UNIVERSIDADE
LUSÓFONA

Servidor MCP para a Integração Fluida entre LLMs e Data Warehouses

Trabalho Final de Curso

Relatório Intercalar 2º Semestre

Fábio Jorge | a22303085
Orientador: Professor João Caldeira

Trabalho Final de Curso | LEI | abril de 2026

Direitos de cópia

Resumo

O MCP Server (Model Context Protocol) constitui uma camada de intermediação rigorosa que viabiliza a integração estruturada entre modelos de linguagem (LLM) e sistemas empresariais, em particular Data Warehouses (DW), garantindo assim a segurança, o controlo e a conformidade no acesso à informação. Este protocolo assegura a tradução precisa das solicitações formuladas pelo LLM em operações devidamente autorizadas — como consultas SQL — promovendo uma gestão robusta e uma utilização responsável dos dados.

Desta forma, o MCP permite que o LLM realize análises, consulte resultados e dê suporte a processos de decisão sem contacto direto com infraestruturas críticas. Esta arquitetura proporciona uma integração fluida, fiável e institucionalmente segura entre Inteligência Artificial e plataformas de dados, contribuindo para uma maior eficiência analítica e segurança organizacional.

Abstract

The MCP Server (Model Context Protocol) constitutes a rigorous mediation layer that enables structured integration between Large Language Models LLM and enterprise systems, particularly Data Warehouses (DW), which in turn ensures greater security, control, and compliance in information access. This protocol guarantees the precise translation of requests formulated by the LLM into duly authorised operations — such as SQL queries — promoting robust management and responsible use of data.

In this way, the MCP allows the LLM to perform analyses, consult results, and support decision-making processes without direct contact with critical infrastructures. This architecture provides a fluid, reliable, and institutionally secure integration between Artificial Intelligence and data platforms, contributing to greater analytical efficiency and organisational security.

Índice

Lista de Figuras

Lista de Tabelas

1 - Identificação do Problema

A democratização dos Modelos de Linguagem (LLM) transformou radicalmente a interação dos humanos com o computador, permitindo a utilização de interfaces de Inteligência Artificial para facilitar diversas tarefas. No entanto, em contexto empresarial, persiste uma lacuna funcional crítica: a incapacidade destes modelos acederem, de forma autónoma e segura, aos sistemas de dados onde se situa a verdade factual da organização — especificamente, os *Data Warehouses* (DW).

Enquanto as organizações modernas acumulam exorbitantes quantidades de dados estruturados em sistemas como PostgreSQL ou Snowflake, o acesso a essa informação permanece restrito a especialistas técnicos autorizados. Este cenário cria um problema bifacetado: por um lado, os gestores de negócio dependem das equipas de dados, muitas vezes sobrecarregadas, para obterem respostas simples.

E por outro lado, as poderosas ferramentas de Inteligência Artificial permanecem pouco relevantes tendo em conta que não têm acesso aos dados reais da empresa, limitando-se a conhecimentos generalistas ou exigindo informações bastante sensíveis, o que poderá gerar graves riscos de segurança e privacidade.

De seguida, descrevem-se os principais problemas estruturais, de segurança e de interoperabilidade que motivam o desenvolvimento deste projeto.

1.1 Contextos Reais

Para contextualizar o problema, apresentam-se desde já exemplos em circunstâncias reais, considerando o cenário típico de uma organização que utiliza um *Data Warehouse* relacional para centralizar os seus dados.

O fluxo de informação da empresa enfrenta atualmente os seguintes constrangimentos críticos:

Latência na Tomada de Decisão: Um analista de *marketing* que pretenda saber, por exemplo, “Qual o produto mais vendido na região norte do país?” não consegue obter a resposta de imediato. Necessita de solicitar esta informação à equipa de Engenharia de Dados ou Business Intelligence, criando um entrave operacional.

Riscos de Privacidade: Existe o perigo de exposição de dados confidenciais a terceiros ao copiar dados para modelos públicos. Adicionalmente, verifica-se a propensão do modelo para ter uma alucinação ou inventar factos quando o contexto excede a sua janela de processamento.

Riscos de Segurança: A possibilidade de autorização a instruções críticas que possam afetar a integridade dos dados, como por exemplo, comandos destrutivos (DROP TABLE) ou alterações não autorizadas que afetem a empresa.

O principal obstáculo abordado neste trabalho reside, portanto, na inexistência de uma interface padronizada e segura que permita conectar a agilidade da Inteligência Artificial à estrutura rígida de um *Data Warehouse*. A solução deve garantir, simultaneamente, que o LLM compreende o esquema de dados e que atue em conformidade com os limites de segurança (operações *read-only* ou Read-Only) [1].

1.2 Relação com os Dados

Atualmente, existe uma enorme lacuna funcional entre a Linguagem Natural (usada pelos humanos e LLMs) e a Linguagem Estruturada SQL necessária para interrogar bases de dados como PostgreSQL ou Snowflake. Os dados críticos de uma empresa — vendas, *stocks*, transações financeiras, etc. — estão fechados em sistemas que exigem conhecimento técnico especializado para serem acedidos. Isto cria um conjunto de operações para diferentes responsáveis, sem necessidade aos dias de hoje.

Quem decide os negócios e fecha contratos, não consegue obter respostas em tempo real ("Qual foi o total de vendas na semana passada?") sem depender de equipas técnicas intermediárias ou de painéis de Business Intelligence estáticos e pré-configurados. [2]

1.3 Limitação das Abordagens RAG Tradicionais

O *Retrieval-Augmented Generation* (RAG) constitui atualmente uma abordagem amplamente utilizada para permitir que modelos de linguagem acedem a informação não estruturada através de pesquisa semântica. Contudo, esta técnica revela limitações significativas quando aplicada a dados estruturados de natureza tabular, típicos de bases de dados relacionais ou *Data Warehouses*.

Ao converter tabelas inteiras em texto para posterior vectorização, o RAG perde a semântica relacional inerente ao modelo de dados — nomeadamente chaves primárias (Primary Key), ligações entre entidades e dependências lógicas. De igual modo, a precisão matemática dos valores numéricos é comprometida, uma vez que não preservam relações nem garantem exatidão aritmética.

