INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CAMPUS PIRIPIRI
TECNÓLOGO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CÍCERO ANDRADE SANTOS

AGENTES DE IA PARA AUTOMAÇÃO NA SELEÇÃO DE ESTUDOS EM REVISÕES SISTEMÁTICAS DA LITERATURA EM CIÊNCIA DA COMPUTAÇÃO

PIRIPIRI/PI
2026

CÍCERO ANDRADE SANTOS

AGENTES DE IA PARA AUTOMAÇÃO NA SELEÇÃO DE ESTUDOS EM REVISÕES SISTEMÁTICAS DA LITERATURA EM CIÊNCIA DA COMPUTAÇÃO

Trabalho de Conclusão de Curso (artigo científico) apresentado como exigência parcial para obtenção do diploma do Curso de Tecnólogo em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Piripiri.

Orientador(a): Prof. Me. Jeferson do Nascimento Soares

PIRIPIRI/PI
2026

AGENTES DE IA PARA AUTOMAÇÃO NA SELEÇÃO DE ESTUDOS EM REVISÕES SISTEMÁTICAS DA LITERATURA EM CIÊNCIA DA COMPUTAÇÃO

Cícero Andrade Santos[1]
Prof. Me. Jeferson do Nascimento Soares[2]

RESUMO

A Revisão Sistemática da Literatura (RSL) é um método essencial da pesquisa científica, mas sua etapa de triagem, na qual milhares de estudos são avaliados manualmente, consome muito tempo e está sujeita a inconsistências. As soluções que automatizam essa etapa com Inteligência Artificial costumam exigir conhecimento de programação, o que restringe seu uso. Este trabalho propõe e valida uma arquitetura modular que automatiza a triagem de estudos primários, orquestrada por um agente de IA na plataforma *low-code* n8n, apoiada por um modelo de linguagem que roda localmente, na máquina do próprio pesquisador. A pesquisa adota o paradigma *Design Science Research* e valida o artefato em três estudos de caso que replicam revisões existentes, com perfis distintos de fonte de dados e de informação disponível para a decisão. No caso de maior cobertura, o sistema alcançou revocação de 98% sobre os estudos recuperáveis, o que indica que recupera quase todos os artigos relevantes, e reduziu o esforço manual de triagem em pelo menos 28% e até 91% nos cenários avaliados, conforme o grau de confiança adotado pelo pesquisador. Os resultados indicam que a automação da triagem por agentes de IA na própria plataforma *low-code* n8n é viável e reduz o trabalho manual, desde que o pesquisador permaneça responsável pela decisão final nos casos ambíguos.

Palavras-chave: Revisão Sistemática da Literatura; Triagem Automatizada de Artigos; Inteligência Artificial; Modelos de Linguagem; *Low-Code*.

ABSTRACT

The Systematic Literature Review (SLR) is an essential method in scientific research, but its screening stage, in which thousands of studies are manually assessed, is time-consuming and prone to inconsistencies. Solutions that automate this stage with Artificial Intelligence usually require programming skills, which limits their adoption. This work proposes and validates a modular architecture that automates the screening of primary studies, orchestrated by an AI agent on the low-code platform n8n, supported by a language model that runs locally, on the researcher's own machine. The research follows the Design Science Research paradigm and validates the artifact through three case studies replicating existing reviews, each with a distinct profile of data source and information available for the decision. In the case with the broadest coverage, the system reached 98% recall over the recoverable studies, meaning it retrieved almost all relevant articles, and reduced manual screening effort by at least 28% and up to 91% in the scenarios evaluated, depending on the level of confidence adopted by the researcher. The results indicate that automating screening with AI agents on the n8n low-code platform itself is feasible and reduces

[1]Discente do curso de Tecnólogo em Análise e Desenvolvimento de Sistemas (IFPI).
[2]Docente orientador do IFPI - Campus Piripiri.

manual work, provided that the researcher remains responsible for the final decision in ambiguous cases.

Keywords: Systematic Literature Review; Automated Article Screening; Artificial Intelligence; Language Models; Low-Code.

Data de aprovação: 16/07/2026

1 INTRODUÇÃO

A produção científica cresce em ritmo acelerado. Estima-se que o número de publicações aumente cerca de 4% ao ano, o que faz o acervo mundial dobrar de tamanho a cada 17 anos (Bornmann; Haunschild; Mutz, 2021). Para o pesquisador, isso significa que localizar e ler tudo o que já foi publicado sobre um assunto ficou uma tarefa cada vez mais complexa.

Diante desse cenário, a Engenharia de Software Baseada em Evidências (*Evidence-Based Software Engineering* — EBSE)[3] orienta que as decisões técnicas e gerenciais no desenvolvimento de *software* considerem os resultados de pesquisas publicadas, e não apenas a experiência prática dos profissionais (Dybå; Kitchenham; Jørgensen, 2005). Para reunir e organizar essas evidências, a EBSE recorre à Revisão Sistemática da Literatura (RSL), que é o método de levantamento e não o objeto da decisão. Esse método identifica, avalia e sintetiza a produção científica sobre uma questão específica a partir de um protocolo predefinido. O registro detalhado das etapas nesse plano de trabalho garante que outros pesquisadores possam repetir o processo e verificar os resultados (Kitchenham; Charters, 2007).

O principal obstáculo na execução de uma RSL está no custo operacional da triagem. Após a busca nas bases científicas, o pesquisador se depara com milhares de artigos e precisa avaliar o título e o resumo de cada um. Esse processo é considerado um dos grandes gargalos para a produção de revisões sistemáticas de qualidade (Carver *et al.*, 2013). A atividade é demorada e exaustiva, o que compromete a consistência das decisões à medida que o volume de leitura avança (Kitchenham *et al.*, 2010).

As ferramentas atuais oferecem suporte limitado. O *StArt* (Fabbri *et al.*, 2016) e o *Parsifal* (Parsifal, 2024) organizam as etapas da revisão e controlam o andamento do trabalho, porém a leitura e a decisão sobre cada artigo continuam sendo feitas manualmente. Já soluções que usam Inteligência Artificial (IA) para sugerir as decisões, como o *EmbedSLR* (Matysik; Wiśniewska; Frankowski, 2025), automati-

[3] A sigla EBSE deriva da denominação em inglês e não deve ser confundida com SBSE (*Search-Based Software Engineering*), traduzida como Engenharia de Software Baseada em Busca, campo distinto que aplica algoritmos de busca e otimização a problemas de Engenharia de Software.

zam essa leitura, mas exigem que o pesquisador saiba programar em linguagens como *Python* ou R. Falta, portanto, uma alternativa que automatize a triagem e, ao mesmo tempo, possa ser usada por quem não programa.

Este trabalho propõe preencher esse espaço com um sistema de triagem automática. A proposta é construir um sistema de triagem automática sobre a plataforma n8n, uma ferramenta que permite montar fluxos de trabalho conectando blocos visuais na tela, com pouca ou nenhuma escrita de código. Dentro desse fluxo, um programa de IA capaz de interpretar texto lê o título e o resumo de cada artigo e sugere se ele deve ser incluído na revisão, excluído, ou encaminhado ao pesquisador para uma análise mais cuidadosa. A atuação da IA restringe-se a essa etapa de triagem: ela produz uma sugestão de classificação, e nada é excluído da revisão de forma definitiva sem passagem pela revisão humana. A decisão final permanece, portanto, com o pesquisador.

A investigação é guiada por três questões de pesquisa: (i) é possível automatizar a triagem de artigos em revisões sistemáticas usando IA em uma plataforma sem código? (ii) essa automação reduz o esforço manual do pesquisador? (iii) a qualidade das decisões da IA é suficiente para a prática real?

O objetivo geral é projetar e validar essa arquitetura, que automatiza a importação dos arquivos das bases, a padronização dos dados, a remoção de artigos repetidos e a triagem propriamente dita, na área de Computação. Como objetivos específicos, o trabalho se propõe a: desenvolver os módulos de importação, padronização e remoção de repetidos na plataforma n8n; configurar um modelo de IA treinado para compreender textos, conhecido como Grande Modelo de Linguagem (*Large Language Model*, LLM), para realizar a triagem; e validar o sistema em três estudos de caso, cada um com características diferentes de fonte de dados e de informação disponível para a decisão.

O restante do trabalho está organizado da seguinte forma: o Capítulo 2 apresenta o referencial teórico; o Capítulo 3 descreve a metodologia e o desenvolvimento do artefato; o Capítulo 4 apresenta os resultados e a discussão; e o Capítulo 5 expõe as conclusões e os trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta os fundamentos conceituais que sustentam o desenvolvimento e a avaliação da arquitetura proposta. São abordados a Engenharia de Software Baseada em Evidências, as RSLs, o protocolo PRISMA, o DSR, os LLMs, a validação humana, os sistemas multiagentes e as plataformas *low-code*, além dos trabalhos relacionados à automação da triagem de estudos.

2.1 ENGENHARIA DE SOFTWARE BASEADA EM EVIDÊNCIAS

A EBSE é um paradigma que atua sobre o corpo de estudos científicos publicados na área de Computação, propondo que decisões técnicas e gerenciais sejam subsidiadas por dados empíricos, e não pela intuição ou pela experiência pessoal isolada. Importada da medicina baseada em evidências, a EBSE propõe que as melhores decisões resultam da integração entre os achados científicos disponíveis, a experiência profissional do tomador de decisão e o contexto específico de cada projeto (Kitchenham; Charters, 2007). A RSL é o principal instrumento desse paradigma para identificar, avaliar e sintetizar os achados existentes sobre um tema (Dybå; Kitchenham; Jørgensen, 2005). Cabe registrar que a EBSE entra aqui como contexto que motiva o problema, e não como objeto de estudo: o foco deste trabalho é a etapa de triagem da RSL, e é a essa etapa que a arquitetura proposta se dedica.

2.2 REVISÃO SISTEMÁTICA DA LITERATURA

A RSL é um método de pesquisa secundária que segue um protocolo reprodutível para localizar, selecionar, avaliar e sintetizar estudos primários relevantes a uma linha de pesquisa (Kitchenham; Charters, 2007). A questão de pesquisa específica é definida a partir dessa linha, e somente então orienta a estratégia de busca e os critérios de inclusão e exclusão de estudos. A principal vantagem da RSL em relação às revisões narrativas tradicionais, que sintetizam a literatura de forma livre, sem critérios explícitos de busca ou seleção, é a transparência do processo: cada decisão é documentada, o que permite auditoria e replicação por outros pesquisadores.

O processo de RSL é estruturado em três grandes fases (Kitchenham; Charters, 2007). Na fase de planejamento definem-se a questão de pesquisa, a estratégia de busca e os critérios de inclusão e exclusão. A fase de condução abrange a execução das buscas, a triagem dos resultados e a extração dos dados. A fase de análise e síntese interpreta os dados extraídos e produz o relatório final.

2.2.1 O Protocolo PRISMA

O protocolo *Preferred Reporting Items for Systematic Reviews and Meta-Analyses* (PRISMA) é um conjunto de diretrizes que orienta como os resultados de uma revisão sistemática devem ser relatados, com o objetivo de garantir transparência suficiente para permitir replicação e avaliação crítica (Page *et al.*, 2021). Publicado originalmente em 2009 e atualizado em 2021, o PRISMA é amplamente adotado em RSLs da área de Computação, a exemplo dos trabalhos de Rajapakse *et al.* (2021) e Kitchenham *et al.* (2010).

O componente central do PRISMA é o diagrama de fluxo, definido por Page *et al.* (2021) como o elemento que registra quantos estudos foram identificados nas buscas, quantos foram eliminados por duplicidade, quantos foram excluídos na triagem por título e resumo, e quantos foram incluídos na síntese final. Essa rastreabilidade permite que leitores e avaliadores compreendam as decisões tomadas ao longo do processo seletivo (Page *et al.*, 2021). Neste trabalho, as etapas do fluxo automatizado foram modeladas em conformidade com o fluxo PRISMA, o que permite comparar diretamente os resultados produzidos pelo sistema com os reportados na RSL utilizada como gabarito de comparação (*ground truth*), termo adotado em português no restante do texto.

