP–Moodle (confiabilidade de chamadas e sincronização). Os resultados demonstram ganhos de eficiência na preparação de materiais, apoio à diferenciação pedagógica e redução de esforço docente em tarefas repetitivas. As principais contribuições incluem: uma arquitetura modular replicável para LMS, um conjunto de ferramentas **MCP** focadas em educação e um fluxo RAG otimizado para PDFs académicos. São discutidas limitações (dependência de qualidade dos PDFs, custos de inferência, privacidade) e perspetivas futuras, nomeadamente exportação direta para formatos Moodle, algoritmos de repetição espaçada para flashcards e estudos de eficácia com turmas reais. Este trabalho evidencia a viabilidade de integrar LLMs no ecossistema Moodle para potenciar práticas pedagógicas baseadas em conteúdo e dados.

Palavras-chave: Moodle; MCP; RAG; Qdrant; Gemini; LLM; educação; questionários; resumos; IA generativa

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Listagens

Acrónimos

ADL Advanced Distributed Learning.

AICC Aviation Industry CBT Committee.

API Application Programming Interface.

CBL Challenge Based Learning.

CBT Computer-Based Training.

IMS CC IMS Common Cartridge.

LIME Learning, Interaction, Mentoring, and Evaluation.

LLM Large Language Model.

LMS Learning Management System.

LRS Learning Record Store.

LTI Learning Tools Interoperability.

MCP Model Context Protocol.

MOOC Massive Open Online Course.

PBL Problem-Based Learning.

PjBL Project-Based Learning.

RAG Retrieval-augmented generation.

RTE Run-Time Environment.

SCORM Sharable Content Object Reference Model.

SOA Service-Oriented Architecture.

xAPI Experience API.

IA Inteligência Artificial.

TAM Technology Acceptance Model.

Capítulo 1

Introdução

1.1 Contexto e Motivação

Durante muito tempo, o ensino seguiu uma abordagem tradicional, na qual o professor assumia um papel central e dominante nas interações, enquanto os alunos desempenhavam sobretudo uma função passiva, limitando-se a receber informação. Este modelo, estruturado em torno de manuais escolares e de uma sequência rígida de conteúdos, privilegiava a memorização em detrimento da compreensão profunda e da aplicação prática do saber adquirido [8].

Com a crescente interligação e sofisticação tecnológica da sociedade, estas estratégias revelam-se cada vez menos eficazes na preparação dos estudantes para os desafios contemporâneos [9]. Muitos alunos sentem-se desmotivados, uma vez que as atividades escolares permanecem desligadas de problemas reais, conduzindo a baixos níveis de envolvimento e limitando o desenvolvimento de competências como pensamento crítico e criatividade.

Nos últimos anos, a integração da inteligência artificial (IA), da automação e das tecnologias digitais transformou profundamente a forma como vivemos, trabalhamos e aprendemos. Estas mudanças têm redefinido as competências essenciais do século XXI — como adaptabilidade, pensamento crítico, colaboração e resolução criativa de problemas —, tornando imperativa a evolução dos sistemas educativos. Os modelos de ensino tradicionais, centrados na memorização e na aprendizagem passiva, revelam-se insuficientes para preparar os estudantes a enfrentar este mundo dinâmico [9; 10].

Neste enquadramento, a transformação digital no ensino superior não é apenas uma opção tecnológica, mas uma exigência estrutural de modernização pedagógica. Estudos recentes mostram, contudo, que apenas 25% das universidades analisadas possuem uma estratégia digital clara, enquanto mais de metade lançaram iniciativas isoladas, sem integração num plano institucional consistente [11]. A pandemia de COVID-19 acelerou a digitalização, mas de forma maioritariamente reativa, focada em assegurar continuidade, sem originar transformações sustentáveis ao nível organizacional nem garantir níveis adequados de personalização [12].

Assim, embora a transformação digital represente uma oportunidade incontornável, continua marcada por lacunas de maturidade e pela ausência de estratégias unificadas. Torna-se, por isso, fundamental analisar os sistemas que suportam esta digitalização, em particular os sistemas de gestão de aprendizagem, compreender as suas limitações no apoio à

personalização e identificar oportunidades de inovação como a integração de modelos de linguagem de grande escala e a necessidade de ecossistemas digitais coesos que evitem a fragmentação e promovam experiências de aprendizagem mais eficazes [13; 14; 15].

1.2 Objetivos

O objetivo geral desta dissertação é conceber, implementar e validar um sistema integrado com o Moodle que, recorrendo a técnicas de **RAG**, a uma base de dados vetorial e a modelos de linguagem, apoie de forma eficaz a atividade letiva. Pretende-se assegurar a recuperação semântica rigorosa de conteúdos, a geração automática de materiais pedagógicos e uma integração técnica robusta, com benefícios tangíveis para docentes e estudantes em termos de eficiência, qualidade e personalização do ensino.

Para concretizar este objetivo geral, definem-se os seguintes objetivos específicos:

- Implementar a integração entre **MCP** e Moodle, garantindo comunicação eficiente, segura e sincronizada.
- Desenvolver um sistema de recuperação semântica com base em Qdrant, uma base de dados vetorial orientada a similaridade semântica,para assegurar respostas fundamentadas nos materiais do curso.
- Criar mecanismos de geração automática de conteúdos educativos, como resumos, questionários e flashcards, explorando modelos de linguagem de última geração.
- Disponibilizar estas funcionalidades numa interface web intuitiva, de apoio direto à atividade letiva.
- Validar a solução em contexto educativo, avaliando desempenho técnico, utilidade pedagógica e perceção de docentes e estudantes.

1.3 Questões de Investigação

A definição das questões de investigação permite transformar o problema identificado em perguntas concretas que orientam o desenvolvimento do trabalho e asseguram a ligação entre os objetivos definidos e a metodologia adotada. Neste âmbito, foram formuladas as seguintes questões centrais:

- **Q1**: Como integrar de forma eficaz o protocolo **MCP** com sistemas Moodle já existentes?
- **Q2**: Que abordagens de **RAG** são mais adequadas para suportar a recuperação semântica de conteúdo educativo?
- **Q3**: De que modo é possível personalizar conteúdos de aprendizagem com base em perfis individuais de estudantes?
- **Q4**: Qual o impacto da solução proposta na eficiência da atividade letiva dos professores?

Estas questões orientam a investigação ao longo da dissertação, servindo de referência para a proposta metodológica apresentada no Capítulo 3 e para a análise crítica dos resultados descrita no Capítulo 4.

1.4 Metodologia

A metodologia seguida nesta dissertação combina diferentes abordagens que procuram garantir o rigor científico e, em simultâneo, a aplicabilidade prática da solução desenvolvida.

Em primeiro lugar, foi utilizada uma **abordagem de desenvolvimento iterativo com prototipagem rápida**, permitindo construir versões sucessivas do sistema, identificar problemas precocemente e incorporar melhorias de forma ágil.

Adotou-se igualmente uma **metodologia de investigação em contexto educativo**, que alia a intervenção prática no ambiente de aprendizagem à análise crítica dos seus efeitos, promovendo um ciclo contínuo de planeamento, implementação e reflexão.

A avaliação da solução combinou **métodos qualitativos e quantitativos**. Foram recolhidas métricas de desempenho técnico (relevância e latência do **RAG**, robustez da integração MCP–Moodle), bem como perceções de docentes e estudantes, permitindo avaliar a utilidade pedagógica da proposta.

Por fim, a **validação foi conduzida através de casos de uso específicos**, representativos de cenários reais de ensino superior, como a geração de resumos de conteúdos e a criação de questionários. Estes casos permitiram comprovar a relevância da solução e a sua aplicabilidade prática.

Em conjunto, estas opções metodológicas asseguram uma investigação equilibrada, combinando experimentação técnica, análise pedagógica e validação em contexto real.

1.5 Estrutura da Dissertação

A presente dissertação encontra-se organizada em cinco capítulos principais, complementados por referências bibliográficas e eventuais anexos. A estrutura segue uma progressão lógica, que inicia pela contextualização do tema e culmina na apresentação das conclusões e propostas de trabalho futuro.

Para além deste capítulo introdutório, o **Capítulo 2** apresenta a revisão do estado da arte e os trabalhos relacionados, reunindo a literatura e os estudos mais relevantes sobre o tema. São abordados os sistemas de gestão de aprendizagem (LMS), os principais protocolos e padrões de interoperabilidade, e a aplicação da inteligência artificial no contexto educativo, com especial enfoque nos modelos de linguagem de grande escala (LLMs) e nos sistemas de recuperação aumentada por geração (RAG). O capítulo termina com a análise comparativa de iniciativas semelhantes e a identificação das lacunas que o presente trabalho pretende colmatar.

O **Capítulo 3** descreve o desenho e a especificação do sistema desenvolvido, incluindo a arquitetura proposta, a integração entre o Moodle e o Model Context Protocol (MCP), o sistema de recuperação aumentada por geração (RAG), a utilização de modelos de linguagem e a implementação da interface de utilizador.

O **Capítulo 4** apresenta a validação e os resultados obtidos, descrevendo os casos de uso implementados, os testes realizados, as métricas de desempenho técnico e pedagógico, bem como a análise crítica dos benefícios e limitações da solução.

Por fim, o **Capítulo 5** reúne as conclusões principais, destacando as contribuições do trabalho, as limitações identificadas e possíveis linhas de investigação futura.

A dissertação termina com a lista de referências bibliográficas utilizadas e, quando aplicável, com anexos que documentam elementos complementares de suporte ao estudo.

Capítulo 2

Estado da Arte

2.1 Sistemas de Gestão de Aprendizagem (LMS)

Os **LMS** constituem hoje uma infraestrutura essencial no ensino superior, assegurando a disponibilização de conteúdos, a organização de atividades e a monitorização da aprendizagem. A sua origem remonta à década de 1990, com plataformas pioneiras como o Blackboard e o WebCT, criadas para digitalizar tarefas administrativas e disponibilizar recursos de forma centralizada.[16] Estas primeiras soluções eram sobretudo focadas na distribuição de materiais e na gestão de inscrições, funcionando mais como sistemas administrativos do que como ambientes de aprendizagem interativos [3].

Ao longo dos anos 2000, a massificação do *e-learning* e a pressão das universidades para reduzir custos de licenciamento levaram ao crescimento de alternativas *open-source*, entre as quais se destaca o **Moodle** [3]. Criado em 2002 por Martin Dougiamas, o Moodle foi desenvolvido com base numa filosofia pedagógica construtivista social, privilegiando a interação e a aprendizagem colaborativa. O seu caráter gratuito, a arquitetura modular e a forte comunidade internacional de utilizadores e desenvolvedores permitiram que se tornasse rapidamente a plataforma dominante em diversos países. Para além da flexibilidade técnica, o Moodle beneficiou ainda de um modelo de *partners* certificado que garantiu apoio profissional a instituições que necessitavam de serviços comerciais [3; 17]. Atualmente, continua a ser uma das soluções mais difundidas globalmente, utilizada em universidades de referência como a Open University no Reino Unido ou a UNED em Espanha, bem como em diversas instituições públicas e privadas na América Latina e na Ásia [1].

Paralelamente, novas plataformas foram surgindo, refletindo uma evolução tecnológica e pedagógica. O **Canvas**, lançado em 2011, destacou-se pelo modelo baseado em nuvem e pela aposta numa interface mais intuitiva e colaborativa [18]. O **Open edX**, desenvolvido pelo MIT e pela Harvard University, posicionou-se como referência em cursos massivos online (Massive Open Online Course (**MOOC**)) [19], enquanto o **Google Classroom** simplificou a adoção em escolas de menor dimensão e contextos híbridos [20]. Apesar desta diversidade, o **Moodle** mantém-se como uma referência incontornável devido à sua robustez, extensibilidade e capacidade de adaptação a diferentes cenários de ensino [3].

Para além das suas funções administrativas e de gestão de conteúdos, o Moodle tem vindo a ser explorado em abordagens pedagógicas inovadoras, como o Challenge Based Lear-

ning (**CBL**) . Estudos recentes demonstram que a plataforma pode apoiar metodologias centradas no estudante, fornecendo ferramentas de comunicação, colaboração e reflexão que facilitam o desenvolvimento de competências transversais, como o pensamento crítico e a resolução criativa de problemas [21].

Contudo, os **LMS** não estão isentos de críticas. A sua conceção inicial privilegiou a organização de conteúdos, o que limita a sua capacidade de responder às necessidades pedagógicas atuais. Em muitos casos, os percursos de aprendizagem são lineares e iguais para todos os estudantes, independentemente dos seus ritmos ou níveis de conhecimento, resultando numa personalização reduzida. Um estudo de meta-análise sobre o Moodle mostrou que, apesar de os estudantes expressarem níveis elevados de satisfação tecnológica, os docentes tendem a ser mais críticos, apontando dificuldades de adaptação da plataforma às suas práticas e exigências pedagógicas [22]. Esta diferença de perceção entre estudantes e professores confirma que a adoção generalizada dos **LMS** não corresponde, necessariamente, a experiências de qualidade homogéneas.

Do ponto de vista da usabilidade, a literatura aponta desafios relevantes [1]. Interfaces densas, menus extensos e falta de consistência visual podem aumentar a carga cognitiva, dificultando a navegação e desviando a atenção da aprendizagem propriamente dita. Comparações entre plataformas demonstram que soluções como o Canvas apresentam maior facilidade de uso e uma experiência mais fluida, graças ao seu desenho centrado no utilizador e à integração nativa com serviços em nuvem. Em contrapartida, o Moodle depende em grande medida de *plugins* para oferecer funcionalidades avançadas, o que aumenta a complexidade de administração e pode comprometer a estabilidade do sistema [2].

Estas diferenças tornam-se mais visíveis quando se observam comparações entre os principais **LMS** utilizados no ensino superior. A Tabela 2.1 sintetiza os pontos fortes e limitações de quatro plataformas de referência — Moodle, Canvas, Open edX e Blackboard — com base em estudos comparativos recentes [1; 2; 3], destacando a diversidade de soluções disponíveis e as fragilidades persistentes de cada uma.

Plataforma	**Pontos fortes**	**Limitações**
Moodle	Open source, extensível, comunidade ativa	Usabilidade complexa, dependência de plugins, personalização limitada
Canvas	Interface intuitiva, baseado em nuvem, colaboração integrada	Proprietário, menor flexibilidade de personalização
Open edX	Escalável para **MOOC**, open source, apoio institucional (MIT/Harvard)	Instalação complexa, curva de aprendizagem elevada
Blackboard	Amplamente adotado, estável, suporte institucional	Custos elevados, interface desatualizada, *vendor lock-in*

Tabela 2.1: Síntese comparativa entre **LMS**, elaborada a partir de [1; 2; 3].

Outro desafio central está relacionado com a personalização. O Moodle disponibiliza ferramentas de configuração que permitem diferenciar atividades ou aplicar restrições de acesso, mas estas dependem de uma configuração manual intensiva por parte dos docentes. Tentativas de personalização automática, como a introdução de mecanismos baseados em *learning styles*, demonstraram viabilidade técnica [23], mas permanecem conceptualmente frágeis, além de exigirem alterações profundas ao sistema [24]. A tendência mais robusta

tem sido explorar abordagens baseadas em dados objetivos, utilizando *learning analytics* para identificar padrões de desempenho, competências adquiridas e níveis de interação. Estas metodologias permitem construir perfis dinâmicos de aprendizagem e adaptar a sequência de conteúdos de forma mais fundamentada e escalável, evitando as limitações associadas a estilos de aprendizagem [25].

Finalmente, subsistem ainda desafios ligados à carga de trabalho docente e à desigualdade no acesso tecnológico. A elaboração manual de materiais, questionários e atividades interativas continua a ser morosa, o que limita a escalabilidade em contextos com grandes turmas. Em paralelo, a eficácia dos **LMS** depende também das competências digitais dos utilizadores e da disponibilidade de infraestruturas adequadas, fatores que podem agravar desigualdades entre instituições e estudantes.

Apesar da difusão dos Sistemas de Gestão de Aprendizagem, a sua eficácia depende das funcionalidades disponíveis e da forma como respondem às necessidades reais dos utilizadores. Neste sentido, importa analisar de forma crítica as limitações de plataformas específicas como o Moodle, amplamente usado no ensino superior.

2.1.1 Limitações dos sistemas Moodle na geração automática de conteúdo

Apesar da sua difusão global, o Moodle e outros sistemas de gestão de aprendizagem continuam a revelar limitações significativas no que diz respeito à geração automática de conteúdo educativo. Na sua conceção original, estas plataformas foram desenhadas sobretudo para gestão e distribuição de conteúdos, assumindo que a criação de materiais seria da responsabilidade direta do docente. Assim, tarefas como a elaboração de questionários, exercícios ou resumos permanecem essencialmente processos manuais, exigindo tempo e esforço acrescido por parte dos professores.

A literatura recente mostra que a inteligência artificial embutida pode desempenhar um papel importante na criação automática de conteúdos e na personalização da experiência de aprendizagem. Contudo, os sistemas tradicionais de **LLM** raramente incluem estes mecanismos de forma nativa, limitando-se a registar e disponibilizar recursos já preparados [26]. Esta lacuna traduz-se numa dependência estrutural do trabalho humano, comprometendo a escalabilidade dos processos de ensino.

Mesmo quando surgem propostas inovadoras, como a utilização de técnicas de mutação de modelos (model mutation) para gerar automaticamente exercícios baseados em diagramas no Moodle [27], estas continuam a ser soluções especializadas, dependentes de linguagens formais e de processos complexos de configuração. Embora demonstrem viabilidade técnica, a sua adoção generalizada permanece reduzida, evidenciando que a geração automática de conteúdo ainda está longe de ser uma funcionalidade consolidada nos LMS.

Embora o Moodle seja amplamente utilizado no ensino superior, continua a revelar fragilidades quando se trata de personalizar percursos de aprendizagem com base em dados recolhidos na plataforma. Apesar de registar informação como acessos, tempo de permanência ou resultados em atividades, estes indicadores não são transformados em recomendações automáticas ou adaptações do percurso do estudante. Na prática, a personalização depende sobretudo da intervenção manual dos docentes, que precisam de interpretar os dados e ajustar os conteúdos ou atividades.

O estudo de [28] confirma esta limitação ao analisar a utilização do Moodle em cursos personalizados de inglês. Os autores verificaram que, embora existam funcionalidades que permitem alguma adaptação — como ajustar materiais ou diversificar exercícios —, estas exigem grande esforço de configuração por parte do professor e não resultam de um sistema adaptativo baseado em dados. Assim, a personalização em Moodle permanece restrita e pouco dinâmica, não respondendo de forma eficaz à diversidade de ritmos, estilos e necessidades dos estudantes [28].

A criação de materiais educativos em plataformas como o Moodle continua a ser um processo exigente para os docentes, que precisam de investir tempo significativo na elaboração de questionários, exercícios ou conteúdos interativos [29]. Mesmo quando surgem propostas de automação, como a solução apresentada por [27], que permite gerar e corrigir automaticamente exercícios baseados em diagramas, a sua utilização prática envolve um conjunto de desafios técnicos. Embora estas abordagens demonstrem a viabilidade da automação e possam reduzir a carga associada à produção manual de conteúdos, exigem conhecimentos especializados, configurações complexas e continuam limitadas a tipos específicos de materiais.

Deste modo, a dificuldade na criação automática de recursos educativos não reside apenas na ausência de ferramentas eficazes integradas no Moodle, mas também na barreira de entrada que as soluções externas representam. Tal realidade mantém os docentes dependentes de processos manuais ou de sistemas complementares pouco acessíveis, dificultando a adoção generalizada da automação no ensino superior.

A integração de **LLM** em plataformas educativas não constitui apenas uma questão técnica, mas uma condição essencial para que estas tecnologias possam ser utilizadas de forma eficaz e responsável. A literatura recente mostra que, embora os **LLM** ofereçam oportunidades únicas para a personalização da aprendizagem e para a automação de tarefas pedagógicas, o seu potencial só se concretiza plenamente quando existe uma integração eficiente com sistemas já estabelecidos, como o Moodle.

Kaleci [30] analisou em detalhe a integração de ferramentas de inteligência artificial no Moodle 4.5, concluindo que estas podem ampliar significativamente as capacidades da plataforma, nomeadamente através da personalização de percursos de aprendizagem, do fornecimento de feedback adaptativo e da tomada de decisões baseadas em dados. O autor destaca ainda que a arquitetura modular e de código aberto do Moodle o torna particularmente adequado para acolher este tipo de extensões. Contudo, também identifica desafios estruturais que não podem ser ignorados: preocupações com a privacidade e segurança de dados, enviesamentos algorítmicos, exigências de infraestrutura tecnológica e a necessidade de preparar docentes e instituições para uma adoção informada e ética.

Deste modo, a integração eficiente entre **LLM** e **LMS** é mais do que uma oportunidade de inovação: é uma necessidade para evitar a fragmentação de sistemas, garantir a coerência pedagógica e assegurar que os ganhos tecnológicos se traduzem em ecossistemas digitais sustentáveis, inclusivos e eticamente responsáveis.

Embora os **LMS**, e em particular o Moodle, se tenham consolidado como infraestruturas centrais no ensino superior, continuam a apresentar desafios significativos ao nível da usabilidade, da personalização e da integração com novas tecnologias. A análise realizada evidencia que, apesar da sua evolução de ferramentas administrativas para ecossistemas educativos complexos, persistem limitações relevantes: a personalização é frequentemente

restrita, a criação de conteúdos depende de processos manuais e a usabilidade continua a levantar barreiras à adoção plena. Estas fragilidades reforçam a necessidade de soluções capazes de automatizar tarefas pedagógicas e de integrar inteligência artificial de forma nativa, preparando a transição para sistemas mais adaptativos. Neste contexto, torna-se pertinente explorar os protocolos e padrões de interoperabilidade que suportam a ligação entre diferentes ferramentas e plataformas, tema desenvolvido na secção seguinte.

2.2 Metodologias Pedagógicas Baseadas em Desafios e Tecnologia (CBL)

O avanço das metodologias de ensino centradas no estudante tem levado à adoção de abordagens mais ativas, colaborativas e tecnologicamente mediadas. Entre estas, o **CBL** destaca-se pela sua capacidade de integrar a aprendizagem experiencial, o uso significativo da tecnologia e o compromisso social, aproximando o processo educativo da resolução de problemas reais.

O **CBL** é uma metodologia educativa centrada no estudante que promove a aprendizagem ativa através da resolução de problemas do mundo real, integrando competências interdisciplinares e tecnológicas [31; 32]. O modelo baseia-se em princípios de autenticidade, colaboração e ação. Os alunos enfrentam desafios significativos — locais ou globais — relacionados com temas como sustentabilidade ou saúde pública, desenvolvendo competências de pensamento crítico e resolução de problemas. Esta abordagem valoriza a participação de todos os intervenientes (alunos, professores e comunidade) como coaprendentes, numa estrutura flexível e adaptável a diferentes níveis e contextos educativos [33].

Além de promover competências do século XXI, como criatividade, trabalho em equipa e literacia digital [34], o **CBL** enfatiza o uso estratégico da tecnologia para investigar, colaborar e documentar o processo de aprendizagem. A reflexão contínua é outro elemento-chave: os estudantes registam descobertas, analisam erros e desenvolvem portfólios digitais que documentam o progresso [31; 32].

O **framework** do **CBL** divide-se em três fases — **Engage**, **Investigate** e **Act** — que conduzem o estudante desde a identificação de uma grande ideia até à implementação de soluções reais. A aprendizagem é personalizada e orientada para a ação, com o professor a atuar como facilitador, e não como simples transmissor de conhecimento [35; 32]. A integração de tecnologias digitais (como plataformas colaborativas e ferramentas de avaliação automática) é considerada essencial para fomentar a auto-organização e o trabalho cooperativo [36].

Comparativamente com outras metodologias ativas, como Problem-Based Learning (**PBL**) e Project-Based Learning (**PjBL**), o **CBL** distingue-se por colocar a ênfase simultaneamente no processo e no impacto social da solução. Enquanto a PBL tende a explorar problemas teóricos e a PjBL se foca no produto final, o **CBL** promove a co-criação de desafios abertos, incentivando a autonomia do aluno e a ligação entre aprendizagem e ação transformadora [37; 38; 39].

Assim, o **CBL** constitui um modelo pedagógico versátil que integra aprendizagem experiencial, uso significativo da tecnologia e compromisso social, preparando os estudantes para agir de forma crítica e criativa perante problemas complexos da contemporaneidade [35; 32].

A aplicabilidade do **CBL** em ambientes digitais, como o Moodle, tem vindo a ganhar destaque na literatura, que demonstra o potencial destas plataformas para operacionalizar as três fases do **CBL** através de módulos de comunicação, avaliação contínua e colaboração em grupo. Apesar deste interesse crescente, os estudos existentes continuam a ser limitados, evidenciando a necessidade de investigações que articulem a dimensão pedagógica com as possibilidades tecnológicas dos **LMS**. No âmbito desta dissertação, tem sido desenvolvida uma linha de investigação dedicada à integração entre o **CBL**, o Moodle e tecnologias emergentes de apoio à aprendizagem. Como parte deste percurso, foi publicado o artigo *Challenge-Based Learning: A Technology Perspective* [40], que analisa o papel da tecnologia como elemento estruturante do **CBL** e identifica lacunas na sua implementação prática. Adicionalmente, foram submetidos para publicação os manuscritos *Leveraging Moodle to Support the Challenge-Based Learning Framework* e *Empowering Digital Learning Through MCP and Moodle Integration*, que exploram, respetivamente, o potencial do Moodle como suporte ao **CBL** e a integração do **MCP** com mecanismos de **RAG** para personalização e automatização de processos de aprendizagem.

Esta produção científica serve de base conceptual e empírica à presente dissertação, cuja arquitetura proposta representa uma síntese prática dos princípios defendidos nos estudos anteriores. O projeto desenvolvido nesta investigação procura materializar esses contributos, combinando o Moodle, o **MCP** e o **RAG** para responder de forma integrada aos requisitos tecnológicos e pedagógicos do **CBL**. Desta forma, a inteligência artificial deixa de ser apenas uma ferramenta auxiliar e passa a atuar como mediadora ativa do processo de aprendizagem baseada em desafios.

2.3 Protocolos e Padrões de Interoperabilidade

A interoperabilidade é um elemento essencial na construção de ecossistemas digitais de aprendizagem abertos, sustentáveis e adaptáveis. No contexto do ensino superior, onde a flexibilidade e a integração de ferramentas externas são cada vez mais relevantes, os protocolos de e-learning assumem um papel determinante. O **LMS**, enquanto um dos sistemas de gestão de aprendizagem mais amplamente adotados, suporta diversos padrões reconhecidos que asseguram a reutilização, a portabilidade e o rastreio de conteúdos educativos. Entre os mais relevantes encontram-se o Learning Tools Interoperability (**LTI**) [41], o Sharable Content Object Reference Model (**SCORM**) [42], o Experience API (**xAPI**) [43], o IMS Common Cartridge (**IMS CC**) [44] e o Aviation Industry CBT Committee (**AICC**) [45].

O protocolo **LTI**, desenvolvido pelo IMS Global Learning Consortium, permite a integração de aplicações externas em ambientes como o Moodle de forma segura e transparente, através de mecanismos como a autenticação única, o controlo de acesso baseado em perfis e a troca de dados encriptada. A adoção de **LTI** previne desenvolvimentos personalizados dispendiosos e garante compatibilidade com múltiplas plataformas. Projetos como o ceLTIc demonstraram a sua eficácia na integração de ferramentas como o WebPA, facilitando a gestão da avaliação por pares. Do mesmo modo, ferramentas amplamente utilizadas como o Zoom, o Turnitin ou plataformas de grandes editoras (e.g., Pearson, McGraw-Hill) recorrem ao **LTI** para assegurar uma experiência unificada no interior dos **LMS** [46]. Estudos mais recentes reforçam a relevância do **LTI**, ao evidenciar que a sua implementação enfrenta ainda desafios relacionados com autenticação segura, gestão de dados sensíveis e integração institucional, exigindo boas práticas de governança e atuali-

zações constantes [47].

O **SCORM**, criado pela iniciativa Advanced Distributed Learning (**ADL**), constitui um dos padrões mais duradouros no e-learning, oferecendo especificações que asseguram a compatibilidade entre conteúdos e plataformas. Entre as suas vantagens destacam-se a interoperabilidade, a reutilização de conteúdos, o rastreio do progresso do estudante e a longevidade dos materiais educativos. No Moodle, os pacotes **SCORM** são normalmente carregados através de um leitor nativo. Contudo, abordagens mais recentes têm proposto arquiteturas orientadas a serviços (Service-Oriented Architecture (**SOA**)), que externalizam o motor **SCORM** (Run-Time Environment (**RTE**)), permitindo o seu uso partilhado entre múltiplos **LMS** e melhorando a escalabilidade e a manutenção do sistema [42; 48].

A evolução do **SCORM** deu origem ao **xAPI**, um protocolo mais versátil que permite o rastreio de experiências de aprendizagem formais e informais, tanto em contextos online como offline. Com o **xAPI**, atividades realizadas em jogos, simulações, redes sociais ou ambientes colaborativos podem ser monitorizadas, proporcionando uma visão mais completa do percurso de aprendizagem do estudante. Casos como o *Oregon Trail game* ou o modelo Learning, Interaction, Mentoring, and Evaluation (**LIME**) (Learning, Interaction, Mentoring, and Evaluation) demonstram o potencial do **xAPI** para recolher dados complexos e gerar recomendações personalizadas, contribuindo para uma melhoria contínua do processo educativo [43]. Investigações mais recentes confirmam que o **xAPI** representa uma evolução significativa face ao **SCORM**, ao permitir a integração de dados distribuídos e multimodais num Learning Record Store (**LRS**), ampliando as possibilidades de análise para *learning analytics* e sistemas adaptativos [49].

Outro protocolo relevante é o **IMS CC**, também desenvolvido pelo IMS Global, que possibilita a exportação e importação de cursos completos entre diferentes **LMS**. Ao contrário do **SCORM**, o **IMS CC** oferece maior capacidade de reutilização e edição de conteúdos após a importação, integrando fóruns, avaliações e hiperligações externas num único pacote. Estudos comparativos mostram que os pacotes **IMS CC** são automaticamente integrados no Moodle, permitindo reorganização e edição direta no ambiente do curso, enquanto os pacotes **SCORM** permanecem objetos fechados, de difícil adaptação [44].

O **AICC**, criado pelo Computer-Based Training (**CBT**) Committee da indústria da aviação, foi um dos primeiros protocolos de e-learning, destacando-se pela simplicidade, segurança e pela possibilidade de alojar conteúdos em servidores externos. Apesar de ainda ser suportado em algumas plataformas, foi oficialmente descontinuado em 2014. As suas limitações na monitorização do progresso e a falta de atualizações contribuíram para a sua obsolescência face a padrões mais modernos como o **SCORM** e o **xAPI** [50; 45].

A diversidade de protocolos de e-learning analisados mostra que, embora existam padrões consolidados como o **LTI**, o **SCORM** e o **xAPI**, subsistem limitações estruturais. Estes protocolos privilegiam a padronização de formatos e a monitorização de atividades, mas não oferecem mecanismos para personalização adaptativa nem para integração direta com sistemas de inteligência artificial. Assim, a sua utilidade é essencial mas insuficiente: asseguram interoperabilidade técnica, porém não respondem à necessidade crescente de ecossistemas educativos dinâmicos e centrados no estudante. Esta limitação abre espaço para novas abordagens baseadas em **LLM**, capazes de reforçar a personalização e a automação da aprendizagem, cuja relevância será discutida na secção seguinte.

2.4 Modelos de Linguagem de Grande Escala (LLMs)

Os Modelos de Linguagem de Grande Escala **LLM** emergiram recentemente como ferramentas transformadoras no contexto educativo, remodelando a forma como o conhecimento é acedido, entregue e construído. A sua capacidade de processar linguagem natural em larga escala e de gerar saídas coerentes e sensíveis ao contexto abre espaço para um vasto leque de aplicações, que vão desde a avaliação automática até à tutoria personalizada. Uma revisão sistemática de [51], que analisou noventa e quatro estudos publicados nos últimos anos, mostra que os **LLM** estão a ser implementados em múltiplos domínios, incluindo a formação médica, o ensino de inglês como língua estrangeira, o apoio à escrita académica e a preparação para exames. Mais de noventa por cento destes estudos incidem especificamente no ChatGPT (versões 3.5 e 4), confirmando o seu papel central como ferramenta de referência na investigação educacional atual. Em diferentes áreas, os **LLM** são reportados como promotores de feedback imediato, aumento do envolvimento estudantil e suporte à aprendizagem ativa, contribuindo assim para ultrapassar desafios persistentes na escalabilidade da educação personalizada.Para compreender como os LLMs alcançam estas capacidades, é fundamental analisar a sua arquitetura subjacente.

A compreensão do papel dos **LLM** em educação exige não apenas a caracterização geral destes modelos, mas também a análise da sua arquitetura interna. Esta análise permite entender como os **LLM** representam e processam linguagem, fornecendo a base conceptual necessária para discutir soluções como o **RAG**.

2.4.1 Arquitetura dos Modelos de Linguagem de Grande Escala (LLMs)

Os **LLM** representam um avanço significativo no campo da Inteligência Artificial, largamente impulsionados pela arquitetura **Transformer** [52]. Esta arquitetura marcou uma viragem ao abandonar as redes neurais recorrentes (RNNs) em favor de mecanismos de **atenção**, que permitem ao modelo ponderar a importância de diferentes partes da sequência de entrada de forma mais eficiente. A principal vantagem do Transformer reside na sua capacidade de captar relações contextuais complexas e dependências de longo alcance no texto, o que o torna ideal para tarefas como a geração de texto, tradução automática e compreensão profunda da linguagem, especialmente em grandes volumes de dados.

