INSTITUTO FEDERAL
Rondônia

MINISTÉRIO DA EDUCAÇÃO
Secretaria de Educação Profissional e Tecnológica
Instituto Federal de Educação, Ciência e Tecnologia de Rondônia

Campus Vilhena
Coordenação do Curso Superior em Tecnologia em Análise e Desenvolvimento de Sistemas

LUCCA FERNANDES LIVINO

SISTEMA DE TRIAGEM DE CANDIDATOS COM INTELIGÊNCIA ARTIFICIAL BASEADO EM CRITÉRIOS E QUESTIONÁRIOS CONFIGURÁVEIS

Vilhena – RO
2026

LUCCA FERNANDES LIVINO

SISTEMA DE TRIAGEM DE CANDIDATOS COM INTELIGÊNCIA ARTIFICIAL BASEADO EM CRITÉRIOS E QUESTIONÁRIOS CONFIGURÁVEIS

Artigo entregue como Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO), Campus Vilhena, como requisito parcial para obtenção do grau de Tecnólogo, junto ao Curso Superior em Tecnologia em Análise e Desenvolvimento de Sistemas, sob a orientação do professor José Lucas Brandão Montes.

Vilhena – RO
2026

Ficha catalográfica elaborada pelo Sistema Gerador de Ficha Catalográfica do IFRO.

Livino, Lucca Fernandes.
Sistema de triagem de candidatos com inteligência artificial baseado em critérios e questionários configuráveis / Lucca Fernandes Livino. - Vilhena, 2026.
30 f. : il.

Orientador(a): Prof. Esp. Jose Lucas Brandao Montes.

Trabalho de Conclusão de Curso (Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO, Vilhena, 2026.

1. Recrutamento e seleção. 2. Triagem de candidatos. 3. Modelos de linguagem de grande escala. 4. Critérios configuráveis. I. Montes, Jose Lucas Brandao (orient.). II. Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO. III. Título.

Bibliotecário(a) Responsável: Rosilene Maria do Couto Marques, CRB-11/321

LUCCA FERNANDES LIVINO

SISTEMA DE TRIAGEM DE CANDIDATOS COM INTELIGÊNCIA ARTIFICIAL BASEADO EM CRITÉRIOS E QUESTIONÁRIOS CONFIGURÁVEIS

Artigo entregue como Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO), Campus Vilhena, como requisito parcial para obtenção do grau de Tecnólogo, junto ao Curso Superior em Tecnologia em Análise e Desenvolvimento de Sistemas, sob a orientação do professor José Lucas Brandão Montes.

Aprovado em 10/11/2026 pela banca examinadora.

gov.br Documento assinado digitalmente
JOSE LUCAS BRANDAO MONTES
Data: 17/11/2026 22:28:22-0300
Verifique em https://validar.iti.gov.br

Prof. José Lucas Brandão Montes
Orientador

gov.br Documento assinado digitalmente
WESLEY JHONNES RAMOS ROLIM
Data: 17/11/2026 22:46:00-0300
Verifique em https://validar.iti.gov.br

Prof. Wesley Jhonnes Ramos Rolim
Examinador interno

gov.br Documento assinado digitalmente
ALVINO MORAES DE AMORIM
Data: 18/11/2026 11:46:10-0300
Verifique em https://validar.iti.gov.br

Prof. Alvino Moraes de Amorim
Examinador interno

Resumo. *A triagem manual de candidatos pode ser um processo lento e inconsistente, por isso, este trabalho teve como objetivo desenvolver o TrIA, sistema de triagem em que o recrutador define os critérios de cada vaga, seus pesos e um questionário próprio. O trabalho caracteriza-se como pesquisa aplicada, conduzida segundo a Design Science Research. A aplicação web usa Node.js, MongoDB e React, com o método Kanban. Os critérios obrigatórios são conferidos em código contra um catálogo, e um modelo de linguagem de grande escala, o Google Gemini, atribui a cada critério uma nota com a evidência que a sustenta. Com essas notas, o sistema calcula uma média ponderada pelos pesos, e as candidaturas que alcançam uma nota mínima seguem para a análise do recrutador, que não vê a nota; ela é registrada apenas para auditoria. O sistema foi verificado por rodadas manuais com o modelo de linguagem e por testes automatizados. Nessas rodadas, quando o modelo calculava a nota final, os pesos nem sempre foram respeitados, o que levou esse cálculo para o código. Os 1.961 testes da API e os 180 da interface foram aprovados, e o sistema foi implantado em servidor do Instituto Federal de Rondônia. Como limitação, o sistema não foi avaliado com recrutadores ou profissionais de Recursos Humanos, e os resultados mostram que ele funciona como projetado, não que melhora a triagem de uma equipe real. A contribuição está em deixar com o recrutador tanto a definição do que conta na avaliação quanto a decisão sobre cada candidato.*

