Power BI Conversacional com RAG

João Pedro Sá Fernandes[1], F. Jorge Duarte[1], and David Mota[2]

ISEP/IPP - DevScope, Porto
1211681[1],fjd[1]@isep.ipp.pt david.mota@devscope.net[2]

Resumo O crescimento do uso do Microsoft Power BI multiplicou o número de relatórios estratégicos, mas o acesso tradicional via *dashboards* mantém uma barreira de literacia em BI. Este artigo sintetiza o projeto de um protótipo de assistente conversacional que permite interrogar relatórios Power BI em linguagem natural com base numa arquitetura RAG, em contexto empresarial. A solução automatiza a exportação para PDF, armazena os ficheiros no Azure Blob Storage, extrai informações dos mesmos, gera *embeddings* (ChromaDB) e fundamenta respostas com LLM do Azure OpenAI, orquestrada por FastAPI e LangChain, com autenticação corporativa via Azure AD/MSAL. Ensaios conceptuais realizados com dados reais indicam respostas em $< 4\,\mathrm{s}$, redução substancial do tempo de procura de métricas e elevada satisfação dos utilizadores, com citação explícita dos trechos de origem, caracterizando o trabalho como prova de conceito com potencial de validação futura.

Palavras-chave Business Intelligence Conversacional · Power BI · Chatbots · Retrieval-Augmented Generation (RAG) · Azure OpenAI · Inteligência Artificial

1 Introdução

As organizações recorrem a Power BI para análise e decisão, mas o acesso por *dashboards* continua pouco interativo para não especialistas. Pretende-se um fluxo *end-to-end* que exporte, indexe e recupere conteúdo de relatórios, devolvendo respostas auditáveis em linguagem natural. A abordagem adotada conjuga recuperação vetorial e geração (RAG), com armazenamento em Azure Blob, núcleo em FastAPI/LangChain, *vector store* ChromaDB e autenticação MSAL.

1.1 Descrição do problema

Sem integração direta e automatizada entre Power BI e um sistema conversacional, o acesso à informação depende de navegação manual e de equipas de BI, reduzindo a eficiência no dia-a-dia e dificultando respostas rápidas e fundamentadas.

1.2 Objetivos

O objetivo principal deste trabalho é automatizar a extração de relatórios Power BI para a *cloud* e disponibilizá-los a um LLM via RAG, permitindo perguntas em linguagem natural com respostas precisas e citadas. Os objetivos específicos foram:

- Exportação automática de relatórios para PDF;
- Ingestão, *chunking* e *embeddings* persistidos em ChromaDB;
- Recuperação semântica e geração com *prompt* conservador;
- UI *web* com SSO corporativo;
- Avaliação funcional e de desempenho.

2 Estado da Arte

A literatura recente em BI conversacional destaca o valor de interfaces naturais para democratizar o acesso a informação analítica, reduzindo a dependência de literacia em ferramentas de BI e de equipas especializadas [2]. Em particular, assistentes conversacionais ligados a *dashboards* permitem consultar relatórios e indicadores em linguagem natural, aproximando as perguntas dos utilizadores da forma como pensam o negócio [1,2]. No entanto, vários estudos apontam limitações persistentes em atualização contínua de dados, precisão das respostas e mecanismos de segurança e governança [2,5].

O paradigma Retrieval-Augmented Generation (RAG) surgiu como resposta a algumas destas lacunas: combina modelos de linguagem com mecanismos de recuperação de documentos, ancorando as respostas em evidências concretas e mitigando alucinações típicas de LLMs [3]. Em cenários empresariais, em que a veracidade e a rastreabilidade são essenciais, esta abordagem tem ganho tração, embora ainda com pouca documentação detalhada em contextos concretos de Power BI.

2.1 Integração de um *chatbot* com Power BI

A Microsoft tem investido na integração entre soluções conversacionais e Power BI, nomeadamente através de Power Virtual Agents/Copilot Studio e Microsoft Bot Framework [1,4]. Estes esforços procuram tornar os dados empresariais acessíveis de forma intuitiva, permitindo que utilizadores coloquem perguntas sobre *dashboards* e relatórios usando linguagem natural [2,5].