O processo de inferência (ou seja, a utilização do modelo para gerar uma resposta) num LLM segue tipicamente um pipeline que pode ser decomposto nos seguintes componentes-chave:

Tokenização: A primeira etapa consiste em converter o texto de entrada contínuo numa sequência discreta de unidades menores, conhecidas como **tokens**. Dependendo do tokenizador usado, estes podem corresponder a palavras completas ou subpalavras (no caso de algoritmos como o *Byte Pair Encoding*) [53] ou até caracteres individuais. Cada token é depois mapeado para um identificador numérico único, que constitui a representação inicial a ser processada pelo modelo.

Por exemplo, a frase em inglês *"The king ate the apple"* pode ser segmentada em cinco tokens: [`The`, `king`, `ate`, `the`, `apple`]. Este processo é fundamental, pois define o "vocabulário" que o modelo consegue reconhecer e manipular. Ferramentas como o *OpenAI*

Tokenizer[1] permitem visualizar, de forma prática, como uma sequência de texto é convertida em tokens, demonstrando que palavras raras ou expressões em línguas diferentes podem ser divididas em múltiplos segmentos, o que aumenta o custo computacional.

A Figura 2.1 mostra graficamente este processo, evidenciando a passagem da frase original para a sequência de tokens e, por fim, para os respetivos IDs numéricos, que servem como entrada para as camadas seguintes do modelo.

Figura 2.1: Segmentação da frase em tokens com IDs numéricos.

Como se observa, a passagem de texto contínuo para tokens individuais é um processo determinístico mas dependente do vocabulário do modelo. Isto implica que línguas diferentes ou palavras raras podem originar tokenizações mais longas, aumentando o custo computacional e influenciando o desempenho global do **LLM**.

Embeddings: Depois da tokenização, cada identificador numérico é transformado num **vetor denso de alta dimensão**, conhecido como embedding. Estes vetores capturam regularidades estatísticas do uso da linguagem, permitindo que tokens semanticamente semelhantes fiquem representados próximos no espaço vetorial.

Por exemplo:

```
"The"   → [0.12, -0.83,  0.51, ...]
"king"  → [0.77,  0.05, -0.33, ...]
"apple" → [-0.45,  0.67,  0.12, ...]
```

É importante sublinhar que não existe uma semântica explícita para cada dimensão individual — o significado emerge da relação entre vetores. Assim, `king` e `queen` aparecem próximos no espaço, enquanto `apple` se afasta, refletindo a diferença semântica [54].

Ferramentas como o *TensorFlow Embedding Projector*[2] permitem visualizar embeddings de alta dimensão após redução para 2D/3D (com PCA ou t-SNE), ajudando a compreender como o modelo organiza a linguagem internamente.

[1] `https://platform.openai.com/tokenizer`
[2] `https://projector.tensorflow.org/`

A Figura 2.2 apresenta um exemplo ilustrativo de um espaço vetorial bidimensional, onde palavras relacionadas estão agrupadas e conceitos de domínios distintos aparecem afastados.

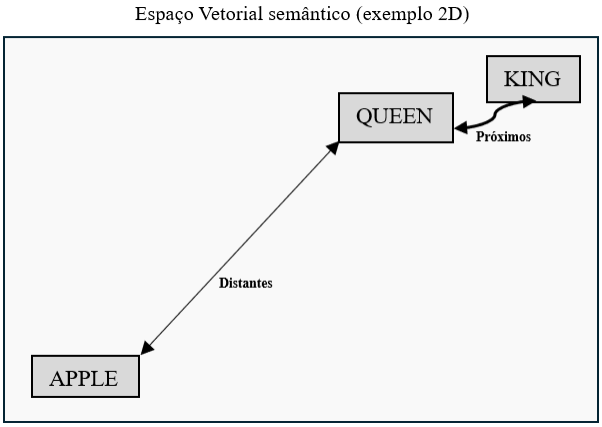


Figura 2.2: Exemplo ilustrativo de um espaço vetorial semântico bidimensional.

A visualização reforça a ideia de que o significado semântico não está contido em cada dimensão isolada, mas sim nas relações entre vetores. Assim, embeddings permitem que o modelo capture regularidades estatísticas da linguagem e estabeleça uma base sólida para operações posteriores, como a atenção contextual.

Blocos Transformer: A sequência de embeddings (tokens + posicionais) é então processada por múltiplos **blocos Transformer** empilhados. Cada bloco é composto essencialmente por dois mecanismos principais:

- **Mecanismo de Atenção Multi-Head:** Este mecanismo permite ao modelo atribuir diferentes pesos de relevância a cada token em relação aos restantes. Com múltiplas "cabeças" de atenção, o modelo pode capturar simultaneamente dependências sintáticas (ex.: sujeito-verbo) e relações semânticas (ex.: analogias e contextos). É esta capacidade que possibilita desambiguar termos polissémicos. Por exemplo, a palavra `apple` assume representações diferentes em "I ate an apple" (ligada a `fruit`) e em "Apple released a new iPhone" (ligada a `iPhone`) [55].

- **Redes Feed-Forward:** Após a etapa de atenção, cada token é processado por uma rede neural feed-forward que introduz não-linearidade, aumentando a expressividade do modelo e permitindo captar padrões mais complexos.

Cada bloco inclui ainda mecanismos de **normalização de camada** e **ligações residuais**, que estabilizam o treino e facilitam a propagação de informação em redes profundas. A repetição iterativa destes blocos assegura que cada representação é enriquecida com informação de toda a sequência, tanto a curto como a longo alcance.

A Figura 2.3 ilustra um exemplo prático de atenção contextual, em que o modelo atribui relevâncias distintas à palavra `apple`, consoante o contexto biológico (fruta) ou tecnológico (empresa).

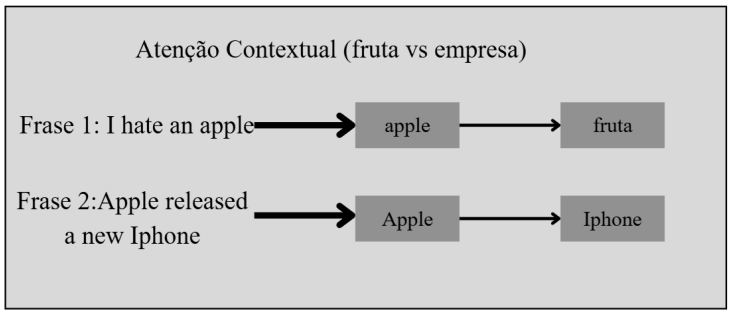


Figura 2.3: Atenção contextual adaptada ao significado da palavra.

Este exemplo evidencia a principal inovação do mecanismo de atenção: a capacidade de ajustar dinamicamente a representação de uma palavra em função do contexto em que ocorre. É esta flexibilidade que distingue os modelos Transformer de arquiteturas anteriores, tornando-os especialmente eficazes em tarefas de desambiguação semântica e compreensão de linguagem natural.

Projeção Linear e Softmax: Após passar por todos os blocos Transformer, a representação final de cada token contém informação contextualizada — ou seja, já não representa apenas a palavra isolada, mas também o seu significado em função das palavras que a rodeiam. Esta representação, contudo, está num espaço vetorial de dimensão interna (ex.: 768, 1024 ou 4096 posições, dependendo do modelo) e precisa de ser convertida num espaço que corresponda ao vocabulário completo do **LLM**.

Para isso, aplica-se uma **projeção linear**, que nada mais é do que uma transformação matricial que mapeia o vetor interno para uma nova dimensão: o número total de palavras/subpalavras conhecidas pelo modelo (ex.: 50 000 tokens). O resultado desta projeção é um vetor de *logits*, ou seja, valores numéricos não normalizados que indicam a "evidência"de cada token ser o próximo na sequência.

Em seguida, aplica-se a função **Softmax**, que transforma estes logits em probabilidades normalizadas. Este passo garante que a soma de todas as probabilidades é igual a 1, permitindo interpretar os valores como distribuições de probabilidade sobre o vocabulário. Por exemplo, se após o cálculo tivermos:

```
Logits: [2.1 (He), 0.9 (She), -0.5 (Car), ...]
Softmax → Probabilidades:
"He": 0.42
"She": 0.18
"It": 0.10
...
```

Observa-se que o modelo atribui maior probabilidade à palavra "He", mas também considera "She"e "It"como alternativas possíveis. A escolha final do token pode ser feita de forma determinística (selecionando sempre o mais provável, *greedy decoding*) ou probabilística (através de métodos de amostragem como *top-k* ou *nucleus sampling*), o que permite equilibrar precisão e diversidade na geração de texto [56; 57].

Estas etapas — **tokenização**, **embeddings**, **blocos Transformer** e **projeção final** com ***Softmax*** — articulam-se num processo integrado que sustenta a capacidade de geração dos **LLM**. A Figura 2.4 ilustra este pipeline de forma esquemática, evidenciando a sequência dos principais componentes descritos anteriormente.

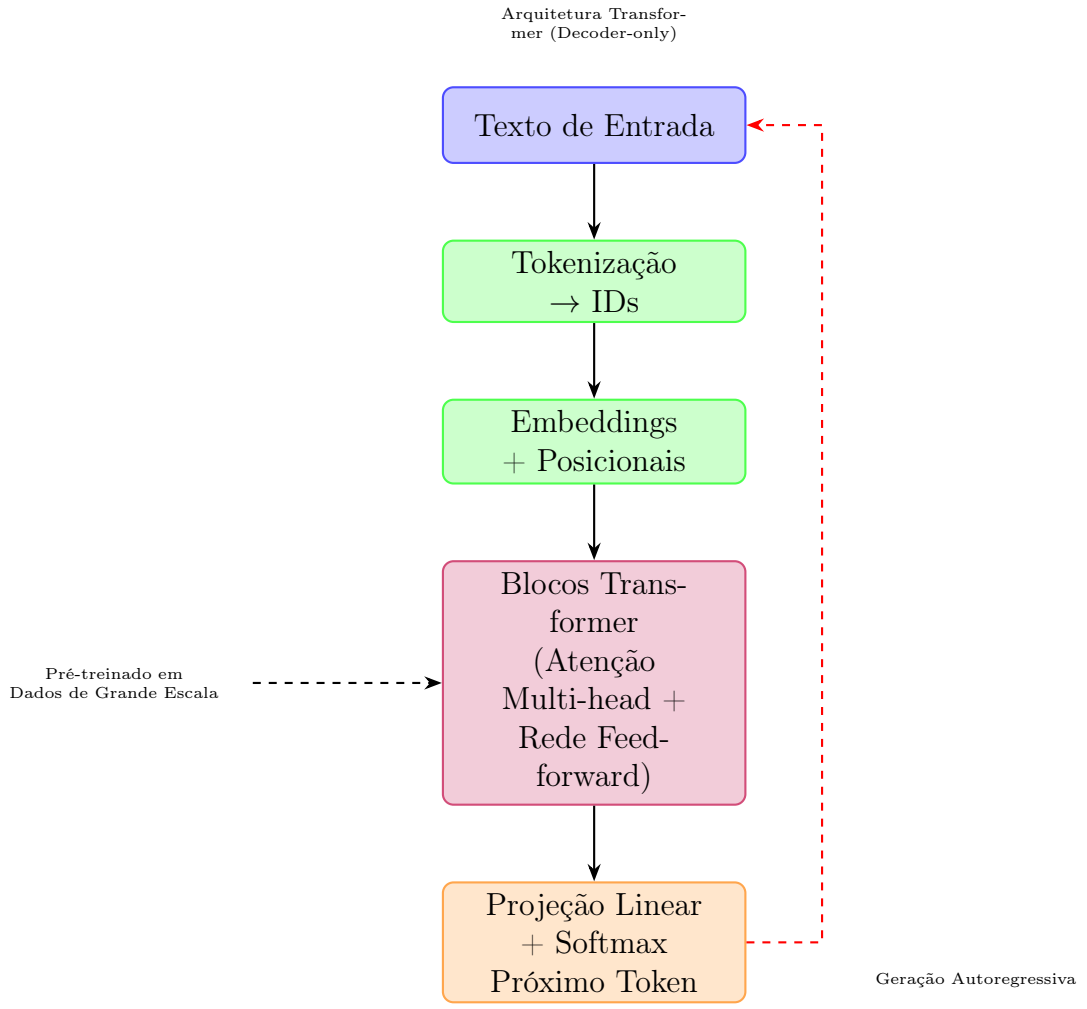


Figura 2.4: Arquitetura **LLM** do tipo Transformer.

A representação esquemática integra todas as etapas discutidas mostrando como estas se encadeiam no processo de geração autoregressiva. Esta visão de conjunto fornece a base conceptual necessária para compreender aplicações mais avançadas, como os mecanismos de **RAG** analisados na secção seguinte.

Esta abordagem é a base dos principais LLMs modernos, como GPT-3, GPT-4, BERT e Gemini, e fornece o enquadramento técnico necessário para compreender aplicações mais avançadas, incluindo geração de conteúdos multimodais, tutoria adaptativa e integração em sistemas educativos [57; 58].

2.4.2 Casos de estudo

Complementando esta perspetiva, em [59], os autores analisam o potencial pedagógico de **LLM** como o ChatGPT, Bing AI e Google Bard. É destacada a versatilidade destes modelos na geração de materiais instrucionais, questionários e planos de aula adaptativos, bem como a sua capacidade de apresentar explicações em diferentes níveis de complexidade, ajustadas a perfis variados de estudantes. Contudo, sublinham que a integração eficaz dos **LLM** na educação exige não apenas implementação técnica, mas também o desenvolvimento de competências digitais por parte de docentes e alunos. Entre estas competências estão a formulação de problemas, a aprendizagem exploratória, a experimentação e a re-

flexão crítica, essenciais para que os **LLM** funcionem como aliados pedagógicos em vez de meros fornecedores de respostas. Evidência empírica recente reforça esta visão, num estudo baseado no Technology Acceptance Model (TAM) [60], os autores reportam que os estudantes avaliaram o Gemini de forma positiva em quatro dimensões de aceitação: utilidade percebida, facilidade de uso, atitude perante a utilização e intenção comportamental de uso continuado. Os participantes enfatizaram o contributo do Gemini para aumentar a eficiência, aprofundar a compreensão dos materiais e estimular a motivação em tarefas colaborativas e exploratórias. Resultados semelhantes foram obtidos por [61], que analisaram a perceção de 120 docentes e 533 estudantes em vários cursos universitários. O estudo mostra que 68% dos professores e 74% dos estudantes reconhecem benefícios dos **LLM**, incluindo melhorias mensuráveis de 30% na interação professor-estudante e 25% no envolvimento estudantil. Contudo, [61] também identificam limitações significativas em tarefas cognitivamente complexas, como a resolução de problemas avançados em matemática ou filosofia, o que sugere que os **LLM** funcionam melhor como apoio pedagógico do que como substitutos de competências humanas especializadas.

O Google Gemini destaca-se como um dos **LLM** multimodais mais avançados, concebido para processar e integrar texto, imagem, áudio, vídeo e documentos estruturados como PDFs. As suas capacidades multimodais, aliadas às opções diferenciadas de implementação — *Nano* (versão leve para dispositivos), *Pro* (equilíbrio entre desempenho e custo) e *Ultra* (configuração mais poderosa) — tornam-no particularmente adequado a contextos educativos diversos [62]. Ao contrário de modelos anteriores, centrados sobretudo em texto, o Gemini expande as fronteiras da educação suportada por IA ao oferecer recursos multimodais, simulações em tempo real, interação multilingue e feedback adaptativo. Do ponto de vista pedagógico, o Gemini oferece benefícios tanto a estudantes como a docentes. Para os primeiros, funciona como tutor personalizado, capaz de adaptar explicações a diferentes necessidades, gerar visualizações e fornecer clarificações imediatas. Para os docentes, facilita a preparação de materiais instrucionais, questionários e planos de aula, ao mesmo tempo que permite a entrega diferenciada de conteúdos e feedback formativo em tempo real [62]. Estas funcionalidades alinham-se com a literatura mais ampla sobre **LLM** na educação, que os reconhece como parceiros no desenho instrucional e na criação de percursos de aprendizagem adaptativos [59; 51].

Apesar do seu potencial, a integração de **LLM** em contextos educativos coloca desafios significativos. A literatura aponta riscos persistentes como imprecisões factuais e "alucinações", que comprometem a fiabilidade dos conteúdos gerados [51; 62]. Questões de integridade académica, incluindo plágio, *ghostwriting* e dependência excessiva, são destacadas como ameaças à credibilidade do ensino superior [59]. Problemas de enviesamento algorítmico também são críticos: [63] alertam que os **LLM** podem reproduzir desigualdades estruturais ligadas à literacia digital, condições socioeconómicas e acessibilidade, exacerbando lacunas em vez de as mitigar. Preocupações com privacidade e comercialização de dados são particularmente relevantes em sistemas multimodais como o Gemini, que processam texto, áudio e imagens sensíveis dos estudantes [62; 63]. De acordo com [61], estas fragilidades exigem a criação de estruturas robustas de governança de dados e políticas institucionais alinhadas com regulamentações como o RGPD, para garantir confiança no uso pedagógico dos **LLM**.

Em termos técnicos, os **LLM** enfrentam ainda limitações estruturais, como a dependência de janelas de contexto finitas, a ocorrência de alucinações e a dificuldade em incorporar informação factual atualizada [64; 4]. Embora parâmetros de geração como *temperature*

e *top-p* possam modular a aleatoriedade e a diversidade das respostas, eles não eliminam os problemas fundamentais de fiabilidade [65; 66].

Os **LLM**, em especial soluções multimodais como o Gemini, abrem novas possibilidades para a educação ao gerar conteúdos, apoiar explicações e oferecer feedback personalizado. Contudo, a literatura identifica riscos importantes: alucinações, fiabilidade incerta e questões éticas relacionadas com privacidade, plágio e enviesamentos algorítmicos. A estas fragilidades acrescem limitações técnicas, como a dependência de janelas de contexto finitas e a dificuldade de atualização factual após o treino, que restringem a aplicabilidade em contextos institucionais. Por essa razão, a integração de **LLM** em ecossistemas educativos exige enquadramentos robustos de governação e literacia digital acrescida por parte de docentes e estudantes. Estas limitações motivaram o surgimento do paradigma de **RAG**, que procura fundamentar a geração em fontes externas verificáveis, apresentado na secção seguinte.

2.5 Retrieval-Augmented Generation (RAG)

A **RAG** emergiu como um paradigma central para estender as capacidades dos **LLM**, ao combinar inferência generativa com acesso a fontes de conhecimento externas. Ao contrário dos modelos tradicionais, que dependem exclusivamente da informação codificada nos parâmetros de pré-treino, a **RAG** introduz um componente de recuperação que injeta dinamicamente evidência contextual no processo de geração. Esta arquitetura responde a limitações críticas, como as alucinações factuais, o conhecimento desatualizado e a cobertura incompleta de domínios especializados [67].

O princípio definidor é, portanto, a integração de dois processos complementares: a recuperação de informação pertinente proveniente de fontes textuais externas ao modelo e a geração de respostas condicionadas tanto pelo conhecimento recuperado como pelo conhecimento paramétrico. Esta hibridização oferece uma resposta prática a uma das maiores críticas feitas aos **LLM**: a sua tendência para gerar respostas plausíveis mas incorretas.

2.5.1 Princípios Fundamentais e Arquitetura do RAG

A arquitetura do RAG combina as capacidades generativas de um LLM com um mecanismo de recuperação de informação, permitindo que o modelo aceda e utilize dados atualizados ou específicos de um domínio no momento da geração da resposta [64]. Esta abordagem visa superar limitações inerentes aos modelos de linguagem estáticos, como a propensão a "alucinações" e a falta de atualização temporal. A Figura 2.5 ilustra, de forma simplificada, o fluxo de dados do modelo RAG, desde a consulta do utilizador até à resposta gerada.

O pipeline pode ser descrito em três etapas centrais:

1. **Recuperação (Retrieval):** Quando um utilizador submete uma consulta, esta é convertida em embeddings semânticos e usada para procurar informações relevantes numa base de dados externa. Esse processo pode recorrer a técnicas lexicais (como BM25), mas é hoje dominado por métodos baseados em embeddings neuronais [68]. Bases vetoriais modernas permitem consultas rápidas em milhões de documentos e devolvem o *retrieved context*.

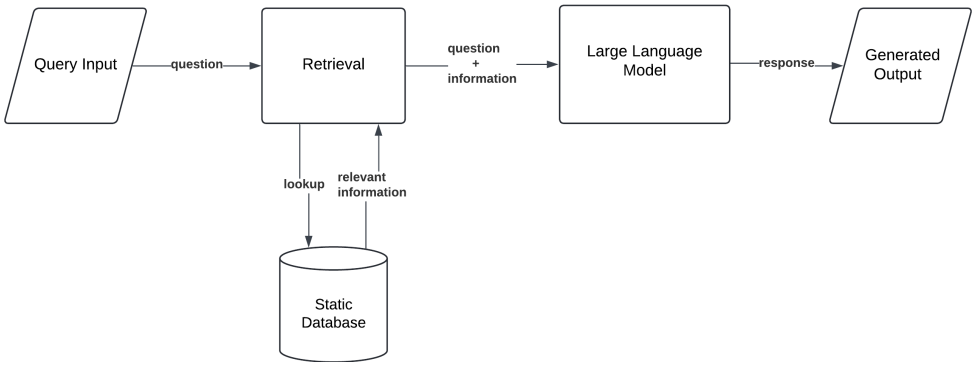


Figura 2.5: Diagrama simplificado da arquitetura RAG.

2. **Aumento (Augmentation):** A informação recuperada é combinada com a consulta original, formando um contexto enriquecido (query + evidência). Esta fase é crucial, pois fornece ao LLM as pistas factuais necessárias para reduzir alucinações e aumentar a precisão da resposta.
3. **Geração (Generation):** O LLM processa a consulta original juntamente com o contexto aumentado, produzindo um output mais fiável e contextualizado, apoiado em evidências explícitas [6].

A eficácia do RAG depende criticamente da qualidade da informação recuperada e da forma como esta é integrada. A escolha do modelo de embedding e a estratégia de busca na base vetorial são determinantes para a relevância do contexto fornecido ao LLM [4; 69].

2.5.2 Componentes Críticos: Embeddings e Bases de Dados Vetoriais

Um elemento central da arquitetura **RAG** é o uso de **embeddings**, representações vetoriais contínuas que captam relações semânticas e sintáticas entre unidades linguísticas [70]. Estas representações permitem mapear consultas, frases ou documentos em espaços de alta dimensão, onde a proximidade entre vetores reflete a semelhança de significado. Assim, duas expressões com sentidos semelhantes ocupam regiões próximas no espaço vetorial, mesmo que não partilhem vocabulário idêntico. Esta propriedade torna possível a recuperação de informação baseada em significado e não apenas em correspondência literal de palavras, sendo um dos mecanismos fundamentais do **RAG** [71; 4].

A evolução dos *embeddings* acompanhou os avanços no processamento de linguagem natural. Primeiras abordagens, como *Word2Vec* [70] e *GloVe* [72], geravam vetores estáticos a nível de palavra, incapazes de lidar com polissemia (ex.: *bank* como rio vs. instituição financeira). Modelos mais recentes, como *BERT* [58] e *Sentence-BERT* [73], introduziram **embeddings contextuais**, nos quais o vetor de uma palavra ou frase depende do seu contexto. Isto permitiu capturar significados mais precisos e melhorar tarefas como recuperação semântica e classificação textual.

Outro aspeto crítico é a **granularidade**. *Embeddings* a nível de palavra são rápidos de calcular, mas limitados em expressividade. Representações a nível de subpalavras mitigam problemas de vocabulário e lidam melhor com línguas morfologicamente ricas. Já os *embeddings* de frases ou documentos oferecem maior robustez semântica, mas aumentam os custos computacionais [73]. A **dimensionalidade** acrescenta outro trade-off: vetores

mais curtos (ex.: 384 dimensões) oferecem menor custo de armazenamento e latência reduzida, mas menor fidelidade semântica; vetores mais longos (ex.: 768 ou 3072 dimensões) capturam melhor as relações semânticas, mas exigem mais memória e tempo de consulta [4; 71].

No projeto em análise, adotou-se o modelo `all-MiniLM-L6-v2` [74], com 384 dimensões, disponibilizado pela biblioteca *Sentence Transformers*. Esta escolha justifica-se por três razões principais: (i) baixo custo computacional, adequado a prototipagem em ambiente académico; (ii) latência reduzida, permitindo respostas em tempo quase real; (iii) desempenho aceitável em benchmarks de similaridade textual [73]. Alternativas mais robustas, como o `all-mpnet-base-v2` (768 dimensões) [75] ou embeddings proprietários em larga escala (e.g., OpenAI `text-embedding-004`, com 3072 dimensões), alcançam maior precisão semântica, mas implicam custos acrescidos em memória, tempo de consulta e dependência de serviços externos.

Para armazenar e consultar esses embeddings, utilizam-se **bases de dados vetoriais**, desenhadas para pesquisa por similaridade em alta dimensão. Estas bases permitem indexar milhões de vetores e realizar consultas do tipo "documentos mais semelhantes a X". Diferem radicalmente das bases relacionais tradicionais: enquanto estas privilegiam consultas exatas sobre dados estruturados, as bases vetoriais recorrem a métricas de distância (e.g., cosseno, euclidiana) para identificar vizinhos próximos em espaços multidimensionais [64].

Entre as soluções mais utilizadas, destacam-se FAISS [76], Qdrant [76], Pinecone [77] e Weaviate [78]. Todas implementam algoritmos de indexação aproximada (ANN – Approximate Nearest Neighbors), com diferentes compromissos entre precisão, latência e escalabilidade. O FAISS, por exemplo, combina estruturas como o IVF (Inverted File Index) [79] com **Product Quantization**, reduzindo espaço de armazenamento com compressão vetorial. O Qdrant utiliza **HNSW (Hierarchical Navigable Small World)**, uma técnica baseada em grafos que assegura pesquisas rápidas mesmo em bases com centenas de milhões de vetores [69]. Estas soluções permitem consultas em tempo quase real, mas os custos de indexação e manutenção crescem proporcionalmente à escala dos dados.

No presente projeto, foi adotada uma abordagem híbrida: Qdrant em nuvem, para suportar escalabilidade e integração com aplicações distribuídas, e Chroma em ambiente local, para prototipagem rápida e acessível. Esta combinação oferece flexibilidade: ambientes de teste beneficiam da simplicidade do Chroma, enquanto cenários mais exigentes recorrem ao Qdrant, que disponibiliza maior robustez em termos de indexação e resiliência. Esta escolha reflete um compromisso entre eficiência e aplicabilidade em contexto educativo, reconhecendo que em ambientes de produção poderia ser vantajoso migrar para embeddings mais expressivos e soluções vetoriais otimizadas para grandes volumes.

A Figura 2.6 ilustra o fluxo simplificado de utilização de embeddings e bases vetoriais num sistema **RAG**, desde a conversão de texto em vetores até à recuperação de documentos relevantes.

Uma diferença estrutural entre o uso de embeddings em **LLM** e em sistemas **RAG** merece destaque. Nos **LLM**, os embeddings fazem parte da arquitetura pré-treinada do modelo e não podem ser alterados sem reconfigurar ou retreinar toda a rede. Em contraste, no **RAG**, o módulo de embeddings é substituível e independente: é possível trocar de MiniLM para MPNet ou para embeddings proprietários (e.g., OpenAI) sem modificar o modelo generativo subjacente. Esta modularidade confere ao **RAG** uma flexibilidade

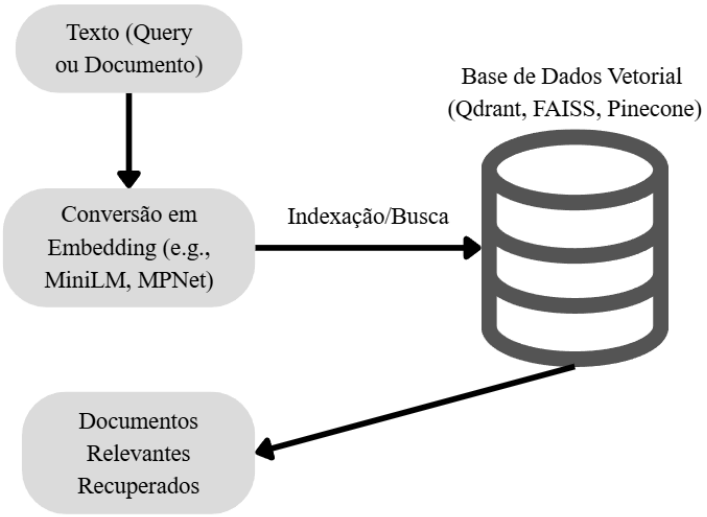


Figura 2.6: Arquitetura **RAG** com base vetorial.

acrescida, permitindo adaptar a precisão semântica e os custos computacionais de acordo com o contexto de aplicação [71; 4].

2.5.3 Variações e Evolução do RAG

A investigação recente tem mostrado que o **RAG** não deve ser entendido como uma técnica única, mas sim como uma **família de arquiteturas em expansão**. De acordo com o survey de [4], é possível distinguir diferentes estágios de sofisticação, que vão do **Naïve RAG**, baseado em recuperação estática numa única passagem, até variantes avançadas que incorporam raciocínio iterativo, integração semântica e planeamento multiagente.

Uma primeira evolução em relação ao modelo mais simples surge com o **Multi-hop RAG**. Nesta abordagem, o sistema encadeia várias consultas de recuperação, permitindo articular evidências dispersas em diferentes documentos. Esta capacidade de raciocínio multi-etapas aproxima a arquitetura da forma como humanos constroem respostas complexas, e tem sido apontada como essencial para domínios de conhecimento denso, como investigação científica e contexto educativo.

Seguindo esta linha, o **M-RAG (Multi-partition RAG)** introduz a ideia de segmentar documentos em múltiplas partições temáticas, reduzindo interferências entre áreas distintas e aumentando a precisão da recuperação [7]. Esta estratégia tem particular utilidade em corpora heterogéneos, como bibliotecas digitais académicas, onde a separação de domínios garante que a informação recuperada permanece relevante para a consulta em causa.

Outra vertente complementar é o **Ontology-Grounded RAG (OG-RAG)**, proposto por [5]. Aqui, os embeddings neuronais são enriquecidos com ontologias formais ou grafos de conhecimento, garantindo maior consistência conceptual e reduzindo ambiguidades. Esta combinação de representações estatísticas com conhecimento simbólico é especialmente crítica em áreas como medicina ou direito, onde relações semânticas explícitas determinam a validade das respostas.

Mais recentemente, a evolução avançou para o conceito de **Agentic RAG**, sistematizado

no survey de [6]. Nestes sistemas, a recuperação deixa de ser um módulo estático e passa a integrar ciclos iterativos de reflexão, planeamento e coordenação multiagente. As capacidades de decompor consultas complexas em subtarefas, refinar sucessivamente os resultados e utilizar ferramentas externas aproximam a RAG de agentes autónomos com raciocínio adaptativo.

A Tabela 2.2 sintetiza estas variações, destacando as principais características e forças de cada paradigma discutido. Esta visão comparativa facilita a compreensão das diferenças práticas entre abordagens, estabelecendo a ligação com as aplicações educativas que se exploram na subsecção seguinte.

Paradigma	**Principais características**	**Forças**
Naïve RAG	- Recuperação lexical simples (TF-IDF, BM25)	- Fácil implementação - Adequado para perguntas factuais diretas
Multi-hop RAG	- Recuperação em múltiplas etapas - Combinação de documentos distintos	- Permite raciocínio multi-etapas - Melhora a cobertura de evidência
M-RAG (Multi-partition)	- Segmentação de documentos em partições temáticas - Redução de interferência entre domínios	- Maior precisão em corpora heterogéneos - Escalabilidade em bases extensas
OG-RAG (Ontology-Grounded)	- Integração de ontologias e grafos de conhecimento - Reforço semântico formal	- Coerência conceptual - Adequado para domínios especializados (médico, jurídico)
Agentic RAG	- Ciclos iterativos de recuperação e geração - Planeamento e coordenação multiagente	- Adaptação dinâmica - Capacidade de raciocínio avançado - Escalável a tarefas complexas

Tabela 2.2: Comparação de variantes de **RAG**, adaptada a partir de [4; 5; 6; 7].

2.5.4 Aplicações Educativas do RAG

Em contextos educativos, a **RAG** tem-se destacado pela ênfase na **fiabilidade** e na **transparência**. Ao ancorar a geração em fontes verificáveis, o sistema fornece não apenas respostas mais precisas, mas também as evidências que as sustentam. Este aspeto é pedagógico e eticamente relevante, pois promove práticas de validação crítica por parte dos estudantes e reforça a confiança docente na utilização de sistemas baseados em IA [80; 67].

A literatura recente identifica vários domínios de aplicação. O primeiro é a **personalização de percursos de aprendizagem**, em que conteúdos são adaptados dinamicamente a partir de repositórios institucionais, alinhando recomendações com currículos específicos [80; 71]. Seguem-se os **sistemas de Q&A pedagógicos**, onde chatbots educativos, ancorados em programas e regulamentos, oferecem respostas factuais e contextualizadas, aumentando a acessibilidade ao conhecimento institucional [81].

Outro domínio em crescimento é a **geração automática de materiais instrucionais**, incluindo resumos, notas de aula e bancos de questões, que aceleram a preparação de recursos e reduzem a carga docente, embora continuem a exigir uma etapa de validação e supervisão humana (curadoria) [82; 71]. Complementarmente, a RAG tem sido aplicada

à **avaliação e feedback**, permitindo criar quizzes, rubricas e relatórios adaptativos, bem como explorar artefactos multimodais, como diagramas e animações [80; 6].