Palavras-chave: Recrutamento e Seleção; Triagem de Candidatos; Modelos de Linguagem de Grande Escala; Critérios Configuráveis.

Abstract. *Manual candidate screening can be a slow and inconsistent process; therefore, this work aimed to develop TrIA, a screening system in which the recruiter defines the criteria of each job, their weights and a job-specific questionnaire. The work is characterized as applied research conducted under Design Science Research. The web application uses Node.js, MongoDB and React, under the Kanban method. Mandatory criteria are checked in code against a catalog, and a large language model, Google Gemini, assigns each criterion a score with the evidence that supports it. From these scores, the system computes a weighted average, and the applications that reach a minimum score move on to the recruiter's review; the recruiter does not see the score, which is recorded only for auditing. The system was verified through manual rounds with the language model and automated tests. In these rounds, when the model computed the final score, the weights were not always respected, which moved this calculation into code. The 1,961 API tests and the 180 interface tests passed, and the system was deployed on a server of the Federal Institute of Rondônia. As a limitation, the system was not evaluated with recruiters or Human Resources professionals, and the results show that it works as designed, not that it improves the screening of a real team. The contribution lies in leaving to the recruiter both the definition of what counts in the evaluation and the decision on each candidate.*

Keywords: Recruitment and Selection; Candidate Screening; Large Language Models; Configurable Criteria.

1. Introdução

A contratação de pessoas está entre as decisões de maior consequência para uma organização, porque o recrutamento e a seleção determinam quais profissionais sustentarão seu crescimento (Garcia; Mendes, 2025). O ponto crítico dessa cadeia aparece logo no começo: quando a procura supera largamente o número de vagas, como ocorre com frequência nos cargos operacionais, a seleção fica lenta e cara, e o tempo que a equipe gasta examinando as candidaturas uma a uma deixa de ir para etapas como a entrevista e a integração de quem foi contratado (Siqueira, 2025). A demora é apenas metade do problema, porque quem avalia manualmente trabalha sob sobrecarga, chega a conclusões diferentes diante do mesmo perfil e sofre com vieses de que nem sempre tem consciência e que interferem no julgamento (Yuksel et al., 2026).

O instrumento usado para decidir também limita o resultado. O currículo em formato livre varia em forma e estilo e não segue padrão entre empresas, de maneira que extrair dele informação confiável permanece um problema em aberto (Barducci et al., 2022), e decidir apenas pelo que ele registra descarta bons perfis e deixa passar perfis inadequados (Stuart; Kaneko; Baldwin, 2026).

Diante desse quadro, a inteligência artificial (IA) consolidou-se como vetor de transformação digital nas organizações (Barelli; Leme, 2024), e quem trabalha na seleção tende a vê-la com bons olhos: em levantamento com 47 profissionais da área, 93,6% apontaram a agilidade do processo seletivo como oportunidade e 83% destacaram a otimização da triagem (Garcia; Mendes, 2025).

O avanço pede cautela, no entanto, porque a automação corre o risco de reforçar os vieses que pretendia corrigir quando os critérios não são expostos nem os algoritmos revisados com regularidade (Coggo, 2025). Parte das ferramentas examina o currículo procurando nele palavras-chave definidas de antemão (Siqueira, 2025), e há relato de que essa correspondência literal se combina a filtros eliminatórios (Yuksel et al., 2026). Com isso, bons candidatos deixam de ser localizados apenas porque não descreveram suas qualificações com as palavras cadastradas na ferramenta, ainda que se saiam bem ao alcançar a entrevista (Siqueira, 2025). Algumas abordagens mais recentes reconhecem essas variações de redação, mas não devolvem ao recrutador o controle sobre o que pesa na comparação (Yu et al., 2026).