2.3 *DESIGN SCIENCE RESEARCH*

O *Design Science Research* (DSR) é um paradigma de pesquisa voltado à criação e avaliação de artefatos tecnológicos como forma de contribuição científica. Diferentemente de metodologias descritivas, que visam compreender fenômenos existentes, o DSR parte de um problema prático identificado para projetar, construir e avaliar uma solução (Hevner *et al.*, 2004). O artefato resultante pode assumir diferentes formas: construtos, entendidos como o vocabulário conceitual e os símbolos que descrevem o problema e a solução; modelos, que representam as relações entre esses construtos; métodos, que definem como conduzir uma atividade; e instâncias, que são implementações executáveis. O artefato deste trabalho enquadra-se nas duas últimas categorias: é um método de triagem apoiada por modelo de linguagem e, ao mesmo tempo, sua instância executável na plataforma n8n.

Hevner *et al.* (2004) organizam o DSR em torno de três atividades centrais: construção, em que o artefato é projetado e implementado com base nos requisitos levantados; avaliação, em que o artefato é testado para verificar se atende aos critérios de utilidade e qualidade; e teorização, em que se extraem contribuições científicas generalizáveis a partir da experiência de construção e avaliação. O DSR é reconhecido como uma metodologia adequada para trabalhos em Computação que visam contribuições tanto práticas quanto científicas (Wazlawick, 2017).

2.4 GRANDES MODELOS DE LINGUAGEM E ENGENHARIA DE *PROMPT*

Os LLMs são sistemas de inteligência artificial treinados em grandes corpora textuais por meio de técnicas de aprendizado profundo, capazes de compreender e gerar linguagem natural com alto grau de coerência semântica (Sciurti *et al.*, 2025). Modelos como o GPT-4 (OpenAI, 2023), o Claude (Anthropic, 2024) e o LLaMA (Touvron *et al.*, 2023) realizam tarefas de classificação, sumarização, tradução e raciocínio lógico a partir de instruções fornecidas em linguagem natural, sem ne-

cessidade de treinamento específico para cada domínio (Sciurti *et al.*, 2025).

A aplicação de LLMs à triagem em RSLs tem produzido resultados relevantes, com diferentes estudos avaliando estratégias de instrução e desempenho dos modelos nessa tarefa (Han; Mathrani; Susnjak, 2025; Guo *et al.*, 2024). Han, Mathrani e Susnjak (2025) demonstraram que modelos adequadamente instruídos classificam resumos científicos com acurácia próxima à de especialistas humanos, desde que o *prompt* contenha critérios explícitos e bem estruturados. Guo *et al.* (2024) avaliaram GPT-4, Claude-3 e Mistral na triagem de títulos e resumos. O estudo mostra que técnicas de Engenharia de *Prompt* combinadas com critérios PICO (acrônimo para *Population, Intervention, Comparison, Outcome*, um esquema usado para estruturar questões de pesquisa em termos de população, intervenção, comparação e resultado) elevam a precisão dos modelos.

A Engenharia de *Prompt* é o campo que se ocupa da formulação de instruções para maximizar a qualidade das respostas dos modelos (Han; Mathrani; Susnjak, 2025). Abordagens baseadas em múltiplos agentes também têm sido exploradas para triagem colaborativa, nas quais instâncias diferentes do modelo revisam e confrontam decisões entre si, o que reduz a taxa de erros (Xie *et al.*, 2025).

A adoção de LLMs em processos científicos exige, porém, validação rigorosa. Delgado-Chaves *et al.* (2025) destacam o risco de falsos negativos, situações em que o modelo descarta estudos relevantes de forma incorreta, e propõem métricas de revocação (*recall*) e precisão para avaliar se a automação compromete a integridade da revisão. Adota-se neste trabalho a forma em português, revocação, em todas as seções seguintes. Essa limitação reforça a necessidade de manter o pesquisador como validador final das decisões sugeridas pela IA, abordagem detalhada na seção seguinte.

2.5 VALIDAÇÃO HUMANA DE DECISÕES AUTOMATIZADAS

A abordagem *human-in-the-loop* consiste em manter o ser humano como agente de decisão final em processos parcial ou totalmente automatizados, revisando, confirmando ou corrigindo as sugestões geradas pelo sistema antes de sua efetivação. Em tarefas de triagem assistida por LLMs, essa abordagem é recomendada como salvaguarda contra falsos negativos e erros de classificação, já que modelos de linguagem podem descartar incorretamente estudos relevantes (Delgado-Chaves *et al.*, 2025). Ao preservar a decisão final com o pesquisador, o *human-in-the-loop* concilia o ganho de eficiência da automação com o rigor metodológico exigido pela EBSE, sem transferir integralmente a responsabilidade da decisão científica para o modelo.

2.6 SISTEMAS MULTIAGENTES

Um agente de *software* é uma entidade computacional capaz de perceber seu ambiente, tomar decisões de forma autônoma e executar ações orientadas a objetivos (Wooldridge, 2009). Sistemas multiagentes são arquiteturas compostas por múltiplos agentes que interagem entre si para resolver problemas complexos, dividindo-os em tarefas especializadas (Chen *et al.*, 2024). Com o avanço dos LLMs, arquiteturas multiagentes passaram a automatizar fluxos cognitivos: Wang *et al.* (2024) descrevem como a orquestração de múltiplos agentes viabiliza tarefas inviáveis para um único modelo, e Xie *et al.* (2025) propõem, no contexto de RSLs, um arcabouço multiagente de dois estágios que reduz o esforço manual de triagem.

Cabe demarcar o que este trabalho classifica como agente. Pela definição de Wooldridge (2009), apenas o módulo de triagem satisfaz o critério de autonomia decisória: é ele que interpreta cada registro e decide entre incluir, excluir ou revisar. Os módulos de ingestão, normalização e deduplicação são determinísticos, pois aplicam regras fixas. Por isso, a arquitetura é descrita como modular orquestrada em torno de um único agente cognitivo de triagem, e não como um sistema multiagente pleno. A expressão “agente de IA”, empregada no título e no resumo em sentido amplo, designa esse mesmo componente; “agente cognitivo de triagem” é a forma técnica adotada a partir daqui, por explicitar que a autonomia decisória se restringe ao módulo de triagem. A extensão para um modelo genuinamente multiagente, com um segundo agente verificador, é tratada como trabalho futuro (Seção 5.4).

2.7 PLATAFORMAS *LOW-CODE* E O N8N

Plataformas *low-code* são ambientes de desenvolvimento que permitem construir aplicações e fluxos automatizados por meio de interfaces visuais, reduzindo a necessidade de escrita de código convencional (Velásquez *et al.*, 2024). Sufi (2023) argumenta que essas plataformas são uma solução viável para pesquisadores que precisam implementar sequências automatizadas de processamento de dados sem domínio de linguagens de programação, e o surgimento de plataformas que combinam recursos *low-code* com IA amplia esse potencial para a pesquisa científica (Sundberg; Holmström, 2023).

O n8n é uma plataforma *low-code* de código aberto para orquestração de fluxos de trabalho (*workflows*), na qual serviços, APIs e lógicas de processamento são conectados por meio de nós visuais interligados. Entre suas características relevantes para este trabalho estão: suporte nativo a requisições HTTP, execução de código *JavaScript* em nós específicos, integração com serviços externos como *Google Sheets* e capacidade de execução local em contêineres *Docker*. Essas ca-

racterísticas tornam o n8n uma base adequada para a orquestração de um agente cognitivo em um fluxo de triagem, combinando flexibilidade técnica com acessibilidade operacional.

2.8 TRABALHOS RELACIONADOS

A literatura sobre suporte à triagem em RSLs divide-se em duas categorias (Velásquez *et al.*, 2024): ferramentas de gestão do processo, que organizam o fluxo sem automatizar a decisão, e abordagens baseadas em IA, que buscam automatizar a etapa cognitiva de classificação. A Tabela 1 sintetiza os principais trabalhos identificados quanto a dois critérios centrais para este estudo: automação da triagem e acessibilidade sem programação.

Tabela 1 – Comparativo dos trabalhos relacionados.

Trabalho	**Automatiza triagem**	**Barreira técnica**
StArt (2016)	Não	Não
Parsifal (2024)	Não	Não
EmbedSLR (2025)	Sim	Sim
Han et al. (2025)	Sim	Sim
Yu et al. (2025)	Parcial	Parcial
otto-SR (2025)	Sim	Sim

Fonte: elaborado pelo autor, com base em: Fabbri *et al.* (2016); Parsifal (2024); Matysik, Wiśniewska e Frankowski (2025); Han, Mathrani e Susnjak (2025); Yu *et al.* (2025); Mili *et al.* (2025).

Como evidencia a Tabela 1, nenhuma das soluções existentes combina automação da triagem com uma interface acessível a pesquisadores sem formação em programação. Ferramentas como *StArt* e *Parsifal* eliminam a barreira técnica, mas mantêm o gargalo operacional da decisão manual. Soluções baseadas em IA, como *EmbedSLR* e *otto-SR*, automatizam a triagem, mas exigem programação em *Python* ou acesso a APIs externas. A proposta deste trabalho busca preencher essa lacuna ao encapsular os algoritmos de triagem em agentes visuais orquestrados pelo n8n, sem necessidade de escrever código (Velásquez *et al.*, 2024).

3 METODOLOGIA

Este trabalho adota o DSR, apresentado no referencial teórico, como abordagem de pesquisa, por tratar-se de uma pesquisa aplicada, voltada a resolver um problema operacional concreto por meio da construção de um artefato computacional. As três atividades centrais do estudo correspondem ao desenvolvimento do sistema de triagem, à sua validação por replicação de RSLs publicadas e à discussão dos resultados. A pesquisa combina análise qualitativa, ao comparar as decisões do sistema com as das revisões originais, e análise quantitativa, expressa nas métricas de precisão, revocação e redução de esforço.

Como o trabalho é apresentado em formato de artigo científico, e não de monografia, as três atividades do DSR não recebem seções próprias e distribuem-se pela estrutura do texto. A construção do artefato corresponde à definição dos requisitos e ao desenvolvimento do sistema, tratados neste capítulo; a avaliação corresponde à estratégia de validação (Seção 3.3) e aos resultados dos três estudos de caso; e a teorização é conduzida na discussão das questões de pesquisa e nas contribuições registradas na conclusão. O ciclo de aperfeiçoamento previsto por Hevner *et al.* (2004), em que a avaliação realimenta a construção, aparece de forma explícita nos ajustes feitos entre o Caso 1 e os Casos 2 e 3.

3.1 DEFINIÇÃO DOS REQUISITOS DO SISTEMA

Com base no problema identificado, foram levantados os requisitos funcionais e não funcionais que o sistema deveria atender.

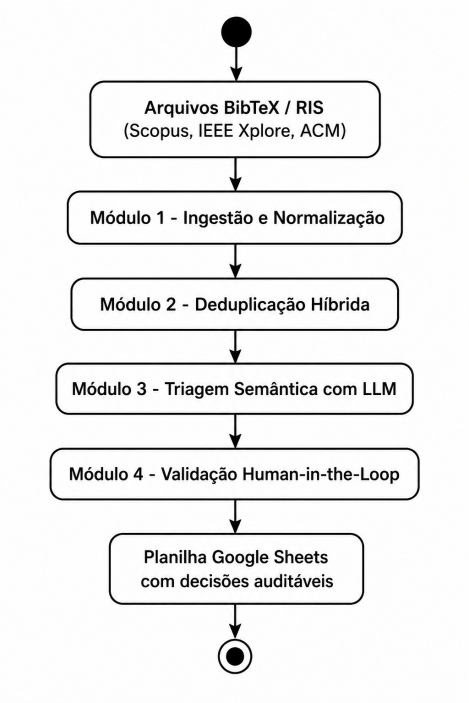
O sistema deve realizar a ingestão automática de arquivos nos formatos *BibTeX* e RIS (*Research Information Systems*, formato-padrão de texto usado para intercâmbio de referências bibliográficas entre gerenciadores) provenientes de múltiplas bases científicas, formatos escolhidos por serem os padrões de exportação oferecidos pelas bases *Scopus*, *IEEE Xplore* e *ACM Digital Library* utilizadas neste trabalho, convertendo-os para um formato JSON (*JavaScript Object Notation*) unificado por meio de normalização dos metadados. Deve também identificar e classificar registros duplicados antes de submetê-los à triagem, realizar a triagem semântica preliminar com base nos critérios de inclusão e exclusão definidos no protocolo da RSL e registrar cada decisão tomada em *log* auditável. Por fim, deve oferecer suporte à revisão humana das sugestões geradas, de modo que o pesquisador possa confirmar, corrigir ou escalar qualquer decisão, conforme a abordagem *human-in-the-loop* apresentada anteriormente.