A Tabela 2.3 sintetiza estes domínios, apresentando exemplos práticos, impactos observados e limitações reportadas na literatura.

Domínio de aplicação	**Exemplos práticos**	**Impactos identificados**
Personalização de percursos de aprendizagem	Adaptação dinâmica de conteúdos a partir de repositórios institucionais [80; 71]	Maior relevância contextual; desafios no alinhamento curricular
Sistemas de Q&A pedagógicos	Chatbots educativos ancorados em programas e políticas de curso [80; 81]	Apoio factual e contextualizado; qualidade dependente de embeddings e fontes
Geração de materiais instrucionais	Resumos, notas de aula, bancos de questões [80; 71]	Redução de carga docente; necessidade de curadoria humana
Avaliação e feedback	Criação de quizzes, rubricas e feedback adaptativo [82; 80]	Eficiência acrescida na avaliação; limitações em evidência não textual (e.g., equações, diagramas)
Multimodalidade e escalabilidade	Produção de artefactos multimédia (diagramas, animações, relatórios) [6]	Expansão para novos domínios; custos de infraestrutura

Tabela 2.3: Síntese das aplicações de **RAG** em contextos educativos: exemplos, benefícios e limitações.

Para além dos exemplos específicos, destacam-se benefícios transversais da **RAG**, nomeadamente:

- **Eficiência**, reduzindo tempos de produção de horas para minutos [82; 68];
- **Personalização**, ao adaptar conteúdos dinamicamente às necessidades dos estudantes [81];
- **Fiabilidade**, mitigando alucinações através de evidência explícita [67; 80];
- **Escalabilidade e multimodalidade**, ao possibilitar a criação de recursos diversificados em larga escala [6].

Estes resultados reforçam o potencial transformador da RAG no ensino, mas também expõem fragilidades que não podem ser ignoradas. Questões técnicas, éticas e pedagógicas emergem como desafios centrais à sua adoção, e serão aprofundadas na subsecção seguinte.

2.5.5 Desafios Técnicos, Éticos e Pedagógicos

Apesar do seu potencial, os sistemas de **RAG** enfrentam desafios críticos que condicionam a sua adoção em larga escala. Estes desafios distribuem-se em três eixos principais: técnico, pedagógico e ético-legal.

Desafios Técnicos. Mesmo quando os mecanismos de recuperação devolvem documentos relevantes, a **integração contextual** continua a ser problemática. Como observam [6], respostas podem tornar-se fragmentadas ou genéricas, sem conseguir articular as evidências num raciocínio coeso. Um exemplo é a dificuldade em transformar artigos recentes sobre Alzheimer em explicações claras sobre implicações clínicas em estágios iniciais, ou a incapacidade de adaptar recomendações agrícolas genéricas a regiões áridas.

Outro problema central é o **raciocínio multi-etapas**. Consultas complexas que exigem encadear múltiplas fontes, como a análise comparativa de políticas de energia renovável na Europa e o seu impacto económico em países em desenvolvimento, tendem a gerar respostas incompletas por falta de mecanismos iterativos de refinamento [6].

Em termos de **latência e escalabilidade**, a pressão sobre bases de dados vetoriais aumenta à medida que crescem os volumes de dados. Estudos como [69; 71] mostram que retrieval e re-ranking tornam-se gargalos em sistemas com milhões de documentos, comprometendo a utilização em tempo real. Singh (2025) ilustra o impacto em cenários críticos, como análises financeiras de alta frequência, onde atrasos de segundos podem anular a utilidade da resposta.

A **qualidade da recuperação** constitui ainda um desafio transversal. Segundo [71], a presença de ruído ou a ausência de documentos relevantes pode induzir erros graves, uma vez que os modelos continuam a gerar respostas mesmo sem evidência. Abordagens como o HyDE (Hypothetical Document Embeddings) ou métodos de re-ranking baseados em modelos generativos (GERE, PARADE, RankT5) têm sido propostas para mitigar estas fragilidades. Contudo, a adoção destas técnicas aumenta a complexidade e os custos de operação.

Por fim, surgem problemas de **eficiência computacional**. Modelos de re-ranking sofisticados, como monoT5 ou RankLLaMA, oferecem ganhos de precisão, mas implicam custos elevados de processamento. Estratégias híbridas que combinam métodos esparsos e densos, ou abordagens de pruning seletivo, têm sido exploradas para reduzir a sobrecarga [71]. Estes trade-offs mostram que a arquitetura técnica da RAG continua em aberto e sujeita a otimização contínua.

Desafios Pedagógicos. No contexto educativo, a dependência de **curadoria docente** mantém-se incontornável. Sistemas de geração de quizzes, notas de aula ou feedback automático exigem supervisão para garantir alinhamento curricular e qualidade das fontes [80]. Persistem ainda dificuldades em lidar com conteúdos multimodais — como equações matemáticas ou diagramas técnicos — que não se traduzem facilmente em embeddings textuais. Estas limitações restringem a aplicabilidade plena da RAG em disciplinas de forte componente visual ou simbólica.

Desafios Éticos e Legais. A utilização de RAG em ambientes educativos levanta também preocupações de privacidade e governação de dados. O estudo de [83] mostra que estudantes e docentes valorizam a personalização proporcionada por memórias RAG, mas expressam receios quanto à retenção de informação pessoal e à falta de transparência sobre políticas de armazenamento. Questões adicionais incluem riscos de plágio, uso indevido de materiais sensíveis e ausência de métricas claras para avaliar impacto educativo [80].

A Tabela 2.4 resume estes desafios, destacando exemplos concretos e as principais referências que os documentam.

Dimensão	Exemplo de desafio	Referências
Técnicos	Integração contextual deficiente: dificuldade em articular documentos recuperados em respostas coesas (ex.: Alzheimer, práticas agrícolas em regiões áridas).	[6]
Técnicos	Limitações em raciocínio multi-etapas: incapacidade de encadear políticas energéticas com impactos económicos em países em desenvolvimento.	[6]
Técnicos	Latência e escalabilidade: degradação em consultas sobre grandes bases; impacto em contextos de tempo real (ex.: análises financeiras).	[69; 71; 6]
Técnicos	Qualidade da recuperação: ruído, ausência de documentos relevantes, necessidade de técnicas como HyDE ou reranking com modelos generativos.	[71]
Pedagógicos	Dependência de curadoria docente: quizzes e materiais gerados requerem supervisão para garantir alinhamento curricular.	[80]
Pedagógicos	Limitações multimodais: dificuldade em lidar com equações, diagramas e conteúdos não textuais.	[80; 71]
Éticos/Legais	Privacidade e retenção de dados: perceções negativas de estudantes e docentes sobre transparência da memória RAG.	[83]
Éticos/Legais	Plágio e uso indevido de materiais sensíveis; ausência de métricas claras de impacto educativo.	[80]

Tabela 2.4: Síntese dos desafios técnicos, pedagógicos e éticos/legais associados à adoção de **RAG** em educação.

Estas fragilidades, identificadas de forma transversal na literatura, evidenciam que a adoção da **RAG** em educação não pode ser dissociada de um enquadramento técnico robusto e de princípios éticos claros.

A **RAG** responde de forma direta a algumas das fragilidades dos **LLM**, ao combinar inferência generativa com recuperação de informação factual. Ao ancorar as respostas em fontes verificáveis, a **RAG** melhora a transparência, a personalização e a fiabilidade dos resultados. Variedades emergentes, como o *M-RAG* [7], o *OG-RAG* [5] e a *Agentic RAG* [4], mostram avanços no controlo de ruído e na coerência semântica, embora ainda careçam de validação empírica robusta em contextos educativos reais. Subsistem também desafios ligados à escalabilidade técnica, ao custo de infraestrutura e à ausência de métricas pedagógicas consolidadas. Neste enquadramento, a integração com mecanismos de interoperabilidade torna-se essencial para operacionalizar estas arquiteturas em **LMS**. É precisamente neste ponto que o **MCP** se revela relevante, funcionando como camada de ligação entre diferentes sistemas, conforme explorado na secção seguinte.

2.6 Model Context Protocol (MCP)

O **MCP** surgiu recentemente como um padrão aberto concebido para orquestrar interações entre **LLM**, ferramentas externas e sistemas de dados. Assenta na especificação *JSON-RPC 2.0*, que define uma comunicação estruturada e bidirecional entre clientes e servidores, permitindo a troca consistente de pedidos, respostas e notificações [84]. Esta arquitetura formaliza três papéis principais — *hosts* (aplicações que executam **LLM** e gerem interações com utilizadores), *clients* (conectores que iniciam pedidos a partir dos hosts) e *servers* (programas que expõem ferramentas, recursos ou *prompts*) — assegurando modularidade, interoperabilidade e extensibilidade sem exigir integrações específicas para cada sistema [85; 86]. Ao separar claramente responsabilidades, o protocolo permite integrar novos serviços de forma escalável e coesa em ambientes heterogéneos. O propósito do **MCP** vai além da eficiência técnica: procura colmatar a fragmentação existente nas soluções de integração de IA, caracterizadas por abordagens ad hoc que dificultam a interoperabilidade, a escalabilidade e a segurança. Vários autores identificam o **MCP** como um passo fundamental para estabelecer uma orquestração transparente, padronizada e segura de aplicações com **LLM**, desempenhando um papel análogo ao do HTTP na evolução da web [85; 86]. De acordo com [87], a sua principal contribuição é reduzir a complexidade da integração, permitindo que sistemas baseados em **LLM** escalem de forma eficaz em infraestruturas institucionais. A especificação oficial reforça esta visão, definindo o protocolo não apenas como um meio de troca de dados, mas como um enquadramento para partilha de contexto, acesso a recursos e coordenação de ferramentas, sustentando ecossistemas de IA adaptativos e interoperáveis [84]. A Figura 2.7 apresenta uma visão de alto nível da arquitetura do **MCP**, mostrando como múltiplos servidores podem ser integrados em paralelo através de clientes embutidos no host.

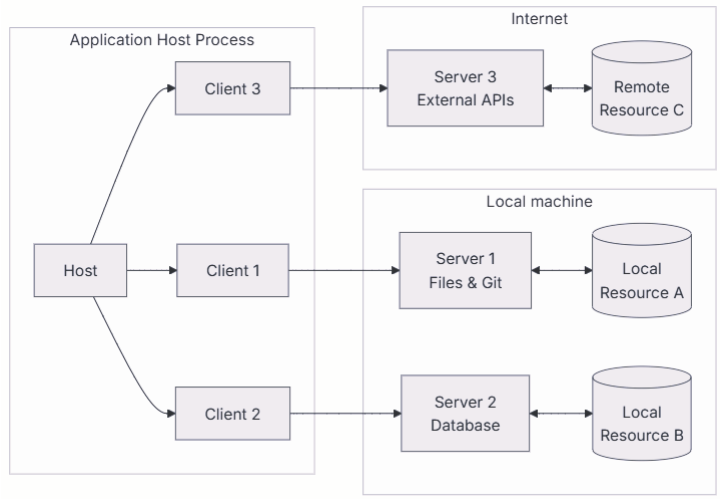


Figura 2.7: Arquitetura do MCP: conexão entre hosts, clientes e servidores.

A especificação define ainda um ciclo de vida das interações dividido em três fases: *inicialização*, com negociação de capacidades; *operação*, em que ocorrem os pedidos e respostas (e.g., invocar uma ferramenta ou aceder a um recurso); e *encerramento*, que garante a terminação limpa da sessão [84].A Figura 2.8 sintetiza este processo, destacando a sequência estruturada que garante previsibilidade e segurança em cada sessão.

O **MCP** é independente do transporte: atualmente suporta tanto *stdio* (onde clientes

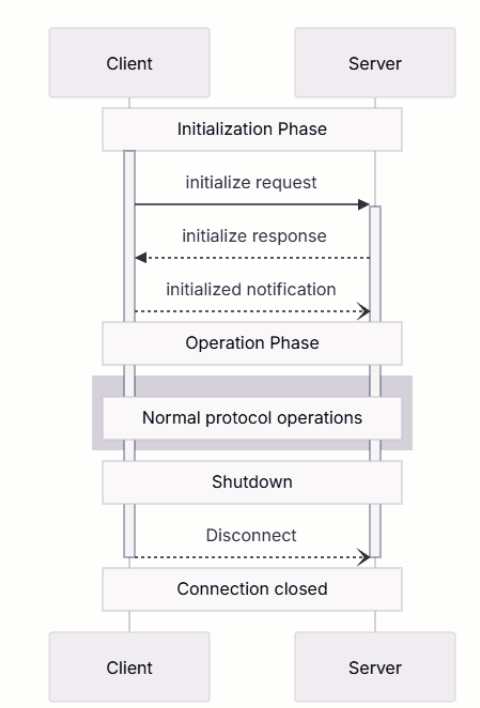


Figura 2.8: Ciclo de vida do **MCP**: inicialização, operação e encerramento.

lançam servidores como subprocessos, trocando mensagens via entrada/saída padrão, adequado a cenários locais) como *HTTP com streaming* (suportando múltiplas ligações concorrentes, eventos enviados pelo servidor e reenvio de mensagens, apropriado a infraestruturas distribuídas) [84]. A extensibilidade é reforçada pela possibilidade de definir transportes adicionais, desde que respeitem a codificação JSON-RPC.

Do ponto de vista funcional, o protocolo organiza a interação em blocos modulares. No lado do servidor, o **MCP** define três primitivas: *tools* (operações definidas por esquemas, invocáveis por **LLM**, como consultar uma base de dados), *resources* (acesso estruturado a informação contextual via URIs, como documentos ou APIs) e *prompts* (modelos parametrizados para fluxos de trabalho consistentes e reutilizáveis) [84]. No lado do cliente, estão previstas funcionalidades como *sampling* (controlo da geração de texto), *elicitation* (estruturas de *prompting*) e *roots* (pontos de entrada para interação com o modelo).

A segurança é um princípio central do **MCP**. O protocolo integra fluxos de autorização compatíveis com OAuth 2.0 e requer consentimento explícito para operações sensíveis, mitigando vulnerabilidades como uso indevido de tokens ou sequestro de sessões [84]. Além disso, todas as ferramentas e recursos expostos por servidores devem ser validados por esquema, garantindo transparência, rastreabilidade e auditabilidade das ações realizadas por IA. Estas medidas estabelecem um modelo de confiança adequado a domínios sensíveis, como a educação.

Ao contrário de protocolos clássicos como o **LTI**, o **SCORM** ou o **xAPI**, concebidos sobretudo para integração de conteúdos e ferramentas de forma estática, o **MCP** foi desenhado para orquestrar interações dinâmicas entre **LLM**, recursos contextuais e serviços externos. Esta característica confere-lhe vantagens conceptuais no domínio da personalização e da adaptabilidade, respondendo a limitações anteriormente identificadas nos **LMS**. Contudo, dada a sua introdução recente, ainda não existem relatos consolidados

de implementações em contextos educativos reais. A sua aplicação em plataformas como o Moodle permanece, por agora, sobretudo a nível exploratório e conceptual, ainda que com elevado potencial de evolução futura.

A relevância do **MCP** em ecossistemas educativos decorre da sua capacidade de oferecer uma infraestrutura padronizada, segura e extensível para integrar **LLM** e recursos digitais em plataformas de aprendizagem. Em ambientes Moodle, por exemplo, o **MCP** pode facilitar o acesso em tempo real a bases de dados académicas, APIs institucionais ou repositórios externos através de *resources*, enquanto permite aos **LLM** gerar avaliações adaptativas ou feedback personalizado invocando *tools*. Do mesmo modo, *prompts* parametrizados podem ser usados para fluxos de trabalho consistentes na criação de questionários, resumos ou ciclos de feedback automático.

De uma perspetiva institucional, o **MCP** oferece vantagens em escalabilidade e governação. A sua arquitetura independente do transporte garante integração em ambientes híbridos (nuvem, servidores locais, aplicações multiplataforma), mantendo a interoperabilidade em ecossistemas digitais cada vez mais complexos. As disposições de segurança do protocolo — como separação de fronteiras de confiança e mecanismos explícitos de autorização — alinham-se com requisitos de proteção de dados educacionais, reforçando a confiança em processos de aprendizagem mediados por IA.

Embora ainda em fase inicial de adoção, o **MCP** surge na literatura como um padrão promissor para superar limitações dos protocolos tradicionais, ao permitir a orquestração direta de ferramentas baseadas em **LLM**. A sua estrutura baseada em jsonrpc 2.0 garante comunicação consistente e modularidade entre cliente e servidor, reduzindo a complexidade da integração com plataformas educativas. Quando articulado com pipelines de **RAG**, o **MCP** oferece um modelo unificado para recuperar e gerar conteúdos educativos de forma controlada e transparente. Esta capacidade de integração entre interoperabilidade e geração contextualizada estabelece o enquadramento para a arquitetura proposta nesta dissertação, apresentada no capítulo seguinte.

2.7 Integração de Tecnologias e Trabalhos Relacionados

A integração de inteligência artificial em plataformas de gestão de aprendizagem tem evoluído rapidamente, impulsionada pela procura de soluções que reduzam a carga de trabalho docente e aumentem a personalização. Em particular, o Moodle tem sido um terreno fértil para experimentação, devido à sua arquitetura modular e natureza *open source*. A presente secção sintetiza as principais abordagens publicadas, analisando as suas contribuições, limitações e relevância para o desenvolvimento da arquitetura MCP–RAG proposta nesta dissertação.

2.7.1 Integração de IA em plataformas Moodle

Nos últimos anos, várias tentativas foram feitas para incorporar **LLM** e mecanismos de geração automática de conteúdos no Moodle. Os autores de [88] analisam um conjunto de *plug-ins* destinados a automatizar a criação de questionários e recursos pedagógicos. Embora demonstrem ganhos de eficiência, estas soluções mantêm uma forte dependência de configuração manual e carecem de algoritmos de personalização baseados em dados de

aprendizagem. De forma semelhante, [30] exploram a integração de ChatGPT no Moodle 4.5, comparando-a com plataformas como Canvas e Blackboard. O estudo demonstra que o Moodle oferece vantagens em flexibilidade, mas também limitações técnicas — sobretudo no tratamento de dados sensíveis e na integração curricular. Estas análises apontam para uma tendência: a maioria das abordagens permanece confinada ao nível do *plug-in*, sem arquitetura de interoperabilidade que permita expansão modular e escalável.

2.7.2 Abordagens híbridas com RAG e LLM

A integração de **RAG** com **LLM** representa um avanço significativo face às soluções puramente generativas. O trabalho de [89] apresenta o *MoodleBot*, um sistema baseado em **RAG** e **LLM** integrado diretamente no Moodle. Testado em cursos de sistemas de informação, obteve 88 % de exatidão e elevada aceitação pelos estudantes, confirmando a utilidade pedagógica da abordagem. Todavia, os autores identificam custos operacionais elevados, necessidade de *fact-checking* manual e resistência docente. Já [81], numa revisão de 47 estudos sobre chatbots educativos com **RAG**, mostram que embora a arquitetura melhore a factualidade e reduza alucinações, a maioria dos protótipos ainda não se encontra integrada em **LMS**, revelando a ausência de padrões de interoperabilidade consistentes.

A Tabela 2.5 sintetiza as soluções mais relevantes, destacando objetivos, tipo de integração e limitações observadas.

Solução/Projeto	**Objetivo**	**Integração com Moodle/IA**	**Limitações**
Younes-Aziz (2024)	Automatizar criação de questionários e recursos	*Plug-ins* Moodle	Configuração manual intensiva; ausência de personalização baseada em dados
Kaleci (2025)	Comparar integração de ChatGPT em vários LMS	ChatGPT no Moodle 4.5 (vs. Canvas, Blackboard, Open edX)	Privacidade; alinhamento curricular; custos de infraestrutura
Neumann et al. (2025)	MoodleBot com RAG + LLM	Integração direta no Moodle; validado com estudantes	Custos de utilização; necessidade de *fact-checking*; resistência docente
Swacha (2025)	Revisão de chatbots educativos com RAG	Implementações experimentais (fora do Moodle)	Soluções fragmentadas; ausência de integração estável em LMS
Nayef-Shaie (2024)	Discussão conceptual da IA em educação	Perspetiva geral (não centrada em Moodle)	Falta de casos práticos; lacunas em escalabilidade e ética

Tabela 2.5: Comparação de soluções existentes para integração de IA.

A análise da tabela confirma a diversidade de esforços e o caráter ainda experimental destas abordagens. Apesar de avanços pontuais, a ausência de uma camada de orquestração comum impede que os resultados sejam generalizáveis ou escaláveis.

2.7.3 Desafios éticos e de governação

Para além das limitações técnicas, a literatura identifica preocupações de natureza ética e institucional. [90] discutem os riscos associados à recolha e tratamento de dados estudantis, ao plágio automatizado e à falta de literacia digital entre docentes e alunos. De igual modo, [30] sublinham que a dependência de serviços externos de IA pode comprometer a soberania dos dados das instituições de ensino. Estes autores convergem na necessidade de políticas de governação claras e de infraestruturas auditáveis, capazes de garantir transparência e conformidade com normas de proteção de dados, como o RGPD. Estas preocupações justificam a busca por soluções interoperáveis e modulares, em que a instituição retenha controlo sobre a comunicação e os fluxos de informação — uma função que o **MCP** pode desempenhar de forma estruturante.

2.7.4 Lacunas e caminhos emergentes

A análise das soluções estudadas permite consolidar um conjunto de limitações transversais que condicionam a adoção plena da inteligência artificial em ambientes de aprendizagem. Em primeiro lugar, verifica-se uma tendência para a **automação parcial**, em que as abordagens se concentram em tarefas isoladas — como a geração de questionários ou resumos — sem abranger de forma integrada o ciclo completo de gestão, avaliação e personalização do conhecimento. Em segundo lugar, observa-se uma **dependência excessiva de configuração manual**, que limita a escalabilidade e reduz o impacto da personalização adaptativa. Em terceiro, surgem restrições de **escalabilidade e desempenho**, sobretudo associadas à latência de consultas vetoriais e aos custos de operação de modelos de linguagem em larga escala. Por fim, mantém-se um défice generalizado em **mecanismos de governação e ética digital**, com lacunas na rastreabilidade, transparência e proteção de dados estudantis [30; 81; 90].

Apesar dos avanços técnicos e das melhorias progressivas em precisão e usabilidade, a literatura revela que a maioria dos protótipos carece de integração profunda com plataformas institucionais e de validação empírica em contextos de ensino reais. Estudos como os de [89] e [81] confirmam a viabilidade de sistemas baseados em **RAG**, mas também evidenciam os desafios logísticos e pedagógicos que impedem a sua adoção generalizada. Além disso, a revisão demonstra que **nenhuma das soluções identificadas recorre a um protocolo de orquestração universal** capaz de unificar a comunicação entre **LMS**, **LLM** e repositórios de conhecimento. Esta ausência de padronização representa uma das lacunas mais críticas do ecossistema atual.

O surgimento do **MCP** introduz, neste contexto, uma possibilidade conceptual promissora: a de estabelecer uma camada de interoperabilidade estável e extensível entre modelos, dados e plataformas educativas. Contudo, a literatura ainda não fornece evidência empírica sobre a sua aplicação em ambientes pedagógicos nem sobre os impactos que tal integração poderá ter na qualidade do conteúdo gerado, na privacidade dos utilizadores ou na eficiência das operações.

Em síntese, o panorama atual revela um cenário fragmentado, com soluções pontuais mas sem uma arquitetura unificadora que assegure automação, personalização e governação de forma integrada. Esta constatação abre caminho a novas linhas de investigação que explorem **modelos de interoperabilidade baseados em protocolos** e **integrações dinâmicas com pipelines de RAG**. O capítulo seguinte apresenta a **arquitetura**

proposta, concebida precisamente para responder a estas lacunas através da integração entre o Moodle, o **MCP** e mecanismos de **RAG**.

Capítulo 3

Desenho e especificação

3.1 Visão geral da arquitetura

A arquitetura proposta foi concebida com o intuito de responder às lacunas identificadas na literatura, nomeadamente a escassez de soluções de automação que possam ser generalizadas para diferentes contextos e a ausência de uma integração eficaz entre modelos de linguagem de grande escala (LLM), mecanismos de recuperação aumentada por geração (RAG) e sistemas de gestão de aprendizagem (LMS). Esta abordagem visa promover uma maior interoperabilidade entre componentes inteligentes, potencializando aplicações mais adaptativas e eficientes no domínio educacional.

O MCP Client, executado em aplicação independente, estabelece ligação com o LMS (Moodle) para autenticação e integração de conteúdos. Através de uma interface web desenvolvida em Flask UI, os estudantes e docentes submetem pedidos que são encaminhados para o servidor. Este módulo gere a sessão dos utilizadores e distingue perfis de acesso (Aluno/Professor), permitindo personalizar a interação e adaptar a experiência de utilização.

O MCP Server concentra a lógica de processamento avançada. É através dele que são expostas ferramentas (`mcp.tools()`) e que se encontra integrado o pipeline de Retrieval-Augmented Generation (RAG). As consultas submetidas pelo utilizador são encaminhadas pelo cliente **MCP**, que decide qual a ferramenta a invocar consoante o tipo de pedido. Nem todas as ferramentas recorrem ao módulo **RAG**; apenas aquelas que requerem acesso contextual a documentos educativos — como geração de resumos, testes ou explicações — passam pelo pipeline de *retrieval*, *augmentation* e *generation*. Neste processo, a questão do utilizador é convertida em embeddings e comparada com uma base vetorial (Qdrant ou Chroma) para identificar documentos relevantes; os textos recuperados são depois combinados com a questão original, formando um contexto enriquecido que é fornecido a um **LLM**. Os documentos utilizados provêm diretamente do Moodle, assegurando que as respostas geradas se mantêm alinhadas com os materiais institucionais e reduzindo o risco de alucinações.

O armazenamento persistente é garantido por uma instância de PostgreSQL em Docker, utilizada tanto pelo MCP Client como pelo MCP Server. Esta base de dados centraliza logs, métricas e registos de sessões, fornecendo suporte a auditoria técnica, monitorização de utilização e análise de desempenho (ex.: tempos médios de resposta, taxas de sucesso

e erro). Embora ainda não esteja implementada a análise pedagógica de conteúdo, a infraestrutura já suporta essa evolução futura.

Do ponto de vista tecnológico, a arquitetura combina os seguintes elementos:

- Moodle, enquanto **LMS** institucional;
- Flask, responsável pela camada de interface web;
- MCP, utilizado como protocolo de interoperabilidade entre cliente, servidor e ferramentas;
- Qdrant/Chroma, enquanto bases vetoriais para recuperação semântica;
- Gemini, OpenAI e Ollama, como fornecedores de **LLM** já integrados no sistema;
- PostgreSQL em Docker, utilizado para armazenamento transacional, incluindo logs, métricas e registos de sessões.

Em termos de escalabilidade, a arquitetura é desenhada de forma modular, permitindo a substituição e configuração independente de componentes já suportados. Isto significa que, no caso dos **LLM**, é possível alternar entre diferentes modelos de fornecedores previamente integrados (como OpenAI, Gemini ou Ollama) sem necessidade de alterações no código, apenas ajustando parâmetros de configuração. No entanto, a integração de novos fornecedores de LLM, ou de novas bases vetoriais além das já previstas (Qdrant e Chroma), implica necessariamente a extensão do código. A facilidade de manutenção é assegurada pela clara separação de responsabilidades entre cliente e servidor, pela utilização de tecnologias containerizadas e pelo uso de protocolos padronizados, que reduzem a dependência de soluções proprietárias. Assim, o sistema pode evoluir de forma incremental dentro do conjunto de integrações já implementadas, mantendo a compatibilidade com o ecossistema institucional.

A comunicação entre módulos é realizada de forma desacoplada, recorrendo a interfaces REST padronizadas e ao protocolo **MCP** para invocação remota de ferramentas. Esta abordagem reduz dependências diretas entre componentes e aumenta a resiliência do sistema, permitindo manutenção e evolução independentes de cada módulo.

A Figura 3.1 apresenta uma visão esquemática desta arquitetura, destacando os principais componentes e fluxos de dados.

Como se observa na Figura 3.1, o MCP Client é o ponto de contacto com o utilizador, interagindo com o Moodle para autenticação e gestão de perfis. O MCP Server integra o pipeline RAG, recorrendo a bases vetoriais para recuperação de documentos pedagógicos e utilizando LLMs na fase de geração. Ambos comunicam com uma instância de PostgreSQL em Docker, que centraliza logs, métricas e dados transacionais, assegurando a monitorização e a evolução futura da plataforma.

3.2 Requisitos do Sistema

O presente capítulo descreve o processo de conceção e desenvolvimento do sistema proposto, concebido para integrar modelos de linguagem e mecanismos de recuperação de informação (**RAG**) no ambiente do Moodle. A sua implementação resulta da necessidade

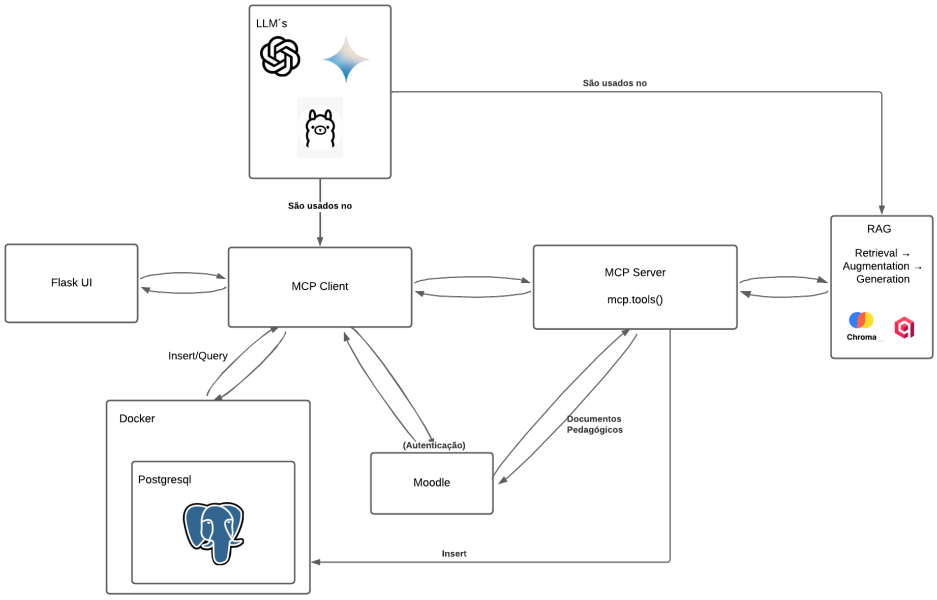


Figura 3.1: Arquitetura geral do sistema. Módulos, fluxos de dados e tecnologias.

de ultrapassar as limitações identificadas na revisão do estado da arte, evidenciando a importância de soluções que combinem personalização, interoperabilidade e automação no contexto educativo. Neste sentido, o sistema foi projetado para servir como uma ponte entre plataformas de gestão de aprendizagem e tecnologias de inteligência artificial, explorando o potencial dos **LLM** na geração e organização de conteúdos pedagógicos. A arquitetura baseia-se num servidor **MCP**, responsável por assegurar a comunicação e a interoperabilidade entre os diferentes componentes. Os requisitos funcionais e técnicos foram definidos tendo em conta os constrangimentos observados — nomeadamente a falta de automação na criação de conteúdos, a ausência de integração nativa de **LLM** e a dificuldade de monitorizar o desempenho destas ferramentas em contexto educativo.

3.2.1 Requisitos Funcionais

Os requisitos funcionais descrevem as principais capacidades que o sistema deve oferecer:

RF1. **Autenticação e gestão de utilizadores:** permitir o acesso seguro de docentes e estudantes, com associação opcional a contas Moodle.

RF2. **Gestão de conversas:** possibilitar que cada utilizador crie, edite e elimine conversas com os modelos de linguagem, mantendo o histórico de interações.

RF3. **Integração com o Moodle:** obter e enviar dados através de webservices, permitindo a recuperação de cursos, recursos e conteúdos educativos.

RF4. **Geração de conteúdo com LLMs:** criar textos, resumos, perguntas e explicações personalizadas com base nos dados do utilizador ou nos materiais do curso.

RF5. **Recuperação de informação com RAG:** pesquisar documentos e materiais institucionais para fundamentar as respostas geradas pelos modelos.

RF6. **Comutação entre provedores de IA:** permitir a utilização de diferentes modelos (Gemini, OpenAI e Ollama) consoante o contexto, o custo e o tipo de tarefa.

RF7. Registo de operações e métricas: armazenar logs detalhados de cada operação, incluindo tipo de ação, modelo utilizado, duração e eventuais erros.

RF8. Interface Web responsiva: disponibilizar uma aplicação acessível via navegador, compatível com dispositivos móveis e com suporte a modo escuro/claro.

3.2.2 Requisitos Não Funcionais

Os requisitos não funcionais estabelecem as propriedades de qualidade que o sistema deve garantir:

RNF1. Escalabilidade: suportar a execução simultânea de múltiplos pedidos sem comprometer o desempenho.

RNF2. Segurança: garantir a confidencialidade e integridade dos dados dos utilizadores, implementando autenticação baseada em tokens, encriptação de comunicações (HTTPS/TLS) e controlo de acesso por níveis de permissão.

RNF3. Desempenho: responder em tempo útil (latência média inferior a 3 segundos em operações típicas).

RNF4. Portabilidade: permitir a execução em ambientes locais ou em servidores remotos, sem dependência de infraestrutura específica.

RNF5. Observabilidade: registar métricas e logs para facilitar a análise de desempenho e a deteção de falhas.

RNF6. Usabilidade: apresentar uma interface simples e coerente, adequada a perfis técnicos e não técnicos, garantindo que o número de interações necessárias para realizar as principais tarefas seja o mais reduzido possível.