Trabalhos mais recentes avançam nessa direção ao organizar a avaliação em rubricas específicas da vaga, com nota e justificativa para cada dimensão avaliada, e ao incorporar recomendações da equipe de recursos humanos entre as entradas da avaliação. Porém, os critérios avaliados e as perguntas dirigidas ao candidato são produzidos pelo próprio modelo, a partir da descrição da vaga, e quem conhece a posição não participa da formulação (Yuksel et al., 2026).

Fora dos sistemas automatizados, a seleção de pessoas já tem formas de avaliar todos os candidatos pelos mesmos parâmetros, como a entrevista estruturada, em que o roteiro e o modo de julgar as respostas são iguais para todos. Uma meta-análise recente a colocou à frente de outros cinco métodos na previsão do desempenho no trabalho, mas, por ser presencial e exigir mais tempo e recursos, ela em geral fica para depois da triagem, feita com outros instrumentos (Berry et al., 2024). Nessa primeira etapa, o ATS (*Applicant Tracking System*) estudado por Puccini et al. (2022) filtra os candidatos com perguntas eliminatórias e questionários configurados pela empresa contratante, porém

os autores concluem que o filtro só funciona com respostas curtas e diretas, porque o sistema não consegue relacionar uma resposta mais longa ao perfil que a vaga procura.

O sistema apresentado neste artigo, denominado TrIA (Triagem com Inteligência Artificial), foi pensado para cobrir o que esse filtro deixa de fora: respostas abertas, avaliadas segundo critérios cujo peso é definido pelo próprio recrutador. A premissa é simples: quem conhece a vaga é quem deve perguntar. Antes de publicar a posição, o recrutador define os critérios pelos quais os candidatos serão avaliados, atribui a cada critério o peso que julga adequado e redige um questionário próprio daquela vaga. O candidato responde a esse questionário ao se inscrever, e as respostas passam a compor a avaliação ao lado do currículo. Cabe à IA analisar os currículos e respostas contra os critérios definidos pelo recrutador, atribuindo a cada critério uma nota acompanhada de uma justificativa.

As notas por critério são combinadas em um valor único, entre 0 e 1, comparado a um limiar de compatibilidade, a nota mínima que a candidatura precisa alcançar. Esse limiar é ajustado pelo papel de suporte, responsável pela configuração da triagem, e vale para todas as vagas. A candidatura que o alcança é promovida automaticamente à fila de análise do recrutador, e a que não alcança permanece entre as inscrições. O recrutador nunca vê esse valor nem a justificativa, que são guardados como registro de auditoria e acessíveis apenas ao papel de suporte, porque exibir um número tenderia a ancorar o julgamento de quem decide.

Diante do problema de pesquisa apresentado, o objetivo geral é desenvolver um sistema de triagem de candidatos com o desenho descrito acima, capaz de apoiar equipes de recursos humanos na avaliação de candidaturas. Os objetivos específicos são:

a) desenvolver o módulo pelo qual o recrutador configura a vaga, define os critérios ponderados e redige as perguntas apresentadas ao candidato;
b) implementar a aplicação dos questionários aos candidatos;
c) implementar o mecanismo de análise automatizada que compara currículo e respostas com esses critérios, atribuindo a cada um uma nota justificada por evidência e reunindo essas notas em um *score* ponderado;
d) encaminhar automaticamente à fila de análise as candidaturas que alcancem o limiar de compatibilidade configurado, preservando o *score* e sua justificativa como registro auditável.

2. Fundamentação Teórica

2.1. Recrutamento e seleção diante da transformação digital

Com a informatização do processo seletivo, o setor de recursos humanos deixou de ocupar posição estritamente operacional e ganhou peso na decisão, passando a atuar de forma orientada por dados (Coggo, 2025). Antes disso, os currículos eram examinados manualmente um a um e controlados em planilhas, o que abria espaço para erros e atrasava o processo (Garcia; Mendes, 2025). Porém, com a análise dos dados automatizada, o tempo que antes era gasto em tarefas repetitivas volta à equipe e pode ser utilizado em etapas que exigem intervenção humana (Siqueira, 2025).