Apesar disso, muitas soluções descritas na literatura e em documentação técnica dependem ainda de processos *batch* ou semiautomáticos para exportar e sincronizar dados, recorrendo a scripts ou integrações pontuais [1,4]. Esta abordagem dificulta a escalabilidade, compromete a atualidade das respostas e raramente explicita como é feita a ligação entre o conteúdo dos relatórios e o texto gerado pelo LLM. Em particular, poucos trabalhos detalham estratégias de *grounding* semântico com citações das fontes, ou a utilização sistemática de arquiteturas RAG integradas com Power BI [3,5].

2.2 Lacunas identificadas

Com base nos trabalhos identificados com a análise do estado da arte, podem sintetizar-se quatro grupos principais de lacunas em BI conversacional com Power BI:

- **Atualização em tempo real**: grande parte das soluções depende de fluxos *batch* ou exportações periódicas, o que reduz a capacidade de responder com dados efetivamente atuais em cenários onde métricas e KPIs mudam rapidamente [2,5].
- **Precisão e contextualização**: muitos *chatbots* baseiam-se em FAQs ou regras fixas, com pouco entendimento contextual e sem ligação clara às fontes de dados; a adoção de RAG é frequentemente mencionada, mas pouco detalhada ao nível de implementação [2,3].
- **Automatização e CI/CD**: integrações mais antigas com Power BI recorrem a scripts ou processos manuais para exportação, o que dificulta a manutenção e a evolução para *pipelines* automatizadas e reprodutíveis [1,4,5].
- **Governança e segurança**: a integração de dados sensíveis de BI exige autenticação corporativa, controlo de acessos, encriptação e auditoria, mas muitos exemplos práticos abordam estes temas de forma superficial [7,4].

Estas lacunas motivam a necessidade de soluções que combinem chatbots, Power BI e RAG com maior preocupação por atualização contínua, transparência das respostas e operacionalização em ambiente empresarial.

2.3 Tecnologias existentes e opções adotadas

Diversas tecnologias suportam a construção de arquiteturas RAG no ecossistema Microsoft. Foram analisadas alternativas para cada componente e justificada a escolha adotada no protótipo:

- **Extração automatizada e *backend***: consideraram-se Azure Functions, GitHub Actions/Azure DevOps e serviços FastAPI expostos via HTTP. Optou-se por FastAPI por concentrar a lógica de exportação e ingestão num único serviço leve, fácil de testar localmente e acionável a partir de outras ferramentas de automatização [6].
- **Armazenamento de documentos**: consideram-se Azure Blob Storage, Azure Data Lake Gen2 e Azure Files. Escolheu-se Azure Blob Storage por oferecer um modelo simples de ficheiros, encriptação nativa e custos reduzidos, adequado ao volume de relatórios previsto [7].
- **Indexação vetorial**: foram avaliadas ChromaDB, FAISS e Weaviate. A opção recaiu em ChromaDB pela integração direta com LangChain, facilidade de utilização e adequação a um protótipo com volume moderado de dados, mantendo a possibilidade de evoluir para soluções mais distribuídas no futuro [8,10].
- **LLM e *embeddings***: seguindo as recomendações da própria Microsoft, utilizaram-se modelos do Azure OpenAI com RAG, combinando *embeddings* para recuperação semântica com modelos generativos para compor respostas em linguagem natural [3].

– ***Frontend* e autenticação**: a camada de interface foi implementada em React com MSAL/Azure AD, garantindo *Single Sign-On* corporativo e integração com o restante ecossistema Microsoft, com possibilidade de publicação em Teams/Outlook via Bot Framework [4,5].

3 Análise e Desenho da Solução

Esta secção faz a ponte entre o problema identificado e a arquitetura proposta, detalhando o modelo de domínio, o fluxo *end-to-end*da solução e as principais componentes que suportam a abordagem RAG com Power BI.

3.1 Descrição geral da solução

A solução organiza-se num fluxo automatizado que exporta relatórios do Power BI para PDF através de scripts desenvolvidos em Python, guarda os ficheiros de forma centralizada no Azure Blob Storage, processa os documentos para extração de texto, criação de *embeddings* e indexação numa base vetorial (ChromaDB), aplica a arquitetura *Retrieval-Augmented Generation* (RAG), com fases de recuperação de contexto e geração de respostas, e disponibiliza as respostas aos utilizadores através de uma aplicação *web* em React.