Quanto aos requisitos não funcionais, a transparência do fluxo de decisão foi estabelecida como critério central: cada decisão produzida pelo sistema deve ser rastreável e acompanhada de justificativa gerada pelo próprio LLM, o que permite ao pesquisador compreender o motivo de cada classificação sugerida. A reprodutibilidade do fluxo de processamento também foi exigida, de modo que a mesma entrada produza sempre os mesmos resultados. A modularidade foi definida como requisito de arquitetura, de modo que cada componente possa ser substituído ou atualizado sem afetar os demais. Por fim, a acessibilidade operacional foi considerada indispensável: o sistema deve ser operável por pesquisadores sem conhecimento técnico em programação.

3.2 DESENVOLVIMENTO DO SISTEMA

O sistema foi desenvolvido sobre a plataforma *low-code* n8n, organizado em quatro módulos sequenciais: ingestão e normalização, deduplicação, triagem semântica e validação humana. A Figura 1 apresenta o fluxo geral do processamento, com os registros bibliográficos percorrendo cada módulo, da entrada dos arquivos até a consolidação das decisões finais em planilha.

Figura 1 – Fluxo geral do sistema de triagem automatizada.



Fonte: Elaborado pelo autor (2026).

3.2.1 Visão Geral da Arquitetura

A arquitetura segue um fluxo sequencial, no qual cada módulo recebe a saída do módulo anterior e produz uma saída padronizada para o módulo seguinte. Por exemplo, o módulo de ingestão entrega ao de deduplicação um conjunto de registros já normalizados em JSON, e este entrega ao de triagem apenas os registros considerados únicos. A orquestração é realizada pelo n8n, que executa os nós em sequência e gerencia o fluxo de dados entre eles. Nós de código *JavaScript* foram utilizados para as lógicas de *parsing*, normalização e deduplicação. A conexão com o modelo de linguagem é feita pelo nó *AI Agent* do n8n, acoplado a um nó *Ollama Chat Model* que aponta para a instância local do Ollama; os resultados finais são exportados para o *Google Sheets* pela integração nativa do n8n.

Todo o sistema executa localmente, na máquina do próprio pesquisador. O n8n é executado em um contêiner *Docker*, o que isola o ambiente e dispensa ins-

talação manual de dependências, e o modelo de linguagem roda via Ollama na mesma máquina. Nenhum dado bibliográfico é enviado a serviços externos, o que preserva a privacidade dos dados e elimina custos de uso de APIs pagas.

A interação do pesquisador com o sistema ocorre por meio de um painel em HTML, que funciona como interface de entrada e acompanhamento do processo. Nesse painel, o pesquisador informa os critérios de inclusão e exclusão da revisão e anexa os arquivos bibliográficos exportados das bases (*BibTeX* ou RIS). Ao acionar a triagem, o painel dispara o fluxo do n8n por meio de uma requisição *webhook*. Concluído o processamento, o painel apresenta, a partir da planilha gerada, o total de registros processados, os contadores de cada decisão (incluir, excluir e revisar) e, para cada registro, o título do trabalho e o grau de confiança da decisão sugerida. O painel atua como auxiliar de operação e visualização; o repositório definitivo dos resultados permanece sendo a planilha no *Google Sheets*.

Em conformidade com a exigência do DSR de que o artefato seja acessível para avaliação por terceiros (Hevner *et al.*, 2004), o código-fonte, os *prompts* completos de cada estudo de caso e os *workflows* do n8n em formato `.json` estão disponíveis publicamente no repositório do projeto[4].

3.2.2 Modelo de Linguagem e Configurações

O modelo de linguagem adotado foi o llama3.2:3b, executado localmente por meio do Ollama, uma ferramenta que permite executar LLMs de código aberto na própria máquina do pesquisador, sem depender de serviços externos. A execução local atendeu a dois critérios: privacidade dos dados bibliográficos processados e independência de custos de API.

A temperatura do modelo foi fixada em zero. Em LLMs, a temperatura controla o grau de aleatoriedade na geração de texto: valores mais altos produzem respostas mais variadas a cada execução, enquanto a temperatura zero aciona a decodificação gulosa (*greedy decoding*), na qual o modelo sempre seleciona o *token* de maior probabilidade a cada passo. Com decodificação gulosa, a mesma entrada produz a mesma decisão de triagem de forma praticamente determinística. Cabe a ressalva de que o determinismo não é absoluto: variações de ordem de operações em ponto flutuante e de *batching* no motor de inferência podem, em casos raros, alterar a saída. Na execução local via Ollama, com lotes controlados, essa variabilidade residual é desprezível, o que atende à reprodutibilidade exigida nos requisitos não funcionais do sistema.

A reprodutibilidade, porém, não é o único motivo da escolha. Três outras razões sustentam a temperatura zero neste contexto. A primeira é a natureza da

[4]Disponível em: `https://github.com/C1ceroAnd/rsl_triagem`.

tarefa: a triagem é um problema de classificação, com um conjunto fechado de saídas possíveis, e não uma tarefa de geração de texto em que a variação entre execuções agrega alternativas úteis. Nesse cenário, a aleatoriedade não produz diversidade proveitosa, apenas ruído sobre a decisão. A segunda é a exigência de auditabilidade herdada do PRISMA e da própria EBSE: se a mesma entrada puder gerar decisões diferentes a cada execução, o registro do processo seletivo deixa de ser verificável por terceiros, e a rastreabilidade que justifica o uso de um protocolo se perde. A terceira é a validade da comparação entre os estudos de caso: com temperatura maior que zero, as diferenças de desempenho observadas entre os Casos 1, 2 e 3 poderiam decorrer da variação estocástica do modelo, e não das características de cada protocolo. Fixar a temperatura em zero elimina essa variável de confusão e permite atribuir as diferenças ao que de fato se pretendia comparar.

Convém não confundir esse mecanismo com a semente aleatória (*seed* = 123) citada nos estudos de caso. São controles de natureza distinta: a temperatura zero governa a geração de texto do modelo, tornando a decisão de triagem reproduzível a cada execução; a semente governa a seleção aleatória da amostra de registros usada na validação manual, e serve apenas para que essa amostragem possa ser repetida por terceiros. Um atua sobre a saída do LLM, o outro sobre o sorteio dos registros avaliados, e nenhum substitui o papel do outro na reprodutibilidade do trabalho.

3.2.3 Ingestão e Normalização

O módulo de ingestão importa os arquivos bibliográficos das bases *IEEE Xplore*, *ACM Digital Library* e *Scopus* e os converte para um formato JSON unificado. Cada base apresenta particularidades de exportação que exigiram estratégias de *parsing* específicas.

Os arquivos da *Scopus* são exportados em *BibTeX* com um cabeçalho de metadados que deve ser removido antes do *parsing*; as entradas são separadas pelo padrão `\n\n@`. Os da *IEEE Xplore* são exportados em formato RIS com entradas concatenadas sem quebra de linha, no padrão `}@`, tratado por expressão regular na separação dos registros. Os da *ACM Digital Library* são exportados em *BibTeX* sem cabeçalho, com o mesmo separador da *Scopus*, e devem ser obtidos diretamente da plataforma para evitar marcações HTML presentes em exportações via portais intermediários.

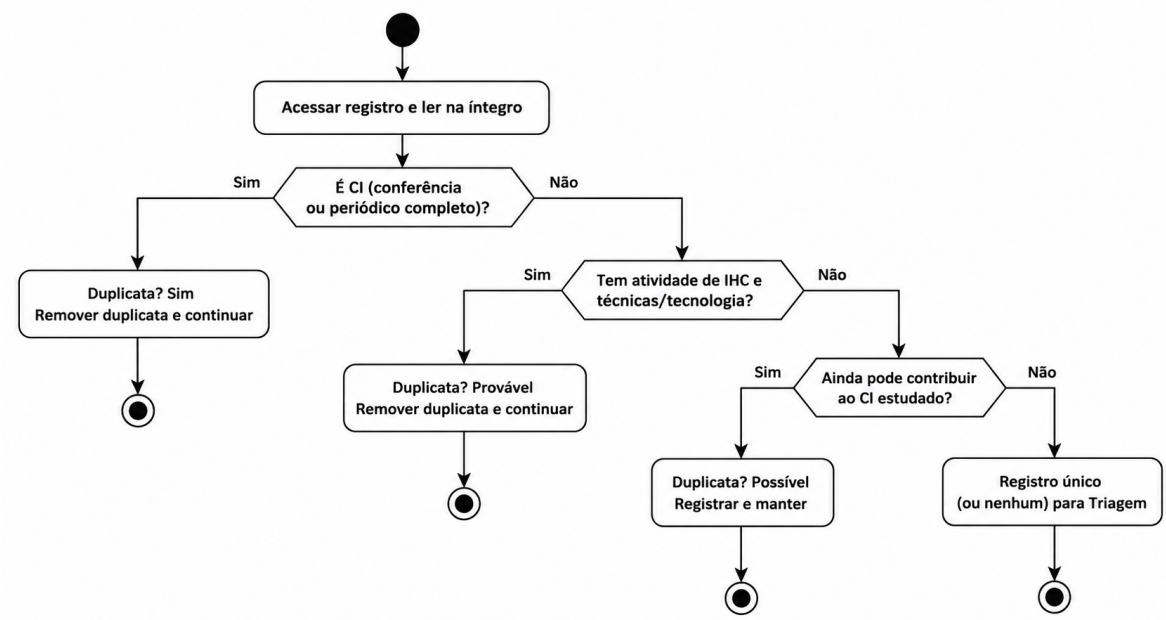
Após o *parsing*, cada registro é convertido para um objeto JSON com campos padronizados: título, autores, ano, *venue*, DOI e resumo. Campos ausentes são preenchidos com valores nulos. Critérios determinísticos de exclusão, como o intervalo de datas do protocolo, são aplicados neste módulo por lógica de código

e não pelo LLM, o que mantém a consistência das decisões e elimina erros de interpretação do modelo.

3.2.4 Deduplicação Híbrida

O módulo de deduplicação identifica registros repetidos provenientes de diferentes bases e os classifica em três categorias, conforme a lógica da Figura 2. Duplicatas exatas têm correspondência direta de DOI e são removidas automaticamente. Duplicatas prováveis apresentam alta similaridade de título e ano, calculada por algoritmo textual, e são marcadas automaticamente ao atingir o limiar definido. Duplicatas suspeitas são casos ambíguos com similaridade parcial ou DOI ausente; esses registros são sinalizados e encaminhados para revisão humana antes de qualquer exclusão.

Figura 2 – Lógica de classificação do módulo de deduplicação híbrida.



Fonte: Elaborado pelo autor (2026).

A classificação em três níveis evita que estudos relevantes sejam descartados automaticamente sem supervisão, e mantém o processo rastreável do início ao fim.

A lógica de classificação da Figura 2 é a mesma nos três estudos de caso, mas sua implementação variou por uma restrição operacional. Nos Casos 2 e 3, a deduplicação foi executada nativamente no n8n, como parte integrada do fluxo. No Caso 1, a deduplicação foi conduzida por um *script Python* externo, de natureza determinística, que aplica exatamente a mesma lógica de três níveis descrita na Fi-

gura 2, sem qualquer intervenção do modelo de linguagem. O recurso a esse *script* não decorre do número de registros em si, pois 935 entradas são poucas para o processamento de texto: o gargalo esteve na execução conjunta, na mesma máquina, da inferência local do LLM e da comparação de similaridade entre todos os pares de registros mantida em memória durante a deduplicação. Para o *hardware* disponível, essa combinação obrigou o processamento em lotes, e a deduplicação sobre a totalidade dos registros passou a ocorrer fora do fluxo do n8n. A diferença de implementação não altera o resultado da classificação, pois as regras de similaridade são idênticas, mas é registrada aqui de forma explícita e retomada entre as limitações do trabalho na Seção 5.3, porque a execução externa reduz a acessibilidade do fluxo para pesquisadores sem conhecimento de programação.

3.2.5 Triagem Semântica com LLM

O módulo de triagem submete cada registro ao modelo de linguagem para uma decisão preliminar de inclusão ou exclusão. O agente recebe um *prompt* estruturado com o título e o resumo do estudo, os critérios de inclusão e exclusão do protocolo e instruções sobre o formato de resposta esperado. O *prompt* completo, comum aos três casos, está disponível no repositório do projeto.