Esse ganho traz alguns obstáculos: quem adota essas soluções pode enfrentar mais problemas culturais e éticos do que técnicos, entre eles a compreensão ainda precária do que a Lei Geral de Proteção de Dados (LGPD) exige (Coggo, 2025). A partir desse ponto surge uma das exigências que orientam o sistema apresentado neste artigo: a decisão automatizada precisa permanecer inspecionável por quem responde por ela.

A automação também não torna a avaliação neutra, como indicam estudos que atribuíram a currículos nomes associados a diferentes gêneros e raças. Em um deles, cinco modelos de linguagem, entre eles uma versão anterior do Gemini, pontuaram cerca de 361 mil currículos fictícios, e as notas variaram conforme esses atributos, com um corte simulado, essas diferenças podem mudar em um a três pontos percentuais a chance de contratação (An et al., 2025). Em outro, que testou a triagem por similaridade entre currículo e vaga, os nomes associados a pessoas brancas foram preferidos na maior parte dos testes, e o nome de homem negro perdeu para o de homem branco em todas as comparações (Wilson; Caliskan, 2024). Como o sistema deste artigo não passou por esse tipo de auditoria, não se afirma aqui que ele reduza o viés, e o que ele oferece é o registro da evidência que levou à nota de cada critério.

2.2. Modelos de linguagem e avaliação de respostas

As perguntas dissertativas do questionário produzem respostas em linguagem natural, que precisam ser julgadas pelo contexto e pelo significado, e não pelas palavras que usam. Uma métrica que leva em consideração apenas os termos em comum com uma resposta esperada acaba punindo quem acerta com vocabulário próprio (Cortes; Vieira; Barone, 2024), e a alternativa de ler cada resposta devolve o processo à triagem manual, lenta e inconsistente.

Os modelos de linguagem de grande escala (*Large Language Model*, LLM), pré-treinados sobre uma grande coleção de textos (Cortes; Vieira; Barone, 2024), abriram outra via, a de o próprio modelo ocupar o papel de juiz. Um levantamento sobre essa prática, conhecida como *LLM-as-a-Judge* (Gu et al., 2025), trata a confiabilidade desse julgamento como uma questão ainda sem resposta.

Deng et al. (2025) mediram essa confiabilidade ao usar um LLM, sem ajuste fino, para corrigir respostas curtas de estudantes a questões de ciências, com a pergunta, a rubrica e uma resposta de referência à disposição do modelo. Mesmo com esse apoio, o modelo foi mais generoso do que os especialistas e classificou como parcialmente corretas muitas respostas que eles haviam considerado irrelevantes. A precisão também caiu conforme a escala ficou mais detalhada, passando de 76% de acerto, quando bastava decidir entre certo e errado, para 57% com cinco categorias. A esse problema de julgamento soma-se a alucinação, limitação conhecida dos LLMs, e por isso a resposta gerada pelo modelo precisa ser conferida com uma fonte confiável (Cortes; Vieira; Barone, 2024). No sistema desenvolvido neste artigo, essas duas limitações estão entre os motivos da escolha de usar o modelo como apoio à avaliação das respostas, deixando a decisão final sobre o candidato com o recrutador.

2.3. Inteligência artificial aplicada à triagem de candidatos

Os sistemas de triagem automatizada encontrados na literatura podem ser comparados pelo modo como definem o que será avaliado em cada candidato. Boa parte