Esta cadeia garante que as respostas do assistente conversacional não são apenas geradas pelo LLM, mas ancoradas em evidência extraída de relatórios reais, reduzindo *alucinações* e permitindo auditoria posterior às fontes consultadas.

3.2 Modelo de domínio

O modelo de domínio inclui as entidades Utilizador, *Query*, ExportJob, BlobFicheiro, Documento, *Chunk*, *Embedding* e Resposta. Um ExportJob invoca a Power BI REST API; o PDF no Azure Blob origina um Documento do qual se extraem *chunks* com *embeddings*. A *Query* recupera *chunks* relevantes e compõe a Resposta citável, preservando a rastreabilidade *Query*→*Chunks*→Resposta.

A Fig. 1 sintetiza estas relações, evidenciando o fluxo lógico desde o pedido do utilizador até à geração de uma resposta fundamentada.

3.3 Componentes da arquitetura

A arquitetura materializa o fluxo descrito na Secção 3.2 num conjunto de componentes que suportam a extração automática de relatórios Power BI, a indexação semântica e a interação conversacional em linguagem natural. A exportação de relatórios Power BI é realizada por scripts em Python que invocam a Power BI REST API para gerar PDFs e acompanham o estado por *polling*, preparando a posterior ingestão. Os ficheiros gerados são guardados num repositório central em Azure Blob Storage, com encriptação em repouso e em trânsito e organização por *containers*, o que facilita a integração com outros serviços Azure. Sobre estes documentos atua o módulo de indexação vetorial com ChromaDB, responsável

pela extração de texto, *chunking*, geração de *embeddings* via Azure OpenAI e persistência na ChromaDB (através de LangChain) para suportar pesquisa semântica eficiente.

O núcleo RAG, suportado por Azure OpenAI, recupera os *chunks* mais relevantes na base vetorial e gera respostas em linguagem natural, privilegiando respostas conservadoras (não sei) quando não existem evidências suficientes. O *backend* em FastAPI expõe *endpoints* para ingestão e consulta, integra Azure Blob, ChromaDB e Azure OpenAI, aplica autenticação com JWT e regista *logs* estruturados para diagnóstico e auditoria. A camada de apresentação é assegurada por um *frontend* em React com MSAL, que disponibiliza uma interface de *chat web* responsiva, com *Single Sign-On* corporativo e comunicação com o *backend* via Axios, permitindo ao utilizador colocar perguntas em linguagem natural e visualizar respostas com citações.

Em conjunto, estes componentes respondem tanto aos requisitos funcionais de exportação, ingestão e consulta como aos requisitos não funcionais de desempenho, segurança e usabilidade identificados no modelo FURPS.

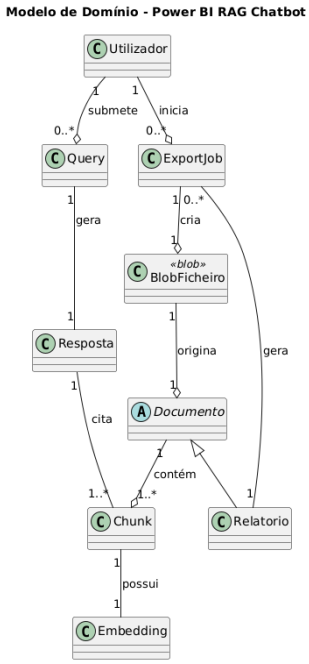

Fig. 1: Modelo de domínio do sistema RAG para Power BI.

3.4 Requisitos funcionais

Requisitos Funcionais (RF):

– RF1: Exportar automaticamente relatórios Power BI para PDF (REST API, *polling* de estado).

– RF2: Armazenar PDFs no Azure Blob com metadados (relatório, página, data).
– RF3: Ingestão: extração de texto, *chunking* e geração de *embeddings*; persistência em ChromaDB.
– RF4: RAG: recuperar trechos relevantes e gerar respostas citadas (não sei quando não tem evidências).
– RF5: Expor rotas *web* em FastAPI para ingestão/consulta com segurança corporativa.
– RF6: UI *web* com MSAL/*SSO*.

3.5 Visão arquitetural (C4)

A arquitetura é descrita combinando C4 e 4+1, para comunicar a solução em níveis progressivos (Contexto → Lógica/Implementação/Processos/Física) e garantir que as decisões técnicas respondem aos requisitos funcionais e não funcionais identificados. Esta articulação permite mostrar, de forma coerente, quem interage com o sistema, como os contentores principais se relacionam e como os fluxos críticos são executados e implantados.