A engenharia de *prompt* seguiu as recomendações de Han, Mathrani e Susnjak (2025): critérios explícitos e bem estruturados são determinantes para a acurácia dos LLMs na triagem. O *prompt* inclui exemplos de classificação para calibrar o comportamento do modelo, pois modelos de menor porte tendem a colapsar os graus de confiança para poucos valores fixos sem referências. A resposta é retornada em JSON com três campos: decisão (`incluir`, `excluir` ou `revisar`), grau de confiança e justificativa textual. Esta última é gerada pelo próprio LLM, que explica em linguagem natural o motivo da decisão sugerida, exigência que atende ao requisito de transparência definido anteriormente. Registros classificados como `revisar` são encaminhados para validação humana, independentemente do grau de confiança.

3.2.6 Organizador de Resultados e Tratamento de Erros

A saída de um LLM em formato textual não é garantidamente um JSON válido, o que exige uma camada de resiliência entre a triagem e o armazenamento. Essa camada, implementada no nó Organizador de Resultados, trata as irregularidades mais frequentes na resposta do modelo antes de gravar o registro. O extrator de JSON emprega contagem de chaves balanceadas para delimitar o objeto de resposta mesmo quando o modelo o envolve em blocos de *markdown* (‘‘‘`json`), e normaliza aspas curvas e vírgulas finais que quebrariam um *parser* convencional.

A captura do grau de confiança tolera variações de nomenclatura do campo, com valor de reserva quando ele não é encontrado.

Cada registro processado recebe um marcador de estado (`status_parse`) que assume um de cinco valores: `OK` para respostas íntegras, `OK_SEM_CONFIANCA` quando a decisão é válida mas o grau de confiança está ausente, `PARSE_PARCIAL` para respostas recuperadas apenas em parte, `ERRO_PARSE` para respostas irrecuperáveis, e `OVERRIDE_TIPO` quando a trava determinística de tipo documental sobrepõe a decisão do modelo. Esse marcador preserva a rastreabilidade das falhas em vez de descartá-las silenciosamente: um registro com `ERRO_PARSE` é encaminhado à revisão humana, não excluído. A introdução dessa camada reduziu a zero os erros de *parse* que, em versões iniciais do fluxo, atingiam dezenas de registros por lote.

3.2.7 Validação *Human-in-the-Loop*

O módulo de validação grava todos os registros processados, com suas decisões, graus de confiança e justificativas, em uma planilha permanente no *Google Sheets*, que funciona como o repositório central e definitivo dos resultados do fluxo, conforme o conceito de *human-in-the-loop* apresentado anteriormente. Cada linha registra o título, os autores, o ano, o DOI, a fonte, a decisão sugerida, o grau de confiança, a justificativa textual e um campo editável para a validação final do pesquisador. O painel HTML descrito anteriormente consome esses dados para exibir o total processado, os contadores por decisão e a confiança de cada registro, mas a revisão e edição finais ocorrem sobre a própria planilha.

A revisão efetiva dos registros sobre a planilha, e não sobre uma interface de triagem dedicada, é uma escolha que privilegia a simplicidade de implementação e a familiaridade do pesquisador com planilhas, ao custo de uma experiência menos ergonômica do que a de ferramentas especializadas de RSL. O desenvolvimento de uma interface de revisão dedicada é registrado como trabalho futuro.

O pesquisador pode confirmar a decisão sugerida, substituí-la ou marcar o registro para análise de texto completo. O histórico completo das decisões é preservado na planilha, o que possibilita a geração do diagrama PRISMA ao final da triagem. O sistema atua como ferramenta de apoio; a decisão final permanece com o pesquisador.

3.3 ESTRATÉGIA DE VALIDAÇÃO

A automação da triagem de RSLs por agentes de IA em plataformas *low-code* é um tema ainda pouco explorado, sem protocolos de avaliação consolidados na literatura. Essa característica de inovação impôs uma dificuldade metodológica concreta: não havia um procedimento de teste e validação já estabelecido para

este tipo de artefato, de modo que a própria estratégia de validação precisou ser construída ao longo do trabalho. A opção adotada foi replicar revisões sistemáticas já publicadas e comparar as decisões do sistema com seus resultados conhecidos, o que exigiu selecionar RSLs cujos dados fossem recuperáveis e reproduzíveis, um critério que descartou vários candidatos antes da definição final dos casos.

A dificuldade de replicar uma RSL publicada não é particularidade deste trabalho. Kitchenham *et al.* (2010) mostram que revisões sistemáticas em Engenharia de Software raramente são reproduzíveis na prática, porque o relato publicado costuma omitir informações necessárias à repetição do processo seletivo. Essa limitação apareceu já na seleção dos casos: das [N] revisões avaliadas como candidatas, a maior parte foi descartada por não publicar a expressão de busca completa, por não disponibilizar a lista dos estudos incluídos ou por recorrer a bases sem exportação estruturada. Uma revisão publicada em 2025 chegou a ser considerada e foi abandonada porque a lista de estudos incluídos não constava do artigo e o contato com os autores não teve resposta; sem essa lista não há gabarito, e a revocação não pode ser calculada. Os três casos efetivamente utilizados são, portanto, os que satisfizeram ao mesmo tempo três condições: protocolo de busca reproduzível, lista de estudos incluídos verificável e base bibliográfica acessível. Essa escassez de revisões replicáveis é, em si, um achado relevante, porque restringe o conjunto de referências disponíveis para avaliar qualquer ferramenta de triagem automatizada.

A validação foi conduzida por meio de três estudos de caso, estratégia recomendada pelo DSR para avaliar artefatos cujo desempenho pode ser aferido por comparação com um resultado de referência (Hevner *et al.*, 2004). Os casos diferem quanto ao tema, à base bibliográfica, ao modo de triagem e à origem do gabarito de comparação. Nos Casos 1 e 2, o gabarito provém de revisões sistemáticas publicadas por terceiros, cujas listas de estudos incluídos existem independentemente deste trabalho. No Caso 3, por não haver RSL publicada sobre o tema, o gabarito foi construído por avaliação manual do próprio pesquisador, o que reduz a força probatória dessa validação e é retomado nas limitações do trabalho. A Tabela 2 sintetiza os três protocolos.

3.3.1 Caso 1 – Replicação de Rajapakse et al. (2021)

A primeira referência de validação foi a RSL de Rajapakse *et al.* (2021), *Challenges and Solutions when Adopting DevSecOps*, escolhida porque seu protocolo de busca está integralmente documentado, o processo seletivo segue o PRISMA e o tema pertence à área de Engenharia de Software. Reproduziu-se o protocolo original nas bases *Scopus*, *IEEE Xplore* e *ACM Digital Library*, e os arquivos exportados foram submetidos ao sistema, com triagem em modo título e resumo. As decisões

Tabela 2 – Comparação dos protocolos de validação nos três estudos de caso.

Caso	**Tema**	**Base(s)**	**Modo de triagem**	**Origem do gabarito**
1	*DevSecOps*	*Scopus*, IEEE, ACM	Título + resumo	Externa (RSL publicada)
2	Computação Desplugada	*Google Scholar*	Somente título	Externa (RSL publicada)
3	*Code Smell* + *Deep Learning*	*Scopus*	Título + resumo	Interna (avaliação manual)

Fonte: Elaborado pelo autor (2026).

automáticas foram então comparadas com os 54 estudos incluídos na revisão original. Por ser o caso de maior volume, foi também o que exigiu o processamento em lotes e a deduplicação externa descritos anteriormente, cujas implicações são retomadas na Seção 5.3.

3.3.2 Caso 2 – Replicação de Santos, Gama e Farias (2019)

A segunda validação replicou o mapeamento sistemático de Santos, Gama e Farias (2019) sobre Computação Desplugada no ensino brasileiro, cujo gabarito reúne 33 estudos. A busca foi conduzida em base única, o *Google Scholar*, com os termos do protocolo original, restrita ao período de 2015 a 2018 e a publicações em português. Duas restrições da base condicionaram o processamento: a exportação não fornece DOI, o que exigiu deduplicação por título em lugar de identificador único; e não fornece resumos, de modo que a triagem foi executada apenas com o título. Essa segunda restrição é estrutural da fonte de dados, não uma escolha de projeto, e limita o desempenho alcançável, pois o modelo decide sem o contexto do resumo (Gusenbauer; Haddaway, 2020).

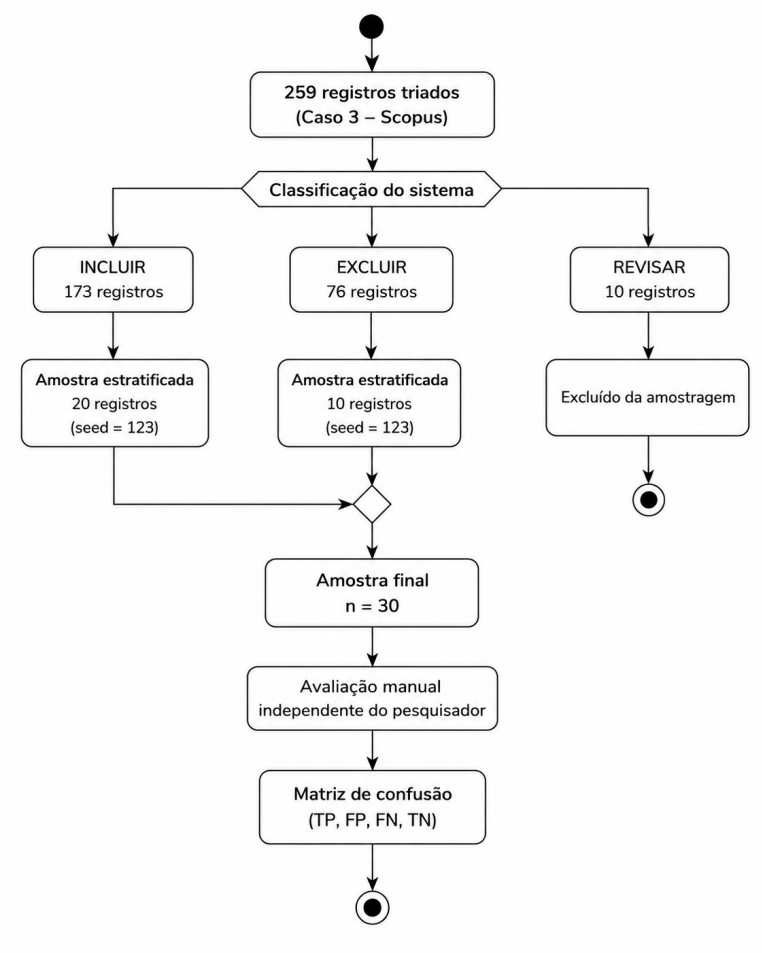
3.3.3 Caso 3 – *Code Smell* e *Deep Learning*

O terceiro caso avalia o sistema em um tema livre, sem RSL publicada como gabarito externo, para testar sua adaptação a um domínio distinto dos anteriores: a detecção de *code smells* por técnicas de *deep learning* e *machine learning*. A busca foi realizada na *Scopus*, base única, restrita ao período de 2022 a 2025, com uma *string* que combina os termos do tema por meio de operadores de proximidade. A *ACM Digital Library* foi descartada nesta etapa por não oferecer suporte a esses operadores, o que impediria uma comparação consistente entre as bases. A triagem foi executada em modo título e resumo.

Como não existe gabarito publicado, a validação foi conduzida por avaliação manual de uma amostra estratificada de 30 registros, extraída com semente aleatória fixa (*seed* = 123) para garantir reprodutibilidade: 20 do grupo `incluir` e

10 do grupo `excluir`. O grupo `revisar` ficou fora da amostragem por representar, por definição, casos de baixa confiança do próprio sistema. Cada registro foi lido e classificado pelo pesquisador de forma independente da decisão do sistema, e as classificações manual e automática foram cruzadas em uma matriz de confusão, conforme a Figura 3.

Figura 3 – Fluxo de amostragem estratificada do Caso 3.



Fonte: Elaborado pelo autor (2026).

Duas limitações condicionam a leitura deste caso. A primeira é o risco de viés de confirmação, já que o avaliador da amostra é o mesmo pesquisador que definiu a busca e os critérios; mitigou-se esse risco pela separação temporal entre a triagem e a avaliação manual. A segunda é a margem de erro amostral de aproximadamente ± 17 pontos percentuais (proporção de 50%, confiança de 95%, com correção para população finita), o que situa os resultados do caso como indicativos de tendência, não como estimativa pontual precisa (Wohlin *et al.*, 2012).