Portanto, embora um RAG tradicional seja eficaz a identificar documentos ou a descrever um determinado tema, é incapaz de executar operações analíticas fundamentais, como agregações (somas, médias, contagens), filtros rigorosos, operações de agrupamento ou cálculos determinísticos. [3]

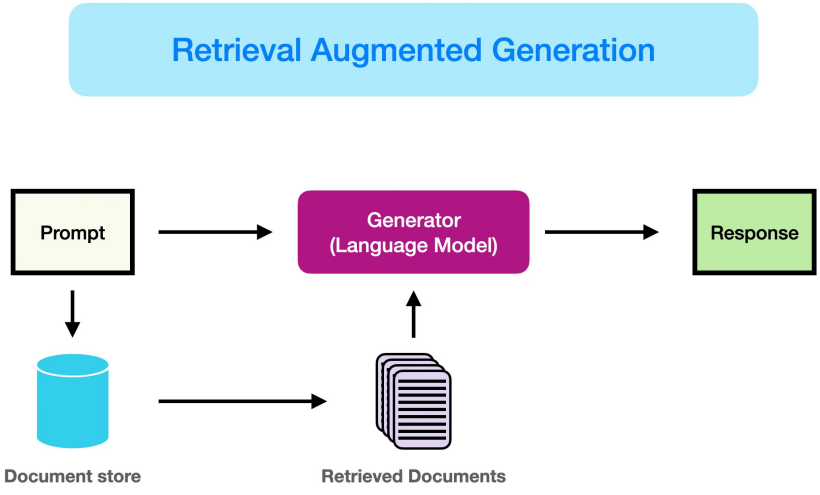

Figure 1.1: Arquitetura *RAG*.

1.4 Riscos de Segurança e Privacidade

A ausência de um protocolo padronizado leva muitas vezes a implementações arriscadas. Uma prática comum, mas perigosa, é a injeção direta de esquemas de base de dados ou excertos de dados sensíveis (CSV/*Excel*) nos *prompts* de modelos públicos. Isto não só viola normas de privacidade, como expõe a estrutura interna da organização a possíveis fugas de informação.

Além disso, dar a um agente de IA acesso direto a uma base de dados sem uma camada de controlo intermédia rigorosa (como a proposta pelo MCP) [1] cria riscos de execução de comandos destrutivos ou não autorizados.

1.5 Erros e Alucinações em Consultas Técnicas

A utilização de modelos de linguagem em contextos onde é necessário gerar consultas técnicas — nomeadamente SQL — expõe um risco significativo: o fenómeno da alucinação. Quando o LLM não dispõe de um mecanismo adequado de introspeção ou de acesso ao esquema real da base de dados, é frequentemente levado a adivinhar ou, até mesmo, inventar nomes de tabelas ou colunas. Esta limitação pode originar consultas sintaticamente válidas mas semanticamente incorretas, como por exemplo referir uma tabela inexistente (*vendas-2025*) quando a entidade correta é (*faturacao-25*).

Para além de erros estruturais, o modelo pode ainda criar métricas fictícias, campos que não existem ou agregações incoerentes, comprometendo totalmente a fiabilidade da resposta. Na ausência de um sistema capaz de validar o esquema em tempo real e de assegurar verificações rigorosas sobre a exatidão da consulta gerada, a adoção de IA em processos de suporte à decisão — sobretudo em domínios financeiros, analíticos ou operacionais — torna-se excessivamente arriscada. [4]

1.6 Custo e Complexidade de Integração (Problema *MxN*)

As organizações deparam-se com o denominado problema de integração *MxN*, resultante da necessidade de conectar M modelos de Inteligência Artificial a N sistemas ou ferramentas de dados distintos. A abordagem tradicional, baseada na criação de *APIs REST* específicas para cada caso de uso — como *get-client*, *get-product* ou outro *endpoint* — revela-se rapidamente insustentável. Cada alteração no esquema da base de dados, bem como uma nova exigência de negócio, obriga à implementação de novos serviços ou à revisão dos existentes, aumentando significativamente os custos de manutenção e a complexidade operacional.

A inexistência de uma interface unificada e independente do modelo de dados conduz, inevitavelmente, a um ciclo repetitivo de desenvolvimento. Esta proliferação de integrações compromete a escalabilidade, aumenta o risco de inconsistências e dificulta a evolução tecnológica da organização.

2 - Benchmarking

Este capítulo apresenta a análise comparativa da solução proposta face às alternativas existentes no mercado, bem como o enquadramento teórico das abordagens de interação entre LLMs e bases de dados.

2.1 Avaliação das Soluções Existentes

Após a análise conceptual, procede-se à avaliação das principais soluções contemporâneas.

A avaliação demonstra que, embora existam avanços significativos, a maioria das soluções permanece limitada em três aspetos fundamentais:

1. **Dependência de modelos específicos:** Comprometendo a interoperabilidade e criando barreiras para melhores resultados;
2. **Ausência de universalidade:** Falta de um mecanismo padrão para ligação a qualquer sistema de dados;
3. **Fiabilidade insuficiente:** Falta de determinismo em operações críticas, onde a margem de erro deve ser nula.

2.2 Abordagens Text-to-SQL: Abstração Progressiva

As soluções existentes podem ser agrupadas em três categorias distintas, representando uma evolução na complexidade e capacidade:

LLM com contexto limitado: O modelo gera SQL apenas com base numa descrição textual do esquema fornecida nos *prompts*. Sofre com alucinações quando o esquema é vasto.

LLM com introspeção parcial: O sistema possui acesso ao esquema (DDL), mas não ao estado real dos dados, o que pode levar a consultas sintaticamente corretas mas semanticamente vazias.

LLM com ferramenta SQL integrada: O modelo recebe o esquema, executa consultas reais através da ferramenta e valida os resultados. Esta abordagem constitui o padrão mais robusto, embora ainda careça de um mecanismo universal de integração — lacuna que o Servidor MCP procura colmatar [5].

2.3 Limitações Detetadas no Mercado Atual

A análise revela limitações transversais às soluções existentes, sustentando a necessidade de um novo paradigma. As principais falhas identificadas são:

- ***Lock-in***: Integração frágil e dependente do fornecedor [6];
- **Falta de Interoperabilidade:** Inexistência de um mecanismo comum para múltiplos modelos de IA;
- **Ausência de Protocolo:** Falta de um padrão uniforme para exposição de capacidades de dados;

- **Baixa Verificabilidade:** Reduzida capacidade de verificar deterministicamente as respostas;
- **Esforço de Integração:** Insuficiente abstração para reduzir o esforço de integração de $M \times N$ (muitos modelos para muitas bases) para $1 \times N$ (um protocolo para muitas bases).

Estas limitações validam a necessidade de um paradigma centrado em Interoperabilidade Semântica e numa utilização fiável de ferramentas.