RNF7. Modularidade: garantir que cada componente (cliente, servidor, base de dados, **RAG**) possa evoluir de forma independente.

Os requisitos não funcionais foram definidos com foco em desempenho, escalabilidade e segurança. Essa abordagem garante que a solução seja viável em ambientes institucionais de grande escala. Por exemplo, a escalabilidade foi considerada essencial para permitir que o sistema possa ser reutilizado com diferentes instâncias de Moodle ou outros **LMS**, enquanto a fiabilidade e a tolerância a falhas asseguram o funcionamento contínuo do servidor **MCP** e das suas ferramentas mesmo em condições adversas. A modularidade e a interoperabilidade, por sua vez, foram incluídas para permitir a substituição ou atualização de componentes (como motores de RAG ou modelos de linguagem) sem necessidade de reconfiguração estrutural.

3.2.3 Arquitetura de Dados

A definição da arquitetura de dados decorre diretamente do estudo da arquitetura **LLM** apresentado anteriormente, refletindo a necessidade de suportar a integração entre componentes de geração, recuperação e orquestração de informação. O modelo de dados foi concebido para assegurar a interoperabilidade entre o Moodle, o servidor **MCP** e os módulos **RAG**, permitindo a gestão eficiente de conteúdos educativos e o armazenamento de contextos semânticos utilizados pelos modelos de linguagem.

Os dados geridos pelo sistema são de natureza heterogénea, abrangendo tanto informação estruturada (utilizadores, operações, métricas) como não estruturada (conteúdo textual e embeddings semânticos). As principais categorias são:

- **Dados de utilizador:** identificadores, preferências e histórico de interação.
- **Dados educativos:** conteúdos provenientes do Moodle (recursos, tópicos, atividades).
- **Dados semânticos:** vetores de embeddings e documentos indexados no módulo **RAG**.
- **Dados operacionais:** métricas de desempenho e registos de atividade armazenados na base de dados PostgreSQL.

Esta arquitetura foi desenhada para garantir que os fluxos de informação entre o Moodle e o **MCP** são rastreáveis, coerentes e adaptáveis, permitindo que os modelos de linguagem operem sobre dados contextuais atualizados. A estrutura de dados resultante foi modelada de modo a representar de forma consistente as entidades envolvidas nas operações de autenticação, gestão de utilizadores, comunicação entre o Moodle e o servidor **MCP**, e registo de interações. A modelação completa da base de dados encontra-se detalhada na secção 3.3. A rastreabilidade entre os requisitos de dados e a estrutura de armazenamento foi assegurada na fase de modelação, garantindo que cada requisito identificado na secção 3.3 tem correspondência direta nas entidades da base de dados relacional. Esta abordagem de engenharia de requisitos assegura coerência e verificabilidade entre o modelo conceptual e a implementação técnica.

3.3 Estrutura da Base de Dados

A camada de persistência de dados do sistema foi implementada em **PostgreSQL**, uma escolha motivada pela sua fiabilidade, escalabilidade e suporte nativo a tipos de dados avançados, como `JSONB`. Este formato permite armazenar e consultar dados semi-estruturados em formato binário, combinando a flexibilidade do *NoSQL* com as garantias transacionais de uma base relacional. No contexto do sistema desenvolvido, o `JSONB` é utilizado para representar objetos dinâmicos, como respostas dos modelos de linguagem ou estruturas de embeddings, mantendo desempenho elevado e capacidade de indexação [91].

A base de dados assegura a rastreabilidade das operações, o armazenamento das interações entre utilizadores e modelos de linguagem e a monitorização de desempenho do sistema. A sua estrutura reflete os princípios de modularidade e interoperabilidade definidos na arquitetura global, funcionando como elo de ligação entre o Moodle, o servidor **MCP** e os pipelines de **RAG**.

A Figura 3.2 apresenta o modelo relacional simplificado da base de dados, gerado a partir da linguagem `DBML` no serviço *dbdiagram.io*. Foram incluídas apenas as tabelas efetivamente utilizadas na implementação final, removendo entidades experimentais como `usage_stats`, `roles`, `quiz_generations`, `user_sessions`, `video_generations` e `user_roles`. O resultado é um esquema conciso e otimizado para a integração de dados provenientes de múltiplos componentes e para a comunicação eficiente entre o servidor **MCP**, o Moodle e os módulos **RAG**.

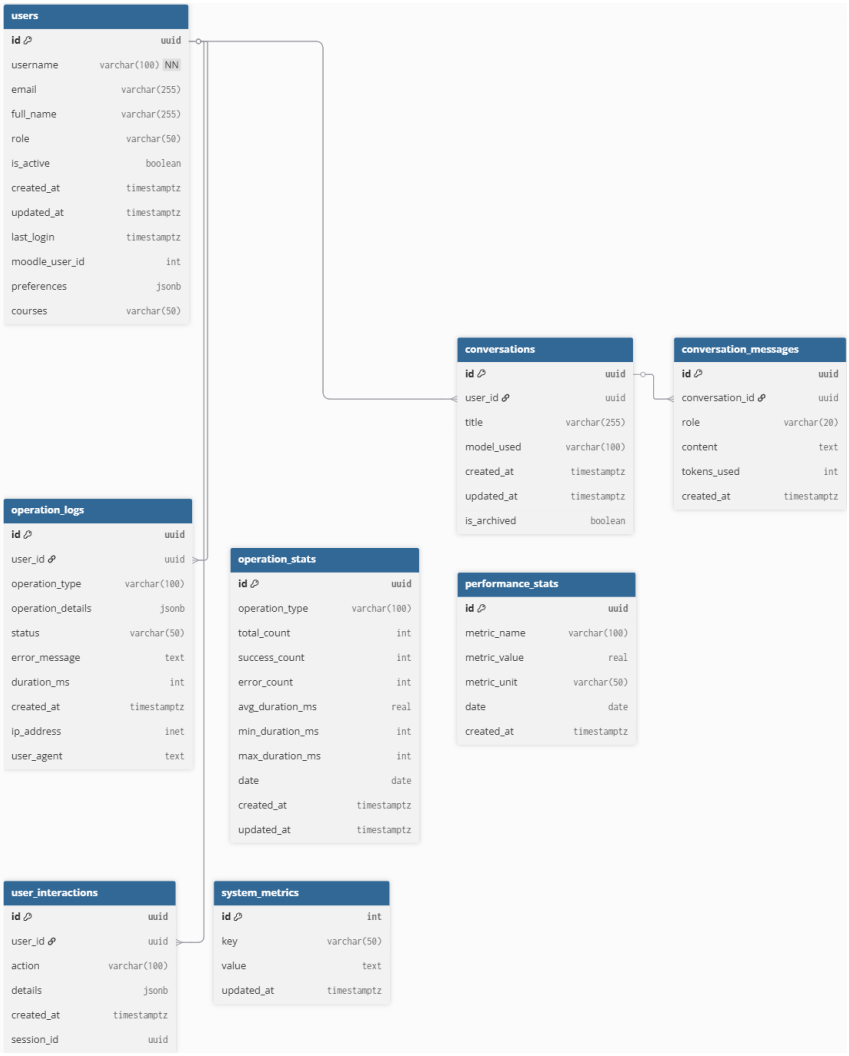


Figura 3.2: Modelo relacional simplificado da base de dados PostgreSQL.

A base de dados é composta por seis grupos funcionais principais:

1. **Utilizadores (users)**: armazena os dados essenciais dos utilizadores autenticados, incluindo identificação, preferências e referências a contas Moodle. Cada registo é identificado por uma chave primária do tipo `UUID`, assegurando unicidade global e interoperabilidade entre instâncias.

2. **Conversações (conversations, conversation_messages)**: regista as interações entre utilizadores e modelos de linguagem. A tabela **conversations** define o contexto geral de cada sessão, enquanto **conversation_messages** guarda as mensagens individuais, o papel de cada interlocutor (**user** ou **assistant**) e o número de tokens utilizados. Esta estrutura permite reconstruir o histórico completo de diálogo, servindo de base à análise e à recontextualização de respostas no módulo **RAG**.

3. **Registos de Operações (operation_logs)**: contém informação detalhada sobre cada operação executada, incluindo tipo, estado, duração e mensagens de erro. É utilizada tanto para auditoria e depuração como para análise estatística do desempenho global. Cada grupo de tabelas está interligado através de chaves estrangeiras,

assegurando integridade referencial e consistência dos dados. Por exemplo, a coluna **user_id** em **conversations** e **operation_logs** referencia a tabela **users**, permitindo associar cada interação e operação a um utilizador autenticado de forma segura. Esta estrutura garante uma visão auditável e coerente das ações do sistema.

4. **Interações de Utilizador (user_interactions)**: armazena ações diretas dos utilizadores, como chamadas de API, submissões de ficheiros ou ativações de ferramentas. Esta tabela fornece contexto comportamental complementar aos registos de operações e é fundamental para compreender padrões de utilização.

5. **Estatísticas e Métricas**: representam a camada de monitorização e análise do sistema. A tabela **operation_logs** regista todas as operações executadas pelo sistema, incluindo o tipo de ação (**rag_query**, **quiz_generation**, **file_upload**), o modelo utilizado, o estado de execução, a duração em milissegundos e eventuais mensagens de erro. Estes registos servem como fonte primária para auditoria, rastreabilidade e análise de desempenho. A tabela **operation_stats** agrega esses dados por tipo e data, sintetizando indicadores de desempenho como número total de execuções, tempos médios, mínimos e máximos de duração, e taxas de erro. Desta forma, é possível identificar padrões de utilização e avaliar a estabilidade das ferramentas integradas. Já a tabela **performance_stats** monitoriza os recursos da infraestrutura, registando métricas como percentagem de CPU, memória total e usada, espaço livre em disco e taxa de utilização. Esta distinção entre métricas operacionais e métricas de sistema permite avaliar tanto a eficiência das operações de alto nível como o comportamento da infraestrutura subjacente.

6. **Variáveis e Contexto de Sistema (system_settings)**: armazena pares **chave valor** utilizados para controlar o estado dinâmico do sistema e a configuração semântica das ferramentas do **MCP**. Nesta tabela são registados os *prompts* associados às diferentes *tools* do servidor, bem como os *prompts* definidos no cliente, permitindo a personalização das interações e a reconfiguração do comportamento dos modelos sem a necessidade de alterar o código. Esta abordagem garante flexibilidade, auditabilidade e consistência na utilização de instruções textuais, assegurando que os modelos de linguagem operam com contextos adaptados às tarefas educativas em curso.

O modelo segue uma filosofia de **minimização de dependências** e **coerência semântica**, permitindo escalar horizontalmente sem comprometer a integridade referencial. O uso de campos **JSONB** assegura a flexibilidade necessária para armazenar dados heterogéneos provenientes de diferentes modelos e versões de **LLM**. A integração entre as tabelas **operation_logs**, **performance_stats** e **conversations** constitui a base para as análises quantitativas realizadas na fase de avaliação.

Esta estrutura fornece o suporte de armazenamento e gestão de operações necessário ao funcionamento do servidor **MCP**, detalhado na secção seguinte. A segurança e confidencialidade dos dados são asseguradas através de mecanismos de controlo de acesso a nível de aplicação, garantindo que apenas utilizadores autenticados e com permissões adequadas possam aceder a dados sensíveis, como históricos de conversação ou métricas operacionais.

3.4 Sistema MCP (Model Context Protocol)

O **MCP** constitui o núcleo de interoperabilidade da arquitetura desenvolvida, operando como camada intermediária entre o LMS Moodle, o módulo de recuperação aumentada (**RAG**) e os modelos de linguagem.

A adoção deste protocolo reflete uma decisão estratégica orientada pela busca de modularidade, padronização e coesão comunicacional entre componentes de inteligência artificial heterogéneos. Desenvolvido pela Anthropic, o MCP baseia-se numa filosofia de interoperabilidade aberta, que facilita a execução estruturada de ferramentas, permitindo a orquestração integrada de agentes, pipelines de dados e sistemas educativos sob uma mesma linguagem técnica.

A sua aplicação no contexto da dissertação reforça a coerência arquitetónica do sistema, proporcionando uma infraestrutura escalável e extensível, apta a evoluir de forma incremental sem comprometer a estabilidade ou a compatibilidade com o ecossistema institucional [92].

3.4.1 Fase de prototipagem e validação

Antes de iniciar a implementação do sistema, foi conduzida uma fase de exploração prática do **MCP**, com o objetivo de compreender o seu modelo de funcionamento interno e o ciclo de comunicação entre cliente e servidor. Esta etapa teve natureza exploratória e visou apenas validar o potencial do protocolo como camada de interoperabilidade para integração futura com o Moodle e com os módulos de inteligência artificial.

Para este efeito, foi utilizado o **SDK oficial do Model Context Protocol para Python**, disponibilizado pela *Anthropic*[1]. O SDK fornece uma implementação de referência denominada `FastMCP`, que permite criar rapidamente um servidor funcional e definir ferramentas.

A Listagem 3.1 apresenta o exemplo `quickstart` incluído no SDK, utilizado nesta fase para compreender o comportamento do protocolo e o registo de ferramentas.

```
1  from mcp.server.fastmcp import FastMCP
2
3  # Creating the MCP server
4  mcp = FastMCP("Demo")
5
6  # Defining a tool
7  @mcp.tool()
8  def add(a: int, b: int) → int:
9      """Add two numbers"""
10     return a + b
```

Listagem 3.1: Protótipo exploratório com o SDK oficial do MCP (FastMCP Quickstart).

A execução deste exemplo permitiu observar a forma como o MCP gere o registo e a invocação de operações. Cada ferramenta (`@mcp.tool()`) define uma função executável que pode ser chamada a partir de um cliente, ilustrando o princípio de modularidade e extensão do protocolo. Esta experiência validou o MCP como base viável para construir a camada de interoperabilidade entre os diferentes componentes educativos do sistema.

[1]Repositório oficial do SDK: `https://github.com/modelcontextprotocol/python-sdk`

3.4.2 Implementação do Servidor MCP

Com a fase exploratória concluída, iniciou-se o desenvolvimento do **servidor MCP**. Este componente representa o núcleo funcional do sistema, sendo responsável por interligar o Moodle, o módulo **RAG**, a base de dados e os modelos de linguagem de forma unificada e extensível.

O servidor foi construído com base na classe `FastMCP` do SDK oficial, expandida com um conjunto de ferramentas registadas através do decorador `@mcp.tool()`. Cada ferramenta implementa uma funcionalidade autónoma e expõe uma interface remota acessível via protocolo MCP, permitindo que clientes externos executem operações no servidor de forma controlada e padronizada. Estas ferramentas cobrem várias áreas funcionais — gestão de modelos, integração com o Moodle, operações de RAG e administração do sistema — e encontram-se documentadas de forma completa no Apêndice A. A Listagem 3.2 apresenta um exemplo real de uma das ferramentas implementadas, *retrieve(prompt: str)*, responsável pela recuperação de informação diretamente a partir da base de conhecimento (documentos indexados). Esta função é invocada sempre que o utilizador formula uma questão relacionada com o conteúdo dos materiais educativos processados pelo módulo **RAG**.

```
1 @mcp.tool()
2 def retrieve(prompt: str) → dict:
3     """Retrieves information from indexed PDFs."""
4     res = qa.invoke({"query": prompt})
5     return {"success": True, "response": res["result"]}
```

Listagem 3.2: Ferramenta MCP simplificada para recuperação de informação no módulo RAG.

Cada ferramenta comunica com o sistema de logging (**PostgresLogger**), que armazena na tabela `operation_logs` a duração da execução, o estado final e as mensagens de erro, quando existentes. Estes dados alimentam as tabelas `operation_stats` e `performance_stats`, utilizadas na análise de desempenho e monitorização da estabilidade do sistema.

O servidor **MCP** funciona, assim, como uma camada de orquestração que abstrai a complexidade técnica dos diferentes componentes — o Moodle, a base vetorial e os modelos de linguagem — disponibilizando um ponto de entrada unificado para o sistema. Cada ferramenta pode ser atualizada ou substituída de forma independente, preservando a modularidade e a escalabilidade da arquitetura.

3.4.3 Comunicação Cliente–Servidor

O cliente do **MCP** é o componente responsável por mediar a comunicação entre a interface Flask, o servidor **MCP** e os modelos de linguagem. A sua principal função é interpretar a intenção do utilizador, construir o contexto conversacional, selecionar a ferramenta apropriada e apresentar uma resposta final formatada.

O método central `process_query()` executa este ciclo completo de interação, combinando a geração linguística do modelo com o controlo lógico do cliente. A Listagem 3.3 mostra uma versão simplificada e anotada da sua implementação.

```python
async def process_query(self, query):
    if not self.is_connected:
        return "Erro: sem ligação ao servidor MCP."

    # 1) Context building
    context = build_conversation_context(self.conversation_history)
    tools = describe_tools(self.tools)
    language_rules = get_language_instructions(self.current_language)

    # 2) Creating the main prompt
    prompt = f"""
    {context}
    Available tools:
    {tools}
    {language_rules}
    IMPORTANT: Always use a tool.
    Current question: {query}
    """

    # 3) First generation. Model chooses TOOL and ARGS.
    text = generate_with_model(self.current_provider, self.current_model, prompt)
    tool_name, tool_args = parse_tool_and_args(text)

    # 4) Remote execution of the tool
    result = await self.session.call_tool(tool_name, tool_args)
    parsed = parse_server_response(result)

    # 5) Second generation. Formatting the final response.
    follow_up_prompt = build_follow_up_prompt(query, parsed)
    final_response = generate_with_model(self.current_provider,
                                         self.current_model,
                                         follow_up_prompt)
    return final_response
```

Listagem 3.3: Processamento simplificado de uma query no cliente MCP.

Este método segue um fluxo de sete fases bem definidas: contextualização, escolha de ferramenta, validação, execução remota, interpretação, formatação e registo da resposta. O cliente garante, assim, uma comunicação bidirecional segura e coerente, mantendo sincronização constante com o servidor e os modelos de linguagem configurados.

A Figura 3.3 representa o ciclo interno de processamento do cliente MCP durante a execução da função `process_query()`. O diagrama mostra como o pedido submetido pelo utilizador na interface Flask é transformado numa sequência estruturada de operações: identificação da ferramenta apropriada, invocação remota no servidor MCP e tratamento da resposta devolvida. Após a execução da ferramenta, ocorre uma etapa automática de *follow-up generativo*, em que o modelo de linguagem sintetiza uma resposta final com base nos resultados recebidos.

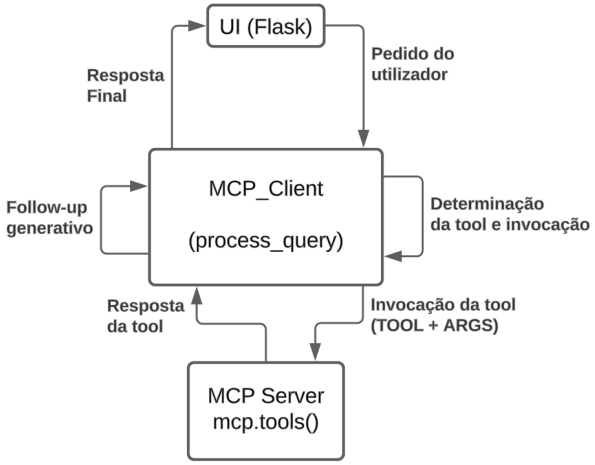


Figura 3.3: Fluxo interno do cliente MCP durante a execução da função `process_query()`.

Este modelo de interação segue a arquitetura *client–host–server* definida no protocolo **MCP** [84]. No sistema desenvolvido, o *host* corresponde à aplicação Flask que gere o contexto da sessão e interage com o utilizador, criando e coordenando instâncias de cliente. O *cliente* **MCP** é responsável por interpretar o pedido do utilizador, selecionar a ferramenta adequada e encaminhar a operação para o servidor correspondente. Por sua vez, o *servidor* **MCP** executa as ferramentas especializadas — como os módulos de **RAG** ou geração de questionários — e devolve os resultados ao cliente para posterior formatação e apresentação. Esta separação clara de responsabilidades garante segurança, modularidade e interoperabilidade, permitindo combinar raciocínio algorítmico e mediação semântica num ciclo de interação controlado e extensível.

3.5 Integração com o Moodle

A integração entre o **MCP** e o **LMS** Moodle constituiu um dos pilares do sistema, garantindo que os modelos de linguagem e os pipelines de RAG pudessem aceder a conteúdos pedagógicos de forma segura e eficiente. Esta integração foi desenhada em três camadas principais: configuração da API Moodle, desenvolvimento de um servidor MCP personalizado e estabelecimento da comunicação cliente–servidor.

3.5.1 Configuração da API Moodle

O Moodle 5.0 disponibiliza uma Application Programming Interface (**API**) baseada em **Web Services**, que permite a comunicação entre aplicações externas de forma controlada e extensível. Esta interface suporta vários protocolos (REST, SOAP, XML-RPC); no presente trabalho foi adotado o protocolo **REST**, devido à sua simplicidade, compatibilidade com Python e suporte nativo ao formato `JSON`. A integração REST é realizada através do ponto de entrada `server.php`, complementado por rotas auxiliares para autenticação e acesso a ficheiros [2].

[2]Documentação Web Services: `https://docs.moodle.org/501/en/Using_web_services`

A configuração dos *webservices* foi realizada integralmente através da interface administrativa do Moodle, seguindo o processo indicado no menu `Site Administration > Server > Web Services`. O processo completo de configuração — incluindo ativação dos protocolos, definição de serviços e geração de tokens — encontra-se documentado no Apêndice B.

Durante a configuração do serviço, foi essencial ativar as opções **"Can download files"** e **"Can upload files"** na secção de permissões do utilizador, garantindo acesso bidirecional a conteúdos. *(Inserir aqui uma imagem ilustrativa destas opções — Figura 3.4).*

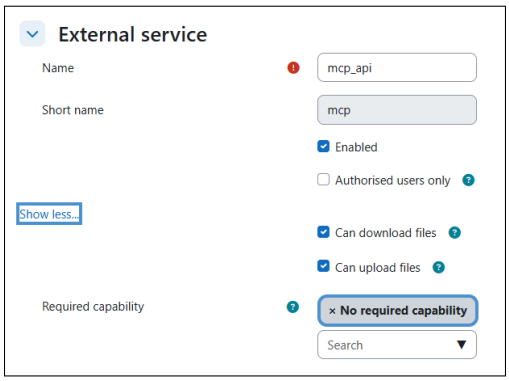


Figura 3.4: Opções de permissões de serviço no Moodle ("Can download/upload files").

Na etapa "Add functions", foram adicionadas as funções necessárias para suportar a integração entre o Moodle e o servidor **MCP**. A Tabela 3.1 resume os *endpoints* REST efetivamente utilizados.

Endpoint / Função REST	**Descrição**	**Utilização no Projeto**
`login/token.php`	Valida credenciais de utilizadores e devolve token de sessão.	Usado apenas para verificação de credenciais dos utilizadores no login inicial.
`core_webservice_get_site_info`	Retorna informações gerais sobre a instância Moodle e o utilizador autenticado.	Verificação de ligação e contexto da instância.
`core_user_get_users`	Pesquisa utilizadores com base em nome, e-mail ou ID.	Identificação e associação de perfis Moodle ao sistema local.
`core_enrol_get_users_courses`	Retorna os cursos em que um utilizador está inscrito.	Mapeamento das inscrições e personalização de conteúdos.
`core_course_get_courses_by_field`	Retorna cursos filtrados por campo (ID, nome, categoria, etc.).	Listagem dinâmica de cursos disponíveis.
`core_course_get_contents`	Obtém todos os módulos e recursos de um curso.	Recolha de conteúdos pedagógicos para o módulo **RAG**.
`pluginfile.php`	Permite o acesso autenticado a ficheiros e recursos binários.	Transferência de materiais (PDFs, slides) para indexação no sistema.

Tabela 3.1: Endpoints REST configurados no serviço Moodle

Após a configuração, o serviço foi testado através do cliente REST interno do Moodle, o qual permite simular chamadas externas de forma segura. Neste passo, foi possível con-

firmar o funcionamento dos endpoints REST e obter o endereço final de serviço, utilizado nas integrações com o servidor **MCP**:

`https://<MOODLE_URL>/webservice/rest/server.php`

O endpoint `login/token.php` foi utilizado exclusivamente para a autenticação de utilizadores no sistema desenvolvido. Durante este processo, a aplicação Flask envia o par `username/password` ao Moodle e, em caso de resposta `200 OK`, o utilizador é considerado autenticado e autorizado a aceder à interface principal. O token devolvido é descartado após a validação, servindo apenas como confirmação de identidade e não sendo utilizado em chamadas subsequentes à API.

```
1 http://<MOODLE_URL>/login/token.php?
2   username=<EMAIL>&
3   password=<PASSWORD>&
4   service=moodle_mobile_app
```

Listagem 3.4: Verificação de credenciais via login/token.php.

Por outro lado, o acesso programático do servidor **MCP** à API Moodle é efetuado através do token permanente gerado no passo 8 ("Create a token for a user"), conforme exemplificado na Listagem 3.5. Este token está associado ao utilizador de serviço criado especificamente para a integração e às funções REST adicionadas ao serviço personalizado. Desta forma, todas as chamadas aos endpoints REST (`server.php`) e de ficheiros (`pluginfile.php`) utilizam esse token, garantindo autenticação segura e persistente sem necessidade de credenciais explícitas.

```
1 http://<MOODLE_URL>/webservice/pluginfile.php/
2   23/mod_resource/content/1/<FILE>.pdf?
3   forcedownload=0&
4   token=<TOKEN>
```

Listagem 3.5: Exemplo real de ficheiro Moodle obtido via pluginfile.php.

A configuração da API REST no Moodle permitiu estabelecer uma base sólida para a integração com o servidor **MCP**. O sistema foi configurado de modo a garantir comunicação autenticada, leitura de dados sobre utilizadores e cursos, e recolha automatizada de ficheiros educativos. Com a API operacional, foi então possível desenvolver o servidor **MCP**, responsável por consumir os endpoints configurados e interligar o Moodle ao módulo de geração de conteúdos inteligentes descrito na secção seguinte.

3.5.2 Integração com o Servidor MCP

A comunicação entre o Moodle e o servidor **MCP** é realizada através das ferramentas registadas no servidor, as quais encapsulam chamadas diretas aos *endpoints* REST configurados. Estas ferramentas funcionam como adaptadores, garantindo a autenticação, a formatação dos parâmetros e o tratamento dos dados devolvidos, permitindo a sua utilização posterior pelo módulo **RAG**.

A Listagem 3.6 apresenta um exemplo de implementação da ferramenta **download_pdfs_from_course()**, responsável por recolher automaticamente todos os ficheiros PDF de um curso Moodle, fazendo uso dos *endpoints* **core_course_get_contents** e **pluginfile.php**.

```python
@mcp.tool()
def download_pdfs_from_course(course_fullname: str) → dict:
    """Retrieves all PDFs associated with a Moodle course."""
    url = f"{MOODLE_URL}/webservice/rest/server.php"
    params = {
        "wstoken": MOODLE_TOKEN,
        "wsfunction": "core_course_get_contents",
        "moodlewsrestformat": "json",
        "courseid": course_fullname
    }

    response = requests.get(url, params=params)
    data = response.json()

    pdf_files = []
    for section in data:
        for module in section.get("modules", []):
            if module.get("modname") == "resource" and module["contents"][0]["
    filename"].endswith(".pdf"):
                file_url = f"{module['contents'][0]['fileurl']}&token={
    MOODLE_TOKEN}"
                pdf_files.append(file_url)

    return {"course": course_fullname, "pdfs": pdf_files}
```

Listagem 3.6: Ferramenta MCP para integração com a API Moodle.

Esta ferramenta ilustra o processo de integração entre o servidor **MCP** e o Moodle. Após receber o nome do curso, o servidor consulta os conteúdos disponíveis através do `core_course_get_contents`, identifica os recursos do tipo PDF e constrói as ligações autenticadas com o token REST configurado. Os ficheiros obtidos são depois transferidos para o pipeline **RAG**, onde são processados e indexados para recuperação semântica.

A Figura 3.5 resume o fluxo de comunicação estabelecido entre os dois sistemas.

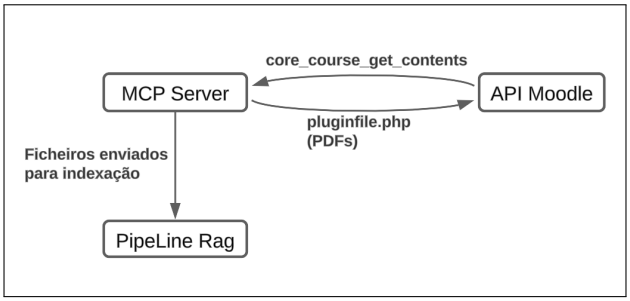


Figura 3.5: Fluxo de comunicação entre o servidor MCP e o Moodle.

Esta integração posiciona o servidor MCP como um middleware inteligente entre o Moodle e a Inteligência Artificial (IA). O processo inicia-se com a recolha estruturada e segura de conteúdos pedagógicos do Moodle, validando a autenticação e as permissões de acesso. Estes dados passam de seguida para o **pipeline RAG** (Recuperação e Geração Aumentada), onde são processados, convertidos em **embeddings semânticos** e inseridos numa **base vetorial** que possibilita a pesquisa e recuperação de informação contextual.

3.6 Sistema RAG para Educação

Após a integração entre o Moodle e o servidor MCP, foi implementado um sistema baseado no paradigma **Retrieval-Augmented Generation (RAG)**, com o objetivo de transformar os conteúdos recolhidos em conhecimento utilizável pelos modelos de linguagem. A adoção deste paradigma no contexto educativo permite gerar respostas fundamentadas e pedagogicamente alinhadas com os materiais institucionais, reduzindo o risco de alucinações e promovendo uma aprendizagem baseada em fontes verificáveis.

3.6.1 Pipeline de Processamento e Pesquisa Semântica

O sistema **RAG** implementado neste trabalho foi concebido para converter os conteúdos educativos provenientes do Moodle em representações vetoriais semanticamente pesquisáveis, permitindo que as respostas geradas pelos modelos de linguagem sejam baseadas em fontes verificáveis. O processo é composto por quatro fases principais: extração, segmentação, vetorização e recuperação.

Na primeira fase, dedicada à extração e pré-processamento, o sistema suporta atualmente apenas ficheiros PDF, formato predominante nos materiais pedagógicos do Moodle. A leitura e extração de texto são realizadas através da **framework LangChain**[3], utilizando o componente `PyPDFLoader`, que interpreta o conteúdo textual e estrutural de cada documento. Para complementar este processo, o módulo `RapidOCRBlobParser` aplica reconhecimento ótico de caracteres (OCR) sobre as imagens embutidas nos PDFs, permitindo capturar texto presente em figuras ou digitalizações. Esta combinação assegura uma extração híbrida e completa, combinando informação textual e visual, e garantindo a integridade dos dados processados.

Após a extração, o texto é dividido em fragmentos menores, designados por *chunks*, através do componente `RecursiveCharacterTextSplitter`. Esta etapa é fundamental para garantir que o conteúdo textual pode ser tratado de forma eficiente e semanticamente coerente. Foram definidos os parâmetros `chunk_size = 800` e `chunk_overlap = 200`, valores que equilibram granularidade e contexto. O tamanho de 800 carateres permite que cada fragmento mantenha coerência semântica suficiente, enquanto a sobreposição de 200 garante continuidade entre secções adjacentes, reduzindo perdas de significado em documentos longos.

Os fragmentos resultantes são depois transformados em vetores numéricos através do modelo `sentence-transformers/all-MiniLM-L6-v2`, disponibilizado pela biblioteca **HuggingFace**. Este modelo gera embeddings de 384 dimensões que representam a proximidade semântica entre conceitos — fragmentos com significado semelhante ficam mais próximos no espaço vetorial. Esta representação vetorial é o elemento central do sistema, permitindo efetuar pesquisas por similaridade e associar perguntas de linguagem natural aos conteúdos mais relevantes.

Os embeddings são armazenados numa base vetorial — **Qdrant** (em modo remoto) ou **Chroma** (em modo local) — que atua como mecanismo de indexação semântica. Durante uma pesquisa, a questão do utilizador é convertida num vetor de consulta (*query vector*) e comparada com os vetores armazenados, utilizando a métrica de distância cosseno. Os fragmentos mais relevantes são recuperados e enviados para o servidor **MCP**, que

[3]Documentação: `https://www.langchain.com/`

os combina com a questão original e fornece o contexto ao modelo de linguagem. Este processo garante que as respostas geradas são factualmente sustentadas pelos documentos originais, reduzindo o risco de alucinações e assegurando consistência pedagógica.

A Listagem 3.7 apresenta o excerto responsável pela leitura, fragmentação e indexação dos documentos PDF no sistema RAG.

```
embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-
    L6-v2")
text_splitter = RecursiveCharacterTextSplitter(chunk_size=800, chunk_overlap=200)

all_texts = []
for pdf_file in Path(PDF_FOLDER).glob("*.pdf"):
    loader = PyPDFLoader(
        file_path=str(pdf_file),
        extract_images=True,
        images_parser=RapidOCRBlobParser(),  # OCR para texto em imagens
    )
    data = loader.load()
    texts = text_splitter.split_documents(data)
    all_texts.extend(texts)

docsearch = Qdrant.from_documents(
    all_texts, embeddings, url=QDRANT_URL, collection_name="moodle_docs"
)
```

Listagem 3.7: Pipeline de processamento e indexação de documentos no módulo RAG.

O fluxo geral do pipeline é apresentado na Figura 3.6. O processo inicia-se com a extração dos PDFs, segue pela segmentação e vetorização, e culmina na recuperação contextual de informação, que alimenta o modelo de geração de linguagem. Este fluxo garante que as respostas produzidas são coerentes, fundamentadas e pedagogicamente alinhadas com os conteúdos originais do Moodle.