3.4 CRITÉRIO DE COMPARAÇÃO

Os três estudos de caso não têm o mesmo peso probatório: os Casos 1 e 2 oferecem validação externa por gabarito publicado, enquanto o Caso 3 oferece validação interna sujeita ao viés já declarado. Na comparação das métricas, adota-se a revocação como critério primário, em razão da assimetria dos custos de erro na triagem: um estudo relevante descartado pelo sistema (falso negativo) é perda irreparável, pois nunca chega à avaliação manual, enquanto um estudo irrelevante incluído (falso positivo) apenas acrescenta trabalho à etapa de validação humana (Han; Mathrani; Susnjak, 2025). Entre dois resultados com o mesmo F1-*score*, prefere-se o de maior revocação.

4 RESULTADOS E DISCUSSÃO

Esta seção apresenta os resultados dos três estudos de caso descritos anteriormente, seguidos da síntese comparativa, das limitações observadas no modelo de linguagem e da discussão, na qual as questões de pesquisa são respondidas individualmente.

4.1 CASO 1 – *DEVSECOPS* (RAJAPAKSE ET AL., 2021)

Este caso replica a RSL de Rajapakse *et al.* (2021) e constitui o cenário de maior cobertura da avaliação, por reunir registros de três bases científicas com título e resumo disponíveis para a triagem. A análise a seguir apresenta o fluxo de processamento, a recuperação dos estudos do gabarito, as métricas de desempenho e os erros observados.

4.1.1 Corpus Processado e Fluxo de Triagem

A execução do sistema processou 935 registros das três bases: *Scopus* (416), *IEEE Xplore* (314) e *ACM Digital Library* (205). A Tabela 3 compara esse volume com os 460 registros da busca original de Rajapakse *et al.* (2021).

Tabela 3 – Comparação do volume de registros por base entre a RSL original e a replicação (Caso 1).

Base	**Original (2020)**	**Replicação (2026)**	**Variação**
Scopus	250	416	+66,4%
IEEE Xplore	134	314	+134,3%
ACM Digital Library	76	205	+169,7%
Total	**460**	**935**	**+103,3%**

Fonte: Elaborado pelo autor (2026), com base em Rajapakse *et al.* (2021).

O crescimento é desigual entre as bases: a ACM mais que dobrou, enquanto

a *Scopus* cresceu de forma mais moderada. Três fatores explicam essa diferença de volume em relação à busca original. O primeiro é o intervalo de quase seis anos entre a coleta de Rajapakse *et al.* (2021), em 2020, e esta replicação, em 2026: nesse período novos trabalhos sobre *DevSecOps* foram publicados e passaram a compor o corpus. O segundo é a indexação retroativa, ou seja, a inclusão, por parte das bases, de artigos antigos que só depois foram catalogados, o que aumenta a contagem mesmo para o intervalo de anos já coberto pela busca original (Bornmann; Haunschild; Mutz, 2021). O terceiro é o ritmo desigual de crescimento entre bases: a ACM ampliou sua cobertura de forma mais acentuada que a *Scopus* no período, o que responde pela variação mais expressiva observada naquela fonte. A soma desses três fatores, e não uma falha de coleta, é o que justifica o volume maior; vale notar que um corpus de entrada mais amplo tende a favorecer a revocação, já que reduz a chance de um estudo relevante ficar de fora da busca. Em contrapartida, essa diferença de composição entre o corpus de 2020 e o de 2026 limita a comparabilidade estrita da replicação, ponto retomado entre as limitações do trabalho na Seção 5.3.

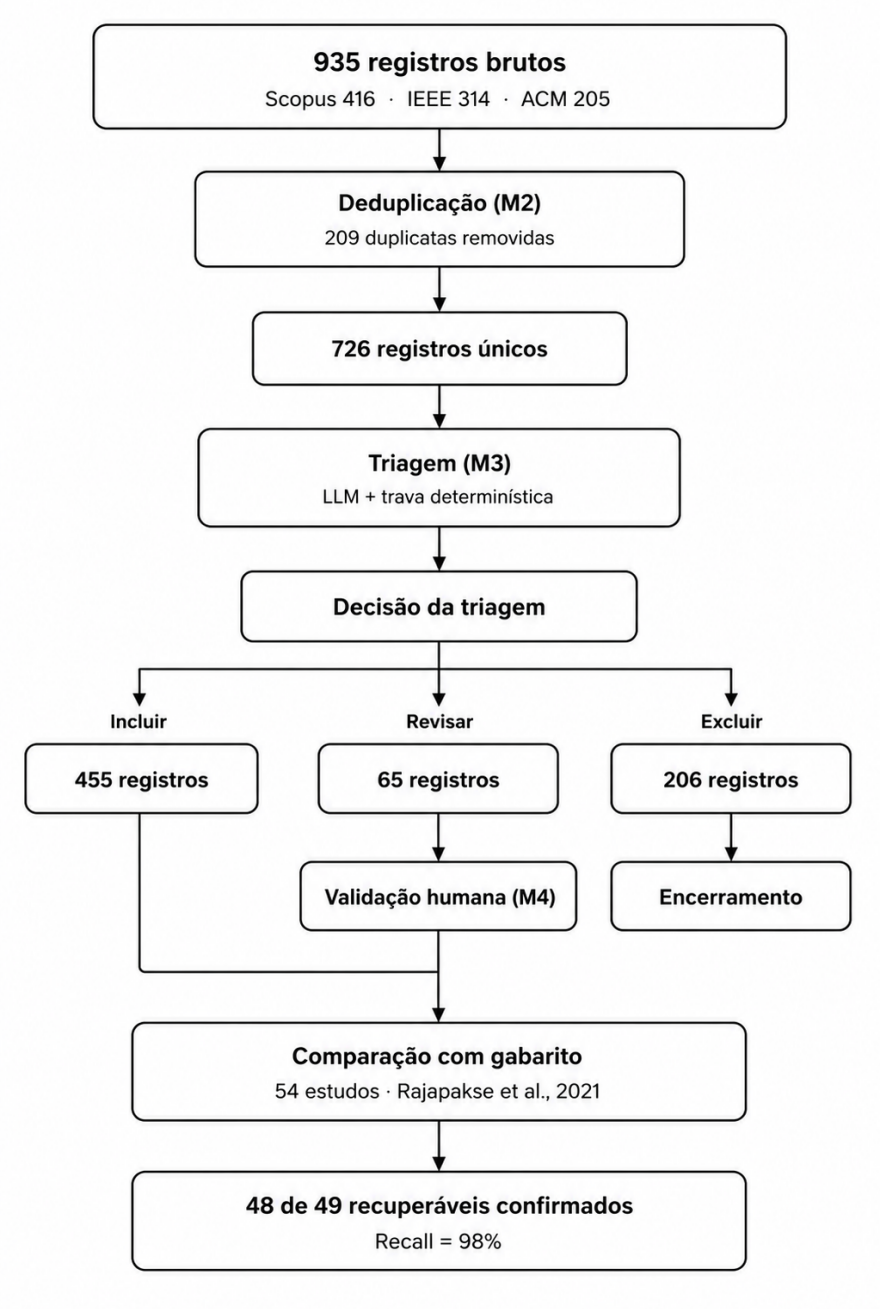
A replicação só foi possível porque Rajapakse *et al.* (2021) publicam o protocolo de busca de forma completa, com os descritores e os operadores booleanos que os combinam. Reproduziram-se essas mesmas expressões em cada base, com as adaptações mínimas de sintaxe que cada plataforma impõe, o que mantém a busca comparável à original. Essa disponibilidade dos termos combinados foi, na prática, um dos critérios que orientaram a escolha do caso: revisões cujos descritores não estivessem descritos em detalhe não permitiriam uma comparação justa de recuperação.

Após o módulo de deduplicação (M2), o conjunto foi reduzido a 726 registros únicos (-209 duplicatas). O módulo de triagem (M3) classificou a totalidade desses 726 registros, combinando a triagem semântica do LLM com a trava determinística de tipo documental já apresentada, em três categorias: 455 como `incluir`, 206 como `excluir` e 65 como `revisar`. Os registros classificados como `revisar` foram encaminhados ao módulo de validação humana (M4). O percurso completo dos registros ao longo das etapas é apresentado na Figura 4.

4.1.2 Análise da Revocação

Os 54 artigos selecionados por Rajapakse *et al.* (2021) serviram de conjunto de referência, ou gabarito. Desses, cinco (P23, P26, P33, P39 e P48) não estavam presentes no corpus coletado, provavelmente por terem sido incluídos na revisão original via *snowballing* ou bases adicionais. Como um artigo ausente da entrada não pode ser recuperado por nenhum método automatizado, o conjunto efetiva-

Figura 4 – Fluxo de triagem do Caso 1 (935 registros brutos até a classificação final).



Fonte: Elaborado pelo autor (2026).

mente recuperável reduz-se a 49 artigos, dos quais o sistema identificou corretamente 48. O único não recuperado (P07) foi excluído por uma alucinação pontual do modelo, que atribuiu ao resumo a menção a um livro protegido por direitos autorais, sem correspondência com o conteúdo real, comportamento documentado como risco de modelos de menor porte (Delgado-Chaves *et al.*, 2025).

A revocação é reportada sobre dois denominadores. Sobre os 54 estudos do gabarito original, é de 88,9% (48/54); sobre os 49 recuperáveis, é de 98,0% (48/49), com intervalo de confiança de 95% entre 89,3% e 99,6% (aproximação de Wilson). O primeiro valor mede o desempenho de ponta a ponta; o segundo isola o módulo de triagem, objeto de avaliação deste trabalho, e a diferença entre eles deve-se apenas aos cinco estudos ausentes do corpus. Cabe notar que o gabarito é o resultado final da RSL original, obtido após avaliação de qualidade e *snowballing*, enquanto o sistema atua só na triagem por título e resumo, o que torna a comparação conservadora.

Uma revocação alta não deve ser lida de forma isolada. Recuperar quase todos os estudos relevantes tem como contrapartida a inclusão de muitos registros irrelevantes, e o custo desses falsos positivos precisa ser dimensionado antes de se afirmar que o resultado é bom. É o que faz a análise de precisão a seguir, que quantifica esse excedente e discute o trabalho de revisão que ele transfere para o pesquisador.

4.1.3 Análise da Precisão

Com 455 dos 726 registros triados classificados como `incluir`, a validação manual de todo o conjunto seria proibitiva no contexto deste trabalho. Adotou-se, portanto, uma amostragem aleatória estratificada de 30 registros, com semente fixada em 123 para garantir a reprodutibilidade da seleção. Cada artigo amostrado foi avaliado manualmente por meio de consulta ao DOI e leitura do título e resumo, verificando se o artigo atendia aos critérios de inclusão definidos no protocolo de Rajapakse *et al.* (2021): estudos que investigam desafios ou soluções para a adoção de *DevSecOps*, publicados entre 2014 e 2021 em inglês.

Dos 30 artigos amostrados, 10 foram confirmados como verdadeiros positivos, resultando em uma precisão de 33,3%. Por se tratar de estimativa sobre amostra de mesmo tamanho que a do Caso 3 (n = 30), essa precisão carrega a mesma incerteza amostral: intervalo de confiança de 95% entre 19,2% e 51,2% (aproximação de Wilson), o que corresponde a uma margem de aproximadamente $\pm$16 pontos percentuais. A precisão do Caso 1 deve, portanto, ser lida como estimativa de ordem de grandeza, não como valor pontual exato, ressalva que se estende, por consequência, ao F1-*score* derivado dela.

A análise dos 20 falsos positivos revelou que a categoria dominante foi a de

artigos sobre práticas de DevOps ou integração contínua sem foco explícito em segurança, presente em 11 dos 20 casos. Os 9 falsos positivos restantes distribuíram-se entre artigos sobre segurança em contextos não relacionados ao ciclo de desenvolvimento (segurança de redes ou IoT, por exemplo), *surveys* genéricos sobre desenvolvimento ágil e artigos que abordam *DevSecOps* tangencialmente, sem tratar de desafios ou soluções de adoção.