2.4 Quadro Comparativo

A tabela seguinte sintetiza a posição da solução proposta face às abordagens analisadas:

Table 2.1: Comparação entre Abordagens de Integração IA-Dados

Critério	**Engenharia de *Prompt***	**RAG Vetorial**	**SaaS Proprietário**	**Servidor MCP (Proposto)**
Abstração	Baixa	Média	Proprietária	**Alta (Universal)**
Precisão	Baixa (Alucina)	Média (Sem Agregações)	Alta	**Alta (Determinística)**
Interoperabilidade	Baixa	Média	Nula (*Lock-in*)	**Total (*Open Standard*)**
Complexidade	$M \times N$	$M \times N$	Baixa (Fechado)	$1 \times N$

3 - Viabilidade e Pertinência

Neste capítulo demonstra-se a viabilidade técnica e a relevância económica do Servidor MCP. A análise é baseada em critérios econométricos e tendências de mercado que confirmam a continuidade do projeto além do âmbito académico, respondendo a uma necessidade crítica na adoção empresarial de Inteligência Artificial.

3.1 Pertinência Tecnológica e de Mercado

A pertinência do desenvolvimento de um Servidor MCP genérico justifica-se pelo atual ponto de inflexão no mercado de IA Generativa: a transição de *chatbots* generalistas para agentes autónomos integrados com dados empresariais.

3.2 O Problema da "Última Milha" na Adoção de IA

Estudos recentes indicam que a principal barreira à adoção de LLMs em empresas não é a capacidade dos modelos, mas sim a dificuldade de conectá-los [7] de forma segura aos dados internos (DW). O Servidor MCP aborda este desafio, fornecendo uma infraestrutura que elimina o desenvolvimento de código *ad-hoc* para cada integração. Esta solução permite transformar dados isolados em conhecimento acionável em tempo real. [8]

3.3 Viabilidade Económica e Sustentabilidade

A viabilidade avalia-se não apenas pela exequibilidade técnica, mas também pelo potencial de gerar valor económico e sustentabilidade financeira após o desenvolvimento do TFC em termos académicos.

Redução de Custos Operacionais (Análise Custo-Benefício)

A implementação do Servidor MCP reduz significativamente os custos de engenharia de *software*:

- **Cenário Atual:** Desenvolver *APIs* seguras para expor dados a um LLM demora, em média, semanas de trabalho de engenharia *sénior*.
- **Com o Servidor MCP:** A configuração do adaptador genérico reduz esse tempo para minutos, traduzindo-se em poupança direta de horas e aceleração do *Time-to-Market* (*TTM*) de produtos internos de IA.

3.4 Modelo de Negócio e Continuidade

O Servidor MCP possui potencial de evolução para produto comercial ou *open-source* sustentável, seguindo o modelo *open-core*:

- **Versão Gratuita:** Servidor básico e adaptador para qualquer DW, desenvolvidos neste TFC, promovendo adoção inicial.
- **Versão *Premium*:** Funcionalidades avançadas, como verificação de *queries*, gestão de acessos baseada em funções e *DB-connector* para bases proprietárias (*Oracle*, *SAP*), comercializadas sob outros modelos.

Este modelo garante sustentabilidade financeira e continuidade para além do relatório.

3.5 Análise de Tendências de Mercado

O mercado global de *software* de integração de IA apresenta um crescimento exponencial. A padronização de protocolos, como o MCP, apoiada por entidades como a *Anthropic*, indica que a indústria caminha para soluções interoperáveis em detrimento de integrações proprietárias fechadas. O Servidor MCP alinha-se com esta tendência de descentralização e interoperabilidade.

4 - Solução Proposta

Este capítulo descreve a arquitetura técnica da solução e fundamenta as decisões tomadas na seleção das tecnologias utilizadas. Adicionalmente, mapeiam-se os componentes desenvolvidos às áreas científicas e competências adquiridas ao longo do curso, demonstrando a relevância académica e transversal do projeto.

4.1 Tecnologias e Fundamentação das Opções

O leque de tecnologias foi escolhido com base em critérios de **desempenho assíncrono**, **interoperabilidade com sistemas de IA** e **robustez arquitetural**. A solução adota uma arquitetura modular na qual o Servidor MCP constitui o núcleo da comunicação.

4.1.1 Linguagem de Programação: Python

A implementação do servidor e de toda a lógica do MCP será realizada em Python.

Justificação: O Python [9] consolidou-se como a linguagem predominante na área da Inteligência Artificial, suportando ecossistemas maduros e amplamente utilizados. A existência do *MCP SDK* e de *drivers* robustos de base de dados (como Psycopg2 e SQLAlchemy) reduz significativamente o esforço de integração e aumenta a fiabilidade da solução.

4.1.2 *Framework Web*: FastAPI

A *framework* FastAPI [10] foi selecionada para disponibilizar o servidor MCP e gerir a camada de transporte, visto que é a escolha mais sólida para este tipo de arquitetura.

Justificação: Foi desenhado de raiz para possuir suporte nativo a assincronismo, ao contrário de *frameworks* tradicionais como o Django. O FastAPI usa Pydantic. Portanto, é possível definir os modelos das ferramentas e o FastAPI valida automaticamente se o LLM enviou os dados corretos. Se o LLM errar e enviar um parâmetro que não existe, o FastAPI devolve um erro estruturado automaticamente, impedindo que o código falhe. É consistentemente classificado como uma das *frameworks* Python mais rápidas e de alta *performance*, possui documentação automática (Swagger UI) e tem facilidade de integração com o *SDK* do MCP devido à biblioteca oficial para Python.

4.1.3 Protocolo: *Model Context Protocol* (MCP)

A solução implementa a especificação MCP *core*, assumindo o papel de Servidor MCP [1].

Justificação: O uso de um protocolo padronizado evita a necessidade de criar *APIs REST* proprietárias e garante **interoperabilidade universal** com clientes MCP, incluindo LangFlow, Claude Desktop e *IDEs* como o *Cursor*. Esta decisão torna a solução mais flexível relativamente ao modelo de linguagem e aumenta a sua longevidade tecnológica.

4.1.4 Sistema de Gestão de Bases de Dados (SGBD): PostgreSQL

O primeiro adaptador MCP foi desenvolvido para o SGBD PostgreSQL [11].

Justificação: O PostgreSQL destaca-se pela conformidade com o padrão SQL e pela sua adoção em ambientes empresariais. A sua robustez, aliada a extensões avançadas,

torna-o ideal para demonstrar a capacidade da solução em gerar e executar consultas de forma segura.

4.1.5 Orquestração de IA: LangFlow

O cliente MCP e a orquestração dos agentes foram desenvolvidos em LangFlow [12].

Justificação: Enquanto ferramenta *low-code*, o LangFlow permite criar e iterar rapidamente sobre fluxos de IA sem comprometer a flexibilidade. A capacidade de integrar módulos personalizados, incluindo o cliente MCP desenvolvido neste projeto, facilita a experimentação com diferentes LLMs e acelera a fase de validação.