dos trabalhos mais recentes sobre o tema, incluindo os mais próximos desta proposta, foi divulgada como *preprint* no arXiv, sem revisão por pares, e por isso é usada aqui para descrever como esses sistemas foram desenhados, e não como prova do desempenho que relatam. Nos que mais se afastam da proposta deste artigo, não há uma lista explícita de critérios, e a avaliação se apoia apenas na descrição da vaga ou nas instruções dadas ao modelo. Por exemplo, no trabalho de Padmaja et al. (2023), quem faz a triagem informa a descrição da vaga, e o sistema ordena os currículos pela semelhança de significado com ela, calculada por similaridade de cosseno. Os que ficam abaixo de uma nota mínima saem da disputa ou perdem prioridade na revisão manual. Essa nota é um único valor de similaridade por currículo. O sistema deste artigo também aplica um corte por nota, mas a nota dele é calculada a partir de critérios avaliados um a um, cada um com a evidência que o sustenta, o que permite identificar qual requisito o candidato atendeu. Essa comparação por semelhança de significado é, aliás, a tendência apontada por uma revisão sistemática sobre a correspondência entre perfis e oportunidades, que também observou que a maioria dos estudos avalia o desempenho do conjunto e explica pouco cada recomendação (Duarte; Souza, 2026). Já no trabalho de Gan, Zhang e Mori (2024), um LLM lê cada currículo e lhe dá uma nota de 0 a 100 acompanhada de um resumo, e os mais bem avaliados seguem para a decisão final. O que se espera do candidato fica apenas na instrução dada ao modelo, que no experimento o coloca no papel de profissional de recursos humanos de uma empresa de tecnologia.

Os trabalhos mais próximos avaliam o candidato por uma rubrica, que Stuart, Kaneko e Baldwin (2026) definem como um conjunto de competências, por exemplo autonomia ou habilidade de programação, cada uma dividida em níveis. No trabalho de Yuksel et al. (2026), o próprio modelo adapta essa rubrica a partir do anúncio da vaga, e cada competência recebe um nível e uma justificativa que aponta a evidência encontrada no material sobre o candidato. O trabalho de Yuksel et al. não descreve pesos entre as competências, e o seu resultado final é uma ordenação dos candidatos, do mais ao menos indicado para a vaga. O sistema deste artigo compartilha com esse trabalho a justificativa apoiada em evidência, dada a cada critério. Difere, porém, em dois pontos: quem define o que será avaliado é o recrutador, e o resultado não ordena os candidatos. O trabalho de Stuart, Kaneko e Baldwin (2026), por sua vez, é um sistema de entrevista automática que parte da mesma constatação deste artigo, a de que o currículo sozinho oferece pouca informação para decidir. Nele, um LLM entrevista o candidato e outro atualiza o nível estimado de cada competência a cada resposta, mas as perguntas são criadas pelo modelo durante a conversa, e a rubrica, feita pelos autores, serve para uma área profissional inteira, e não para uma vaga específica.

Nesses trabalhos, portanto, o que conta na avaliação é definido pelo texto da vaga, pela instrução do modelo, pelo próprio modelo ou pelos autores da pesquisa, e nenhum deles descreve peso por critério ou questionário escrito pelo recrutador. O sistema proposto parte justamente dessa lacuna e deixa a quem conhece a vaga a definição dos critérios, do peso de cada um e das perguntas. Além disso, cada candidatura é avaliada sozinha, de forma individual, sem gerar uma comparação entre diferentes candidatos em uma mesma rodada de avaliação pelo modelo, e o questionário limita o que o modelo lê.

3. Metodologia

O trabalho é de natureza aplicada e produziu um sistema em funcionamento. Quanto ao procedimento, segue a *Design Science Research* (DSR), método em que o próprio pesquisador constrói o artefato que responde ao problema e verifica se ele cumpre o que se propôs, diferentemente do estudo de caso, em que o pesquisador acompanha o fenômeno sem criar a intervenção (Dresch; Lacerda; Cauchick Miguel, 2015). O desenvolvimento juntou duas frentes: a construção da aplicação e a verificação experimental do motor de avaliação, feita em ambiente isolado antes de cada mecanismo entrar no sistema. Essa verificação cumpre aqui o papel de avaliação do artefato, mas se limita a testes técnicos, sem uso do sistema por recrutadores.

3.1. Tecnologias e arquitetura da aplicação

A interface do usuário é construída em React 19 com Vite e TypeScript, apoiada em Tailwind CSS e Radix UI, e consome a interface de programação de aplicações (*Application Programming Interface*, API) do servidor.