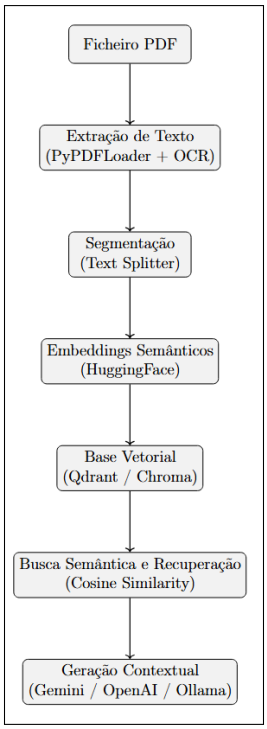


Figura 3.6: Fluxo de comunicação entre o servidor MCP e o Moodle.

A Figura 3.6 representa o fluxo completo do sistema, desde a extração de texto até à geração contextual de respostas. Este pipeline é executado de forma sequencial, adequada a volumes moderados de documentos, garantindo uma conversão fiável dos conteúdos em vetores semanticamente pesquisáveis. Não é aplicada limpeza textual adicional, uma vez que o `PyPDFLoader` realiza normalização básica de caracteres e remoção de metadados estruturais. A métrica de distância cosseno foi selecionada por ser robusta à magnitude dos vetores e refletir apenas a proximidade direcional, o que a torna mais adequada para tarefas semânticas do que a distância euclidiana.

Atualmente, o sistema assume a existência de ficheiros PDF válidos no diretório configurado e executa a indexação de forma integral sempre que são adicionados novos documentos. Futuras versões poderão incorporar um mecanismo de reindexação incremental e paralelização do processamento, otimizando o desempenho em cenários de larga escala.

As consultas semânticas são expostas através da ferramenta, por exemplo, `retrieve()` do servidor **MCP**, que invoca o pipeline RAG sempre que o utilizador submete uma questão. Desta forma, a integração entre o módulo de recuperação e o servidor MCP garante que todas as respostas geradas pelos modelos de linguagem são contextualizadas com base nos conteúdos educativos previamente indexados.

3.6.2 Integração com Qdrant e Chroma

A camada de armazenamento vetorial do sistema foi concebida com uma arquitetura modular, suportando tanto execução local como remota. Esta flexibilidade permite alternar entre duas opções de *backends* vetoriais — **Chroma** (modo local) e **Qdrant** (modo cloud) — garantindo adaptabilidade a diferentes cenários de desenvolvimento e produção, sem necessidade de alterar o código-fonte principal.

A escolha do backend ativo é definida no ficheiro de configuração `.env` do projeto, através da variável `RAG_BACKEND`. Este ficheiro contém também os parâmetros necessários para a ligação à infraestrutura remota, nomeadamente o `QDRANT_URL` e o `QDRANT_API_KEY`. Estas variáveis são lidas no arranque do servidor **MCP**, que inicializa automaticamente o cliente e a pipeline de indexação de acordo com o modo selecionado.

Quando o modo `qdrant` está ativo, é criado um cliente remoto através da biblioteca `qdrant-client`. O sistema verifica a existência da coleção definida em `QDRANT_COLLECTION_NAME` e, se necessário, recria-a com vetores de 384 dimensões e métrica de distância `cosine`, compatível com os embeddings gerados pelo modelo `sentence-transformers/all-MiniLM-L6-v2`. O código seguinte mostra o excerto responsável pela inicialização da ligação e da pipeline de indexação:

```
client = QdrantClient(url=QDRANT_URL, api_key=QDRANT_API_KEY)

if QDRANT_COLLECTION_NAME not in [c.name for c in client.get_collections().
    collections]:
    client.recreate_collection(
        collection_name=QDRANT_COLLECTION_NAME,
        vectors_config=VectorParams(size=384, distance=Distance.COSINE)
    )

# Document processing and indexing pipeline
docsearch = Qdrant(
    client=client,
    collection_name=QDRANT_COLLECTION_NAME,
    embeddings=embeddings
)
```

Listagem 3.8: Inicialização e pipeline de indexação com Qdrant.

Quando o modo `chroma` é selecionado, o sistema utiliza uma instância local persistente armazenada na pasta definida em `CHROMA_DIR`. Esta opção é especialmente útil em ambientes de desenvolvimento ou de ensino, pois não requer autenticação externa nem ligação à Internet. O Chroma é inicializado a partir dos documentos processados, criando o índice vetorial local de forma automática:

```
docsearch = Chroma.from_documents(
    all_texts, embeddings, persist_directory=CHROMA_DIR
)
```

Listagem 3.9: Inicialização local do repositório vetorial Chroma.

Ambos os sistemas partilham a mesma interface de pesquisa e recuperação, o que permite alternar entre backends apenas alterando o valor da variável `RAG_BACKEND`. Esta abordagem assegura que o sistema mantém compatibilidade funcional e resultados consistentes independentemente da infraestrutura utilizada.

Do ponto de vista técnico, a escolha entre uma arquitetura local e uma baseada em nuvem envolve compromissos significativos entre **privacidade** e **escalabilidade**. Evidência recente na literatura confirma que sistemas RAG locais oferecem maior segurança e proteção de dados institucionais, evitando a transmissão de informação sensível para servidores externos [93]. Por outro lado, soluções em nuvem apresentam melhor desempenho e escalabilidade, mas introduzem riscos acrescidos de exposição e custos computacionais elevados

[94; 95]. Estudos mais recentes indicam que abordagens híbridas — combinando processamento local e edge-cloud — conseguem equilibrar desempenho e privacidade, reduzindo atrasos e consumo de recursos sem perda de precisão [96; 95].

A Figura 3.7 ilustra o fluxo de integração entre o sistema RAG e as bases vetoriais Qdrant e Chroma, evidenciando a comutação dinâmica entre modos e o processo unificado de indexação e pesquisa.

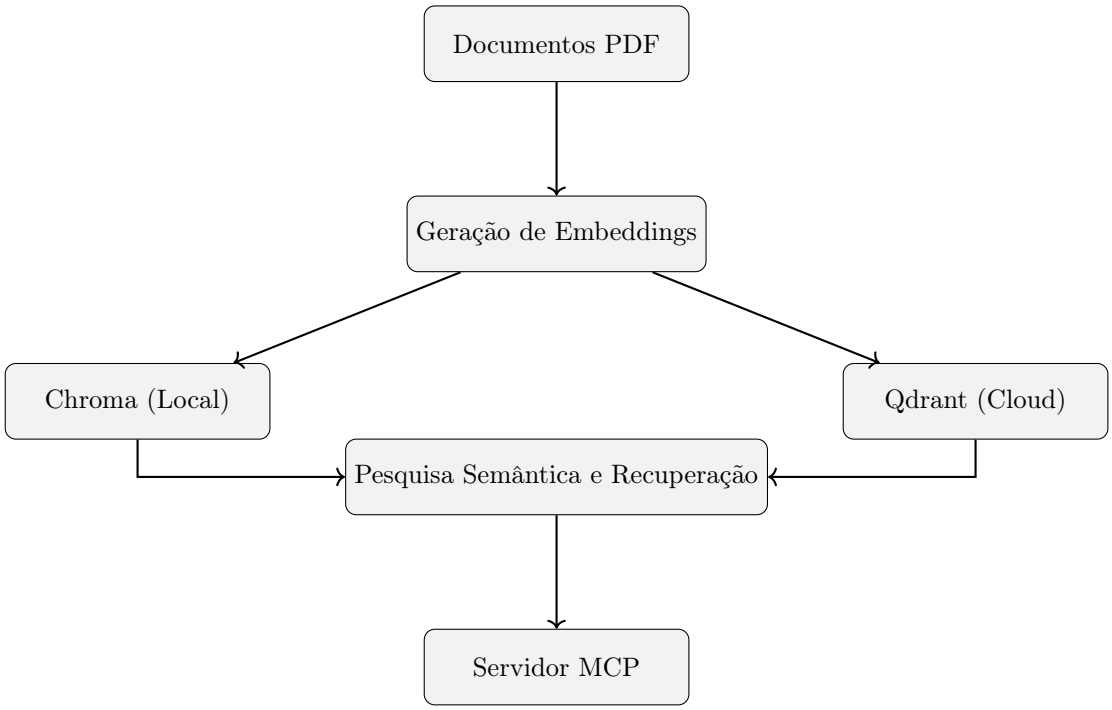


Figura 3.7: Integração modular do sistema RAG com Qdrant (cloud) e Chroma (local).

Durante o desenvolvimento, o **Chroma** foi utilizado como solução de armazenamento local, permitindo uma execução rápida, leve e totalmente offline, adequada a ambientes de teste e contextos institucionais com restrições de privacidade. Em produção, o **Qdrant** oferece maior desempenho e escalabilidade, tirando partido da infraestrutura em nuvem para gerir um volume superior de embeddings e consultas simultâneas. Ambos os backends expõem a mesma interface de pesquisa (`.as_retriever()`), o que permite alternar entre modos sem impactar o funcionamento das ferramentas do servidor MCP.

A inclusão de ambos os modos não resulta apenas de uma decisão técnica, mas de uma opção arquitetónica que visa acomodar diferentes preferências institucionais e requisitos operacionais. O sistema pode assim ser executado localmente, privilegiando a privacidade e o controlo de dados, ou remotamente, maximizando o desempenho e a escalabilidade. Atualmente, a indexação é executada integralmente sempre que há novos documentos; versões futuras poderão incorporar estratégias de atualização incremental e sincronização distribuída entre instâncias locais e remotas.

3.7 Integração com Modelos Gemini, OpenAI e Ollama

A camada de inteligência artificial do sistema baseia-se na integração com diferentes fornecedores de modelos de linguagem de grande escala (**LLM**), permitindo flexibilidade, redundância e adequação a distintos contextos de utilização. Foram integrados três motores principais: **Gemini** (Google Generative AI), **OpenAI** (GPT-4 e variantes) e **Ollama** (instância local compatível com Llama 3 e Mistral). Esta diversidade de modelos permite equilibrar desempenho, custo e disponibilidade, assegurando compatibilidade tanto com ambientes em nuvem como com infraestruturas locais.

3.7.1 Seleção e Configuração de Modelos

A integração de múltiplos modelos de linguagem no sistema visa garantir flexibilidade, acessibilidade e sustentabilidade de custos. Em vez de depender de um único fornecedor, a arquitetura suporta três **LLM** distintos — **Gemini** (Google Generative AI), **OpenAI** (GPT-4 Turbo) e **Ollama** (Llama 3 local) — permitindo ajustar a operação consoante as necessidades pedagógicas, a disponibilidade de recursos e o nível de privacidade requerido.

Esta abordagem multiparadigma assegura que o sistema pode ser executado tanto em ambientes *cloud* (Gemini, OpenAI) como de forma totalmente local (Ollama), sem comprometer a experiência do utilizador. Assim, docentes e instituições podem optar por modelos de baixo custo (como o Gemini), por maior qualidade linguística (OpenAI), ou por execução privada e offline (Ollama), equilibrando **custo**, **desempenho** e **soberania de dados**.

Todos os modelos disponíveis encontram-se definidos no ficheiro de configuração **models.json**, que funciona como catálogo central de metadados. Cada entrada contém informações sobre o `provider`, nome legível, descrição funcional, limite de `tokens` e um índice relativo de custo (`cost_rank`). Um excerto simplificado do ficheiro é apresentado na Listagem 3.10.

```
{
  "gemini-2.5-flash-lite": {
    "provider": "gemini",
    "name": "Gemini 2.5 Flash Lite",
    "description": "An optimized and up-to-date version of Flash Lite. Faster and
     more stable, while maintaining a low cost",
    "max_tokens": 4096,
    "cost_rank": 3
  }
}
```

Listagem 3.10: Excerto simplificado do ficheiro de configuração `models.json`

3.7.2 Seleção e Configuração de Modelos

O sistema suporta três famílias de modelos de linguagem de grande escala (**LLM**): **Gemini** (Google Generative AI), **OpenAI** (GPT) e **Ollama** (execução local). Esta diversidade visa assegurar flexibilidade, permitindo alternar entre execução em nuvem e local conforme os requisitos de privacidade, custo e desempenho. A lista de modelos é mantida no ficheiro `models.json`, que define o `provider`, `name`, `description`, `max_tokens` e `cost_rank` de cada modelo, garantindo uma estrutura unificada de configuração.

A seleção do modelo ativo é realizada exclusivamente através da **interface web** da aplicação, concretamente na secção `Settings`. Esta funcionalidade encontra-se restrita a utilizadores com perfil de administrador, de forma a garantir o controlo e a consistência das configurações do sistema. Quando o utilizador seleciona um novo modelo, a interface cliente invoca o método `set_model()`, responsável por atualizar as variáveis locais `current_model` e `current_provider`. Posteriormente, esta ação desencadeia a execução da ferramenta `set_model()` no servidor, a qual sincroniza a alteração e procede à reconstrução interna do objeto `RetrievalQA`. Desta forma, assegura-se que o pipeline de **RAG** utiliza o mesmo **LLM** configurado na interface, mantendo a coerência entre o lado cliente e o lado servidor. A Listagem 3.11 apresenta uma versão simplificada do código correspondente à função no lado do cliente.

```
def set_model(self, model_name):
    """Define the model to be used (Gemini, OpenAI, or Ollama) and synchronize it with the server."""
    if model_name in ALL_MODELS:
        self.current_model = model_name
        self.current_provider = ALL_MODELS[model_name]["provider"]
        print(f"Modelo definido: {model_name} (provider={self.current_provider})")

        if self.is_connected:
            try:
                # Synchronize model with the MCP server.
                result = run_async(
                    self.session.call_tool("set_model", {"model_name": model_name})
                )
                print(f" Modelo sincronizado com servidor: {model_name}")
            except Exception as e:
                print(f"Erro ao sincronizar modelo: {e}")
        return True
    return False
```

Listagem 3.11: Definição do modelo ativo no cliente MCP.

A implementação do lado do servidor, responsável por atualizar o modelo e reconstruir o pipeline `RetrievalQA`, encontra-se apresentado no Apêndice C.

O parâmetro de geração mais relevante na configuração dos modelos é a **temperature**, que controla o grau de variabilidade nas respostas geradas. Valores baixos (`0.1-0.4`) produzem respostas mais determinísticas e consistentes, enquanto valores altos (`>0.7`) promovem criatividade e diversidade lexical. Considerando que o objetivo do sistema é fornecer respostas pedagógicas e factualmente fundamentadas, foi definido `temperature = 0.4` em todos os modelos, assegurando equilíbrio entre coerência, fluência e precisão. Esta configuração reduz o risco de alucinações e mantém estabilidade entre execuções idênticas, característica essencial para aplicações educativas e avaliações automáticas.

O excerto seguinte ilustra a parametrização do modelo Gemini no servidor **MCP**:

```
model = GoogleGenerativeAI(model=new_model_name, temperature=0.4)
```

Listagem 3.12: Configuração do modelo Gemini com temperature controlada.

Todos os modelos partilham a mesma interface de chamada, permitindo alternância transparente entre fornecedores. No caso do Ollama, o sistema comunica via API local

(`http://localhost:11434/api/chat`), garantindo operação totalmente offline e preservação da privacidade institucional. Assim, a arquitetura suporta três níveis de execução — local, híbrida e cloud — mantendo a coerência da experiência de utilização e a compatibilidade com o pipeline **RAG** descrito anteriormente.

Este mecanismo assegura que tanto o cliente como o servidor utilizam sempre o mesmo modelo e fornecedor, mantendo coerência semântica nas respostas e estabilidade operacional. A atualização é refletida em tempo real, sem necessidade de reiniciar o servidor ou recompilar o código, o que facilita testes comparativos entre modelos e otimizações pedagógicas. A coexistência de diferentes **LLM** no mesmo sistema constitui, assim, uma componente estratégica de flexibilidade e adaptabilidade, permitindo explorar diferentes equilíbrios entre custo, precisão e privacidade em contextos educativos e de investigação.

3.7.3 Prompts Especializados para Educação

Os *prompts* constituem o principal mecanismo de controlo do comportamento dos modelos de linguagem no sistema, orientando a geração de respostas adequadas ao contexto educativo. A qualidade e estrutura destes comandos têm impacto direto na clareza, pertinência e consistência pedagógica das respostas produzidas. De acordo com [97], um *prompt* eficaz deve incluir três componentes essenciais — a descrição da tarefa, o fornecimento de informação contextual e a definição do formato esperado de saída —, assegurando clareza, precisão e reprodutibilidade. No presente sistema, estas diretrizes são operacionalizadas através de uma estrutura tripartida: (i) uma **instrução principal**, que define o objetivo da tarefa (ex.: gerar um resumo, criar um questionário, explicar um conceito); (ii) um **bloco de contexto**, que integra dados do utilizador e excertos relevantes do Moodle; e (iii) um **modelo de resposta**, que especifica o formato esperado (texto explicativo, lista, tabela, etc.). Esta organização segue as boas práticas de [97], garantindo uniformidade, transparência e consistência pedagógica nas interações com os modelos de linguagem.

No sistema desenvolvido, os *prompts* são armazenados na tabela `system_settings`, permitindo que possam ser atualizados dinamicamente através da interface de administração. Existem dois tipos principais de *prompts*: (i) os utilizados pelo **MCP Client**, responsáveis por estruturar a interação entre o utilizador e o modelo; (ii) os utilizados pelas ferramentas do **MCP Server**, que orientam o comportamento do pipeline **RAG** durante a recuperação e geração de respostas.

O exemplo da Listagem 3.13 mostra um dos *prompts* centrais utilizados pelo cliente, responsável por estruturar as instruções enviadas ao modelo antes da primeira geração de texto. O *prompt* inclui exemplos de uso de ferramentas, regras linguísticas (responder sempre em português de Portugal) e formato esperado da resposta, assegurando consistência e controlo interpretativo.

```
1 prompt = f"""
2 Context:
3 You are an educational assistant integrated into a learning environment.
4 Your task is to select and invoke the most appropriate tool from the available
      options to fulfil the user's request efficiently and accurately.
5
6 Available tools:
7 {tool_descriptions}
8
9 Guidelines:
```

```
1. Always respond in European Portuguese.
2. Use precise academic and pedagogical terminology appropriate for higher
    education.
3. Never answer directly, you must always invoke a tool.
4. Follow the exact output format below.

Required format:
TOOL: <tool_name>
ARGS: <json_with_arguments>

Examples:
1. TOOL: retrieve
   ARGS: {{"prompt": "Explain the concept of Challenge-Based Learning."}}

2. TOOL: generate_quiz_with_difficulty
   ARGS: {{"topic": "Artificial Intelligence", "num_questions": 5, "difficulty":
    "medium"}}

Quality criteria:
- Ensure the selected tool matches the user's intent.
- If the request is ambiguous, ask for clarification before proceeding.
- Prioritise conceptual accuracy, clarity, and linguistic correctness.
"""
```

Listagem 3.13: Revised base prompt used in the MCP Client.

Através deste esquema, o modelo é instruído a agir como intermediário inteligente, interpretando o pedido do utilizador e escolhendo a ferramenta apropriada, em vez de responder de forma direta. Esta abordagem garante rastreabilidade, consistência terminológica e alinhamento com o propósito educativo da plataforma.

3.8 Interface web e experiência do utilizador

3.8.1 Desenvolvimento do MCP Client (CLI → Flask)

O **MCP Client** constitui a camada de interação entre o utilizador e o servidor **MCP**, sendo responsável por enviar as instruções do utilizador, interpretar as respostas e apresentar os resultados de forma estruturada. O desenvolvimento deste componente seguiu uma evolução incremental em duas fases: uma versão inicial em linha de comandos (CLI) e uma versão final baseada em interface web construída com **Flask**.

A versão inicial, desenvolvida em Python puro, teve como principal objetivo validar a comunicação com o servidor MCP e testar o comportamento do método central **process_query()**. Nesta fase, o cliente operava em modo de consola, permitindo ao utilizador inserir perguntas diretamente no terminal e visualizar as respostas geradas pelos modelos de linguagem. Apesar de funcional, esta abordagem apresentava limitações significativas em termos de usabilidade e gestão de contexto, nomeadamente a ausência de histórico, autenticação e controlo de sessões.

Com base nesses constrangimentos, procedeu-se à migração para uma arquitetura web suportada por **Flask**, que passou a oferecer uma interface gráfica moderna e responsiva. Esta nova camada manteve o núcleo de comunicação do cliente MCP — incluindo as funções de ligação, envio de queries e processamento de respostas — mas encapsulou-o

em rotas HTTP e componentes visuais que facilitam a interação. A comunicação entre o navegador e o cliente é realizada através de pedidos `fetch()` assíncronos em JavaScript, garantindo uma experiência fluida sem necessidade de recarregar a página.

A migração para Flask foi motivada por três objetivos principais:

1. **Melhorar a experiência do utilizador:** a nova interface apresenta conversas em formato de chat, com histórico persistente e visualização estruturada de respostas, proporcionando uma interação mais natural com os modelos de linguagem.
2. **Permitir a gestão de perfis e permissões:** o sistema passou a suportar diferentes papéis de utilizador (*Aluno*, *Professor* e *Administrador*), com acesso diferenciado às ferramentas MCP e às funcionalidades administrativas.
3. **Integrar monitorização e estatísticas:** através de dashboards integrados, a interface permite visualizar logs de utilização, tempo médio de resposta e número de operações executadas, promovendo transparência e controlo operacional.

Esta evolução consolidou o **MCP Client** como um ponto de interação acessível e escalável, combinando a flexibilidade técnica do protocolo MCP com uma experiência de utilização moderna e intuitiva. A integração total entre cliente e servidor garante que as operações executadas — desde a submissão de queries até à geração de respostas — são tratadas de forma síncrona e segura, independentemente do tipo de utilizador ou do modelo de linguagem em execução.

3.8.2 Design e Acessibilidade

A interface foi desenvolvida em **Flask**, com design responsivo e suporte a internacionalização em Português e Inglês, permitindo alternância entre *light mode* e *dark mode*. O desenho da aplicação foi inspirado na experiência conversacional de plataformas modernas como o **ChatGPT**, privilegiando a simplicidade visual, o foco no diálogo e a redução de elementos distrativos. A autenticação é realizada através das credenciais do Moodle, assegurando integração direta com a infraestrutura institucional. Durante o processo de login, o sistema verifica se é a primeira vez que o utilizador se autentica e, com base no endereço de e-mail institucional, determina automaticamente o seu perfil — **Professor** ou **Aluno**. Esta diferenciação permite ajustar o acesso a determinadas funcionalidades, garantindo coerência entre as permissões pedagógicas e o papel do utilizador no sistema.

Após a autenticação, o utilizador é redirecionado para a interface principal (Figura 3.8), que apresenta um ambiente de chat interativo com histórico persistente de conversas, um **botão para estabelecer a ligação ao servidor MCP** (com indicação do estado da conexão) e uma barra lateral de navegação para gestão de sessões. No canto esquedo, dentro da caixa de texto, o círculo azul com o ícone de "+" permite adicionar novos documentos ao módulo **RAG**, processo restrito a perfis de **Professor** e **Administrador**. Esta funcionalidade possibilita o carregamento de conteúdos pedagógicos (em formato PDF) para posterior indexação e recuperação semântica pelo sistema, ampliando a base de conhecimento acessível durante o diálogo.

A página principal também disponibiliza a seleção de idioma da plataforma, permitindo alternar entre **Português** e **Inglês**. Esta opção visa facilitar a utilização por estudantes internacionais, nomeadamente os que participam em programas de mobilidade académica como o *Erasmus+*, garantindo inclusão linguística e acessibilidade. O sistema suporta tanto o modo *dark* como *light*, oferecendo uma experiência personalizada, confortável e adaptada a diferentes preferências visuais.

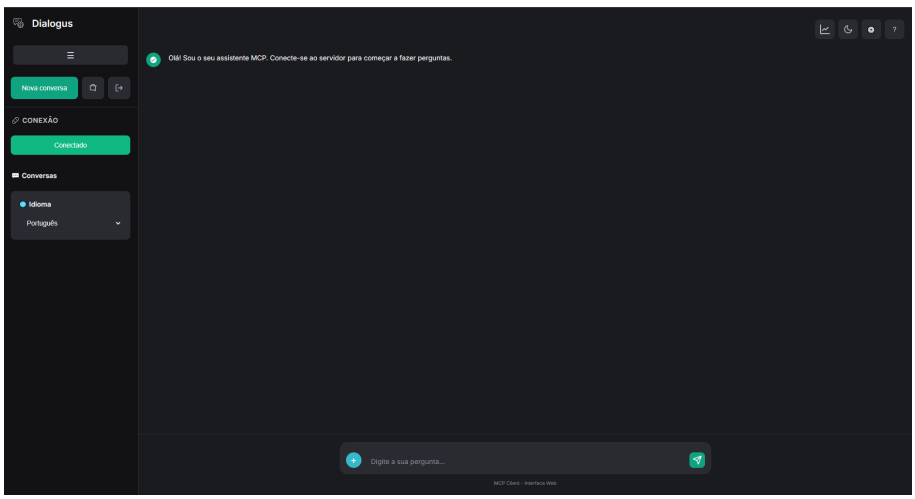


Figura 3.8: Interface principal do sistema em modo escuro.

A versão em modo claro encontra-se incluída no Apêndice D.

A página de configurações, acessível apenas ao perfil de **Administrador**, está representada na Figura 3.9. Nesta secção é possível alterar o modelo de linguagem ativo (*Gemini*, *OpenAI* ou *Ollama*), o backend RAG (*Chroma* ou *Qdrant*) e editar diretamente os *prompts*

utilizados pelo sistema. As alterações são aplicadas em tempo real e registadas na base de dados `system_settings`, permitindo rastreabilidade e auditoria de configurações.

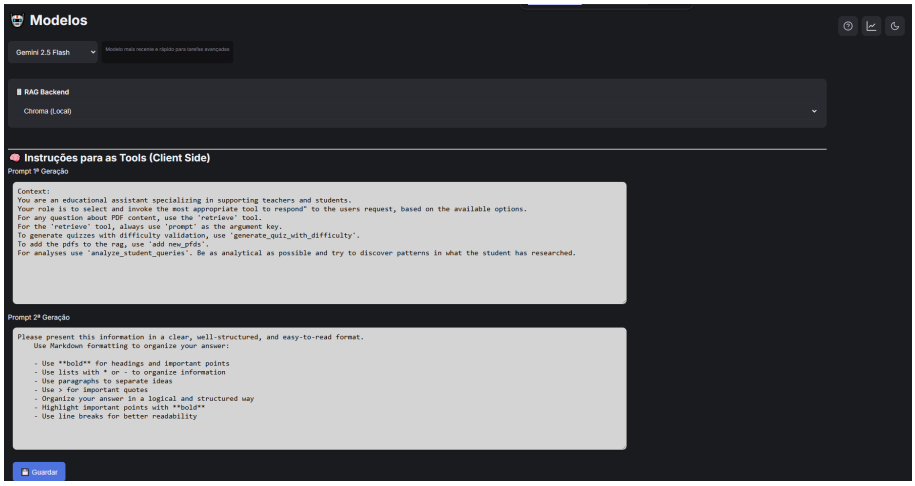


Figura 3.9: Página de configurações administrativas (*Settings*).

Por fim, o **Dashboard de Estatísticas** (Figura 3.10) apresenta métricas de desempenho do sistema, incluindo número de operações executadas, taxa de sucesso e falhas, uso de disco, memória e distribuição de tipos de operação. Este painel foi desenvolvido para facilitar a análise de eficiência e estabilidade da aplicação em contexto real.

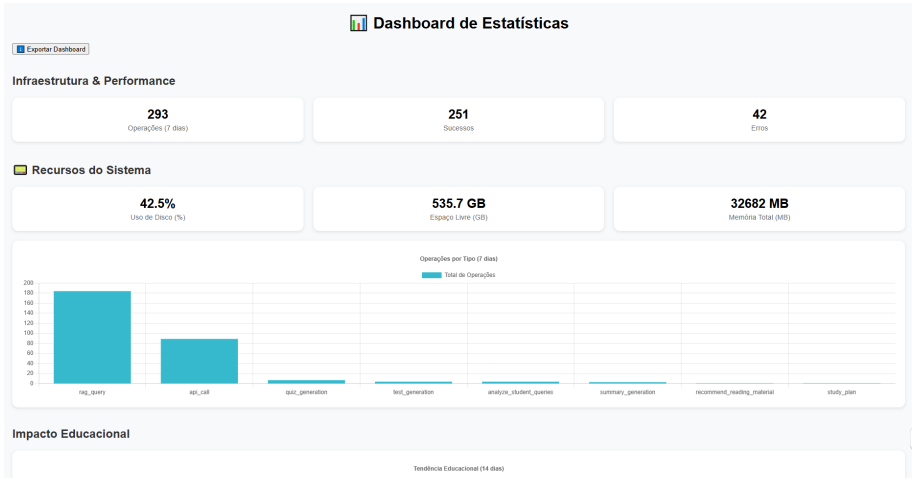


Figura 3.10: Painel de estatísticas e monitorização do sistema.

A interface adota uma estrutura centrada no utilizador, com tipografia clara, contraste adequado e disposição adaptável a diferentes tamanhos de ecrã. O uso consistente de ícones, cores neutras e separação visual entre blocos de mensagem contribui para uma experiência acessível e intuitiva. As opções de tema claro e escuro foram concebidas não apenas por motivos estéticos, mas também para promover a **acessibilidade visual** em contextos de iluminação variados e reduzir a fadiga ocular durante sessões prolongadas de utilização.

Em conjunto, estas decisões de design asseguram uma coerência visual e funcional em

toda a plataforma, garantindo que o utilizador — seja aluno, professor ou administrador — interage com o sistema de forma fluida, inclusiva e consistente.

3.8.3 Funcionalidades Principais

A interface web desenvolvida em Flask integra todas as funcionalidades essenciais do sistema, oferecendo uma experiência interativa, responsiva e adaptada a diferentes perfis de utilizador. O sistema suporta múltiplas conversas, histórico de interações, integração com ferramentas MCP (quiz, resumo, flashcards), gestão de permissões por função e visualização de estatísticas através de *dashboards* dedicados.

As principais funcionalidades podem ser sintetizadas da seguinte forma:

1. **Autenticação via Moodle** — o acesso é realizado com as credenciais institucionais do utilizador, garantindo validação e sincronização de perfis.
2. **Gestão de conversas** — o utilizador pode iniciar novas sessões, consultar o histórico e interagir em tempo real com os modelos de linguagem (Figura 3.8).
3. **Integração com ferramentas MCP** — acesso direto a funcionalidades pedagógicas, como `practice_quiz()`, `generate_summary()` e `flashcards()`, adaptadas ao papel do utilizador.
4. **Controlo de permissões** — a aplicação distingue os papéis de **Aluno**, **Professor** e **Administrador**, limitando o acesso a funcionalidades sensíveis e configurando automaticamente as ferramentas disponíveis.
5. **Configuração administrativa** — disponível apenas para o Administrador, permite alterar o modelo de linguagem, o backend RAG e os *prompts* de sistema, em tempo real (Figura 3.9).
6. **Dashboard de estatísticas** — disponível apenas para o Administrador, apresenta métricas sobre o desempenho do sistema, número de operações executadas, taxa de sucesso, consumo de recursos e impacto educativo (Figura 3.10).

Gestão de Permissões e Configuração Administrativa

O sistema implementa um modelo hierárquico de permissões composto por três perfis principais:

- **Aluno:** pode colocar questões, consultar conteúdos e utilizar ferramentas básicas de apoio ao estudo (RAG, resumo, flashcards).
- **Professor:** além das funções do aluno, pode carregar novos ficheiros, gerar questionários e consultar estatísticas de utilização.
- **Administrador:** tem acesso total, podendo alterar configurações globais, gerir utilizadores e editar os *prompts* base das ferramentas MCP.

Esta lógica de controlo é aplicada automaticamente durante a autenticação, utilizando a variável de sessão `session["role"]`, e é refletida na interface. Por exemplo, ferramentas como `clear_rag()` ou `export_logs()` só são apresentadas no painel do administrador, enquanto funções pedagógicas, como `download_pdfs_from_course()` ou `retrieve()`, permanecem acessíveis a alunos e professores.

A Tabela 3.2 sintetiza esta correspondência entre os perfis de utilizador e as ferramentas MCP disponíveis, destacando de forma clara as permissões atribuídas a cada nível de acesso.

Ferramenta MCP	**Admin**	**Professor**	**Aluno**
`set_model`	X		
`get_current_model`	X		
`set_rag_backend`	X		
`get_rag_backend`	X		
`add_new_pdfs`	X	X	
`download_and_add_pdf`	X		
`download_pdfs_from_course`	X	X	
`clear_rag`	X		
`assign_role`	X		
`list_users`	X		
`deactivate_user`	X		
`practice_quiz`	X		X
`recommend_reading_material`	X	X	X
`analyze_student_queries`	X	X	
`retrieve`	X	X	X
`generate_quiz_with_difficulty`	X	X	
`generate_dev_questions`	X	X	X
`study_plan_generator`	X	X	X
`generate_lesson_summary`	X	X	X
`interactive_flashcards`	X	X	X
`generate_test`	X	X	X
`export_logs`	X		
`system_health_check`	X		

Tabela 3.2: Distribuição das ferramentas MCP por perfil de utilizador.

Esta estrutura de permissões garante que cada perfil de utilizador acede apenas às ferramentas compatíveis com o seu papel pedagógico ou administrativo, preservando a segurança e a coerência funcional do sistema. A lógica de acesso foi desenhada para refletir as necessidades reais de um ambiente educativo digital, equilibrando autonomia e controlo técnico. Com base nesta hierarquia, a interface disponibiliza opções configuráveis específicas para cada nível, sendo a gestão global reservada ao perfil de Administrador.