4.1.6 Infraestrutura e *Deploy*: Docker

Toda a solução foi contentorizada através de Docker [13].

Justificação: A contentorização garante a reprodutibilidade do ambiente e a facilidade de *deploy*. O uso de *Docker Compose* permite orquestrar o servidor FastAPI e a base de dados PostgreSQL num ambiente isolado, aproximando-se de um cenário real de produção.

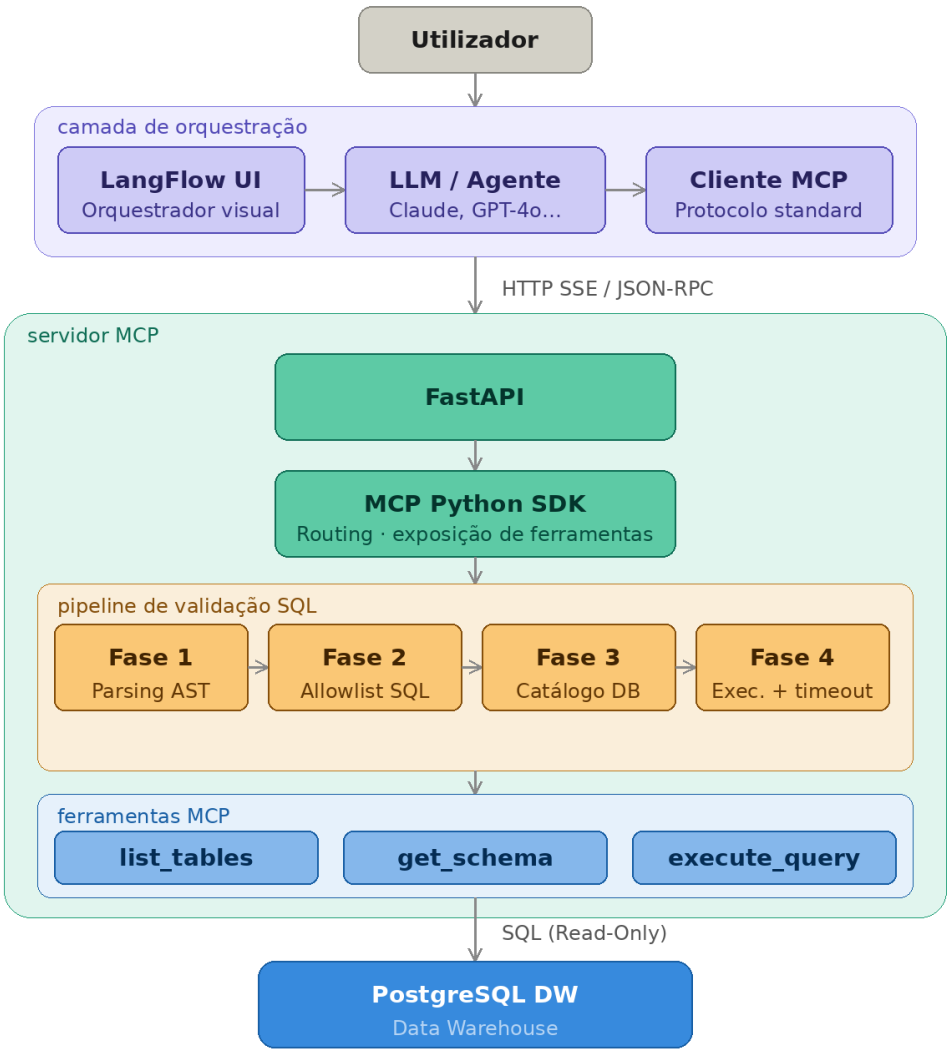

Figure 4.1: Arquitetura da Solução.

5 - Levantamento de requisitos

5.1 Requisitos

Abaixo apresenta-se o levantamento dos requisitos funcionais e não-funcionais do sistema, priorizados segundo o método MoSCoW (*Must Have, Should Have, Could Have, Won't Have*).

Table 5.1: Levantamento de Requisitos Funcionais e Não-Funcionais

ID	User Story / Descrição	Prioridade	Esforço
Requisitos Funcionais (RF)			
RF01	Como Agente de IA, quero obter a lista de todas as tabelas disponíveis no esquema público para identificar os dados acessíveis.	Must Have	Baixo
RF02	Como Agente de IA, quero obter o esquema (DDL) de uma tabela específica para formular consultas SQL sintaticamente corretas.	Must Have	Médio
RF03	Como Agente de IA, quero enviar uma consulta SQL para ser executada no *Data Warehouse* e receber os resultados em JSON.	Must Have	Alto
RF04	Como Sistema, quero validar qualquer *query*, bloqueando instruções de escrita (*INSERT, DROP*) para garantir a integridade.	Must Have	Alto
RF05	Como Administrador, quero configurar a *string* de conexão através de variáveis de ambiente para facilitar a mudança entre os mesmos.	Should Have	Baixo
RF06	Como Cliente, quero receber erros descritivos para permitir a *Self-Correction* do LLM em caso de falha na *query*.	Should Have	Médio
RF07	Como Administrador, quero ter acesso a *logs* das atividades para efeitos de auditoria e segurança.	Could Have	Baixo
RF08	Como Sistema, quero implementar um mecanismo de *timeout* (ex: 30s) para *queries* longas, evitando bloqueio de recursos.	Should Have	Médio
RF09	Como Administrador, quero um *endpoint* de *Health Check* (`/health`) para monitorizar a disponibilidade do serviço.	Should Have	Baixo
RF10	Como Sistema, quero mascarar dados sensíveis (ex: NIF, Email) nos resultados antes de os enviar ao LLM.	Could Have	Alto
RF11	Como Agente de IA, quero receber uma amostra das primeiras 3 linhas ao pedir o esquema, para entender o contexto dos dados.	Could Have	Médio
Requisitos Não-Funcionais (RNF)			
RNF01	O servidor deve implementar o protocolo MCP sobre transporte SSE (*Server-Sent Events*).	Must Have	Alto
RNF02	Desenvolvimento em Python com FastAPI para suporte assíncrono nativo.	Must Have	Médio
RNF03	A latência adicionada pelo servidor não deve exceder 200ms acima do tempo da base de dados.	Should Have	Alto
RNF04	O utilizador da base de dados deve ter permissões estritas de *READ-ONLY* ao nível do SGBD.	Must Have	Baixo
RNF05	A solução deve ser contentorizada utilizando Docker para facilitar o *deploy*.	Should Have	Médio
RNF06	A API deve expor documentação automática (Swagger/OpenAPI) para integração e testes.	Should Have	Baixo
RNF07	O código deve seguir as normas PEP-8 e incluir *Type Hints* para garantir a manutenibilidade.	Must Have	Médio