Nível 2 Vista lógica. O *chatbot web* comunica com o *backend* FastAPI, que orquestra a cadeia RAG com LangChain e Azure OpenAI. Os relatórios são exportados do Power BI e armazenados no Azure Blob; durante a ingestão, o *backend* extrai texto dos PDFs, segmenta em *chunks*, gera *embeddings* e persiste-os na base vetorial ChromaDB para pesquisa semântica. Na consulta, realiza-se a procura por similaridade na ChromaDB e o contexto recuperado fundamenta a resposta gerada pelo LLM, com prioridade a respostas ancoradas em evidência, conforme ilustrado na Fig. 2. Esta vista evidencia os contentores e as suas responsabilidades: interface (React+MSAL), lógica e APIs (FastAPI+LangChain), armazenamento de documentos (Azure Blob), indexação e recuperação (ChromaDB) e serviços externos de IA (Azure OpenAI).

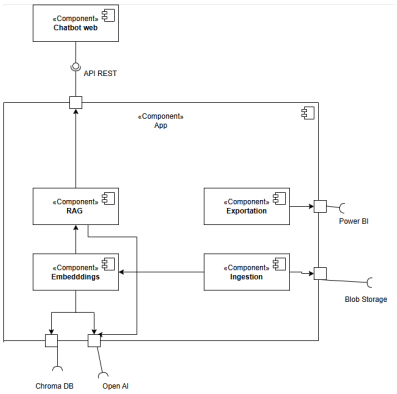

Fig. 2: C4 Nível 2 Diagrama de vista lógica.

3.6 FURPS requisitos não funcionais

O modelo FURPS orientou os requisitos não funcionais da solução, com foco em: usabilidade (UI responsiva em React com pesquisa em linguagem natural e *SSO* via MSAL); confiabilidade (autenticação segura com AAD/JWT, *logging* estruturado e rastreabilidade *Query*→*Chunks*→Resposta); desempenho (latência média $< 4\,\text{s}$ e recuperação vetorial $< 40\,\text{ms}$); e suportabilidade (código modular em FastAPI/LangChain com testes unitários e de integração).

3.7 Contributo face ao estado da arte

Face a este panorama, a abordagem proposta diferencia-se ao combinar, num único fluxo *end-to-end*, a exportação automatizada de relatórios Power BI via REST API, o armazenamento seguro em Azure Blob Storage, a indexação vetorial com ChromaDB e a geração de respostas com Azure OpenAI num cenário RAG, com autenticação corporativa e citação explícita dos trechos utilizados. Em contraste com soluções focadas em FAQs ou em integrações pouco detalhadas, o protótipo enfatiza *grounding* nas fontes e rastreabilidade *Query*→*Chunks*→Resposta, respondendo diretamente às lacunas identificadas na literatura em termos de atualização, precisão e governança.

4 Implementação da Solução

Esta secção descreve a construção prática do sistema, mostrando como as decisões de análise e desenho se materializaram em código e serviços *cloud*. São abordadas a *pipeline* RAG (ingestão, indexação semântica e consulta), os serviços de *backend* em FastAPI, a interface *frontend* em React com MSAL, a exportação via Power BI REST API com armazenamento em Azure Blob e os aspetos de segurança e operação.

4.1 Visão geral da *pipeline* RAG

A Fig. 3 apresenta uma visão de alto nível da *pipeline* RAG adotada. A solução materializa uma *pipeline* em duas fases bem definidas:

- **Ingestão e indexação**: PDFs são exportados do Power BI e armazenados no Azure Blob; o texto é extraído, segmentado em *chunks* e transformado em vetores densos (`text-embedding-ada-002` via Azure OpenAI); os vetores e metadados ficam persistidos na ChromaDB para recuperação posterior.
- **Consulta**: a pergunta do utilizador é convertida em *embedding*, usada para procurar *top-k* por similaridade na ChromaDB e os trechos recuperados fundamentam a resposta gerada pelo LLM, com prioridade a respostas ancoradas em evidência.

Esta implementação reflete as recomendações para reduzir alucinações e tornar as respostas auditáveis, permitindo ao utilizador inspecionar os trechos que suportam cada resposta.