Esse padrão indica que o modelo interpretou os critérios de inclusão de forma permissiva: incluiu artigos que mencionam práticas de DevOps ou segurança sem que haja interseção explícita entre os dois domínios. Uma regra de filtragem adicional, exigindo que o artigo aborde explicitamente a integração de práticas de segurança no ciclo de desenvolvimento de *software*, reduziria esse tipo de falso positivo, embora não tenha sido aplicada nesta versão do sistema.

Esse caso deixa claro que as métricas não medem só o modelo: elas dependem de quão operacionalizáveis são os critérios de inclusão e exclusão. Um critério como o intervalo de anos é objetivo e simples de aplicar, e por isso foi movido para a filtragem determinística. Já um critério como "o artigo trata da integração entre segurança e desenvolvimento" exige um julgamento que nem sempre está explícito no título e no resumo, e é justamente onde o modelo tropeça. Quanto mais subjetivo o critério, mais o desempenho depende da qualidade da instrução no *prompt* e menos ele reflete uma limitação intrínseca do LLM. Isso significa que os números aqui reportados são específicos do protocolo avaliado: critérios mais nítidos tenderiam a elevar a precisão, e critérios mais ambíguos a reduzi-la, ainda que o modelo e a arquitetura permanecessem os mesmos.

4.1.4 Redução de Esforço

A redução de esforço estimada neste trabalho é uma medida de volume, não de tempo cronometrado: quantifica quantos registros deixam de exigir decisão manual, não quantos minutos de trabalho são poupados. Essa distinção é importante para não superestimar o ganho.

Sem o sistema, o pesquisador avaliaria individualmente os 726 registros únicos. O limite superior teórico de redução ocorre quando o pesquisador confia integralmente nas classificações `incluir` e `excluir` do sistema e revisa manualmente apenas os 65 registros `revisar`: nesse cenário, a redução é de 91,0% ($(726 - 65)/726$). Esse número, porém, pressupõe confiança total na saída do sistema, o que a precisão de 33,3% não sustenta: cerca de dois em cada três registros `incluir` são falsos positivos, e um pesquisador rigoroso reavaliaria os 455 registros dessa categoria antes de aceitá-los.

O cenário conservador, portanto, é mais realista: o pesquisador revisa os 455 registros `incluir` e os 65 `revisar`, confiando apenas nas exclusões. Nesse

caso, o sistema elimina a avaliação dos 206 registros `excluir`, uma redução de 28,4% do volume total. A revocação de 98% torna essa confiança nas exclusões defensável: a chance de um estudo relevante ter sido descartado é baixa.

A redução efetiva situa-se entre esses dois limites (28,4% e 91,0%), e sua posição exata depende de quanto o pesquisador confia nas inclusões sugeridas, variável que este trabalho não mediu experimentalmente. Uma avaliação do tempo real de triagem humana, com e sem o sistema, e da taxa de correção das sugestões na etapa M4, é registrada como trabalho futuro necessário para converter esta estimativa de volume em uma medida de esforço efetivo.

Ainda assim, mesmo o limite inferior de 28,4% representa ganho operacional relevante para a triagem.

O F1-*score* do Caso 1 foi de 49,7%, abaixo do relatado por soluções com modelos maiores (Yu *et al.* (2025) reportam F1-*score* acima de 80% com modelos de 7B ou mais parâmetros), mas consistente com um modelo de 3B sem ajuste fino (Sciurti *et al.*, 2025). Sob o critério de prioridade da revocação, o valor não invalida o resultado: com revocação de 98%, o sistema recupera quase todos os estudos relevantes, e o excedente de falsos positivos é absorvido pela validação humana (Delgado-Chaves *et al.*, 2025).

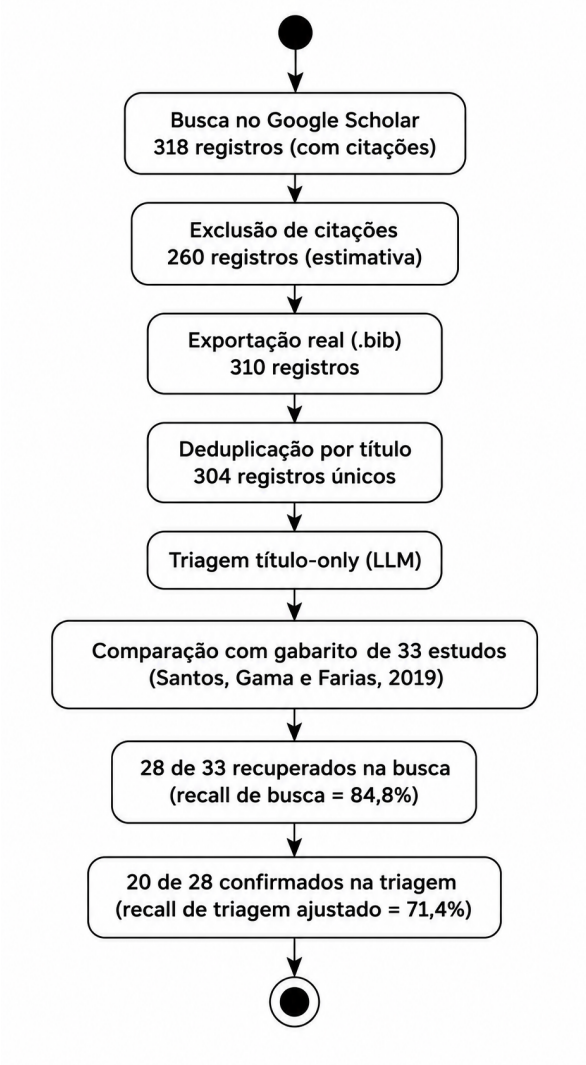
4.2 CASO 2 – COMPUTAÇÃO DESPLUGADA (SANTOS, GAMA E FARIAS, 2019)

O protocolo do Caso 2 foi descrito anteriormente. A cadeia de redução dos registros exige uma distinção entre dois tipos de número: as contagens exibidas pela interface do *Google Scholar*, que são estimativas não confiáveis (Gusenbauer; Haddaway, 2020), e a contagem real dos registros efetivamente exportados em *BibTeX*. A interface indicou 318 resultados com citações e cerca de 260 após aplicar o filtro de exclusão de citações, mas esses valores são apenas estimativas da plataforma. O número que importa para o fluxo é o dos registros de fato exportados: 310 registros no arquivo *BibTeX*, reduzidos a 304 registros únicos após deduplicação por título. É sobre esses 304 que toda a triagem e as métricas a seguir são calculadas. A Figura 5 apresenta o fluxo completo, da busca até a comparação com o gabarito de 33 estudos.

Dos 33 estudos do gabarito de Santos, Gama e Farias (2019), 28 foram recuperados pela busca no *Google Scholar*: revocação de busca de 84,8% (IC 95%: 69,1% a 93,3%). Os cinco estudos não recuperados (E10, E16, E25, E26 e E29) não constituem falha de triagem: são registros ausentes do corpus de entrada, portanto irrecuperáveis por qualquer módulo de classificação, pela mesma razão discutida no Caso 1.

Sobre o conjunto de 304 registros triados em modo somente-título, o sistema confirmou 20 dos 33 estudos do gabarito: revocação de triagem bruta de 60,6%.

Figura 5 – Fluxo de busca e recuperação do gabarito no Caso 2.



Fonte: Elaborado pelo autor (2026).

Ajustando o cálculo para o universo efetivamente recuperável (28 estudos, já descontadas as cinco perdas de busca), a revocação de triagem sobe para 71,4% (20/28, IC 95%: 52,9% a 84,7%). Os intervalos de confiança amplos, aqui e nos demais casos, refletem os tamanhos amostrais modestos e reforçam que os valores devem ser lidos como estimativas de tendência.

Os 8 estudos do gabarito recuperados na busca mas não confirmados na triagem dividem-se em dois grupos, conforme a causa do erro:

- Grupo A – corrigível (3 estudos: E05, E15, E27): erros do modelo, como classificar incorretamente um estudo primário como revisão secundária. Poderiam ser corrigidos com ajuste de *prompt* ou troca de modelo.
- Grupo B – estrutural (5 estudos: E01, E03, E06, E08, E28): casos em que a informação necessária para a decisão correta não está contida no título. Nenhum ajuste de *prompt* recupera uma informação que o modelo nunca recebeu.

Essa divisão confirma a limitação estrutural do modo somente-título antecipada anteriormente: cinco dos oito falsos negativos de triagem (62,5%) são inerentes ao modo de operação imposto pela base de dados, não ao comportamento do modelo. A revocação de triagem ajustada de 71,4% deve, portanto, ser lida como o teto aproximado de desempenho alcançável em modo somente-título para este domínio, e não como um indicador de qualidade do modelo em si.

4.3 CASO 3 – *CODE SMELL* E *DEEP LEARNING*

O protocolo do Caso 3 foi descrito anteriormente, incluindo a ressalva sobre o risco de viés de confirmação e a margem de erro de aproximadamente ± 17 pontos percentuais decorrente do tamanho amostral. O fluxo de amostragem foi apresentado na Figura 3.

Da amostra de 30 registros avaliados manualmente, a Tabela 4 apresenta a matriz de confusão resultante do cruzamento entre a decisão do sistema e a avaliação manual do pesquisador.

Tabela 4 – Matriz de confusão da amostra do Caso 3 (n = 30).

	Manual: relevante	**Manual: não relevante**
Sistema: incluir	15 (VP)	5 (FP)
Sistema: excluir	6 (FN)	4 (VN)

Fonte: Elaborado pelo autor (2026).

A partir da matriz de confusão, a concordância global entre sistema e avaliação manual foi de 63,3% (19/30). Restrita à classe `incluir`, a precisão foi de 75,0%

(15/20, IC 95%: 53,1% a 88,8%) e a revocação de 71,4% (15/21, IC 95%: 50,0% a 86,2%), resultando em F1-*score* de 73,2%.

Os 11 erros da amostra concentram-se em dois padrões, e não se distribuem aleatoriamente:

- Falsos positivos (5 casos: A07, A09, A12, A16, A18): o modelo classificou como `incluir` estudos secundários (*surveys* ou revisões) que deveriam ser excluídos pelo critério de exclusão correspondente. O modelo não reconheceu de forma consistente o padrão textual que caracteriza um estudo secundário.
- Falsos negativos (6 casos: A21, A23, A24, A25, A29, A30): o modelo classificou como `excluir` estudos primários válidos, por aplicar a exigência de que *todos* os critérios de inclusão fossem satisfeitos de forma mais rígida do que o pretendido pelo protocolo.

Os dois padrões de erro têm naturezas distintas e apontam para intervenções diferentes: o primeiro pede reforço do critério de reconhecimento de estudo secundário no *prompt*; o segundo pede flexibilização da lógica de combinação de critérios, hoje tratada como conjunção estrita.

4.4 SÍNTESE COMPARATIVA DOS TRÊS CASOS

A Tabela 5 consolida as métricas dos três casos.

Tabela 5 – Síntese comparativa das métricas dos três estudos de caso.

Caso	**Revocação**	**Precisão**	**F1**	**Tipo de gabarito**	**Perfil**
1	98,0%	33,3%	49,7%	Externo	Revocação máxima
2	71,4% (ajustado)	–	–	Externo	Limitado por modo somente-título
3	71,4%	75,0%	73,2%	Interno	Equilíbrio revocação/precisão

Fonte: Elaborado pelo autor (2026).

A leitura desta tabela deve seguir o critério de prioridade da revocação estabelecido anteriormente, e não o F1-*score* isoladamente. Dois pontos merecem destaque.

Primeiro, o Caso 1 tem a maior revocação (98%) porque seu protocolo, com título, resumo e base ampla de três fontes, fornece ao modelo o máximo de contexto disponível neste trabalho. É o cenário de referência para uso em produção, quando o custo de um falso negativo é inaceitável.

Segundo, o empate de revocação entre os Casos 2 e 3 (71,4% em ambos) não é coincidência informativa por si só, mas resulta de causas completamente diferentes: no Caso 2, o teto é imposto pela ausência estrutural de resumo no modo somente-título; no Caso 3, o teto resulta de uma regra de combinação de critérios mais rígida do que o necessário. Dois protocolos distintos convergem para o mesmo número por razões distintas, o que reforça a utilidade de reportar revocação e precisão separadamente, em vez de reduzir a comparação a uma única métrica combinada.

O Caso 3 tem o maior F1-*score* (73,2%) por apresentar o melhor equilíbrio entre revocação e precisão, não por ser o caso de "melhor desempenho" em sentido absoluto. Sob o critério de prioridade da revocação adotado neste trabalho, o Caso 1 permanece a referência preferencial sempre que a omissão de estudos relevantes for inaceitável, mesmo com precisão mais baixa.