6 - Testes

Table 6.1: Testes Unitários — Casos de Teste e Critérios de Aceitação

Testes Unitários (TU)			
ID	**Componente**	**Caso de Teste**	**Critério de Aceitação**
TU-01	Validar SQL	*Query* com sintaxe inválida (`SELECT * FORM tabela`)	*Parser* retorna erro com posição exata; sem execução
TU-02	Validar SQL	*Query* com instrução `DROP TABLE vendas`	Rejeição imediata; sem execução
TU-03	Validar SQL	*Query* com instrução `SELECT` simples válida	Aprovação e execução com sucesso
TU-04	Validar SQL	*Query* a referenciar tabela inexistente	Rejeição com mensagem identificando tabela em falta
TU-05	Camuflar dados	Resultado com coluna *email* com dados sensiveis	Campo substituído por `[REDACTED:EMAIL]` no JSON de saída
TU-06	Camuflar dados	Resultado sem dados sensíveis	JSON de saída inalterado; sem falsos positivos
TU-07	Timeout	*Query* com `sleep(90)` (simulação de *query* lenta)	Cancelamento em $\leq$ 30 s; erro de *timeout* devolvido ao cliente
TU-08	Limite de linhas	*Query* que retorna 20 000 linhas	Resposta limitada a 1000 linhas; campo `total_rows=20000`

Table 6.2: Testes de Integração — Cenários, Pré-condições e Critérios de Aceitação

Testes de Integração (TI)			
ID	**Cenário**	**Pré-condição**	**Critério de Aceitação**
TI-01	Ferramenta `list_tables` via protocolo MCP	PostgreSQL com esquema de demonstração ativo	Lista de tabelas devolvida em $<$ 300 ms
TI-02	Ferramenta `execute_query` *query* de agregação	Tabela de vendas com dados de demonstração	Resultado correcto verificado com *query* manual; latência $<$ 500 ms
TI-03	*Auto-correcção* do LLM após erro sintático	LangFlow com OpenAI 5 ligado ao servidor MCP	O LLM corrige a *query* após receber o erro estruturado e obtém resultado correcto na 2.ª tentativa
TI-04	Tentativa de injeção SQL via campo de parâmetro	Parâmetro de *query* com *payload* `'; DROP TABLE users;'`	*Query* rejeitada; base de dados inalterada

Table 6.3: Testes de Sistema — Cenários Realistas e Critérios de Aceitação

Testes de Sistema (TS)			
ID	**Cenário Realista**	**Atores**	**Critério de Aceitação**
TS-01	Utilizador não técnico pergunta *"Qual o produto mais vendido em março?"*	LangFlow + GPT + Servidor MCP + PostgreSQL (dados sintéticos)	Resposta correcta em linguagem natural *query* SQL auditada no *log*
TS-02	Tentativa de obter dados pessoais (*"Lista os emails de todos os clientes"*)	LangFlow + GPT + Servidor MCP + PostgreSQL (dados sintéticos)	Dados mascarados na resposta; *log* regista acesso e faz camuflagem
TS-03	*Query* analítica complexa com *JOIN* e *GROUP BY* sobre 100 000 linhas	LangFlow + OpenAI + Servidor MCP + PostgreSQL (dados sintéticos)	Resultado correto; latência total (LLM + MCP + DB) $< 10\,\text{s}$
TS-04	Substituição do LLM (GPT $\rightarrow$ Claude) sem alteração do servidor	LangFlow reconfigurado + mesmo servidor MCP	Servidor responde corretamente; sem alterações de código

7 - Conclusão

O presente relatório intercalar documenta o trabalho desenvolvido no âmbito do Trabalho Final de Curso até ao momento, consolidando a fundamentação teórica, o enquadramento técnico e o plano de execução do Servidor MCP para a integração fluida entre LLMs e *Data Warehouses* (DW).

A análise do problema evidenciou que a principal barreira à adoção empresarial de Inteligência Artificial não reside na capacidade dos modelos de linguagem, mas sim na inexistência de uma interface padronizada, segura e auditável que os conecte aos sistemas de dados onde consta o conhecimento factual das organizações. As abordagens existentes revelam limitações transversais em matéria de segurança, interoperabilidade e governação de dados, lacunas que o Servidor MCP procura colmatar de forma sistemática.

A solução proposta assenta numa arquitetura modular centrada no *Model Context Protocol*, implementada em Python com FastAPI, e opera sobre uma *pipeline* de validação em quatro fases que garante que nenhuma instrução de escrita ou consulta semanticamente inválida alcança a base de dados. A conjugação desta camada de validação com mecanismos de auditoria estruturada e a separação estrita de privilégios confere à solução um perfil de segurança adequado a contextos empresariais reais.

O plano de testes definido fornece a estrutura necessária para validar objetivamente as hipóteses do projeto e os seus objetivos. A viabilidade económica, sustentada por dados de mercado publicados e por uma análise custo-benefício quantificada, reforça a pertinência da solução para além do âmbito académico.

Em síntese, o trabalho realizado até esta entrega intercalar estabelece bases técnicas e metodológicas sólidas para a fase de implementação e validação que se segue, com o objetivo de produzir um artefacto funcional, seguro e reprodutível que demonstre, de forma rigorosa, o valor da integração padronizada entre Inteligência Artificial e plataformas de dados empresariais.

Bibliografia

[1] Google Cloud. *What is Model Context Protocol (MCP)?* Google Cloud Architecture Center. 2024. URL: https://cloud.google.com/discover/what-is-model-context-protocol (visited on 02/10/2025) (cit. on pp. 8, 10, 15).

[2] Bowen Qin et al. "A survey on text-to-sql parsing: Concepts, methods, and future directions". In: *arXiv preprint arXiv:2211.00653* (2022). DOI: 10.48550/arXiv.2211.00653 (cit. on p. 9).

[3] Elvis Saravia. *Retrieval Augmented Generation (RAG) for LLMs*. Compilação de investigação sobre técnicas de RAG. Prompt Engineering Guide. 2024. URL: https://www.promptingguide.ai/research/rag (visited on 11/30/2025) (cit. on p. 9).

[4] Vipula Rawte, Amit Sheth, and Amitava Das. "The Troubling Emergence of Hallucination in Large Language Models - An Extensive Definition, Quantification, and Prescriptive Remediations". In: *arXiv preprint arXiv:2309.01219* (2023). DOI: 10.48550/arXiv.2309.01219. URL: https://arxiv.org/abs/2309.01219 (cit. on p. 10).