O servidor é uma API REST em Node.js com Express. A organização segue quatro camadas com responsabilidades isoladas: as rotas recebem a requisição HTTP e a encaminham aos *middlewares* de autenticação e permissão, os *controllers* validam a entrada, os *services* concentram a regra de negócio, e os *repositories* isolam o acesso aos dados.

Os dados ficam em MongoDB, com Mongoose, escolha que responde à variação na estrutura dos questionários: o número de perguntas, os tipos admitidos e as opções de resposta mudam de vaga para vaga, porque quem os define é o recrutador. Um esquema de tabelas fixas deixaria colunas vazias na maioria das vagas, ou exigiria uma tabela de atributos genéricos que jogaria de volta para o código a montagem de cada questionário. O modelo por documentos aceita essa variação sem nenhum dos dois custos.

A autenticação usa a biblioteca better-auth, a API é documentada em Swagger a partir de anotações no próprio código, e o ambiente de execução roda em contêineres Docker.

3.2. O desenho da avaliação automatizada

A avaliação de cada candidatura reúne três entradas em uma única estrutura. A primeira são os critérios da vaga, definidos pelo recrutador: cada critério tem um nome e um peso percentual, que indica sua importância na nota final, e pode ser marcado como obrigatório. A segunda é o perfil do candidato, formado pelas habilidades, formações, experiências e certificações que ele cadastra em campos próprios do sistema. A terceira são as respostas que o candidato dá ao questionário elaborado pelo recrutador para a vaga, com perguntas dissertativas, de múltipla escolha e de verdadeiro ou falso, que complementam o perfil com o que o recrutador quer saber especificamente sobre aquela vaga.

Antes de qualquer chamada ao modelo, um *gate* determinístico, isto é, uma verificação feita por regras fixas em código, confere os critérios que o recrutador marcou como obrigatórios. O processo tem três passos. Ao cadastrar a vaga, o recrutador indica o tipo de cada critério obrigatório e escolhe o requisito em um catálogo de habilidades, áreas de formação, áreas de atuação e certificações, completando-o, conforme o tipo, com o nível mínimo, o grau da formação ou os meses de experiência exigidos. Ao montar o perfil, o candidato escolhe as suas habilidades,

formações, experiências e certificações no mesmo catálogo e informa o nível, o grau, a situação e a duração de cada uma. Por fim, no início da avaliação de cada candidatura, o *gate* compara os dois lados pelo item do catálogo, que guarda os sinônimos de cada item e as relações entre eles, como a de que quem domina PostgreSQL domina SQL. O Quadro 1 acompanha esses passos em uma vaga de exemplo.

Tipo	**O recrutador exige**	**O candidato informa**	**O *gate* confere**
Habilidade	SQL, nível mínimo intermediário	PostgreSQL, nível avançado	PostgreSQL implica SQL, e avançado supera intermediário: atende
Formação	Graduação em Computação, aceitando curso em andamento	Graduação em Análise e Desenvolvimento de Sistemas, cursando	O curso é sinônimo da área, o grau é o mesmo e a situação é aceita: atende
Experiência	12 meses em desenvolvimento de software	Estágio em desenvolvimento por 8 meses e desenvolvedor *back-end* por 6 meses	Os dois cargos pertencem à área e somam 14 meses: atende
Certificação	AWS Certified Cloud Practitioner	Nenhuma certificação	Requisito não encontrado no perfil: não atende

Quadro 1. Exemplo de verificação dos critérios obrigatórios de uma vaga de desenvolvedor pelo *gate*

No exemplo, a falta da certificação basta para que a candidatura seja marcada como incompatível, com nota zero e o motivo registrado, e ela continua entre as inscrições, sem ser promovida. Assim, um requisito eliminatório não depende de julgamento probabilístico, e nenhuma chamada ao modelo é gasta com candidatura que já está fora. Comparar pelo item do catálogo corrigiu um erro da primeira versão do *gate*, que comparava os nomes como texto e errava sem aviso, aceitando Java onde a vaga pedia JavaScript e recusando PostgreSQL onde ela pedia SQL.