4.5 COMPORTAMENTOS OBSERVADOS NO MODELO

Três comportamentos do `llama3.2:3b` apareceram de forma consistente nos três casos. O primeiro foi o colapso do grau de confiança: em vez de graduar a confiança como solicitado no *prompt*, o modelo retornou valores fixos por decisão (`incluir` = 90, `excluir` = 95, `revisar` = 50) mesmo após reforços no *prompt*, o que impediu o uso dessa coluna como variável de análise. O padrão sugere que o modelo, nesse porte, não estima de fato o próprio grau de certeza: ele parece associar um número mais ou menos fixo a cada rótulo de decisão, como se esse valor fosse parte do formato da resposta e não uma medida calculada. Em outras palavras, a confiança relatada diz mais sobre qual decisão foi tomada do que sobre quão segura ela é, e por isso não pôde ser tratada como uma probabilidade confiável. Modelos maiores tendem a calibrar melhor esse valor, hipótese que se registra entre os trabalhos futuros. O segundo foi a aplicação estrita demais de critérios conjuntivos, origem dos seis falsos negativos do Caso 3. O terceiro foi o reconhecimento inconsistente de estudos secundários (*surveys* e revisões), origem dos cinco falsos positivos do Caso 3 e de parte dos do Caso 1. Esses comportamentos são coerentes com o porte reduzido do modelo e são retomados, como limitações, na Seção 5.3.

4.6 DISCUSSÃO

Esta seção retoma os resultados apresentados para responder, individualmente, a cada uma das três questões de pesquisa formuladas na introdução.

4.6.1 QP1: É possível automatizar a triagem em plataforma sem programação?

Sim. O sistema foi integralmente desenvolvido, executado e validado sobre a plataforma n8n, e os três estudos de caso foram processados sem que a decisão de triagem dependesse de código escrito pelo pesquisador. Os nós de código *JavaScript* presentes no fluxo limitam-se à leitura dos arquivos das bases e à padronização dos dados; a decisão sobre incluir, excluir ou revisar cada artigo cabe ao modelo de linguagem, configurado por meio de instruções em texto.

Essa separação é o que sustenta a resposta afirmativa. Um pesquisador que deseje adaptar o sistema a outra revisão precisa alterar apenas os critérios de inclusão e exclusão, fornecidos como dados de entrada, sem modificar a lógica do fluxo. Uma ressalva, porém, deve ser registrada: no Caso 1, o volume de registros exigiu que a remoção de duplicatas fosse executada por um *script* externo, o que reintroduz, nesse ponto específico, a barreira técnica que a proposta busca eliminar.

Convém dimensionar o peso dessa ressalva. O *script Python* não faz parte do desenho do sistema: ele é um contorno para uma limitação de *hardware*, aplicado a apenas um dos três casos, e a lógica que executa é a mesma já implementada em nós *JavaScript* nos Casos 2 e 3. Ou seja, não há impedimento conceitual para que a deduplicação rode inteiramente dentro do n8n, apenas uma restrição de memória do equipamento usado no desenvolvimento. Nesse sentido, o *script* é um problema real para a acessibilidade, mas circunstancial e não estrutural: em uma máquina com mais recursos, ou com o processamento em lotes integrado ao próprio fluxo, ele deixa de ser necessário. Essa é, justamente, uma das direções registradas em trabalhos futuros.

4.6.2 QP2: A automação reduz o esforço manual do pesquisador?

Sim, embora a magnitude da redução dependa de quanto o pesquisador confia nas sugestões do sistema. No Caso 1, dos 726 registros únicos, apenas 65 foram classificados como `revisar` e exigem análise obrigatória. Se o pesquisador aceitar as demais decisões, a redução do volume de leitura chega a 91,0%. No cenário oposto, em que ele revisa manualmente todos os registros sugeridos para inclusão e confia apenas nas exclusões, a redução é de 28,4%.

O valor real situa-se entre esses extremos e depende do grau de confiança adotado. Cabe destacar que a revocação de 98% torna a confiança nas exclusões defensável, pois é baixa a chance de um estudo relevante ter sido descartado. Esta é uma medida de volume, não de tempo: o trabalho não mediu quantos minutos o pesquisador de fato economiza, o que fica registrado como trabalho futuro.

4.6.3 QP3: A qualidade das decisões é suficiente para uso real?

A resposta depende do perfil da revisão e do que se considera qualidade. Adotando a revocação como critério primário, pela assimetria de custos já discutida, o Caso 1 alcançou 98,0% sobre os estudos recuperáveis, o que significa que o sistema encontrou praticamente todos os artigos relevantes presentes no corpus. Esse valor supera os 86% relatados por Han, Mathrani e Susnjak (2025) e aproxima-se dos 99% de Yu *et al.* (2025), obtidos com modelos de porte muito maior.

A precisão, por outro lado, ficou em 33,3% no Caso 1, abaixo da maioria dos estudos comparáveis. O resultado é esperado para um modelo de 3 bilhões de parâmetros sem ajuste fino para o domínio, e seu impacto prático é limitado: um falso positivo custa apenas o tempo de uma leitura adicional na etapa de validação humana, ao passo que um falso negativo eliminaria um estudo relevante de forma definitiva.

Os três casos delimitam os cenários de uso. O Caso 1, com título e resumo disponíveis e três bases, é o cenário de referência quando a omissão de estudos é inaceitável. O Caso 3 apresenta o melhor equilíbrio entre revocação e precisão, com F1-*score* de 73,2%. O Caso 2 revela o teto do sistema quando a fonte de dados não fornece resumos, situação em que o desempenho é limitado pela informação disponível, não pelo modelo. Em conjunto, os resultados indicam que a arquitetura automatiza a triagem com qualidade suficiente para uso prático, desde que o pesquisador permaneça responsável pela validação dos casos ambíguos. As limitações que restringem a generalização desses achados são sistematizadas na Seção 5.3.

4.7 CUSTOS E VIABILIDADE PRÁTICA

Duas perguntas de ordem prática ficaram em aberto nos resultados: quanto custaria trocar o modelo local por um modelo pago acessado via API, e qual é o custo computacional de executar a ferramenta no formato atual.

A primeira pode ser respondida com uma estimativa de ordem de grandeza a partir do volume do Caso 1. Na triagem, cada registro consome cerca de 900 *tokens* de entrada, entre critérios, título, resumo e instruções, e cerca de 150 de saída, correspondentes à resposta em JSON. Para os 726 registros únicos do caso, o total fica em torno de 0,65 milhão de *tokens* de entrada e 0,11 milhão de saída. Aos preços praticados em 2026, um modelo econômico como o GPT-4o mini, a US$ 0,15 por milhão de *tokens* de entrada e US$ 0,60 de saída, processaria todo o Caso 1 por aproximadamente US$ 0,16; um modelo mais robusto como o GPT-4o, a US$ 2,50 e US$ 10,00 por milhão, custaria cerca de US$ 2,72. Projetada para

revisões maiores, a ordem de grandeza gira em torno de US$ 0,23 por mil registros no modelo econômico e US$ 3,75 por mil no modelo robusto. Em valores absolutos, portanto, o custo de API é baixo e não representa, sozinho, um obstáculo à adoção.

Essa economia, contudo, tem contrapartidas. Enviar títulos e resumos a uma API externa expõe os dados a um serviço de terceiros, cria dependência de conexão e da disponibilidade do provedor e introduz um custo recorrente que cresce com o tamanho da revisão. A opção pelo modelo local, neste trabalho, não se deveu ao custo, mas à privacidade dos dados e à independência de infraestrutura externa. A estimativa acima apenas mostra que um pesquisador disposto a aceitar aquelas contrapartidas trocaria de modelo por um valor modesto e teria acesso ao desempenho dos modelos maiores.

A segunda pergunta diz respeito ao custo computacional da ferramenta como ela está. Esse custo concentra-se na inferência local do LLM. O n8n, o *parsing* e a deduplicação são operações de texto leves, que rodam em qualquer máquina comum. O peso está no modelo de linguagem, que exige memória e processamento proporcionais ao seu porte, e foi esse peso que restringiu a escolha ao `llama3.2:3b` e obrigou o processamento em lotes no caso de maior volume. Na prática, a ferramenta é executável em um computador pessoal razoável, sem GPU dedicada, ao custo de um tempo de processamento maior. Uma máquina com mais memória e com GPU comportaria modelos maiores e execução mais rápida, sem qualquer alteração na arquitetura proposta.

4.8 EVOLUÇÃO ITERATIVA DO SISTEMA

Seguindo o ciclo de avaliação e aperfeiçoamento do DSR (Hevner *et al.*, 2004), a execução do Caso 1 revelou três inconsistências, corrigidas antes dos Casos 2 e 3. A primeira era um filtro de ano *hardcoded* no nó Organizador que sobrescrevia decisões do LLM e excluía artigos indevidamente; foi removido, já que os arquivos são pré-filtrados por data na exportação. A segunda era a não propagação do campo de tipo pelo módulo de limpeza, que fazia livros serem classificados como `incluir`; resolveu-se com a função `normalizarTipo()`, que força `excluir` para os tipos livro e relatório. A terceira, decorrente da anterior, tratava teses e dissertações como livros, apesar de fazerem parte do gabarito do Caso 2; corrigiu-se com um mapeamento próprio para trabalhos acadêmicos.

As três correções ilustram uma propriedade relevante da arquitetura: a separação entre filtragem determinística e semântica permitiu corrigir cada erro de forma isolada, sem alterar o comportamento do LLM nem reexecutar o fluxo do início, o que reforça o valor da modularidade como requisito de projeto.

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho propôs e validou uma arquitetura modular de IA, orquestrada em torno de um agente cognitivo de triagem e construída sobre a plataforma *low-code* n8n, para automatizar a etapa de triagem de estudos primários em RSLs na área de Computação. A validação foi conduzida em três estudos de caso com perfis distintos: replicação de uma RSL sobre *DevSecOps* com base ampla e modo título mais resumo (Caso 1), replicação de uma RSL sobre Computação Desplugada em base única e modo somente-título (Caso 2), e uma validação interna sobre um domínio livre sem RSL publicada como gabarito (Caso 3).

5.1 SÍNTESE DOS RESULTADOS

O Caso 1 alcançou revocação de 88,9% sobre os 54 estudos do gabarito original e de 98,0% sobre os 49 recuperáveis (excluídos 5 ausentes de todas as bases), com precisão de 33,3% e F1-*score* de 49,7%. A redução de esforço manual situa-se entre 28,4% e 91,0%, conforme o pesquisador reavalie ou aceite as inclusões sugeridas pelo sistema. O Caso 2 expôs o teto de desempenho do modo somente-título, imposto pela ausência estrutural de resumo nas exportações do *Google Scholar*: revocação de busca de 84,8% e revocação de triagem ajustada de 71,4%. O Caso 3 obteve revocação de 71,4%, precisão de 75,0% e F1-*score* de 73,2% sobre uma amostra de 30 registros, com margem de erro amostral de aproximadamente ± 17 pontos percentuais.

As três questões de pesquisa foram respondidas de forma afirmativa na discussão dos resultados: a automação é viável em plataforma sem programação, reduz o esforço manual do pesquisador e mantém qualidade suficiente para uso prático, desde que a validação dos casos ambíguos permaneça sob responsabilidade humana.

5.2 CONTRIBUIÇÕES

A primeira contribuição é a arquitetura modular orquestrada para triagem automatizada de RSLs, com módulos especializados e auditáveis (ingestão e normalização, deduplicação híbrida, triagem semântica com um agente de LLM e validação humana) que podem ser adaptados a diferentes protocolos de revisão sem alteração de código.

A segunda é o princípio de critérios via *payload*: os critérios de inclusão e exclusão de cada estudo de caso são fornecidos ao sistema como dados de entrada (JSON), não codificados diretamente no *prompt* do LLM. Essa separação permitiu executar os três estudos de caso sobre o mesmo fluxo, trocando apenas o conteúdo dos critérios e mantendo toda a lógica de orquestração inalterada.

A terceira é a validação empírica em múltiplos perfis de evidência. Diferentemente de trabalhos que avaliam sistemas de triagem em um único cenário, esta pesquisa combinou dois gabaritos externos (RSLs publicadas, com listas de estudos incluídos verificáveis por terceiros) e uma validação interna com viés declarado e margem de erro quantificada. Essa combinação mostra tanto a capacidade quanto os limites da arquitetura em diferentes condições de rigor.