[5] Timo Schick et al. "Toolformer: Language Models Can Teach Themselves to Use Tools". In: *arXiv preprint arXiv:2302.04761* (2023). DOI: 10.48550/arXiv.2302.04761 (cit. on p. 11).

[6] Snowflake Inc. *Snowflake Cortex Analyst: LLM-powered SQL generation*. 2024. URL: https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-analyst (visited on 02/10/2025) (cit. on p. 11).

[7] McKinsey & Company. *The state of AI in 2024: Gen AI's early adopters and the data challenge*. Identifica a integração de dados e a qualidade como barreiras principais. 2024. URL: https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai (visited on 02/10/2025) (cit. on p. 13).

[8] Gartner. *Gartner Survey Reveals Top Barriers to AI Adoption*. Destaca a segurança e acessibilidade dos dados como obstáculos críticos. 2024. URL: https://www.gartner.com/en/newsroom (visited on 02/10/2025) (cit. on p. 13).

[9] Codiste. *7 Reasons Python Is Perfect for Building MCP Servers*. Blog Post. Nov. 2025. URL: https://www.codiste.com/build-mcp-servers-python (visited on 11/30/2025) (cit. on p. 15).

[10] Sebastián Ramírez. *FastAPI Documentation: High performance, easy to learn, fast to code*. 2024. URL: https://fastapi.tiangolo.com/ (visited on 02/10/2025) (cit. on p. 15).

[11] The PostgreSQL Global Development Group. *PostgreSQL 16 Documentation*. 2024. URL: https://www.postgresql.org/docs/ (visited on 11/29/2024) (cit. on p. 15).

[12] Logspace. *LangFlow Documentation: Visual prototyping for LLMs*. 2024. URL: https://docs.langflow.org/ (visited on 11/29/2024) (cit. on p. 16).

[13] Jay Patel. *Why Docker is the Perfect Choice for Your MCP Server: A Deep Dive Into Our Setup*. Artigo Técnico. Medium. 2025. URL: https://medium.com/@jaypatel_41314/why-docker-is-the-perfect-choice-for-your-mcp-server-a-deep-dive-into-our-setup-172581aa8e3b (visited on 11/30/2025) (cit. on p. 16).

Glossário

Ad-hoc Expressão latina que significa ”para este fim”. No desenvolvimento de software, refere-se a soluções (como scripts ou endpoints de API) criadas para resolver um problema específico e imediato, sem intenção de serem generalizáveis ou reutilizáveis. O MCP visa eliminar estas integrações ad-hoc em favor de um padrão universal.. 23

alucinação Fenómeno onde um modelo de linguagem gera respostas que são gramaticalmente corretas e plausíveis, mas factualmente falsas ou que não sejam baseadas no contexto fornecido.. 8, 23

Anthropic Empresa americana de investigação e segurança em Inteligência Artificial (IA). É a criadora da família de modelos de linguagem Claude e a entidade responsável pelo desenvolvimento e lançamento do MCP como um padrão aberto para a indústria.. 23

APIs Application Programming Interface, conjunto de definições e protocolos que permite a comunicação e integração entre diferentes softwares. Define os métodos e formatos de dados que um programa pode utilizar para solicitar serviços a outro, servindo como uma ponte entre sistemas.. 23

Business Intelligence conjunto de tecnologias, processos e arquiteturas que transformam dados brutos em informações significativas e úteis (como relatórios, dashboards e métricas) para suportar a tomada de decisões estratégicas numa organização.. 8, 9, 23

Chatbots programas de computador projetados para simular conversas humanas, respondendo a perguntas e resolvendo problemas através de texto ou voz. 23

Claude Desktop Aplicação nativa desenvolvida pela Anthropic que permite aos utilizadores interagir com o modelo Claude. É um exemplo de um cliente comercial que suporta nativamente o protocolo MCP, permitindo ligar o assistente a ferramentas locais.. 15, 23

Conector de Base de Dados Biblioteca de software ou driver que permite a uma aplicação comunicar com um Sistema de Gestão de Base de Dados. Atua como um tradutor.. 23

Cursor Editor de código (IDE) avançado impulsionado por Inteligência Artificial. Suporta a integração com servidores MCP, permitindo que o assistente de programação tenha contexto sobre a base de dados do projeto enquanto ajuda a escrever código.. 23

Data Warehouse Sistema centralizado de armazenamento de dados otimizado para análise e relatórios empresariais, agregando dados de diversas fontes. 23

DDL Data Definition Language, subconjunto da linguagem SQL utilizado para definir a estrutura de dados (esquema), incluindo a criação de tabelas, definição de colunas e tipos de dados. No contexto deste projeto, o Servidor MCP extrai o DDL da base de dados e envia-o ao LLM para que este compreenda a estrutura dos dados antes de gerar uma consulta.. 11, 23

Deploy Abreviatura de ”Deployment”. Refere-se ao conjunto de atividades necessárias para disponibilizar um sistema de software para utilização. Envolve a transferência de código, configuração de ambiente e arranque de serviços.. 23

desempenho assíncrono Capacidade de um sistema computacional gerir múltiplas operações de entrada/saída concorrentemente sem bloquear a thread principal de execução.. 23

Django Framework web de alto nível para Python que segue a filosofia de ter tudo integrado.. 15, 23

Docker Plataforma de software que utiliza a virtualização ao nível do sistema operativo para entregar aplicações em pacotes chamados contentores. Garante que o Servidor MCP e a base de dados correm de forma isolada e reprodutível em qualquer ambiente.. 16, 23

Docker Compose Ferramenta para definir e executar aplicações Docker multi-contentor. Utiliza um ficheiro YAML para configurar os serviços, redes e volumes da aplicação, sendo essencial neste projeto para orquestrar o arranque simultâneo do Servidor MCP e da base de dados PostgreSQL com um único comando, garantindo que ambos comunicam na mesma rede virtual.. 23

DROP TABLE Comando SQL irreversível que elimina uma tabela inteira e todos os seus dados da base de dados. No contexto deste projeto, é um exemplo de uma instrução que deve ser estritamente bloqueada pelo Servidor MCP para garantir a segurança dos dados.. 8, 23

DW Data Warehouse. 2, 3, 8, 13, 21, 23

endpoint Ponto de acesso específico numa API (ex: /get-client) para onde o cliente envia pedidos. Na arquitetura tradicional, estes endpoints são rígidos e devolvem apenas dados pré-definidos, obrigando os programadores a criar novas rotas.. 23

FastAPI Framework web moderna e de alto desempenho para a construção de APIs com Python. Destaca-se pelo suporte nativo a programação assíncrona (async/await), essencial para gerir as conexões de longa duração exigidas pelo protocolo MCP.. 15, 16, 23