A Figura 3.9 mostra a página dedicada à configuração administrativa. Através desta secção, o administrador pode:

- Selecionar o **modelo de linguagem** ativo (*Gemini*, *OpenAI* ou *Ollama*);
- Definir o **backend RAG** em uso (*Qdrant* ou *Chroma*);
- Editar os **prompts base** utilizados pelas ferramentas MCP e pelos clientes conectados.

Os valores são guardados na tabela `system_settings`, que armazena exclusivamente os prompts de sistema e de utilizador. Esta abordagem permite ajustar dinamicamente

o comportamento das ferramentas sem necessidade de reiniciar o servidor, mantendo a consistência entre cliente e servidor.

Em conjunto, o mecanismo de controlo de permissões, a parametrização de modelos e o painel de estatísticas conferem à aplicação uma elevada flexibilidade e transparência operacional, tornando a interface adequada tanto para contextos de ensino individual como institucional.

Esta estrutura de permissões, aliada à interface administrativa e ao painel de controlo, assegura uma gestão equilibrada entre autonomia pedagógica e supervisão técnica. Desta forma, o sistema **MCP** mantém coerência funcional, segurança operacional e flexibilidade para acomodar diferentes perfis de utilizador no ecossistema educativo.

3.9 Persistência e Otimização

A camada de persistência e otimização foi projetada para assegurar a integridade dos dados, a escalabilidade do sistema e a manutenção contínua do desempenho. Nesta fase, foram implementadas soluções de armazenamento, containerização e monitorização que garantem a reprodutibilidade dos resultados e a fiabilidade do sistema em ambiente de produção.

3.9.1 Integração PostgreSQL e Containerização Docker

A persistência de dados no sistema foi implementada utilizando uma instância **PostgreSQL 15**, executada em ambiente **Docker**. Esta decisão assegura isolamento, reprodutibilidade e facilidade de implantação, permitindo que o ambiente de base de dados possa ser configurado e iniciado automaticamente em qualquer máquina, sem dependências externas.

O serviço de base de dados é definido no ficheiro `docker-compose.yml`, apresentado na Listagem 3.14. O contêiner `postgres` armazena os dados de forma persistente através de um volume dedicado (`postgres_data`), e inclui um ficheiro `init.sql` responsável pela criação automática das tabelas essenciais à primeira execução. Adicionalmente, foi incluído o serviço `pgAdmin`, que fornece uma interface web de gestão e monitorização da base de dados.

```
version: '3.8'
services:
  postgres:
    image: postgres:15-alpine
    container_name: rag_postgres
    restart: unless-stopped
    environment:
      POSTGRES_DB: rag_system
      POSTGRES_USER: rag_user
      POSTGRES_PASSWORD: rag_password_secure_2024
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql
      - ./backups:/backups
    healthcheck:
```

```
      test: ["CMD-SHELL", "pg_isready -U rag_user -d rag_system"]
      interval: 30s
      timeout: 10s
      retries: 3
    networks:
      - rag_network

  pgadmin:
    image: dpage/pgadmin4:latest
    container_name: rag_pgadmin
    restart: unless-stopped
    environment:
      PGADMIN_DEFAULT_EMAIL: admin@example.com
      PGADMIN_DEFAULT_PASSWORD: admin_password_2024
    ports:
      - "8080:80"
    depends_on:
      postgres:
        condition: service_healthy
    networks:
      - rag_network

volumes:
  postgres_data:
  pgadmin_data:

networks:
  rag_network:
    driver: bridge
```

Listagem 3.14: Configuração do serviço PostgreSQL e pgAdmin via Docker Compose.

A integração da aplicação com o PostgreSQL é realizada através de uma camada de abstração implementada no módulo `postgres_logger.py`. Este componente é responsável por registar todas as operações executadas no sistema — incluindo consultas, geração de respostas e erros —, assegurando rastreabilidade e auditoria completa das interações. Os parâmetros de ligação à base de dados são definidos através de variáveis de ambiente no ficheiro `.env`, garantindo segurança e flexibilidade de configuração.

```
db_config = {
    "host": os.getenv("PG_HOST", "localhost"),
    "port": int(os.getenv("PG_PORT", "5432")),
    "database": os.getenv("PG_DATABASE", "rag_system"),
    "user": os.getenv("PG_USER", "rag_user"),
    "password": os.getenv("PG_PASSWORD", "rag_password_secure_2024"),
}
postgres_logger = PostgresLogger(db_config)
```

Listagem 3.15: Inicialização do PostgreSQL Logger no cliente MCP

Cada operação é registada na tabela `operation_logs`, contendo informações como o tipo de operação, o utilizador, o estado (sucesso/erro), a duração e detalhes técnicos. A função `log_operation()` permite inserir estes registos de forma estruturada, recorrendo à serialização em `JSON`. Um excerto simplificado do método é apresentado na Listagem 3.16.

```
def log_operation(self, operation_type, user_id=None, details=None,
                  status="success", error_message=None, duration_ms=None):
```

```
    try:
        with self.get_connection() as conn:
            with conn.cursor() as cursor:
                cursor.execute("""
                    INSERT INTO operation_logs (
                        id, user_id, operation_type, operation_details,
                        status, error_message, duration_ms, created_at
                    ) VALUES (%s, %s, %s, %s, %s, %s, %s, NOW())
                """, (
                    str(uuid.uuid4()), user_id, operation_type,
                    Json(details), status, error_message, duration_ms
                ))
                conn.commit()
                return True
    except Exception as e:
        self.file_logger.error(f"Erro ao registar operação: {e}")
        return False
```

Listagem 3.16: Função simplificada de registo de operações no PostgreSQL

Este mecanismo é utilizado em múltiplos pontos do sistema, permitindo medir o tempo de execução de ferramentas, contabilizar erros e gerar estatísticas de desempenho. Os dados registados alimentam as tabelas `operation_stats` e `performance_stats`, analisadas na secção 3.9.2, que suportam o módulo de monitorização e otimização do sistema.

A adoção combinada de **PostgreSQL** e **Docker** proporcionou um ambiente de desenvolvimento e produção altamente portável, reprodutível e resiliente. Esta abordagem garante que a base de dados é inicializada automaticamente com as tabelas necessárias, enquanto os contêineres podem ser reiniciados ou migrados sem perda de dados, reforçando a confiabilidade e manutenção contínua do sistema.

3.9.2 Monitorização, Métricas e Refinamento do Sistema

A monitorização do sistema foi concebida para fornecer uma visão abrangente sobre o desempenho operacional, a fiabilidade das ferramentas e o comportamento global dos módulos integrados (Moodle, MCP, RAG e LLMs). Esta camada de controlo combina dois mecanismos complementares: **registos locais em ficheiros de log** e **métricas persistentes na base de dados PostgreSQL**.

Sistema de Logs Local

O sistema de registo local foi implementado através do módulo `logging` da biblioteca padrão do Python, configurado no ficheiro `MCP_Server_new.py`. O método `setup_logging()` cria automaticamente o diretório `logs/` e gera ficheiros diários com o formato `rag_server_<data>.log`. O formato de saída inclui o carimbo temporal, o nível de severidade, o nome do módulo e a mensagem, conforme exemplificado na Listagem 3.17.

```
def setup_logging():
    """Configure the log system."""
    log_dir = Path("logs")
    log_dir.mkdir(exist_ok=True)

    formatter = logging.Formatter(
        '%(asctime)s - %(name)s - %(levelname)s - %(message)s'
```

```
    )

    file_handler = logging.FileHandler(
        f"logs/rag_server_{datetime.now().strftime('%Y%m%d')}.log"
    )
    file_handler.setFormatter(formatter)

    console_handler = logging.StreamHandler()
    console_handler.setFormatter(formatter)

    logger = logging.getLogger('RAG_Server')
    logger.setLevel(logging.INFO)
    logger.addHandler(file_handler)
    logger.addHandler(console_handler)

    return logger
```

Listagem 3.17: Configuração do sistema de logs no servidor MCP.

Cada evento significativo do sistema — como a geração de questionários, o carregamento de PDFs, a execução de ferramentas ou a deteção de erros — é registado através de diferentes níveis de severidade (`INFO`, `WARNING`, `ERROR`). Exemplos de registos emitidos durante a execução incluem:

```
logger.info(f"QUIZ_GENERATION | Topic: {topic} | Questions: {num_questions} |
    Success")
logger.warning(f"QUIZ_GENERATION | Topic: {topic} | No content found | Duration:
    {duration:.2f}s")
logger.error(f"PDF_DOWNLOAD | URL: {url} | Error: {e} | Duration: {duration:.2f}s
    ")
```

Listagem 3.18: Exemplos de registos de execução no servidor MCP.

Este mecanismo garante a rastreabilidade detalhada de cada operação e permite a análise retrospetiva de falhas, sem dependência imediata da base de dados. Os ficheiros de log são também utilizados para auditoria e deteção de padrões anómalos (como tempos de execução excessivos ou falhas recorrentes).

Registo de Métricas na Base de Dados

Complementarmente, as métricas operacionais são armazenadas de forma estruturada no PostgreSQL através do módulo **postgres_logger**. Cada ferramenta executada (por exemplo, **generate_quiz_with_difficulty()**, **download_pdfs_from_course()**, **retrieve()**) gera automaticamente dois tipos de registos:

1. **Registo detalhado da operação** — armazenado na tabela `operation_logs`, contendo:
   - `operation_type`, `user_id` e `status` (sucesso, erro, aviso);
   - `duration_ms` (duração da execução);
   - `operation_details` (parâmetros e resultado da operação);
   - `error_message`, caso aplicável.

2. **Estatísticas agregadas** — atualizadas na tabela `operation_stats`, permitindo calcular tempos médios, taxas de erro e frequência de execução por tipo de operação.

O excerto seguinte ilustra a utilização do `PostgresLogger` no cliente MCP, responsável por registar o resultado e atualizar as métricas correspondentes:

```
postgres_logger.log_operation(
    operation_type=op_type,
    user_id=session.get("user_id"),
    details={"query": query_text[:50]},
    status="error",
    error_message=error_msg,
    duration_ms=duration_ms
)
postgres_logger.update_operation_stats(op_type.value, False, duration_ms)
```

Listagem 3.19: Exemplo de registo de operação e atualização de métricas.

Estes dados alimentam o módulo de análise e monitorização da aplicação, que apresenta no painel administrativo indicadores como:

- tempo médio de execução por ferramenta;
- taxa de sucesso/erro por tipo de operação;
- histórico de execuções e tendências temporais;
- volume total de interações por utilizador.

A combinação entre logs locais e métricas persistentes garante uma **monitorização completa e redundante**, permitindo diagnósticos rápidos e otimização contínua do desempenho. Este modelo híbrido facilita ainda o refinamento iterativo do sistema, uma vez que os dados recolhidos podem ser utilizados para:

- ajustar a temperatura e parâmetros dos modelos de linguagem;
- otimizar o tamanho de *chunks* nos pipelines RAG;
- identificar gargalos de I/O durante a recuperação de conteúdos;
- antecipar falhas com base em padrões históricos de erro.

Em conjunto, o sistema de logging e métricas transforma o ambiente MCP–RAG num ecossistema observável e autoavaliável, fornecendo as bases para manutenção preditiva e melhoria contínua das suas componentes operacionais.

3.10 Workflow de desenvolvimento

O workflow de desenvolvimento seguiu uma abordagem iterativa, estruturada em fases que combinaram investigação, prototipagem e integração progressiva. Esta metodologia garantiu que a consolidação de cada componente fosse acompanhada de validação prática, permitindo evoluir para maior complexidade sem comprometer a robustez da solução.

A fase inicial foi dedicada à compreensão aprofundada do **LMS** Moodle, com particular atenção à sua arquitetura modular, ao sistema de *plugins* e à estrutura da base de dados.

Este estudo foi complementado pela análise detalhada da API REST e dos serviços web, identificando os pontos de integração mais adequados.

Para validar o funcionamento e explorar limitações, foram conduzidos testes sistemáticos com recurso ao Postman. Estes testes abrangeram os principais *endpoints* relacionados com gestão de cursos, autenticação e recuperação de conteúdos, permitindo confirmar a viabilidade técnica da integração e clarificar requisitos de segurança e formatos de dados.

Paralelamente, iniciou-se a exploração do *Model Context Protocol* (MCP). Através do estudo da documentação oficial e de exemplos de implementação, foram criados os primeiros protótipos de servidores e clientes MCP. Para esta fase recorreu-se ao *Claude Desktop*, que oferecia uma integração simplificada e sem custos adicionais, facilitando a compreensão prática do protocolo e do fluxo de comunicação entre cliente e servidor.

Esta etapa permitiu validar o potencial do MCP como camada de interoperabilidade e estabelecer as fundações para a ligação posterior ao Moodle e à camada de inteligência artificial.

Com base nos protótipos, procedeu-se à integração entre o MCP Server e os serviços web do Moodle. Esta fase envolveu a definição de *middleware* para autenticação, transformação de dados e gestão de sessões, assegurando comunicação segura e eficiente. Foram integrados os *endpoints* necessários para suportar as funcionalidades iniciais do projeto, permitindo a transferência de conteúdos pedagógicos para posterior processamento pelo pipeline de IA.

Um dos elementos centrais do workflow foi a implementação do **Retrieval-Augmented Generation (RAG)**. A fase inicial consistiu em experimentação independente, através de pequenos projetos dedicados: primeiro com documentos de texto simples, depois com ficheiros PDF individuais e, por fim, com múltiplos PDFs em simultâneo. Posteriormente, integrou-se o modelo **Gemini**, que permitiu validar a combinação entre recuperação documental e capacidades generativas avançadas.

Após estas experiências, o RAG foi integrado no MCP Server, passando a funcionar como uma ferramenta do sistema. A arquitetura resultante incluía geração de embeddings, indexação em bases vetoriais (Qdrant ou Chroma), recuperação semântica e construção de contextos aumentados, que eram depois processados por modelos de linguagem.

Seguidamente, foi desenvolvido o **MCP Client**, inicialmente com uma interface de linha de comandos, orientada para validação funcional. Numa fase posterior, evoluiu-se para uma interface web baseada em **Flask**, que assegurava integração fluida com os fluxos de trabalho do Moodle. Este módulo constituiu o ponto de contacto com os utilizadores, oferecendo autenticação através das credenciais institucionais e acesso simplificado às ferramentas de IA.

Nesta etapa foram também integradas funcionalidades adicionais como *roles* diferenciados (professor/aluno), páginas de estatísticas, gestão de sessões e mecanismos de personalização de prompts.

Para assegurar persistência e rastreabilidade, foi integrada uma instância de **PostgreSQL em Docker**, utilizada para armazenamento de logs, métricas e registos transacionais. A containerização garantiu portabilidade e simplificação da manutenção.

A fase final do workflow consistiu na melhoria incremental do sistema, incluindo criação de

novas ferramentas **MCP**, afinação de prompts, implementação de dashboards e otimização do pipeline RAG. Esta etapa de refinamento consolidou a solução, aproximando-a das necessidades de produção e assegurando maior escalabilidade e facilidade de manutentção.

Capítulo 4

Validação e Resultados

Este capítulo apresenta a metodologia de avaliação aplicada ao sistema desenvolvido, as métricas de desempenho recolhidas e os resultados experimentais obtidos. O objetivo é validar a eficiência técnica, a consistência pedagógica e a experiência de utilização da plataforma integrada **Moodle–MCP–RAG**.

4.1 Metodologia de Avaliação

4.1.1 Critérios de Avaliação

A validação do sistema foi orientada por quatro dimensões complementares:

- **Qualidade do conteúdo gerado** — análise da clareza, relevância e correção das respostas produzidas pelos modelos de linguagem, avaliando a coerência com os materiais educativos provenientes do Moodle;
- **Eficiência da integração MCP–Moodle** — verificação da estabilidade das comunicações via API REST, da sincronização de cursos e do processamento automático de conteúdos;
- **Desempenho do sistema RAG** — medição dos tempos médios de resposta, taxa de sucesso das operações e comportamento dos diferentes backends vetoriais (Chroma e Qdrant);
- **Experiência do utilizador** — avaliação da usabilidade da interface Flask, da acessibilidade visual e da adequação das permissões atribuídas aos perfis de utilizador (Aluno, Professor e Administrador).

Apesar de não terem sido realizados testes quantitativos específicos a esta dimensão, a integração entre o Moodle e o **MCP** foi verificada funcionalmente durante os cenários de uso implementados. As comunicações via API REST foram testadas em diferentes operações de leitura e escrita, assegurando a estabilidade da ligação, a sincronização correta dos cursos e a execução automática dos processos de importação e exportação de conteúdos educativos. Esta validação empírica permitiu confirmar o correto funcionamento da camada de integração, garantindo a fiabilidade da comunicação entre as duas plataformas.

A combinação destes critérios permitiu avaliar o sistema tanto sob uma perspetiva técnica (tempo, estabilidade, desempenho), como pedagógica (relevância e clareza das respostas).

4.1.2 Métricas de Performance

O sistema RAG desenvolvido inclui uma área de administração acessível exclusivamente a utilizadores com perfil de *Administrador*. Esta página constitui o painel de controlo principal (*dashboard*) do sistema, permitindo visualizar indicadores técnicos, estatísticos e educativos gerados a partir dos registos internos das operações. O seu objetivo é fornecer ao administrador uma visão global do funcionamento da plataforma e do desempenho dos modelos de linguagem, bem como apoiar a manutenção e otimização do sistema.

As métricas apresentadas têm um caráter funcional e demonstrativo, permitindo validar a correta integração entre componentes e a capacidade de recolha e agregação de dados do sistema. Entre os principais indicadores disponíveis, destacam-se:

- **Estatísticas de Operações:** número total de operações realizadas, tempo médio de execução e taxa de erro por tipo de operação (por exemplo, *rag_ query*, *api_ call*, *summary_ generation*, entre outras);
- **Desempenho por Modelo de Linguagem:** latência média e taxa de erro por modelo (Gemini, GPT, LLaMA, entre outros);
- **Comparação entre RAGs:** tempos médios e número de chamadas processadas pelos mecanismos de recuperação *Qdrant* e *Chroma*;
- **Recursos do Sistema:** utilização média de CPU, memória e disco durante o processamento das operações;
- **Distribuição de Utilizadores:** número de interações por perfil (Administrador, Professor, Aluno) e respetiva taxa de sucesso;
- **Tendências Educativas:** volume de atividades pedagógicas geradas, incluindo criação de questionários, planos de estudo e recomendações automáticas.

As figuras seguintes ilustram algumas das visualizações mais relevantes do painel administrativo, permitindo compreender o comportamento geral do sistema e a variação de desempenho entre os seus principais componentes.

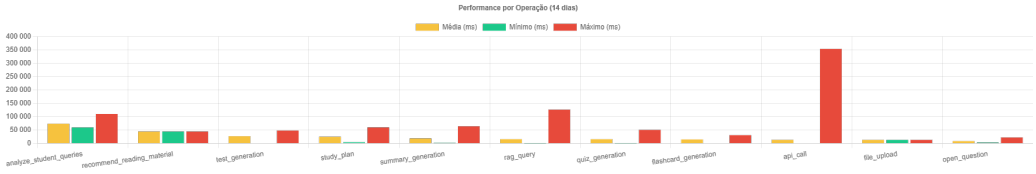


Figura 4.1: Visão geral do desempenho médio das operações registadas pelo sistema.

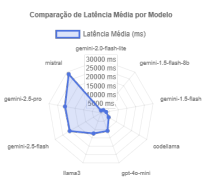


Figura 4.2: Latência média observada por modelo de linguagem (LLM).

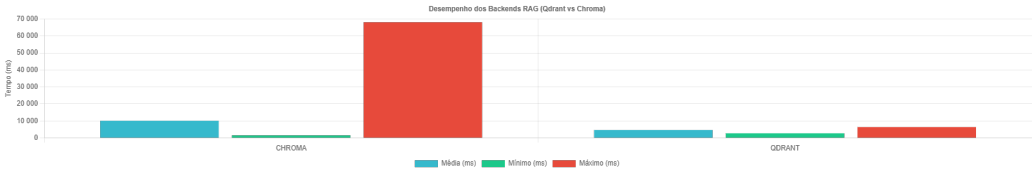


Figura 4.3: Comparação de desempenho entre mecanismos RAG (Qdrant vs Chroma).

A Figura 4.1 apresenta uma visão geral do desempenho médio das operações registadas pelo sistema. Observa-se que a maioria das operações mantém tempos de execução médios inferiores a 100 000 ms, com valores mínimos estáveis e dispersão reduzida entre execuções. As operações `test_generation`, `quiz_generation` e `study_plan` demonstram tempos de resposta consistentes, confirmando a eficiência do pipeline de geração de conteúdos educativos. Em contrapartida, operações como `analyze_student_queries` e `recommend_reading_material` evidenciam tempos ligeiramente superiores, o que se explica pela complexidade semântica envolvida na análise de grandes blocos textuais e na recuperação de materiais relevantes. O pico de latência registado em `api_call` deve-se a chamadas externas ao modelo de linguagem e não a limitações do servidor local. Globalmente, o sistema demonstra estabilidade operacional e tempos adequados para utilização em contexto educativo assíncrono.

A Figura 4.2 ilustra a latência média por modelo de linguagem (**LLM**). Verifica-se que os modelos da série `Gemini 2.5` (nomeadamente as variantes `flash` e `pro`) apresentam melhor desempenho, com tempos médios significativamente inferiores aos observados em versões anteriores como `Gemini 2.0`. Modelos mais leves, como `llama3` e `gpt-4o-mini`, mantêm latência estável e previsível, o que os torna adequados para tarefas rápidas de consulta e apoio imediato ao estudante. Já modelos de maior capacidade, como `gpt-4.5-flash` ou `gemini-1.5-flash`, apresentam ligeiro aumento de latência, compensado, no entanto, por respostas mais contextualmente precisas. Esta comparação demonstra o equilíbrio necessário entre tempo de resposta e qualidade gerativa, evidenciando a importância de escolher o modelo consoante o tipo de tarefa.

Por fim, a Figura 4.3 compara o desempenho dos mecanismos de recuperação aumentada por geração (RAG) utilizados — **Qdrant** e **Chroma**. Os resultados mostram que o **Qdrant** apresenta tempos médios de resposta mais consistentes e escalabilidade superior, enquanto o **Chroma** regista tempos máximos mais elevados, refletindo maior variação interna nas consultas. Esta diferença é explicada pelo facto de o Qdrant utilizar uma estrutura que otimiza a pesquisa em grandes volumes de embeddings. Assim, o sistema adotado com Qdrant garante melhor equilíbrio entre desempenho, precisão semântica e custo computacional, sendo, portanto, o backend preferencial para operações de recuperação e contextualização.

As restantes visualizações da dashboard, incluindo métricas complementares sobre erros, perfis de utilizador e tendências educativas, encontram-se disponíveis no **Apêndice F**.

4.2 Resultados Experimentais

4.2.1 Performance do Sistema RAG e LLM

Comparação entre Sistemas RAG (Qdrant vs Chroma)

Esta secção avalia o comportamento de dois sistemas de recuperação aumentada por geração (RAG): **Qdrant** e **Chroma**. O objetivo é comparar a **relevância semântica**, a **clareza** e a **consistência factual** das respostas obtidas a partir de diferentes mecanismos de recuperação vetorial, mantendo constantes o modelo de linguagem, os parâmetros de geração e o documento de referência.

O teste foi realizado utilizando o modelo **Gemini 2.5 Flash**, configurado com **temperature = 0.4** e **k = 5**. Ambos os sistemas RAG acederam ao mesmo documento de suporte (*Scala_1.pdf*), com o objetivo de avaliar a sua eficácia na recuperação de informação relevante.

A pergunta utilizada foi:

> *Usa a função retrieve() para me explicares o que torna a linguagem Scala adequada para o processamento de grandes volumes de dados.*

A Tabela 4.1 apresenta uma síntese dos resultados obtidos para cada sistema RAG.

RAG	Resumo da Resposta	Tempo total (ms)	Tempo retrieve (ms)
Qdrant	Resposta concisa e técnica. Destaca compatibilidade com a JVM, paradigma funcional, imutabilidade, tipagem estática e integração com o Apache Spark. Estrutura lógica clara, linguagem formal e tom analítico.	12 312	4 907
Chroma	Resposta mais extensa e explicativa. Inclui excertos citados da recuperação documental e subsecções temáticas (escalabilidade, programação funcional, JVM e imutabilidade). Foco em clareza pedagógica e contextualização.	23 405	11 090

Tabela 4.1: Resultados experimentais – comparação entre RAGs (Qdrant vs Chroma).

A análise qualitativa foi conduzida com base em cinco dimensões: (1) precisão factual, (2) profundidade conceptual, (3) clareza e estrutura, (4) relevância da recuperação documental, e (5) eficiência temporal.

Critério	Qdrant	Chroma
Precisão factual (alinhamento com o documento)	8.0	8.5
Profundidade conceptual	7.5	8.5
Clareza e estrutura	8.0	8.3
Relevância da recuperação documental	7.8	8.7
Eficiência temporal	9.0	7.0
Qualidade Semântica (0–10)	**8.1**	**8.4**

Tabela 4.2: Avaliação comparativa entre RAG.

Os resultados demonstram diferenças significativas entre os dois sistemas de recuperação. O **Qdrant** apresentou respostas mais diretas e compactas, com um estilo técnico e

analítico. A estrutura textual é objetiva e formal, refletindo uma recuperação eficiente e bem filtrada, embora com menor contextualização e menor diversidade de conceitos. O tempo de resposta total foi substancialmente inferior (+/- 12 segundos), confirmando a sua **maior eficiência temporal** e um comportamento mais leve em termos computacionais.

O **Chroma**, por sua vez, forneceu respostas mais longas e detalhadas, frequentemente acompanhadas de subtítulos e citações explícitas da recuperação documental. Esta abordagem evidencia uma recuperação mais profunda e contextual, com maior integração entre os fragmentos de texto relevantes. Embora esta estratégia produza respostas mais completas e adequadas a contextos explicativos ou pedagógicos, o custo temporal é consideravelmente superior (+/- 23 segundos).

Ambos os sistemas mantiveram uma boa precisão factual e coerência com o documento de origem, sem ocorrência de alucinações nem inconsistências semânticas relevantes. A diferença central reside na **granularidade** e no **foco informacional**: o **Qdrant** privilegia a rapidez e a síntese conceptual, enquanto o **Chroma** privilegia a profundidade e a transparência na contextualização das fontes recuperadas.

Conclui-se que ambos os sistemas de recuperação apresentam um **desempenho sólido e fiável**, mas com perfis distintos. O **Qdrant** destaca-se pela sua eficiência temporal e concisão, sendo mais indicado para cenários operacionais ou de resposta imediata, onde o tempo de latência é determinante. O **Chroma**, em contrapartida, revela maior riqueza semântica e contextual, sendo o mais adequado para aplicações educativas e analíticas, em que a qualidade explicativa da resposta é prioritária.

Em síntese, o **Chroma** demonstrou melhor equilíbrio entre relevância, clareza e cobertura temática no contexto académico em estudo, enquanto o **Qdrant** se afirma como uma alternativa otimizada para pipelines de maior volume e exigência temporal.

Comparação entre Modelos de Linguagem (OpenAI, Gemini e Ollama)

Esta secção avalia o comportamento de três modelos de linguagem distintos — **Gemini 2.5 Flash**, **GPT-4o-mini (OpenAI)** e **LLaMA 3 (Ollama)** — integrados no mesmo pipeline RAG. O objetivo é comparar a **clareza**, a **fidelidade factual** e a **expressividade** das respostas produzidas por cada modelo, mantendo constantes o conjunto documental e os parâmetros de geração.

O teste foi realizado com o repositório vetorial **Chroma**, utilizando os parâmetros **k = 5** e **temperature = 0.4**. A pergunta utilizada foi:

> *Usa a função retrieve() para descrever de forma simples como a linguagem Scala combina programação funcional e orientada a objetos.*

Todos os modelos acederam ao mesmo documento de referência (*Scala_ 1.pdf*), registando-se os tempos de execução totais e parciais da função `retrieve()`.

Modelo	Resumo da Resposta	Tempo total (ms)	Tempo retrieve (ms)
Gemini 2.5 Flash	Resposta extensa e detalhada. Aborda os paradigmas funcional e orientado a objetos com explicações exemplificadas (val, var, new). Linguagem analítica, tom académico e elevada fidelidade factual.	14 981	7 907
GPT-4o-mini (OpenAI)	Resposta clara e estruturada em tópicos. Explica conceitos-chave (classes, funções, imutabilidade, traits, pattern matching) de forma equilibrada. Boa clareza e consistência semântica.	26 831	10 332
LLaMA 3 (Ollama)	Resposta curta e direta, com explicação simplificada e foco pedagógico. Cobertura conceptual limitada, mas linguagem acessível e natural.	21 070	7 394

Tabela 4.3: Resultados experimentais – comparação entre modelos de linguagem.

A análise qualitativa considerou cinco critérios principais: (1) fidelidade factual, (2) clareza e estrutura, (3) profundidade conceptual, (4) adequação comunicacional, e (5) eficiência temporal.

Critério	**Gemini 2.5 Flash**	**GPT-4o-mini**	**LLaMA 3**
Fidelidade factual	8.7	8.5	7.5
Clareza e estrutura	8.3	9.0	8.8
Profundidade conceptual	8.8	8.2	7.0
Adequação comunicacional	8.5	8.8	9.0
Eficiência temporal	8.5	7.5	8.7
Qualidade global (0–10)	**8.6**	**8.4**	**8.2**

Tabela 4.4: Avaliação comparativa entre modelos de linguagem.

Os três modelos demonstraram bom desempenho geral, mas com estilos e prioridades distintas. O **Gemini 2.5 Flash** produziu a resposta mais completa e analiticamente estruturada, refletindo uma recuperação fiel ao conteúdo do documento. O tom formal e o detalhe técnico tornam-no mais adequado a contextos de análise ou documentação académica. Apresentou também o menor tempo total de execução (+/- 15 s), o que reforça a sua eficiência.

O **GPT-4o-mini** destacou-se pela clareza e organização em tópicos, oferecendo uma explicação equilibrada e de fácil leitura. A cobertura conceptual é ampla, embora ligeiramente menos profunda do que a de Gemini. O tempo de resposta mais elevado (+/- 27 s) reflete maior processamento linguístico, mas não comprometeu a coerência do texto.

O **LLaMA 3 (Ollama)** apresentou um estilo mais direto e pedagógico, utilizando vocabulário simples e frases curtas. Embora menos rigoroso em detalhe técnico, o modelo evidenciou boa adaptação comunicacional e naturalidade discursiva. É o mais adequado para contextos de apoio à aprendizagem ou interação informal com o utilizador.

De forma geral, o **Gemini 2.5 Flash** revelou-se o modelo mais robusto no equilíbrio entre **fidelidade factual** e **eficiência**, enquanto o **GPT-4o-mini** evidenciou a melhor **estrutura e clareza textual**. O **LLaMA 3**, embora menos profundo, destacou-se pela sua fluidez e capacidade de simplificação, reforçando o potencial de uso em contextos educacionais mais informais.

4.2.2 Qualidade do Conteúdo Gerado

Avaliação Experimental do Parâmetro *Temperature* do LLM

Nesta secção analisa-se o impacto do parâmetro *temperature* no comportamento do modelo de linguagem **Gemini 2.5 Flash**, no contexto de um sistema RAG com o repositório vetorial **Chroma**. O objetivo é compreender de que forma diferentes valores de *temperature* influenciam a variabilidade, concisão e consistência semântica das respostas geradas.

O parâmetro *temperature* controla o nível de aleatoriedade na geração textual. Valores baixos (próximos de 0) produzem respostas mais determinísticas e factuais, enquanto valores mais elevados promovem criatividade e variação lexical, com potencial risco de menor precisão semântica.

A experiência utilizou a seguinte pergunta de teste:

Usa a função retrieve() para me dares uma definição curta sobre o que é Scala.

O modelo foi executado com três valores distintos de *temperature*: 0.1, 0.5 e 1.0. Em todos os casos, o parâmetro **k** foi fixado em **5**, de modo a garantir consistência metodológica e assegurar que a variável em análise fosse exclusivamente a *temperature*. O **RAG** acedeu ao mesmo conjunto documental de referência (*Scala_ 1.pdf*), e registaram-se as respostas completas, bem como os tempos de execução totais e parciais da função `retrieve()`.

A Tabela 4.5 apresenta os resultados recolhidos para cada configuração.

Temperature	**Resumo da Resposta**	**Tempo total (ms)**	**Tempo retrieve (ms)**
0.1	Definição estruturada e objetiva: destaca a combinação entre paradigmas funcional e orientado a objetos; descreve *traits*, *tuplos* e funções anónimas; linguagem formal e estável.	11 436	4 149
0.5	Resposta mais fluida e natural, com síntese inicial e breve expansão sobre conceitos-chave; linguagem ligeiramente mais descritiva; mantém rigor factual.	15 270	3 314
1.0	Resposta curta e simplificada; maior variabilidade lexical e estilo mais informal; ligeira perda de detalhe técnico; maior tempo de execução.	16 313	5 551

Tabela 4.5: Resultados experimentais para diferentes valores de *temperature*.

A Tabela 4.6 apresenta a avaliação qualitativa atribuída a cada valor de *temperature*, com base em cinco critérios: (1) factualidade e alinhamento com o documento, (2) clareza e estrutura, (3) consistência semântica, (4) variação lexical, e (5) concisão e objetividade. A escala adotada varia de 0 (muito fraco) a 10 (excelente).