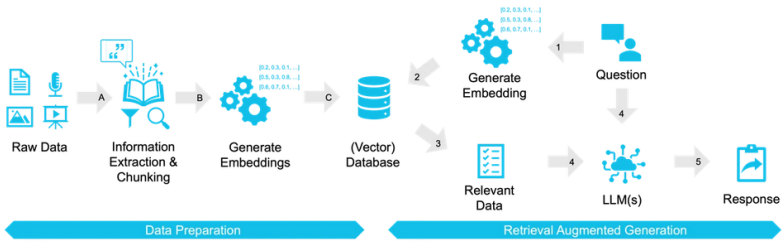

Fig. 3: *Pipeline* RAG utilizada no protótipo. [9]

4.2 Serviços e integração

O *backend* em FastAPI integra os serviços *cloud* e a lógica RAG, expondo *endpoints* para ingestão de PDFs (extração de texto, *chunking*, geração de *embeddings* e indexação em ChromaDB) e para consulta (receção da pergunta, recuperação semântica e chamada ao LLM). A utilização de FastAPI facilita a definição de rotas, a documentação automática da API e a criação de testes com `TestClient`, deixando preparada a evolução para *pipelines* automatizadas de *deploy* e CI/CD.

A interface *frontend* foi desenvolvida em React e comunica com o *backend* via Axios, disponibilizando uma interface *web* responsiva onde o utilizador pode colocar perguntas e visualizar respostas com as respetivas citações. A autenticação é gerida com MSAL/Azure AD, garantindo *Single Sign-On* corporativo e reutilização de identidades existentes, enquanto o *frontend* apresenta *feedback* visual sobre o estado das operações (carregamento, sucesso, erro) e organiza o histórico de interações de forma simples.

A exportação de relatórios Power BI para PDF é automática e transparente para o utilizador: o *backend* invoca a Power BI REST API, realiza *polling* até o ficheiro ficar pronto e guarda o PDF no Azure Blob Storage com metadados adequados, ficando este disponível para ingestão. Os documentos são armazenados com encriptação em repouso e em trânsito, e os *embeddings* são persistidos na ChromaDB com metadados que permitem ligar cada resposta aos trechos de origem, suportando consultas por proximidade vetorial e rastreabilidade *Query*→*Chunks*→Resposta.

Ao nível da segurança e operação, rotas sensíveis são protegidas com JWT emitidos após autenticação em Azure AD, e o sistema regista *logs* estruturados por pedido, incluindo informação sobre o utilizador autenticado, tempos de resposta, erros e identificadores de correlação. Estes registos suportam monitorização contínua, diagnóstico de problemas e auditoria em contexto empresarial, completando a implementação prática da arquitetura definida na Secção 3.

5 Avaliação e Testes

5.1 Metodologia

Esta secção apresenta a metodologia de avaliação adotada, estruturada em três camadas: testes unitários e de integração com `pytest`/`TestClient` no nú-

cleo RAG e serviços, testes de desempenho focados na segmentação de documentos, recuperação vetorial e latência de resposta, e um cenário *end-to-end* ingestão→consulta com dados reais. Em conjunto, estes testes visam comprovar a robustez técnica da solução e o cumprimento dos requisitos definidos.

5.2 Resultados

Alguns dos principais resultados obtidos através dos testes implementados foram: a *pipeline* RAG interpreta perguntas em linguagem natural, localiza conteúdo relevante e gera respostas coerentes fundamentadas em trechos reais dos documentos, reconhecendo corretamente quando não existe informação disponível. O tempo médio de resposta ficou abaixo de 4 s. Relativamente ao desempenho, a segmentação de documentos grandes (100 000 caracteres) ocorre em menos de 1 s e o *retrieval* semântico apresenta tempos < 40 ms (top-k). Relativamente à qualidade, as respostas mantêm *grounding* com citações claras e satisfação percebida através de testes internos. A cobertura de testes é $\sim$63% (núcleo RAG $> 80\%$), recomendando-se ampliar E2E/CI.