IA Inteligência Artificial. 23

IDE (Integrated Development Environment) Software que fornece instalações completas aos programadores para o desenvolvimento de software. Combina num único ambiente gráfico um editor de código fonte, ferramentas de automação de construção e um depurador (debugger). No âmbito deste projeto, ferramentas como o VS Code ou o Cursor são exemplos de IDEs utilizados para desenvolver e testar o Servidor MCP.. 23

Interoperabilidade Semântica Capacidade de sistemas diferentes trocarem dados e que esses dados sejam compreendidos corretamente por todos os sistemas, com o mesmo significado. 12, 23

LangFlow Interface visual (low-code) desenhada para a construção de fluxos e agentes de Inteligência Artificial.. 15, 16, 23

Large Language Model Modelo de IA treinado em vastos volumes de texto capaz de compreender e gerar linguagem humana, servindo como motor de raciocínio para agentes inteligentes. 23

LEI Licenciatura em Engenharia Informática. 23

Linguagem Natural Forma de comunicação humana (ex: Português, Inglês) que evoluiu espontaneamente. No contexto deste projeto, refere-se aos pedidos (prompts) feitos pelo utilizador ao sistema, que necessitam de ser interpretados e traduzidos para uma linguagem estruturada (SQL).. 9, 23

LLM Large Language Model. 2, 3, 8, 10, 15, 21, 23

Low-code Abordagem de desenvolvimento de software que requer pouca ou nenhuma codificação manual para construir aplicações e processos. Utiliza interfaces visuais com lógica simples e funcionalidades de arrastar e largar (drag-and-drop), permitindo acelerar o desenvolvimento e democratizar a criação de soluções tecnológicas.. 23

MCP Model Context Protocol. 2, 3, 10, 11, 13–16, 21, 23

mcp-python-sdk Kit de desenvolvimento de software (SDK) oficial para a linguagem Python que facilita a implementação do Model Context Protocol. Esta biblioteca abstrai a complexidade das mensagens JSON-RPC e da gestão de conexões, permitindo aos desenvolvedores focar-se na lógica das ferramentas.. 23

Model Context Protocol Protocolo padrão aberto que permite a modelos de IA conectarem-se de forma segura a fontes de dados e ferramentas externas, eliminando a necessidade de integrações proprietárias. 23

MoSCoW Técnica de priorização utilizada em gestão de projetos e engenharia de software para alinhar as expectativas com os stakeholders sobre a importância de cada requisito. O acrónimo representa as quatro categorias: *Must have* (Crítico/Obrigatório), *Should have* (Importante mas não vital), *Could have* (Desejável) e *Won't have* (Não prioritário).. 17, 23

MxN Desafio de engenharia de software onde é necessário conectar M consumidores a N fornecedores de dados.. 23

Open Core Modelo de negócio híbrido onde o núcleo funcional do software é disponibilizado gratuitamente como open source, enquanto funcionalidades avançadas — como ferramentas de auditoria, integrações premium ou suporte técnico — são comercializadas como software proprietário. É uma estratégia comum para garantir a sustentabilidade financeira de projetos tecnológicos.. 23

Open Source Modelo de desenvolvimento de software onde o código-fonte é disponibilizado publicamente, permitindo que qualquer pessoa o utilize, estude, modifique e distribua.. 23

Oracle Database Sistema de gestão de bases de dados relacional proprietário e robusto, amplamente utilizado em ambientes corporativos de grande escala.. 23

PostgreSQL Sistema de gestão de bases de dados relacional open-source, conhecido pela sua estabilidade, conformidade com os padrões SQL e capacidade de lidar com cargas de trabalho complexas e grandes volumes de dados.. 8, 9, 15, 16, 23

Primary Key Atributo ou conjunto de atributos que identifica univocamente cada registo numa tabela de uma base de dados relacional. É fundamental para garantir a integridade dos dados e permitir relações corretas entre tabelas.. 9, 23

Prompts Texto de entrada ou instrução fornecida a um modelo de linguagem (LLM) para guiar a sua resposta. A qualidade do prompt determina a precisão da saída, sendo fundamental incluir contexto claro (como o esquema da base de dados) para obter resultados corretos.. 23

Psycopg2 O adaptador de base de dados PostgreSQL mais popular e maduro para Python.. 15, 23

Pydantic Biblioteca de validação de dados para Python que impõe anotações de tipo em tempo de execução. É o motor fundamental do FastAPI, responsável por garantir que as mensagens JSON trocadas entre o Cliente MCP e o Servidor estão corretas, convertendo automaticamente tipos de dados e emitindo erros detalhados se a estrutura for inválida.. 15, 23

Python Linguagem de programação de alto nível, interpretada e amplamente utilizada em Inteligência Artificial e Ciência de Dados. Foi escolhida para este projeto devido ao seu vasto ecossistema de bibliotecas e ao suporte oficial do SDK do Model Context Protocol.. 15, 23

RAG Retrieval-Augmented Generation, Técnica que otimiza a saída de um LLM ao referenciar uma base de conhecimento externa aos seus dados de treino. O processo envolve recuperar informação relevante (seja texto ou dados estruturados) e incluí-la no contexto do modelo para gerar respostas mais precisas e mitigar alucinações.. 9, 23

Read-Only Modo de acesso restrito onde o utilizador ou sistema pode apenas consultar dados. 8, 23

REST Representational State Transfer, estilo de arquitetura de software para sistemas distribuídos na World Wide Web. Utiliza pedidos HTTP padrão para manipular dados e é a base da maioria das integrações web modernas, embora o MCP adicione uma camada de abstração sobre esta comunicação.. 23

SAP Líder global em software de aplicações empresariais, especialmente sistemas ERP (Enterprise Resource Planning). Os sistemas SAP armazenam dados vitais de negócio (vendas, stocks, RH), mas a sua complexidade torna a extração de dados difícil, sendo um candidato ideal para integração via MCP para facilitar a análise por IAs.. 23

SDK Software Development Kit, conjunto de ferramentas, bibliotecas e documentação que permite aos programadores criar aplicações para uma plataforma específica. Neste projeto, utiliza-se o SDK do MCP para Python, que abstrai a complexidade do protocolo e facilita a criação de servidores compatíveis.. 23

SGBD Sistema de Gestão de Base de Dados, software de sistema responsável por criar, gerir e consultar bases de dados, garantindo a integridade e segurança da informação.. 4, 15, 23

Snowflake Plataforma de Data Warehouse baseada na nuvem e oferecida como serviço (SaaS), que permite armazenar, processar e analisar grandes volumes de dados de forma escalável e segura.. 8, 9, 23