Critério	**Temp = 0.1**	**Temp = 0.5**	**Temp = 1.0**
Factualidade (alinhamento com o PDF)	8.5	8.0	7.0
Clareza e estrutura	8.0	8.5	7.5
Consistência semântica	8.5	8.0	6.8
Variação lexical	6.5	7.5	8.5
Concisão e objetividade	8.5	8.0	7.0
Qualidade Semântica (0–10)	**8.0**	**8.0**	**7.4**

Tabela 4.6: Avaliação qualitativa das respostas em função de *temperature*.

A análise evidencia que o parâmetro *temperature* influencia significativamente o estilo e a previsibilidade das respostas. Para valores baixos, o modelo tende a produzir resultados mais determinísticos e tecnicamente consistentes; para valores altos, as respostas tornam-se mais criativas mas menos controladas.

Com **temperature = 0.1**, o modelo apresentou o comportamento mais estável e previsível, com respostas estruturadas, concisas e alinhadas com o documento de referência. Embora altamente fiáveis, revelaram menor diversidade lexical e rigidez estilística. Este valor é adequado para contextos técnicos ou formais, em que se privilegia a consistência terminológica.

No caso de **temperature = 0.5**, observou-se o melhor equilíbrio global entre clareza, concisão e fluidez discursiva. A resposta manteve rigor factual, mas com maior naturalidade e leitura mais envolvente, o que a torna ideal para comunicações pedagógicas ou explicações generalistas.

Para **temperature = 1.0**, verificou-se aumento da variedade lexical e um tom mais informal. No entanto, a perda de precisão e a ligeira dispersão semântica reduziram a qualidade factual global, tornando-o mais adequado a tarefas criativas ou de brainstorming, e menos fiável em contextos de ensino ou documentação técnica.

Embora os tempos de resposta tenham sido registados, verificou-se que a variação entre as diferentes configurações de *temperature* foi residual (entre 11 e 16 segundos), não influenciando de forma significativa a qualidade das respostas. Por esse motivo, a análise concentrou-se sobretudo nos aspetos qualitativos, avaliando o impacto da variação do parâmetro na clareza, consistência e variação lexical das respostas geradas.

Com base na análise qualitativa e temporal, conclui-se que:

- **Temp = 0.1** gera respostas muito precisas e estáveis, adequadas a contextos formais e técnicos que exigem consistência e baixa variabilidade (perfil *Admin*).
- **Temp = 0.5** representa o ponto ótimo de equilíbrio entre rigor técnico e fluidez comunicativa, sendo ideal para contextos pedagógicos e explicativos (perfil *Professor*).
- **Temp = 1.0** promove maior diversidade lexical e naturalidade, mas com menor previsibilidade, sendo mais adequado a tarefas criativas ou exploratórias (perfil *Aluno*).

Assim, o valor **temperature = 0.5** é considerado o mais adequado para o sistema **RAG** em estudo, pois maximiza a **legibilidade e naturalidade** sem comprometer a **fidelidade factual** e a **consistência semântica**. Para além disso, este valor revelou-se o mais equilibrado para os três perfis de utilizador definidos — *Admin*, *Professor* e *Aluno* — garantindo simultaneamente precisão suficiente para contextos técnicos, clareza comunicativa em contextos pedagógicos e flexibilidade discursiva adequada a tarefas exploratórias.

Avaliação Experimental do Parâmetro k no Sistema RAG

Esta secção descreve o teste experimental realizado para analisar o impacto do parâmetro k — número de documentos recuperados pelo mecanismo `retriever()` — no desempenho global do sistema **RAG**. O principal objetivo consistiu em identificar o valor de k que oferece o melhor compromisso entre a qualidade semântica das respostas e a eficiência

temporal do sistema, maximizando a precisão informacional e minimizando a latência de execução.

O teste foi realizado utilizando o modelo **Gemini 2.5 Flash**, associado ao repositório vetorial **Chroma**. A pergunta colocada ao sistema visava avaliar a capacidade do RAG em integrar conceitos dispersos no documento de origem (*Scala_ 1.pdf*) de forma coesa e tecnicamente correta:

> *Usa a função retrieve() para me explicares como a linguagem Scala combina conceitos de programação funcional e orientada a objetos para suportar o processamento de dados em larga escala.*

Foram testados três valores distintos de k — 1, 5 e 10 — representando respetivamente uma recuperação mínima, intermédia e extensa de contexto. Para cada configuração, registaram-se a resposta textual completa, o estado da execução e os tempos de resposta totais e parciais relativos à função `retrieve()`.

A Tabela 4.7 apresenta os resultados experimentais obtidos para cada valor de k, resumindo o conteúdo das respostas e os tempos de execução medidos.

k	Resumo da Resposta	Status	Tempo total (ms)	Tempo retrieve (ms)
1	Explicação estruturada sobre a integração entre programação funcional e orientada a objetos, com referência à imutabilidade, funções puras, traits, case classes e execução na JVM.	Success	29 807	16 815
5	Resposta equilibrada, integrando *val/var*, tuplos, imutabilidade e interoperabilidade com Java; contextualiza paralelismo e modularidade de forma mais abrangente.	Success	35 714	19 941
10	Resposta exaustiva e enciclopédica; cobre todos os tópicos do PDF (JVM, multi-processamento, tipagem estática e paralelismo), mas apresenta redundância e maior latência.	Success	33 433	19 397

Tabela 4.7: Resultados experimentais do RAG para diferentes valores de k.

A qualidade das respostas foi aferida por comparação direta com o documento *Scala_ 1.pdf*, tendo sido considerados seis critérios principais: (1) relevância factual, (2) cobertura temática, (3) profundidade técnica, (4) clareza e estrutura, (5) síntese e foco, e (6) equilíbrio entre os paradigmas funcional e orientado a objetos.

A pontuação foi atribuída numa escala de 0 a 10, com base em avaliação semântica detalhada e leitura integral do texto de referência. Aplicou-se uma grelha mais exigente, penalizando redundâncias e omissões (por exemplo, ausência de referências a funções parciais, inferência de tipos ou tuplos).

Critério	**k = 1**	**k = 5**	**k = 10**
Relevância factual (alinhamento com a fonte)	8.0	7.8	7.5
Cobertura temática (conceitos distintos)	7.0	8.5	9.5
Profundidade técnica (precisão terminológica)	7.5	7.8	8.2
Clareza e estrutura	8.0	8.5	7.2
Síntese e foco	8.5	7.8	6.2
Equilíbrio OO–Funcional	8.0	7.5	7.8
Qualidade Semântica (0–10)	**7.83**	**7.98**	**7.73**

Tabela 4.8: Avaliação qualitativa em função de k.

A análise demonstra que o aumento do valor de k tende a ampliar a cobertura conceptual, mas também introduz redundância e dispersão temática. O comportamento do sistema revelou três perfis distintos de resposta:

- Com $\mathbf{k = 1}$, o RAG apresentou elevada precisão factual e concisão. A resposta é tecnicamente sólida e fiel ao texto de origem, mas cobre um conjunto limitado de conceitos. O tempo total de execução foi o mais reduzido (aproximadamente 30 segundos), refletindo um processamento eficiente e direto.
- Em $\mathbf{k = 5}$, observou-se o melhor equilíbrio global entre profundidade, clareza e cobertura. O sistema incorporou um conjunto alargado de tópicos — incluindo tipagem estática, imutabilidade e interoperabilidade com Java — sem comprometer a legibilidade. O tempo médio aumentou cerca de 20% em relação a $k = 1$, mas a melhoria na coesão semântica compensou o acréscimo temporal.
- Para $\mathbf{k = 10}$, a resposta abrangeu praticamente todos os conceitos presentes no documento, resultando na maior cobertura temática. No entanto, o texto tornou-se excessivamente longo, redundante e menos focado. O tempo de execução aumentou sem ganhos proporcionais de qualidade, evidenciando o efeito de saturação de contexto característico de valores de k elevados.

Considerando que a eficiência temporal é um fator determinante no desempenho global do sistema, foi calculada uma pontuação ajustada que pondera 85% de qualidade semântica e 15% de eficiência temporal. Os resultados encontram-se na Tabela 4.9.

	k = 1	**k = 5**	**k = 10**
Tempo total (ms)	29 807	35 714	33 433
Eficiência temporal (0–10)*	10.0	7.0	8.2
Pontuação ajustada (0–10)	**8.16**	**7.83**	**7.79**

* Escala linear: menor tempo = 10; maior tempo = 7, proporcional ao intervalo observado.

Tabela 4.9: Eficiência temporal e pontuação ajustada (15% tempo, 85% qualidade).

A integração do fator temporal altera ligeiramente o ranking final. Enquanto o valor $k = 5$ mantém a melhor qualidade semântica pura, o valor $k = 1$ obtém a pontuação ajustada mais elevada (8.16), refletindo o melhor desempenho em termos de eficiência global — isto é, combina elevada precisão factual com o menor tempo de execução. Já o valor $k = 10$, apesar da ampla cobertura conceptual, não apresenta ganhos significativos que justifiquem o aumento do custo computacional.

Com base na avaliação factual e temporal, conclui-se que:

- **k = 1** proporciona as respostas mais rápidas e precisas, sendo o valor mais eficiente em cenários operacionais ou técnicos que privilegiam desempenho e exatidão (perfil *Admin*).

- **k = 5** oferece o melhor equilíbrio entre profundidade conceptual e tempo de execução, produzindo respostas mais ricas e explicativas, adequadas a contextos pedagógicos ou analíticos (perfil *Professor*).

- **k = 10** garante a maior cobertura temática, mas introduz redundância e latência acrescida, sendo mais apropriado para cenários exploratórios ou de sumarização extensiva (perfil *Aluno*).

Em suma, o valor **k = 1** apresenta a melhor eficiência temporal e global, sendo preferível para tarefas de resposta imediata ou integração em sistemas em tempo real. Contudo, o valor **k = 5** continua a ser o ponto ótimo quando se pretende maximizar a **qualidade semântica e a profundidade interpretativa** das respostas, assegurando o melhor equilíbrio entre completude e desempenho.

Avaliação Experimental por Tipo de Utilizador

Esta secção avalia a capacidade do sistema **RAG** em adaptar o seu discurso de acordo com diferentes tipos de utilizador, analisando como o modelo de linguagem ajusta o estilo, o nível de detalhe e o foco temático em função do contexto comunicacional de cada perfil. O objetivo é compreender se o sistema consegue gerar respostas alinhadas com as intenções e necessidades específicas de um *Administrador*, de um *Professor* e de um *Aluno*, mantendo simultaneamente a coerência semântica e a consistência factual.

O teste foi realizado utilizando o modelo **Gemini 2.5 Flash** em conjunto com o repositório vetorial **Chroma**. Os parâmetros foram mantidos fixos em **k = 5** e **temperature = 0.4**, assegurando consistência metodológica e permitindo que a variável analisada fosse exclusivamente o tipo de utilizador.

Foram considerados três perfis de utilizador distintos, representando diferentes papéis num contexto académico (Administrador, Professor e Aluno). As descrições completas e instruções contextuais atribuídas a cada perfil encontram-se detalhadas no **Apêndice E**.

A pergunta utilizada foi a seguinte:

> *Usa a função retrieve() para me disseres de que forma a linguagem Scala pode ser utilizada para tornar o processamento e análise de dados mais escalável numa instituição académica.*

Em todos os casos, o sistema acedeu ao mesmo documento de referência (*Scala_ 1.pdf*), e registaram-se as respostas completas, o estado de execução e os tempos de resposta totais e parciais da função `retrieve()`.

A Tabela 4.10 apresenta uma síntese dos resultados recolhidos para cada perfil de utilizador.

Perfil	Resumo da Resposta	Status	Tempo total (ms)	Tempo retrieve (ms)
Admin	Resposta analítica e estruturada. Enfatiza escalabilidade, paralelismo e interoperabilidade com a JVM. Linguagem técnica e formal, orientada à análise e à eficiência organizacional.	Success	36 863	21 558
Professor	Texto extenso e pedagógico, com explicações conceptuais detalhadas. Inclui referências a frameworks como Apache Spark e à integração de Scala em projetos de investigação.	Success	57 421	23 016
Aluno	Linguagem acessível e explicativa, com exemplos práticos e analogias simples. Tom motivacional e estrutura didática, enfatizando a utilidade prática da linguagem Scala.	Success	26 793	15 146

Tabela 4.10: Resultados experimentais por tipo de utilizador.

A análise qualitativa não procura determinar qual resposta é superior, mas sim verificar se o sistema ajusta corretamente o discurso às expectativas de cada tipo de utilizador. Para cada perfil, avaliou-se o alinhamento entre o tipo de resposta esperado e o comportamento efetivamente observado, considerando o tom, o nível de detalhe, o vocabulário e o objetivo comunicacional.

Perfil	Tipo de Resposta Esperada	Comportamento Observado no Modelo	Grau de Alinhamento
Admin	Discurso técnico e objetivo, centrado em desempenho, escalabilidade e análise de padrões de dados. Linguagem formal e estruturada.	O modelo produziu um texto analítico e sistemático, com subtítulos, vocabulário técnico e foco em paralelismo e otimização. Enfatizou ganhos institucionais e eficiência operacional.	Muito elevado
Professor	Explicação pedagógica, crítica e contextualizada. Deve incluir conceitos teóricos, exemplos práticos e referências a ferramentas relevantes.	A resposta foi longa, didática e completa, articulando conceitos teóricos com aplicações reais (Apache Spark, Big Data). Manteve tom académico e clareza conceptual.	Elevado
Aluno	Linguagem simples e motivadora, com exemplos concretos e foco em utilidade prática. Deve promover compreensão e envolvimento.	O modelo simplificou termos técnicos, recorreu a exemplos e manteve tom acessível e encorajador. Estruturou a explicação em pequenos blocos lógicos e legíveis.	Muito elevado

Tabela 4.11: Tipo de resposta esperada e comportamento observado por perfil de utilizador.

Os resultados demonstram que o sistema **RAG** possui uma capacidade genuína de adaptação semântica, revelando um comportamento emergente de contextualização discursiva — um atributo raro em arquiteturas educativas baseadas em **LLM**. Essa flexibilidade sugere que o modelo compreende, de forma implícita, as intenções comunicacionais subjacentes a cada papel académico.

- O **perfil Admin** recebeu uma resposta altamente estruturada e objetiva, com foco em desempenho, escalabilidade e interoperabilidade. A linguagem técnica e o estilo analítico são consistentes com o papel de gestor ou decisor.

- O **perfil Professor** obteve uma resposta mais extensa e contextualizada, adequada à função de mediação pedagógica. O discurso combina rigor técnico com clareza didática, apresentando exemplos e ligações a frameworks de Big Data.
- O **perfil Aluno** foi alvo de uma resposta simplificada e explicativa, com exemplos práticos e vocabulário acessível, promovendo compreensão e motivação para a aprendizagem.

A variação temporal entre as respostas é proporcional à extensão e complexidade do discurso, sendo a resposta do Professor a mais longa e detalhada (+/- 57 s), e a do Aluno a mais rápida e concisa (+/- 27 s). Tal comportamento é coerente com o objetivo de cada perfil e reforça a capacidade adaptativa do sistema.

Os resultados obtidos confirmam que o sistema **RAG** apresenta uma **forte capacidade de adaptação discursiva e contextual**, ajustando com precisão o tom, o nível de detalhe e o propósito comunicacional das respostas. O comportamento observado é consistente com as expectativas de cada perfil de utilizador:

- **Admin**: respostas analíticas, precisas e orientadas a resultados — adequadas a contextos técnicos e administrativos.
- **Professor**: explicações detalhadas e fundamentadas, com foco em clareza conceptual e contextualização pedagógica.
- **Aluno**: linguagem acessível, estruturada e motivacional, centrada na compreensão e na aplicação prática.

Em síntese, o sistema demonstrou uma capacidade de **personalização semântica eficaz**, produzindo respostas diferenciadas e adequadas a cada tipo de utilizador, sem perda de coerência factual. Os parâmetros adotados ($\mathbf{k = 5}$ e **temperature** $= \mathbf{0.4}$) revelaram-se estáveis para este comportamento, confirmando também a robustez do valor **temperature** $= \mathbf{0.5}$ identificado no teste anterior como ponto ótimo para os três perfis.

4.3 Análise de Limitações e Desafios

Apesar dos resultados positivos e da estabilidade geral observada nos testes experimentais, foram identificadas várias limitações e desafios que devem ser considerados em futuras iterações do sistema.

- **Limites dos modelos de linguagem (LLM):** embora os modelos avaliados (Gemini, GPT e LLaMA) tenham apresentado boa coerência e precisão factual, verificou-se variação significativa no estilo discursivo e na profundidade das respostas, especialmente em valores de *temperature* mais elevados. O modelo *Gemini 2.5 Flash*, utilizado como referência em grande parte dos testes, revelou excelente equilíbrio entre fluidez, precisão factual e tempo de resposta, embora algumas respostas apresentassem simplificação excessiva em contextos analíticos mais complexos. Por contraste, modelos mais robustos como o *GPT-4o-mini* demonstraram maior consistência semântica, mas com latência superior e custos potenciais mais elevados.
- **Escalabilidade do RAG:** o mecanismo de recuperação ainda realiza reindexações completas sempre que há atualização de conteúdos, não suportando indexação incremental. Esta limitação tem impacto direto no desempenho de sistemas educativos

com grandes volumes de documentos, como repositórios de cursos Moodle. Além disso, verificou-se que o *Chroma* apresenta maior riqueza contextual nas respostas, mas com latências mais elevadas do que o *Qdrant*, o que reforça a necessidade de um mecanismo híbrido ou adaptativo.

- **Métricas e monitorização:** embora o painel administrativo disponibilize indicadores detalhados sobre latência, carga do sistema e utilização por perfil, nem todas as métricas distinguem automaticamente o backend ativo (*Chroma* ou *Qdrant*). Esta limitação dificulta análises temporais mais granulares e sugere a necessidade de instrumentação adicional para correlação direta entre tipo de operação, modelo e repositório vetorial utilizado.

- **Adaptação por perfil de utilizador:** os testes demonstraram diferenças claras no estilo e complexidade das respostas conforme o contexto (*Admin*, *Professor*, *Aluno*). No entanto, a diferenciação ainda depende de *prompts* estáticos, não existindo um mecanismo dinâmico de personalização contínua com base no histórico de interações ou preferências do utilizador.

- **Segurança e privacidade:** os dados de utilização e os registos de logs armazenam temporariamente identificadores de utilizador e conteúdo textual. Para utilização institucional, será necessário implementar anonimização automática e políticas de retenção compatíveis com o *Regulamento Geral sobre a Proteção de Dados* (RGPD).

- **Dependência e custos operacionais:** a utilização de APIs proprietárias (Gemini e OpenAI) implica custos financeiros associados ao processamento por token e à manutenção de chaves de acesso, o que pode tornar a operação contínua inviável em contextos educativos com grande volume de consultas.[1] Embora estas soluções ofereçam elevada disponibilidade e desempenho consistente, representam uma dependência direta de infraestruturas externas e de modelos comerciais sujeitos a políticas de preço variáveis.

  Alternativas locais, como *Ollama* ou *Mistral*, reduzem esta dependência e permitem maior controlo sobre os dados, mas requerem recursos computacionais significativos — nomeadamente GPUs com elevada capacidade de memória (>=16 GB VRAM) e sistemas otimizados para inferência paralela.[2] A execução local de modelos de grande dimensão aumenta o consumo energético, exige configurações especializadas e reduz a escalabilidade do sistema em ambientes com hardware limitado. Assim, a escolha entre soluções *cloud* e locais deve equilibrar custo, privacidade e capacidade de processamento disponível.

Apesar destas limitações, os resultados obtidos confirmam a viabilidade técnica e educacional da arquitetura proposta. O sistema demonstrou consistência na recuperação e geração de conteúdos, boa estabilidade operacional e potencial de adaptação a diferentes contextos de ensino e perfis de utilizador. Estes resultados evidenciam que é possível desenvolver um ecossistema de apoio ao ensino baseado em IA que seja modular, configurável e alinhado com princípios de transparência e ética digital.

[1] Preços atualizados disponíveis em: `https://ai.google.dev/gemini-api/docs/pricing?hl=pt-br` e `https://chatgpt.com/pricing` (consultado em 2025-10-10).

[2] Modelos e respetivos requisitos técnicos podem ser consultados em: `https://huggingface.co/models` (consultado em 2025-10-10).

4.3.1 Sustentabilidade Operacional

A sustentabilidade do sistema RAG–LLM depende diretamente do equilíbrio entre custos operacionais, desempenho computacional e autonomia tecnológica. Durante a fase experimental, observou-se que a utilização de modelos em *cloud* — nomeadamente o *Gemini 2.5 Flash* e o *GPT-4o-mini* — assegurou tempos de resposta consistentes e elevada estabilidade. Contudo, os custos cumulativos associados ao processamento por token podem tornar-se significativos em cenários de uso intensivo, como cursos com dezenas de alunos ou interações diárias.[3]

Por outro lado, a execução local de modelos através de frameworks como o *Ollama* elimina os custos por token e reforça o controlo sobre os dados, mas requer infraestrutura dedicada. Mesmo modelos de tamanho médio (7B–13B parâmetros) demandam GPUs com grande capacidade de memória e armazenamento rápido, o que limita a adoção em instituições com recursos restritos.[4] Além disso, o tempo de inferência local tende a ser mais elevado, o que pode comprometer a experiência do utilizador em contextos interativos, sobretudo quando a carga simultânea é elevada.

Do ponto de vista da sustentabilidade económica e energética, este trabalho defende a adoção de um modelo híbrido, no qual modelos locais sejam utilizados para tarefas básicas de recuperação ou sumarização, e os modelos em *cloud* sejam reservados a operações mais complexas e de maior valor pedagógico. Esta abordagem oferece um compromisso eficiente entre custo, desempenho e soberania de dados, promovendo uma integração responsável da IA generativa em ambientes educativos.

Com a constante atualização dos modelos e a variação das políticas de preços das APIs comerciais, recomenda-se que esta estratégia seja periodicamente reavaliada, de modo a garantir a viabilidade e a sustentabilidade a longo prazo do ecossistema proposto.

[3] Preços atualizados disponíveis em: `https://ai.google.dev/gemini-api/docs/pricing?hl=pt-br` e `https://chatgpt.com/pricing` (consultado em 2025-10-10).

[4] Modelos e respetivos requisitos técnicos podem ser consultados em: `https://huggingface.co/models` (consultado em 2025-10-10).

Capítulo 5

Conclusão

5.1 Conclusões Principais

Este trabalho demonstrou a viabilidade técnica e pedagógica da integração entre o *Moodle* e modelos de linguagem de grande escala (*Large Language Models* – LLMs), através do***MCP*** e de um pipeline Retrieval-Augmented Generation (RAG) otimizado para conteúdos académicos. A arquitetura proposta revelou-se modular, extensível e adaptável a diferentes contextos educativos, permitindo a construção de um ecossistema digital coeso no qual a inteligência artificial atua como mediadora ativa do processo de aprendizagem.

Os resultados obtidos validam a abordagem em três dimensões essenciais: (i) **eficiência operacional**, com redução significativa do tempo necessário para a criação de materiais pedagógicos; (ii) **qualidade do conteúdo gerado**, sustentada pela recuperação semântica precisa e pelo uso de *prompts* especializados; e (iii) **robustez técnica**, comprovada pela integração estável entre o servidor **MCP**, o *Moodle* e as bases vetoriais *Qdrant* e *Chroma*.

A interface web desenvolvida em Flask consolidou estes resultados, ao integrar autenticação via *Moodle*, gestão de sessões e conversas, seleção dinâmica de modelos (*Gemini*, *OpenAI* e *Ollama*) e um painel interativo de métricas de desempenho. O sistema *Dialogus* demonstrou que é possível automatizar tarefas pedagógicas repetitivas — como a elaboração de resumos, questionários e *flashcards* — mantendo a coerência e o alinhamento com os materiais originais do curso.

Em síntese, este projeto contribui para a modernização das plataformas de gestão da aprendizagem, promovendo uma utilização responsável e pedagógica da inteligência artificial generativa e estabelecendo uma base sólida para a investigação futura em ecossistemas educativos inteligentes.

5.2 Limitações do Estudo

Apesar dos resultados positivos, foram identificadas limitações de natureza técnica e operacional que revelam os limites atuais da integração entre LLMs e plataformas *LMS*.

Uma limitação relevante relaciona-se com a **impossibilidade de efetuar o *upload* automático dos questionários gerados** diretamente para o *Moodle*. Apesar de múltiplas

tentativas e da análise detalhada da documentação da API, o comportamento do *endpoint* de importação de *quizzes* revelou-se inconsistente e insuficientemente documentado. Esta limitação expõe a necessidade de um reforço da interoperabilidade entre o *Moodle* e sistemas externos, sobretudo no que respeita ao suporte de dados estruturados em formato JSON.

Outra limitação identificada prende-se com a **dependência de serviços pagos de inferência**, em particular do *OpenAI*, uma vez que já não existem planos com créditos gratuitos disponíveis. Este fator evidencia a questão da **sustentabilidade económica** das soluções baseadas em IA generativa e reforça a pertinência de explorar alternativas de menor custo, como modelos de código aberto executados localmente.

Por fim, não foi possível concluir a **implementação da ferramenta de geração de vídeos educativos**, devido ao custo elevado e à complexidade associada aos modelos multimodais disponíveis (como o *Gemini Veo* ou o *RunwayML*). Esta limitação ilustra os desafios atuais da integração de IA multimodal em ambientes educativos, onde o equilíbrio entre qualidade, custo e desempenho continua a ser difícil de alcançar.

Estas limitações não invalidam os resultados obtidos; pelo contrário, ajudam a clarificar as fronteiras tecnológicas e económicas do sistema, fornecendo uma base crítica para evoluções futuras.

5.3 Trabalho Futuro

As perspetivas de desenvolvimento futuro organizam-se em três eixos complementares: técnico, pedagógico e institucional.

No plano técnico, destaca-se a **expansão do ecossistema de modelos** e de mecanismos de recuperação. A integração de novos LLMs, como o *Claude*, o *Gemma* ou o *Mistral*, bem como a experimentação de alternativas RAG com diferentes estratégias de *retrieval* e indexação, permitirá comparar custos, latência e adequação semântica.

Em paralelo, é recomendada a **criação de novas ferramentas MCP** orientadas para o ensino, nomeadamente uma *tool* de geração de vídeos educativos baseada em conteúdos do curso e outra de apoio à personalização de percursos de aprendizagem, utilizando *learning analytics* e perfis dinâmicos de estudantes.

No plano institucional, propõe-se o reforço da **integração administrativa com o Moodle**, permitindo que o sistema — quando configurado com as devidas permissões — possa criar cursos, registar utilizadores e sincronizar recursos automaticamente. Esta evolução deve, contudo, manter a filosofia modular do *Dialogus*, garantindo que a camada de inteligência pedagógica continua independente da gestão administrativa da plataforma.

5.4 Contributos Finais

Esta dissertação apresenta um contributo tangível para o avanço dos ambientes de aprendizagem digitais, demonstrando que é possível combinar *Machine Learning*, engenharia de software e princípios pedagógicos num sistema coerente e funcional.

O desenvolvimento do *Dialogus* não se limitou à integração técnica entre o *Moodle*, o **MCP** e os modelos de linguagem de grande escala. A arquitetura foi concebida de forma

a sustentar, de modo explícito, os princípios do *Challenge-Based Learning* (CBL), integrando tecnologia e pedagogia para apoiar as três fases do modelo — *Engage*, *Investigate* e *Act*. Na fase de *Engage*, o sistema permite contextualizar os desafios e promover a reflexão inicial através da interação orientada por modelos de linguagem, apoiada por recursos disponibilizados no Moodle. Durante a fase de *Investigate*, o módulo **RAG** fornece mecanismos de pesquisa e recuperação semântica que estimulam a exploração autónoma e colaborativa de informação relevante. Por fim, na fase de *Act*, as ferramentas de geração de conteúdos — como questionários, planos de estudo e resumos — auxiliam o estudante na criação de soluções fundamentadas, transformando o conhecimento adquirido em ação concreta. Assim, o *Dialogus* materializa os princípios do CBL através de uma integração coerente entre infraestrutura tecnológica e intencionalidade pedagógica, reforçando o papel ativo do estudante no processo de aprendizagem.

Neste contexto, o *Dialogus* atua como mediador tecnológico do CBL, ao automatizar a geração de materiais, sugerir leituras relevantes, criar questionários e sintetizar conteúdos de apoio. Estas funcionalidades ampliam a capacidade do docente em conceber experiências de aprendizagem contextualizadas e adaptadas aos diferentes perfis de estudantes, reduzindo simultaneamente o esforço associado a tarefas repetitivas. Assim, a inteligência artificial deixa de ser apenas uma ferramenta auxiliar e passa a integrar-se como um agente pedagógico ativo no processo de ensino-aprendizagem.

Para além da sua relevância técnica, o trabalho destaca-se pela articulação entre eficiência tecnológica e utilidade educacional, propondo uma visão de futuro em que os sistemas de apoio à docência não substituem o professor, mas potenciam a sua capacidade de criar ambientes de aprendizagem personalizados, sustentados e baseados em evidência.

O *Dialogus* constitui, assim, um passo sólido na transição para ecossistemas educativos inteligentes e colaborativos, reforçando a importância da integração ética, económica e tecnicamente sustentável da inteligência artificial no ensino superior e consolidando a ligação entre inovação tecnológica e metodologias pedagógicas ativas como o CBL.

Capítulo 6

Resultados Científicos

Este capítulo reúne os artigos científicos desenvolvidos no âmbito desta dissertação. Cada publicação é brevemente descrita e integra a ligação para a respetiva versão publicada, de forma a permitir o seu acesso e consulta integral. Estas contribuições representam um avanço relevante na investigação sobre a integração de inteligência artificial e metodologias ativas de aprendizagem em plataformas digitais de ensino superior.

As publicações abrangem diferentes vertentes do problema, desde a revisão do estado da arte e a análise de limitações dos sistemas de gestão de aprendizagem tradicionais, até à proposta e validação de soluções baseadas em protocolos de interoperabilidade e ***LMS***. A abordagem adotada permitiu não só aprofundar o enquadramento teórico e tecnológico da aprendizagem aumentada por inteligência artificial, como também demonstrar a aplicabilidade prática das soluções concebidas em contextos reais de ensino.

Assim, os artigos aqui apresentados constituem a base científica e experimental da presente dissertação, refletindo a evolução e maturação da investigação desenvolvida. Em conjunto, evidenciam uma trajetória consistente de exploração, modelação e validação de sistemas educativos inteligentes, alinhados com os princípios de inovação, personalização e sustentabilidade pedagógica que orientam este trabalho.

6.1 Artigo 1 - *Challenge-Based Learning: A Technology Perspective*

Revista: International Journal of Technological Learning, Innovation and Development
Data de publicação prevista: Agosto 2026
Link: `https://doi.org/10.1504/ijtlid.2026.10071046`
Autores: Ricardo Silva, Óscar Oliveira, Bruno Oliveira [40]

O artigo aborda a ausência de uma estrutura tecnológica clara e replicável para a implementação do ***CBL***. Os autores identificam como principais limitações a escassa documentação técnica sobre as ferramentas digitais usadas, a dificuldade de docentes e estudantes em operacionalizar o modelo com apoio tecnológico, a carência de estudos empíricos de larga escala que avaliem o impacto do **CBL**, e a persistente desigualdade digital que limita a sua adoção. Além disso, destaca-se a integração insuficiente entre pedagogia e

tecnologia, resultando em práticas fragmentadas e de difícil replicação. O artigo propõe uma abordagem sistemática que mapeia tecnologias a cada fase do **CBL**, enfatizando o uso de ferramentas colaborativas, plataformas digitais de investigação e a necessidade de políticas públicas e formação docente que sustentem a adoção do modelo.

6.2 Artigo 2 - *Leveraging Moodle to Support the Challenge Based Learning Framework*

Revista submetida:Educational Technology Research and Development
Autores: Ricardo Silva, Óscar Oliveira, Bruno Oliveira [98]

O artigo posiciona-se na intersecção entre tecnologia educacional e metodologias ativas, examinando a dificuldade de operacionalizar o CBL em plataformas digitais concebidas para aprendizagem linear, como o Moodle. Os autores identificam três dimensões críticas: (i) a pedagógica, marcada pela dificuldade de aplicar as fases "Engage–Investigate–Act" e pela ausência de estratégias de avaliação adequadas; (ii) a tecnológica, relacionada com a falta de ferramentas e interoperabilidade com sistemas externos; e (iii) a organizacional, que inclui resistência institucional e necessidade de formação docente. Propõe-se um mapeamento entre as fases do **CBL** e as funcionalidades do Moodle, sugerindo a integração de ferramentas externas via LTI, SCORM e xAPI, bem como a criação de um repositório público de apoio técnico (`https://github.com/RicardoKermit/CBL-Tech-Tools.git`). O artigo conclui que o Moodle pode servir como infraestrutura-base para o CBL, desde que configurado estrategicamente, complementado por ferramentas colaborativas e sustentado por políticas institucionais de inovação pedagógica.

6.3 Artigo 3 - *Empowering Digital Learning Through MCP and Moodle Integration*

Revista submetida: Educational Technology Research and Development
Autores: Ricardo Silva, Óscar Oliveira, Bruno Oliveira [99]

Este trabalho analisa as limitações dos ***LMS***, como o Moodle, na criação de experiências de aprendizagem dinâmicas, personalizadas e interoperáveis. Identifica-se uma fragmentação tecnológica resultante de abordagens isoladas de integração de IA, a falta de personalização adaptativa, a dependência de tarefas manuais por parte dos docentes e as dificuldades na integração de ***LMS*** e bases vetoriais de forma segura e padronizada. O artigo evidencia ainda desafios éticos relacionados com privacidade, fiabilidade e literacia em IA. Como solução, propõe a integração do ***MCP*** com o Moodle, criando um sistema de orquestração entre IA, bases de dados vetoriais e plataformas educativas. Esta arquitetura automatiza a geração de conteúdos (resumos, questionários e recursos multimédia), reduz o esforço docente, permite personalização adaptativa em tempo real e mantém compatibilidade com as infraestruturas existentes.