A quarta é a demonstração de viabilidade com modelo local compacto: o uso do `llama3.2:3b` via Ollama mostra que é possível atingir revocação elevada em ao menos um perfil de triagem sem depender de serviços de API externos, o que preserva a privacidade dos dados bibliográficos e elimina custos recorrentes de uso.

5.3 LIMITAÇÕES

Sete limitações merecem registro, organizadas por origem.

Duas limitações são do modelo de linguagem, documentadas de forma consistente nos três casos: o `llama3.2:3b` aplica critérios conjuntivos rígidos de forma mais estrita do que o pretendido, o que gera falsos negativos quando o protocolo exige múltiplas condições simultâneas (Caso 3); e não reconhece de forma confiável estudos secundários (*surveys* e revisões), o que gera falsos positivos nos Casos 1 e 3. A coluna de confiança gerada pelo modelo também não pôde ser usada como variável de análise, por colapsar em valores fixos por categoria de decisão independentemente do caso avaliado.

Uma limitação é estrutural da fonte de dados, não do sistema: o Google Scholar não exporta resumos no formato *BibTeX*, impondo um teto de revocação no Caso 2 que nenhum ajuste de *prompt* recupera, pois a informação necessária para a decisão correta nunca chega ao modelo.

Uma limitação é metodológica: a validação do Caso 3 foi conduzida pelo próprio pesquisador, que também definiu a *string* de busca e os critérios de triagem, o que introduz risco de viés de confirmação. A amostra de 30 registros sobre uma população de 259 produz margem de erro de aproximadamente ± 17 pontos percentuais, o que situa os resultados desse caso como indicativos de tendência, não como estimativa precisa.

Duas limitações decorrem da capacidade de *hardware* da máquina utilizada no desenvolvimento. A primeira restringiu a escolha do modelo de linguagem: modelos de maior porte, que tendem a apresentar melhor desempenho nas tarefas de triagem, não puderam ser executados localmente por exigirem mais memória e capacidade de processamento do que o equipamento disponível comportava, o que motivou a adoção do `llama3.2:3b`, de apenas 3 bilhões de parâmetros. A segunda afetou o Caso 1, de maior volume: os arquivos precisaram ser processados

em lotes, pois o conjunto completo excedia a memória disponível. Esse processamento em lotes impediu que a deduplicação ocorresse dentro do fluxo do n8n, sobre a totalidade dos registros, de modo que a remoção de duplicatas desse caso foi realizada após a conclusão do processamento, por um *script Python* externo aplicado à planilha consolidada. Além de reduzir a acessibilidade do sistema para pesquisadores sem conhecimento de programação, essa solução externa contraria parcialmente o requisito de acessibilidade definido anteriormente, e sua integração nativa ao fluxo depende de um ambiente com mais recursos computacionais.

Uma limitação é de comparabilidade da replicação. O corpus recuperado em 2026 não é idêntico ao que Rajapakse *et al.* (2021) coletaram em 2020: além dos trabalhos publicados no intervalo, a indexação retroativa das bases incorpora artigos antigos catalogados posteriormente, o que altera a composição do conjunto mesmo dentro da janela de anos definida no protocolo original. As métricas do Caso 1 comparam, portanto, a triagem sobre um corpus mais amplo com um gabarito construído sobre um corpus menor. O efeito é assimétrico e atua nos dois sentidos: a entrada mais ampla favorece a revocação, porque reduz a chance de um estudo do gabarito ficar de fora da busca, e penaliza a precisão, porque acrescenta registros relevantes para o tema que jamais poderiam constar de um gabarito fechado em 2020. Parte da precisão de 33,3% observada no Caso 1 decorre dessa assimetria, e não apenas do comportamento do modelo.

5.4 TRABALHOS FUTUROS

Seis direções de extensão são identificadas a partir das limitações e dos resultados obtidos.

A primeira direção é a evolução para uma arquitetura genuinamente multiagente, com a introdução de um segundo agente de LLM atuando como verificador: confrontando a decisão do agente de triagem, sobretudo nos casos classificados como `incluir`, esse verificador poderia reduzir os falsos positivos que hoje limitam a precisão, conforme o padrão de verificação cruzada descrito por Xie *et al.* (2025). Essa extensão transformaria a arquitetura modular atual, orquestrada em torno de um único agente cognitivo, no sistema multiagente pleno discutido no referencial teórico.

A segunda é a substituição por um modelo de maior porte (`qwen2.5:7b` ou `llama3.1:8b`), para verificar se as três limitações comportamentais documentadas (colapso de confiança, rigidez em critérios conjuntivos e reconhecimento de estudo secundário) decorrem do porte do modelo ou do desenho do *prompt*.

A terceira é a validação externa do Caso 3 por um avaliador independente do pesquisador, com cálculo de concordância inter-avaliador (Cohen's Kappa), para reduzir o risco de viés de confirmação identificado anteriormente.

A quarta é a medição do esforço efetivo na etapa M4: registro do tempo real de triagem humana com e sem o sistema, e da taxa de correção das sugestões pelo pesquisador, para converter a estimativa de redução de esforço baseada em volume, apresentada nos resultados, em uma medida de tempo efetivamente poupado.

A quinta é a integração nativa da deduplicação ao n8n. A incorporação da lógica de deduplicação híbrida diretamente em nós *JavaScript* eliminaria a dependência do *script Python* externo, tornando o fluxo completo operável dentro da plataforma *low-code* sem etapas manuais de pós-processamento.

A sexta é a ampliação para outras áreas e protocolos de RSL. Os três casos deste trabalho pertencem todos à Engenharia de Software e à Ciência da Computação, de modo que a generalização demonstrada é intra-Computação. A validação com RSLs de outras grandes áreas do conhecimento permitiria avaliar até onde a arquitetura proposta generaliza para fora da Computação, e identificar ajustes necessários nos módulos de *parsing* e normalização para bases ou formatos ainda não cobertos.

REFERÊNCIAS

ANTHROPIC. *The Claude 3 Model Family: Opus, Sonnet, Haiku*. [*S. l.: s. n.*], 2024. `https://www.anthropic.com/news/claude-3-family`.

BORNMANN, L.; HAUNSCHILD, R.; MUTZ, R. Growth rates of modern science: a latent piecewise growth curve approach to model publication numbers from established and new literature databases. *Humanities and Social Sciences Communications*, v. 8, n. 1, p. 224, 2021. DOI: `10.1057/s41599-021-00903-w`.

CARVER, J. C. *et al.* Identifying barriers to the systematic literature review process. *In:* 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. [*S. l.*]: IEEE, 2013. p. 203–212.

CHEN, J. *et al. Multi-Agent Systems for Dataset Adaptation in Software Engineering: Capabilities, Limitations, and Future Directions*. [*S. l.: s. n.*], 2024. arXiv preprint arXiv:2511.21380.

DELGADO-CHAVES, F. *et al. LLM4ScreenLit: Recommendations on assessing the performance of large language models for screening literature in systematic reviews*. [*S. l.: s. n.*], 2025. arXiv preprint arXiv:2511.12635.

DYBÅ, T.; KITCHENHAM, B. A.; JØRGENSEN, M. Evidence-based software engineering for practitioners. *IEEE Software*, IEEE, v. 22, n. 1, p. 58–65, 2005.

FABBRI, S. *et al.* Improvements in the StArt tool to better support the systematic review process. *In:* PROCEEDINGS of the 20th International Conference on Evaluation and Assessment in Software Engineering. [*S. l.: s. n.*], 2016. p. 1–5.

GUO, W. *et al. Large language models streamline automated systematic review: A preliminary study*. [*S. l.: s. n.*], 2024. arXiv preprint arXiv:2502.15702.

GUSENBAUER, M.; HADDAWAY, N. R. Which academic search systems are suitable for systematic reviews or meta-analyses? Evaluating retrieval qualities of Google Scholar, PubMed, and 26 other resources. *Research Synthesis Methods*, v. 11, n. 2, p. 181–217, 2020. DOI: 10.1002/jrsm.1378.

HAN, B.; MATHRANI, A.; SUSNJAK, T. *Evaluating prompting strategies and large language models in systematic literature review screening: Relevance and task-stage classification*. [*S. l.: s. n.*], 2025. arXiv preprint arXiv:2510.16091.

HEVNER, A. R. *et al.* Design Science in Information Systems Research. *MIS Quarterly*, v. 28, n. 1, p. 75–105, 2004.

KITCHENHAM, B. *et al.* Repeatability of systematic literature reviews. *In:* 14TH International Conference on Evaluation and Assessment in Software Engineering (EASE). [*S. l.*]: BCS Learning & Development, 2010.

KITCHENHAM, B.; CHARTERS, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Keele, UK, 2007.

MATYSIK, S.; WIŚNIEWSKA, J.; FRANKOWSKI, P. K. EmbedSLR: an open-source Python framework for efficient embedding-based screening and bibliometric validation in systematic literature review. *SoftwareX*, Elsevier, v. 32, p. 102416, 2025.

MILI, J. *et al. Automation of Systematic Reviews with Large Language Models*. [*S. l.: s. n.*], 2025. medRxiv preprint. DOI: 10.1101/2025.06.13.25329541.

OPENAI. GPT-4 Technical Report. *arXiv preprint*, 2023. DOI: 10.48550/arXiv.2303.08774. eprint: 2303.08774.

PAGE, M. J. *et al.* The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *BMJ*, v. 372, n71, 2021. DOI: 10.1136/bmj.n71.

PARSIFAL. *Parsifal: Perform Systematic Literature Reviews*. [*S. l.: s. n.*], 2024. https://parsif.al/. Acesso em: 2025.

RAJAPAKSE, R. N. *et al.* Challenges and solutions when adopting DevSecOps: A systematic review. *Information and Software Technology*, Elsevier, v. 141, p. 106700, 2021. DOI: 10.1016/j.infsof.2021.106700.

SANTOS, A.; GAMA, R.; FARIAS, C. Computação Desplugada no ensino da Computação no Brasil: um mapeamento sistemático da literatura. *In:* ANAIS da XIX Escola Regional de Computação Bahia, Alagoas e Sergipe (ERBASE). [*S. l.*]: SBC, 2019. p. 565–574.

SCIURTI, A. *et al.* Compact large language models for title and abstract screening in systematic reviews: An assessment of feasibility, accuracy, and workload reduction. *Research Synthesis Methods*, 2025. DOI: 10.1017/rsm.2025.10044.

SUFI, F. Algorithms in low-code-no-code for research applications: a practical review. *Algorithms*, MDPI, v. 16, n. 2, p. 108, 2023.

SUNDBERG, L.; HOLMSTRÖM, J. Democratizing artificial intelligence: How no-code AI can leverage machine learning operations. *Business Horizons*, Elsevier, v. 66, n. 6, p. 777–788, 2023.

TOUVRON, H. *et al.* LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint*, 2023. DOI: 10.48550/arXiv.2302.13971. eprint: 2302.13971.

VELÁSQUEZ, A. P. *et al.* Systematic literature review of low-code and its future trends. *In:* 2024 12th International Conference in Software Engineering Research and Innovation (CONISOFT). [*S. l.*]: IEEE, 2024.

WANG, Z. *et al.* A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, Springer Nature, v. 1, n. 1, p. 9, 2024. DOI: 10.1007/s44336-024-00009-2.

WAZLAWICK, R. S. *Metodologia de pesquisa para ciência da computação*. Rio de Janeiro: Elsevier Brasil, 2017. v. 2.

WOHLIN, C. *et al. Experimentation in Software Engineering*. [*S. l.*]: Springer, 2012. DOI: 10.1007/978-3-642-29044-2.

WOOLDRIDGE, M. *An Introduction to MultiAgent Systems*. 2. ed. [*S. l.*]: John Wiley & Sons, 2009.

XIE, B. *et al.* Doing Systematic Literature Reviews With Artificial Intelligence Tools: What, Why, and How. *Innovation in Aging*, v. 9, Supplement_1, 2025. DOI: 10.1093/geroni/igaf122.1472.

YU, C. *et al. Leveraging LLMs for Semi-Automatic Corpus Filtration in Systematic Literature Reviews*. [*S. l.: s. n.*], 2025. arXiv preprint arXiv:2510.11409.