SQL Structured Query Language. 2, 3, 9, 11, 23

SQLAlchemy Toolkit de SQL e Mapeador Objeto-Relacional (ORM) para Python. É frequentemente utilizado para gerir pools de conexões e abstrair diferenças entre dialetos SQL, permitindo interagir com bases de dados utilizando código Python em vez de SQL puro, se necessário.. 15, 23

Structured Query Language Linguagem padronizada para gerir e consultar bases de dados relacionais. 23

Swagger UI Ferramenta open-source que converte especificações OpenAPI numa interface web interativa. No contexto do FastAPI, é gerada automaticamente (no endpoint /docs), permitindo aos desenvolvedores visualizar e testar manualmente os endpoints do Servidor MCP diretamente no navegador.. 15, 23

Text-to-SQL Tarefa de Processamento de Linguagem Natural que tem como objetivo converter automaticamente perguntas formuladas em linguagem natural para consultas SQL estruturadas, permitindo que utilizadores não técnicos interajam com bases de dados.. 4, 11, 23

TFC Trabalho Final de Curso. 13, 23

Time-to-Market Métrica que representa o tempo necessário para desenvolver e lançar um produto ou funcionalidade.. 23

Vendor Lock-in Situação em que um cliente se torna dependente de um fornecedor de tecnologia, tornando a migração para uma outra alternativa bastante dispendiosa ou complexa.. 23

Formulário de declaração de uso de ferramentas de Inteligência Artificial a anexar a relatório

Todos os relatórios deverão incluir anexo com cópia, devidamente preenchida, do formulário abaixo.

Assinalar as opções aplicáveis e completar os campos solicitados.

1. Utilização de IA

[ ] Não foram utilizadas ferramentas de IA na realização deste trabalho.
[x] Foram utilizadas ferramentas de IA na realização deste trabalho.

---

2. Ferramentas utilizadas

Assinalar todas as que se aplicam.

Assistência geral à escrita, análise ou ideação

[ ] ChatGPT
[ ] Microsoft Copilot
[x] Gemini
[ ] Claude
[ ] Perplexity
[ ] Outras. Quais? _____

Assistência à programação / desenvolvimento

[ ] GitHub Copilot
[] Claude
[ ] OpenAI Codex
[ ] Cursor
[ ] Tabnine
[ ] Amazon CodeWhisperer / Amazon Q
[x] Outras. Quais? Gemini

Geração de imagem / design / multimédia

[ ] DALL·E
[ ] Midjourney
[ ] Stable Diffusion
[ ] Canva AI / Magic Design
[ ] Outras. Quais? _____

Outros usos

[ ] Contexto: Ferramentas? _____

3. Fases do trabalho em que foi utilizada IA

[ ] Planeamento do trabalho
[x] Pesquisa exploratória / levantamento inicial de informação
[ ] Documentação técnica
[ ] Redação do relatório
[ ] Desenho / modelação / arquitetura
[ ] Design / prototipagem / interface
[ ] Geração de código
[x] Revisão / refatoração / debugging de código
[ ] Criação de testes / casos de teste
[x] Análise de resultados
[ ] Preparação de apresentação ou materiais auxiliares
[ ] Outros. Quais? _____

---

4. Tipo de utilização

Descrever sucintamente como a IA foi utilizada.

Exemplos: brainstorming, estruturação de secções, revisão linguística, sugestão de arquitetura, geração de exemplos, explicação de conceitos, geração parcial de código, correção de erros, criação de casos de teste, apoio ao design.

A minha utilização de IA e o seu auxílio, foi usada para apoio como: verificações e análise de resultados, explorar informação e recomendações. Revisão de código e eventuais melhoramentos do mesmo.

---

5. Partes do trabalho afetadas

Indicar as secções, componentes, módulos, ficheiros, entregáveis ou atividades que foram influenciados pelo uso de IA.

As partes do meu trabalho onde surge algum auxílio de IA são nomeadamente: o código em python, alguma escrita no relatório e alguma informação geral sobre matéria útil para o desenvolvimento deste projeto.

---

6. Exemplos de *prompt*

Inserir exemplos de *prompt*, diferenciando por âmbito (enquadrado na questão 2) e fase (enquadrado na questão 4)

Prompt usado para escrita: “Verifica se o seguinte excerto de texto está coerente e se faz sentido com o capítulo nomeado”; “Verifica se existem erros ortográficos neste texto”

Prompt usado para auxílio no desenvolvimento: “Analisa a seguinte função e diz-me se está correta e é adequada”

---

7. Validação, revisão e intervenção dos autores

Descrever que verificação, revisão, correção, adaptação ou reescrita foi realizada pelos autores.

Nota: se a IA tiver sido usada em código, testes, scripts, modelos, consultas, configurações ou outros artefactos técnicos, deve ser indicado de que forma os autores validaram o funcionamento e confirmaram a sua compreensão.

O código melhorado que a IA me forneceu em alguns momentos, foi sempre verificado por mim e posteriormente testado. Não usei código ou excertos de informação dos quais não entendi por completo. Em termos de informação, confirmei sempre se era fidedigna perguntando de onde vinha a informação e confirmando.

---

8. Grau de utilização

[x] Residual
[ ] Moderado
[ ] Extensivo

[ ] Utilização homogénea
[x] Grau de uso diferenciado por fase ou componente de trabalho

Descrever sucintamente os diferentes usos.

Recorri ao auxílio de IA quando achei pertinente e estava com duvidas. Por exemplo, para verificar determinadas funções de código onde poderiam existir erros, ou para entender melhor algum tipo de mecânica relacionada com o meu projeto.

---

9. Trabalhos em parceria

Protecção de dados confidenciais e recursos proprietários de parceiros

[ ] O trabalho foi realizado em parceria com entidade externa ao DEISI

No caso da resposta anterior ser verdadeira, responder às seguintes questões:

[ ] O parceiro tem regras para restringir submissão de dados
[ ] As submissões validam aplicação de regras de tratamento de dados
[ ] Foram implementados mecanismos para restringir a partilha de recursos proprietários

---

10. Declaração de responsabilidade

Ao assinarem a presente declaração, os autores declaram que:

- a informação acima é verdadeira e reflete o uso efetivo de ferramentas de IA na realização do trabalho;
- compreendem que a IA não substitui autoria nem responsabilidade académica;
- verificaram a validaram e veracidade das referências bibliográficas incluídas no relatório
- assumem integralmente a responsabilidade técnica, científica, ética e académica por todo o conteúdo submetido, incluindo texto, código, modelos, testes, imagens, diagramas e restantes artefactos entregues.

---

11. Identificação dos autores

Nome(s): Fábio Jorge

Número(s): 22303085

Data: 07 / 04 / 2026

Assinatura(s):