Capítulo 7

Bibliografia

[1] F. Al-Dhief, A. Nasser, S. Tharikh, H. Nasser, A. Almuslih, M. Albadr, and M. Mohamed, "Review of learning management systems: history, types, advantages, and challenges," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 33, 9 2024.

[2] M. G. R. Grossi, M. Elias, C. M. Chamon, and D. Leal, "The educational potentialities of the virtual learning environments moodle and canvas: a comparative study," *International Journal of Information and Education Technology*, vol. 8, pp. 514–519, 2018.

[3] E. Costello, "Opening up to open source: looking at how moodle was adopted in higher education," *Open Learning: The Journal of Open, Distance and e-Learning*, vol. 28, pp. 187–200, 2013.

[4] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," 2024.

[5] K. Sharma, P. Kumar, and Y. Li, "Og-rag: Ontology-grounded retrieval-augmented generation for large language models," 2024.

[6] A. Singh, A. Ehtesham, S. Kumar, and T. T. Khoei, "Agentic retrieval-augmented generation: A survey on agentic rag," 2025.

[7] Z. Wang, S. X. Teo, J. Ouyang, Y. Xu, and W. Shi, "M-rag: Reinforcing large language model performance through retrieval-augmented generation with multiple partitions," 2024.

[8] M. Hadžimehmedagić and A. Akbarov, *Traditional vs modern teaching methods: Advantages and disadvantages*, vol. 129. Cambridge Scholars Publishing, 2014.

[9] M. Nichols and K. Cator, *Challenge Based Learning*. Apple Inc., 2008.

[10] M. Nichols, K. Cator, and M. Torres, *Challenge Based Learner User Guide*. Digital Promise, 2016.

[11] A. Fernández, B. Gómez, K. Binjaku, and E. K. Meçe, "Digital transformation initiatives in higher education institutions: A multivocal literature review," *Education and Information Technologies*, vol. 28, pp. 12351–12382, 2023.

[12] K. I. B. Qolamani and M. M. Mohammed, "The digital revolution in higher education: Transforming teaching and learning," *QALAMUNA: Jurnal Pendidikan, Sosial, Dan Agama*, vol. 15, pp. 837–846, 2023.

[13] M. M. Amin, S. N. W. Shamsuddin, and W. M. A. F. W. Hamzah, "Exploring learning engagement factors influencing behavioral, cognitive, and emotional engagement and challenges in learning management systems," *The International journal of Multimedia & Its Applications*, 2025.

[14] K. Spriggs, M. C. Lau, and K. Passi, "Personalizing education through an adaptive lms with integrated llms," 2025.

[15] Z.-Y. Liu, N. Lomovtseva, and E. Korobeynikova, "Online learning platforms: Reconstructing modern higher education," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 15, p. 4, 8 2020.

[16] H. coates, R. james, and G. baldwin, "A critical examination of the effects of learning management systems on university teaching and learning," *Tertiary Education and Management*, vol. 11, pp. 19–36, 2005.

[17] N. Shandra, "Moodle as an open source educational platform in teaching foreign languages," *Prospects and Innovations of Science*, 9 2024.

[18] Instructure Inc., "Canvas lms — cloud-based learning management system," 2024. Accessed: 10 October 2025.

[19] Open edX Project, "About the open edx platform — powered by edx and mit," 2024. Accessed: 10 October 2025.

[20] Google for Education, "Google classroom — a free and easy tool for learning," 2024. Accessed: 10 October 2025.

[21] M. Portuguez Castro and M. G. Gómez Zermeño, "Challenge based learning: Innovative pedagogy for sustainability through e-learning in higher education," *Sustainability*, vol. 12, no. 10, 2020.

[22] G. García-Murillo, P. Novoa-Hernández, and R. S. Rodríguez, "Technological satisfaction about moodle in higher education—a meta-analysis," *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, vol. 15, pp. 281–290, 2020.

[23] I. Katsaris and N. Vidakis, "Adaptive e-learning systems through learning styles: A review of the literature," *Advances in Mobile Learning Educational Research*, vol. 1, 10 2021.

[24] M. K. Ishak and N. B. Ahmad, "Enhancement of learning management system with adaptive features," in *2016 Fifth ICT International Student Project Conference (ICT-ISPC)*, pp. 37–40, 2016.

[25] A. Pérez-Suay, S. V. Vaerenbergh, P. D. Diago, A. B. Pascual-Venteo, and F. J. Ferri, "Data-driven modeling through the moodle learning management system: An empirical study based on a mathematics teaching subject," *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, vol. 18, pp. 19–27, 2023.

[26] A. A. A. Ahmed and A. Ganapathy, "Creation of automated content with embed-

ded artificial intelligence: A study on learning management system for educational entrepreneurship," 2021.

[27] P. Gómez-Abajo, E. Guerra, and J. de Lara, "Automated generation and correction of diagram-based exercises for moodle," *Computer Applications in Engineering Education*, vol. 31, pp. 1845–1866, 2023.

[28] R. Prasetya and N. Nugraha, "English educational moodle-based environment on actualizing personalization virtual course," *Premise Journal of English Education*, vol. 11, p. 362, 9 2022.

[29] I. S. Mintii, S. V. Shokaliuk, T. A. Vakaliuk, O. V. Merzlykin, and M. M. Mintii, "Development of a standard moodle course to optimize the teacher's work in distance education," *Universal Journal of Educational Research*, vol. 8, no. 12, pp. 6659–6666, 2020.

[30] D. Kaleci, "Integration and application of artificial intelligence tools in the moodle platform: A theoretical exploration," *Journal of Educational Technology and Online Learning*, vol. 8, pp. 100–111, 2025.

[31] M. Aakeshova and G. Aripzhan, "Structure and peculiarities of the challenge based learning approach," *Yasaiu University Reporter*, vol. 120, pp. 160–172, 06 2021.

[32] M. Nichols, K. Cator, and M. Torres, *Challenge Based Learning Guide*. Digital Promise, 11 2016.

[33] B. Nicholls, "Challenge based learning: A real-world approach for secondary students to solve complex problems using geoscience knowledge and skills," *Terrae Didatica*, vol. 14, pp. 369–372, 10 2018.

[34] S. Perna, M. Recke, and M. Nichols, "Challenge based learning: A comprehensive survey of the literature," *The Challenge Institute*, 2005 2023.

[35] EduTrends, *Challenge Based Learning*. Educational Innovation do Tecnológico de Monterrey, 10 2015.

[36] D. Gibson, L. Irving, and K. Scott, "Technology-enabled challenge-based learning in a global context," *Collaborative learning in a global world*, pp. 450–450, 2018.

[37] W. Gaskins, J. Johnson, C. Maltbie, and A. Kukreti, "Changing the learning environment in the college of engineering and applied science using challenge based learning," *International Journal of Engineering Pedagogy (iJEP)*, vol. 5, p. 33–41, 02 2015.

[38] M. Z. I. Nizami, V. W. Xue, A. W. Y. Wong, O. Y. Yu, C. Yeung, and C. H. Chu, "Challenge-based learning in dental education," *Dentistry Journal*, vol. 11, no. 1, 2023.

[39] E. Vilalta-Perdomo, J. Membrillo-Hernández, R. Michel-Villarreal, G. Lakshmi, and M. Martínez-Acosta, "*Introduction–The Lay of the Land*," in *The Emerald Handbook of Challenge Based Learning*, pp. 1–11, Emerald Publishing Limited, 2022.

[40] R. Silva, O. Oliveira, and B. Oliveira, "Challenge-based learning: A technology perspective," *International Journal of Technological Learning, Innovation and Development (IJTLID)*, 2026. Published.

[41] R. Queiros and J. Leal, "Using the learning tools interoperability framework for lms integration in service oriented architectures," 10 2011.

[42] A. Parmar, "Paper review on sharable content object reference model (scorm): Framework for e-learning standard," *2012 Second International Conference on Advanced Computing & Communication Technologies*, pp. 409–411, 2012.

[43] K. C. Lim, "Case studies of xapi applications to e-learning," 2016.

[44] P. V. Vieira, A. Costa, and A. L. A. Raabe, "Scorm x common cartridge: Um estudo comparativo," *Anais do Computer on the Beach*, vol. 3, pp. 61–70, 2012.

[45] M. Bures and I. Jelinek, "Using aicc to create reusable adaptive hypermedia e-learning content," in *2005 International Conference on Cyberworlds (CW'05)*, pp. 4 pp.–391, 2005.

[46] S. P. Vickers, "Ims learning tools interoperability a briefing paper,"

[47] D. Bigler, J. Manz, K. Lee, D. Fischer, and G. Hagel, "Learning environment interoperability in software engineering education," in *2024 36th International Conference on Software Engineering Education and Training (CSEE&T)*, pp. 1–5, 2024.

[48] G. Costagliola, F. Ferrucci, and V. Fuccella, "Scorm run-time environment as a service," in *Proceedings of the 6th international conference on Web engineering*, pp. 103–110, 2006.

[49] S. Pekkola and H.-M. Järvinen, "Tracking learning experiences with xapi," 2023.

[50] J. Szepesi, "E-learning szabványok összehasonlítása: Aicc, scorm, xapi, cmi5," *Tudományos és Műszaki Tájékoztatás*, vol. 67, no. 5, pp. 280–285, 2020.

[51] B. Dong, J. Bai, T. Xu, and Y. Zhou, "Large language models in education: A systematic review," in *2024 6th International Conference on Computer Science and Technologies in Education (CSTE)*, pp. 131–134, 2024.

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[53] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (K. Erk and N. A. Smith, eds.), (Berlin, Germany), pp. 1715–1725, Association for Computational Linguistics, Aug. 2016.

[54] Cloudflare, "What are embeddings?," 2024. Acedido em 26 de setembro de 2025.

[55] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, "What does BERT look at? an analysis of BERT's attention," in *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP* (T. Linzen, G. Chrupała, Y. Belinkov, and D. Hupkes, eds.), (Florence, Italy), pp. 276–286, Association for Computational Linguistics, Aug. 2019.

[56] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018.

[57] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, and ..., "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[58] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2019.

[59] H. Saiedian, "Leveraging large language models in education: Enhancing learning and teaching," in *2023 ASEE Midwest Section Conference*, no. 10.18260/1-2-119-46353, (University of Nebraska-Lincoln, Lincoln, Nebraska), ASEE Conferences, July 2024. https://peer.asee.org/46353.

[60] Majidah, G. Rullyana, and R. Triandari, "Google gemini as a learning assistant: Exploring student perceptions," *Jurnal PAJAR (Pendidikan dan Pengajaran)*, vol. 9, pp. 163–172, 8 2025.

[61] X. Zhang, X. Zhang, and H. Liu, "Reflections on enhancing higher education classroom effectiveness through the introduction of large language models," *Journal of Modern Educational Research*, vol. 3, p. 19, 2024. Open Access.

[62] M. Imran and N. Almusharraf, "Google gemini as a next generation ai educational tool: a review of emerging educational technology," *Smart Learning Environments*, vol. 11, p. 22, 5 2024.

[63] K. Z. Zhou, Z. Kilhoffer, M. R. Sanfilippo, T. Underwood, E. Gumusel, M. Wei, A. Choudhry, and J. Xiong, ""the teachers are confused as well": A multiple-stakeholder ethics discussion on large language models in computing education," 2024.

[64] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," 2021.

[65] M. Renze, "The effect of sampling temperature on problem solving in large language models," in *Findings of the Association for Computational Linguistics: EMNLP 2024* (Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, eds.), pp. 7346–7356, Association for Computational Linguistics, 11 2024.

[66] M. Peeperkorn, T. Kouwenhoven, D. Brown, and A. Jordanous, "Is temperature the creativity parameter of large language models?," 2024.

[67] M. Cheng, Y. Luo, J. Ouyang, Q. Liu, H. Liu, L. Li, S. Yu, B. Zhang, J. Cao, J. Ma, D. Wang, and E. Chen, "A survey on knowledge-oriented retrieval-augmented generation," 2025.

[68] S. Yumuşak, "An information-theoretic framework for retrieval-augmented generation systems," *Electronics*, vol. 14, 2025.

[69] X. Xie, H. Liu, W. Hou, and H. Huang, "A brief survey of vector databases," in *2023 9th International Conference on Big Data and Information Analytics (BigDIA)*, pp. 364–371, 2023.

[70] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[71] Y. Huang and J. Huang, "A survey on retrieval-augmented text generation for large language models," 2024.

[72] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (A. Moschitti, B. Pang, and W. Daelemans, eds.), (Doha, Qatar), pp. 1532–1543, Association for Computational Linguistics, Oct. 2014.

[73] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," 2019.

[74] Sentence-Transformers Team, "all-minilm-l6-v2." `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`, 2021. Accessed: 12 October 2025.

[75] Sentence-Transformers Team, "all-mpnet-base-v2." `https://huggingface.co/sentence-transformers/all-mpnet-base-v2`, 2021. Accessed: 12 October 2025.

[76] E. Öztürk and A. Mesut, "Performance analysis of chroma, qdrant, and faiss databases," *UNITECH – SELECTED PAPERS*, 2024.

[77] Pinecone Systems, Inc., "Pinecone vector database documentation." `https://docs.pinecone.io/`, 2024. Accessed: 12 October 2025.

[78] Semi Technologies, "Weaviate documentation." `https://docs.weaviate.io/weaviate`, 2025. Accessed: 12 October 2025.

[79] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The faiss library," 2025.

[80] F. Pan, Q. Zhou, W. Guo, and H. Yang, "A survey on retrieval-augmented generation in applications of education and teaching," in *2025 7th International Conference on Computer Science and Technologies in Education (CSTE)*, pp. 803–807, 2025.

[81] J. Swacha and M. Gracel, "Retrieval-augmented generation (rag) chatbots for education: A survey of applications," *Applied Sciences*, vol. 15, 2025.

[82] S. Dakshit, "Faculty perspectives on the potential of rag in computer science higher education," in *Proceedings of the 25th Annual Conference on Information Technology Education*, pp. 19–24, Association for Computing Machinery, 2024.

[83] S. Zhang, R. Ma, Y. Ma, S. Li, Y. Xu, X. Yi, and H. Li, "Understanding users' privacy perceptions towards llm's rag-based memory," 2025.

[84] M. C. Protocol, "Getting started with the model context protocol," 2025. Accessed: 2025-08-17.

[85] X. Hou, Y. Zhao, S. Wang, and H. Wang, "Model context protocol (mcp): Landscape, security threats, and future research directions," *ArXiv*, vol. abs/2503.23278, 2025.

[86] P. Patil, "Inside the mcp protocol: Revolutionizing data communication and system interoperability," *World Journal of Advanced Research and Reviews*, vol. 26, pp. 3055–3071, 2025.

[87] M. D. Patil and V. V. Lokhande, "Model context protocol (mcp) enabling scalable ai data integration," *International Journal for Multidisciplinary Research (IJFMR)*, vol. 7, p. 1, 4 2025.

[88] Y.-A. Bachiri, H. Mouncif, and B. Bouikhalene, "Artificial intelligence empowers gamification: Optimizing student engagement and learning outcomes in e-learning and moocs," *International Journal of Engineering Pedagogy (iJEP)*, vol. 13, pp. 4–19, December 2023. Section: Papers.

[89] A. T. Neumann, Y. Yin, S. Sowe, S. Decker, and M. Jarke, "An llm-driven chatbot in higher education for databases and information systems," *IEEE Transactions on Education*, vol. 68, no. 1, pp. 103–116, 2025.

[90] N. S. Alotaibi, "The impact of ai and lms integration on the future of higher education: Opportunities, challenges, and strategies for transformation," *Sustainability*, vol. 16, no. 23, 2024.

[91] PostgreSQL Global Development Group, "Postgresql 16 documentation." `https://www.postgresql.org/docs/current/datatype-json.html`, 2025. Accessed: 12 October 2025.

[92] Anthropic, "Model context protocol: Technical overview." `https://github.com/modelcontextprotocol`, 2024. Acedido em 2025-10-07.

[93] A. Donvir, P. Yadav, S. Panyam, and R. Joshi, "Leveraging rag for enhanced business intelligence with local llms," *2025 IEEE World AI IoT Congress (AIIoT)*, pp. 0655–0661, 2025.

[94] W. Zhou, Y. Yan, and Q. Yang, "Dgrag: Distributed graph-based retrieval-augmented generation in edge-cloud systems," *ArXiv*, vol. abs/2505.19847, 2025.

[95] Y. Cheng, L. Zhang, J. Wang, M. Yuan, and Y. Yao, "Remoterag: A privacy-preserving llm cloud rag service," *ArXiv*, vol. abs/2412.12775, 2024.

[96] J. Li, C. Xu, L. Jia, F. Wang, C. Zhang, and J. Liu, "Eaco-rag: Towards distributed tiered llm deployment using edge-assisted and collaborative rag with adaptive knowledge update," 2024.

[97] S. Tripathi, D. Alkhulaifat, S. Lyo, R. Sukumaran, B. Li, V. Acharya, R. McBeth, and T. S. Cook, "A hitchhiker's guide to good prompting practices for large language models in radiology.," *Journal of the American College of Radiology : JACR*, vol. 22 7, pp. 841–847, 2025.

[98] R. Silva, O. Oliveira, and B. Oliveira, "Leveraging moodle to support the challenge-based learning framework," *Discover Education, Springer Nature*, 2025. Submetido.

[99] R. Silva, O. Oliveira, and B. Oliveira, "Empowering digital learning through mcp and moodle integration," *Educational Technology Research and Development, Springer Nature*, 2025. Submetido.

Apêndice A

Ferramentas implementadas no Servidor MCP

A Tabela A.1 apresenta a lista completa das ferramentas `@mcp.tool()` desenvolvidas no servidor **MCP**. Estas ferramentas constituem a base operacional do sistema, permitindo a gestão de modelos, integração com o Moodle, geração de conteúdos educativos e monitorização do desempenho.

Ferramenta	Descrição / Função principal
`set_model(model_name)`	Define o modelo de linguagem ativo (Gemini, OpenAI ou Ollama).
`get_current_model()`	Retorna o modelo atualmente selecionado.
`set_rag_backend(backend)`	Define o backend **RAG** (Qdrant ou Chroma).
`get_rag_backend()`	Obtém o backend **RAG** atualmente configurado.
`add_new_pdfs()`	Procura novos ficheiros PDF disponíveis no diretório configurado e adiciona-os ao **RAG**.
`download_and_add_pdf(file_url)`	Faz o *download* de um ficheiro PDF a partir de um URL (geralmente proveniente do Moodle) e envia-o para indexação.
`download_pdfs_from_course(course_fullname)`	Recolhe automaticamente todos os PDFs de um curso Moodle e processa-os para integração no **RAG**.
`clear_rag()`	Limpa a base vetorial e remove todos os embeddings armazenados.
`assign_role(email, role)`	Atribui um papel (Admin, Professor ou Aluno) a um utilizador.
`list_users(limit)`	Lista os utilizadores registados no sistema.
`deactivate_user(email)`	Desativa um utilizador e bloqueia o respetivo acesso.
`practice_quiz(topic)`	Gera exercícios de prática rápida sobre um tema selecionado.
`recommend_reading_material(topic)`	Sugere materiais de leitura complementares (artigos, vídeos, livros).
`analyze_student_queries()`	Analisa padrões de perguntas colocadas pelos alunos para identificar dificuldades frequentes.
`retrieve(prompt)`	Recupera informação diretamente dos PDFs indexados (base de conhecimento).
`generate_quiz_with_difficulty(topic, num_questions, difficulty)`	Gera questionários personalizados com base num tema e nível de dificuldade.
`generate_dev_questions(topic)`	Cria perguntas de desenvolvimento baseadas num tema específico.
`study_plan_generator(topic)`	Cria planos de estudo personalizados com base nos materiais disponíveis.
`generate_lesson_summary(topic)`	Gera resumos automáticos de lições ou documentos.
`interactive_flashcards(topic)`	Cria cartões interativos de revisão para estudo autónomo.
`generate_test(topic)`	Cria testes avaliativos com diferentes níveis de dificuldade.
`export_logs()`	Exporta registos de operação da tabela `operation_logs` para auditoria.
`system_health_check()`	Verifica o estado geral do sistema (base de dados, **RAG**, Moodle, disco).

Tabela A.1: Lista completa de ferramentas (`@mcp.tool()`) implementadas no servidor **MCP**.

Apêndice B

Configuração dos Web Services do Moodle

A Tabela B.1 apresenta as etapas seguidas para configurar a API REST do Moodle 5.0, baseando-se na documentação oficial da plataforma.

Passo	**Estado / Opção**	**Descrição**
1	`Enable web services`	Ativar os serviços web nas `Advanced features`.
2	`Enable protocols`	Selecionar apenas o protocolo REST, desativando SOAP e XML-RPC.
3	`Create a specific user`	Criar um utilizador dedicado para representar o sistema externo (cliente **MCP**).
4	`Check user capability`	Atribuir permissões adequadas ao utilizador (ex.: `webservice/rest:use`).
5	`Select a service`	Criar um novo serviço com as opções "Enable" e "Authorised users" ativadas.
6	`Add functions`	Adicionar as funções REST pretendidas.
7	`Select a specific user`	Associar o utilizador do serviço criado ao novo serviço.
8	`Create a token for a user`	Gerar um token de acesso para o utilizador de serviço.
9	`Enable developer documentation`	Ativar a documentação REST para consulta dos endpoints.
10	`Test the service`	Validar a configuração através do cliente REST interno do Moodle.

Tabela B.1: Etapas de configuração da API REST no Moodle 5.0.

Apêndice C

Código de Alteração e Atualização de Modelos no Servidor MCP

Este apêndice apresenta a implementação integral da ferramenta `set_model()` e da função auxiliar `update_model()` incluídas no ficheiro `MCP_Server_new.py`. Estas funções são responsáveis por atualizar o modelo de linguagem em uso no servidor, garantir a sincronização com o cliente e reconstruir internamente o pipeline de recuperação e geração aumentada (**RAG**) com o novo modelo ativo. A parametrização define o valor de `temperature = 0.4`, assegurando equilíbrio entre consistência, fluência e precisão nas respostas geradas.

```
@mcp.tool()
def set_model(model_name: str) → dict:
    """
    Change the Language Model (LLM) used by the server. This tool should not
    normally be called directly from the wizard.
    """
    if update_model(model_name):
        return {
            "success": True,
            "response": f"Modelo alterado para {ALL_MODELS[model_name]['name']}",
            "details": {"tool": "set_model", "model_name": model_name}
        }
    else:
        return {
            "success": False,
            "response": f"Erro: Modelo '{model_name}' não é válido.",
            "details": {"tool": "set_model", "model_name": model_name, "error": "
    invalid_model"}
        }

def update_model(new_model_name: str) → bool:
    """It updates the Gemini, OpenAI, or Ollama model used by the MCP server."""
    global model, qa, current_model_name
    if new_model_name not in ALL_MODELS:
        return False
    try:
        model_info = ALL_MODELS[new_model_name]
        if model_info["provider"] == "gemini":
            model = GoogleGenerativeAI(model=new_model_name, temperature=0.4)
        elif model_info["provider"] == "openai":
```

```
            model = ChatOpenAI(
                model=new_model_name,
                api_key=os.getenv("OPENAI_API_KEY"),
                temperature=0.4
            )
        elif model_info["provider"] == "ollama":
            from langchain_community.chat_models import ChatOllama
            model = ChatOllama(
                model=new_model_name,
                base_url="http://localhost:11434",
                temperature=0.4
            )

        # RAG pipeline reconstruction
        qa = RetrievalQA.from_chain_type(
            llm=model,
            chain_type="stuff",
            retriever=retriever,
            chain_type_kwargs={"prompt": custom_prompt},
        )

        current_model_name = new_model_name
        return True

    except Exception as e:
        print(f"Erro ao atualizar modelo: {e}")
        return False
```

Listagem C.1: Implementação set_model() e update_model().

Esta implementação reforça a modularidade do sistema, permitindo alternar dinamicamente entre modelos de diferentes fornecedores (Gemini, OpenAI e Ollama) sem comprometer a consistência do pipeline. A atualização é efetuada em tempo real, garantindo que tanto o servidor **MCP** como o cliente utilizam o mesmo modelo configurado, o que assegura coerência na geração de respostas e evita discrepâncias entre contextos de execução.

Apêndice D

Interfaces Complementares do Sistema

A Figura D.1 apresenta a versão em *modo claro* da interface principal do cliente **MCP**. Esta versão foi concebida para garantir conforto visual em ambientes com elevada luminosidade, mantendo a mesma estrutura funcional do modo escuro — incluindo histórico de conversas, ligação ao servidor e área de entrada de perguntas. A interface adapta-se automaticamente à preferência do utilizador, podendo ser alternada manualmente através do ícone de modo visual presente na barra superior.

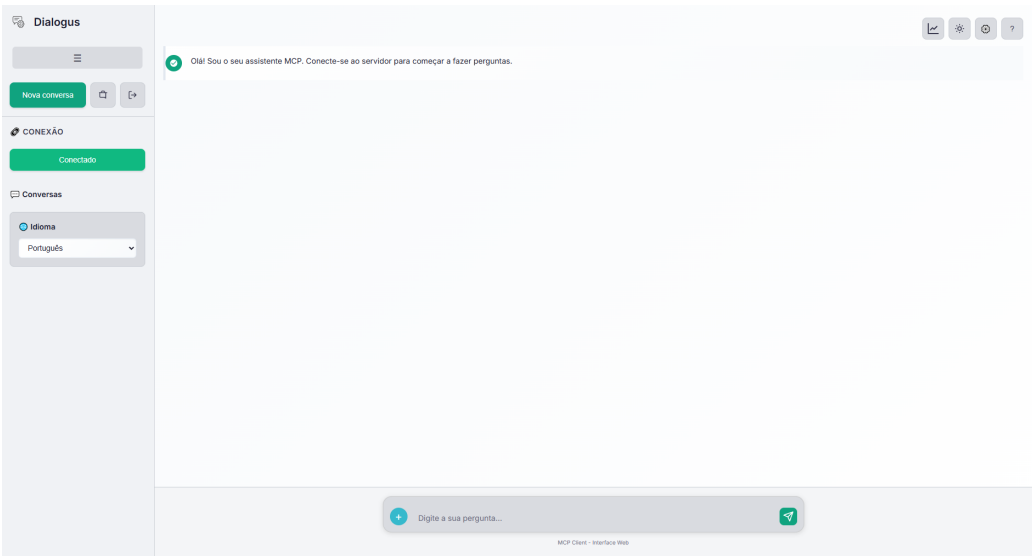
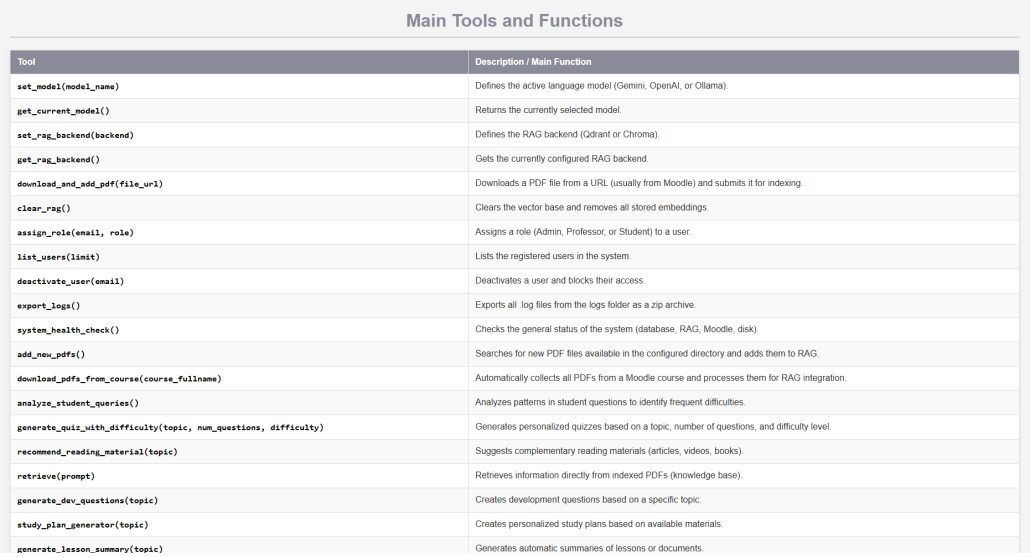


Figura D.1: Interface principal do sistema em modo claro (*Light Mode*).

A Figura D.2 apresenta a lista das ferramentas do MCP a que o utilizador tem *permissão de acesso*.



Main Tools and Functions

Tool	Description / Main Function
`set_model(model_name)`	Defines the active language model (Gemini, OpenAI, or Ollama).
`get_current_model()`	Returns the currently selected model.
`set_rag_backend(backend)`	Defines the RAG backend (Qdrant or Chroma).
`get_rag_backend()`	Gets the currently configured RAG backend.
`download_and_add_pdf(file_url)`	Downloads a PDF file from a URL (usually from Moodle) and submits it for indexing.
`clear_rag()`	Clears the vector base and removes all stored embeddings.
`assign_role(email, role)`	Assigns a role (Admin, Professor, or Student) to a user.
`list_users(limit)`	Lists the registered users in the system.
`deactivate_user(email)`	Deactivates a user and blocks their access.
`export_logs()`	Exports all .log files from the logs folder as a zip archive.
`system_health_check()`	Checks the general status of the system (database, RAG, Moodle, disk).
`add_new_pdfs()`	Searches for new PDF files available in the configured directory and adds them to RAG.
`download_pdfs_from_course(course_fullname)`	Automatically collects all PDFs from a Moodle course and processes them for RAG integration.
`analyze_student_queries()`	Analyzes patterns in student questions to identify frequent difficulties.
`generate_quiz_with_difficulty(topic, num_questions, difficulty)`	Generates personalized quizzes based on a topic, number of questions, and difficulty level.
`recommend_reading_material(topic)`	Suggests complementary reading materials (articles, videos, books).
`retrieve(prompt)`	Retrieves information directly from indexed PDFs (knowledge base).
`generate_dev_questions(topic)`	Creates development questions based on a specific topic.
`study_plan_generator(topic)`	Creates personalized study plans based on available materials.
`generate_lesson_summary(topic)`	Generates automatic summaries of lessons or documents.

Figura D.2: Interface principal do sistema em modo claro (*Light Mode*).

Tal como no modo escuro, o layout segue os princípios de **design responsivo**, assegurando compatibilidade com diferentes resoluções e dispositivos. A consistência cromática entre os dois modos permite preservar a coerência visual e a identidade gráfica da plataforma em todos os contextos de utilização.

Apêndice E

Interfaces Complementares do Sistema

Para avaliar a adaptação contextual do sistema **RAG**, foram definidos três perfis de utilizador representando diferentes papéis num ambiente académico. Cada perfil foi configurado com uma descrição textual explícita, utilizada como *prompt conditioning* antes da execução dos testes experimentais.

- **Admin**: *"I'm an administrator and I want objective answers and statistics on platform usage. Be as analytical as possible and try to discover patterns in what the student has researched, if you are asked to do an analysis."*
- **Professor**: *"I'm a teacher and I want to prepare the best pedagogical, critical, and reference-rich content for my students."*
- **Aluno**: *"I'm a student motivated to learn. I need clear explanations, simple examples, and practical exercises."*

Estas descrições foram utilizadas de forma consistente em todos os testes de adaptação discursiva, assegurando uniformidade na avaliação do comportamento do modelo por tipo de utilizador.

Apêndice F

Visualizações Complementares da Dashboard

As figuras seguintes apresentam indicadores adicionais disponibilizados pela página administrativa, complementando as visualizações principais do Capítulo 4. Para facilitar a leitura comparativa, algumas métricas relacionadas são apresentadas lado a lado.

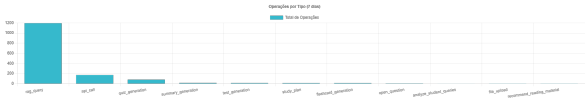


Figura F.1: Distribuição e volume de operações por tipo.

Figura F.2: Utilização relativa dos mecanismos de recuperação (Qdrant vs. Chroma).

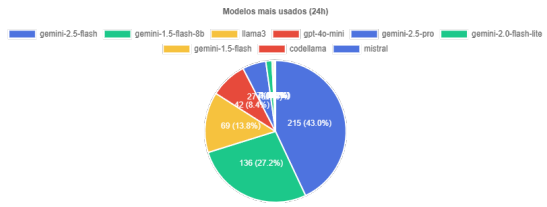


Figura F.3: Distribuição de chamadas por modelo de linguagem (LLM).

Figura F.4: Taxa de erro por modelo de linguagem.

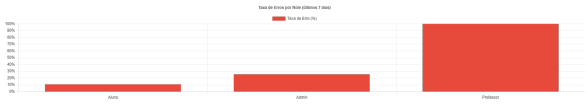


Figura F.5: Erros segmentados por perfil de utilizador.

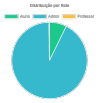


Figura F.6: Distribuição de interações por tipo de utilizador.

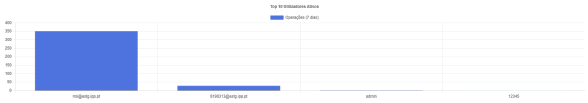


Figura F.7: Utilizadores com maior número de interações.

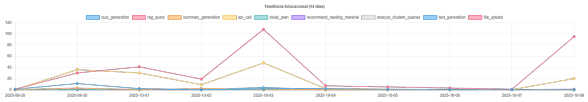


Figura F.8: Tendências educativas e volume de atividades geradas.