6 Conclusão

O projeto demonstrou a viabilidade de integrar relatórios Power BI numa cadeia RAG apoiada por serviços Azure, cobrindo exportação automática, armazenamento em Azure Blob Storage, indexação com *embeddings* e geração com Azure OpenAI, orquestrados por FastAPI/LangChain e expostos numa interface React com autenticação corporativa. Ensaios com dados reais indicam tempos médios de resposta inferiores a 4 s e processamento de cerca de 20 relatórios PDF, com redução substancial do esforço de procura de métricas e fundamentação explícita nas fontes. Neste sentido, o sistema materializa uma prova de conceito tecnicamente sólida de BI conversacional com Power BI e RAG, ainda em fase de validação organizacional.

6.1 Objetivos concretizados

Globalmente, os objetivos foram maioritariamente cumpridos: a exportação automática para Azure Blob foi implementada mas não totalmente validada por dependência de licença Power BI Pro; a *pipeline* RAG (ingestão, *chunking*, *embeddings*, recuperação) encontra-se funcional; o *bot* React com MSAL atinge latências médias de cerca de 3,7 s; e foram aplicados mecanismos de segurança com AAD, JWT e RBAC, ainda sem auditoria externa. Ensaios internos indicam reduções de cerca de 80 % no tempo de pesquisa de métricas.

6.2 Limitações

Apesar da cobertura funcional alcançada, subsistem limitações relevantes para uso em produção. A exportação via Power BI REST API está implementada e

integrada com Azure Blob Storage, mas não foi validada *end-to-end* por falta de licença Power BI Pro, requisito para geração de PDFs. O canal conversacional permanece limitado ao *frontend web* em React, sem publicação em Microsoft Teams/Outlook via Bot Framework. A *vector store* em ChromaDB com persistência local é adequada ao protótipo, mas não garante alta disponibilidade nem escalabilidade horizontal. A cobertura de testes ronda 63% (núcleo RAG $> 80\%$), ainda sem *pipeline end-to-end* nem CI/CD automatizada, e não existem métricas de utilização nem *feedback loop* dos utilizadores, o que dificulta a afinação progressiva do sistema.

6.3 Trabalho futuro e apreciação final

A solução cumpriu a finalidade proposta de permitir consultas em linguagem natural com respostas fundamentadas, sustentando ganhos práticos no acesso a informação de BI e preparando a evolução para contextos empresariais. Como trabalho futuro, recomenda-se: validar o fluxo de exportação com licenciamento adequado, publicar o agente em canais corporativos (Teams/Outlook), migrar a base vetorial para serviços geridos e integrar CI/CD com testes *end-to-end*, e realizar ensaios com utilizadores finais em ambientes realistas, incorporando *feedback loop* e métricas de utilização. Estes passos são essenciais para consolidar a transição do protótipo para operação e medir, de forma sistemática, o impacto da abordagem RAG na acessibilidade e precisão da análise em Power BI.

References

1. V. V. Gurbade, P. Verma, S. Gundewar, V. S. Bramhe: Building a Data Lake for Power BI in the Cloud: A Review on Utilizing Cloud Storage Services for Large Datasets. *Proc. 2024 2nd DMIHER Int. Conf. on Recent Trends in Engineering, Science & Management (RTESM)* (2024).
2. M. Arslan, S. Munawar, C. Cruz: Business insights using RAG–LLMs: a review and case study. *Journal of Decision Systems* (2024).
3. L. Benaddi, A. Souha, C. Ouaddi, A. Jakimi, B. Ouchao: Towards a unified metamodel for developing the conversational agents for smart tourism. *Procedia Computer Science* 236 (2024), 241–247.
4. S. M. A. Khan: Microsoft Azure Bot Service. *Technical article*, ResearchGate (2023).
5. H. Andersson: Retrieval-Augmented Generation with Azure OpenAI. Bachelor Thesis, Mälardalen University, Sweden (2024).
6. FastAPI Documentation (2024). `https://fastapi.tiangolo.com/`
7. Microsoft Learn Azure Blob Storage (2024). `https://learn.microsoft.com/azure/storage/`
8. LangChain Docs Chroma Integration (2024). `https://python.langchain.com/docs/integrations/vectorstores/chroma`
9. CrateDB: RAG Pipelines for Chatbots (2024). `https://cratedb.com/use-cases/chatbots/rag-pipelines`
10. J. Johnson, M. Douze, H. Jégou: Billion-Scale Similarity Search with GPUs (FAISS White Paper) (2019). `https://faiss.ai`