Somente a candidatura aprovada no *gate* segue para a avaliação pelo modelo, que usa *prompting* estruturado: uma rubrica textual, fixa no código, orienta o julgamento do modelo, e a saída volta em JSON validado por *schema*. Para cada critério da vaga, o modelo atribui uma nota de 0 a 1 acompanhada do trecho do perfil ou das respostas que a sustenta. O *score* global não é pedido ao modelo: o sistema o calcula em código como a média dessas notas ponderada pelo peso que o recrutador definiu para cada critério. Com isso, o peso entra na decisão por uma conta fixa e verificável, e não pela interpretação do modelo. A rubrica também proíbe comparar o candidato com os demais, de modo que cada candidatura é julgada sozinha contra a vaga. Mesmo com o formato declarado ao provedor, o sistema confere a saída antes de usá-la. Quando o texto devolvido não pode ser convertido em JSON, ou quando alguma nota fica fora do intervalo de 0 a 1, a resposta é descartada e a avaliação passa ao modelo seguinte da cascata, descrita adiante. Se nenhum modelo devolver uma resposta válida, a candidatura fica pendente, sem nota, e pode ser avaliada de novo.

As respostas ao questionário são tratadas conforme o tipo: a dissertativa é avaliada pelo modelo a partir do texto, enquanto a múltipla escolha e a verdadeiro ou falso chegam já corrigidas em código, contra a alternativa que o recrutador marcou como correta. A rubrica manda não revisar essa correção, para que o resultado de uma pergunta com resposta conhecida não dependa do modelo que atendeu a rodada.

A temperatura, parâmetro que controla quanto as respostas do modelo variam, foi fixada em zero depois de se comparar a mesma requisição repetida em zero e em um valor mais alto. Esse valor reduz a diferença entre execuções, mas não a elimina.

O acesso ao modelo segue uma cascata, isto é, uma sequência de modelos do mesmo provedor em que o seguinte só é usado quando o anterior não consegue responder, por ter atingido o limite diário de uso ou por ter devolvido uma resposta inválida. Os modelos são da família Gemini, do Google (Google, 2026b): o gemini-3.1-flash-lite responde primeiro, e o gemini-3.5-flash-lite é a reserva. O limite diário só se renova no dia seguinte, e cada modelo tem uma cota própria, de modo que esgotar um não afeta o próximo. Já o limite de requisições por minuto se resolve em poucos segundos e, por isso, não leva à troca de modelo: o sistema espera e tenta de novo, com intervalos cada vez maiores.

Cada avaliação registra o modelo que de fato respondeu, que pode não ser o primeiro da lista, para que o registro corresponda ao que aconteceu. Se o provedor falhar, ninguém é reprovado: a candidatura fica pendente e pode ser avaliada de novo.

A promoção pelo limiar, descrita na Introdução, só atinge candidaturas ainda não analisadas por uma pessoa: o recrutador continua sendo quem decide no fim, e voltar atrás no estado de uma candidatura já avaliada apagaria o trabalho dele.

3.3. Verificação experimental e processo de desenvolvimento

O motor de avaliação não foi escrito diretamente na aplicação: cada mecanismo passou antes por um processo de verificação em ambiente isolado, feito para responder dúvidas sobre como o modelo se comporta. Os registros dessa etapa são datados e reúnem, para cada rodada, as condições da execução, a medição, o resultado e os dados brutos, o que facilita saber depois como um resultado foi obtido. Cada rodada seguiu o mesmo protocolo: partia de uma pergunta escrita antes da execução, fixava as condições (modelo, temperatura, versão da rubrica e conjunto de candidaturas de teste, todas fictícias) e repetia a mesma requisição mais de uma vez quando o objetivo era observar variação entre execuções. Salvo onde se indica outro, todas as rodadas usaram o gemini-3.1-flash-lite.

Esse processo determinou as escolhas que a aplicação herdou, a começar pela comparação inicial entre dois provedores: o Gemini e o Groq, plataforma de inferência que executava o modelo aberto Llama 3.3 70B (llama-3.3-70b-versatile). O Groq foi descartado por algumas razões observadas: mais afirmações incorretas em tom seguro e saída estruturada mais complexa de se obter. Manter esse outro provedor exigiria mais código para obter um dado menos confiável.

A calibração da escala foi feita sobre três candidaturas montadas para representar situações diferentes: um perfil claramente compatível com a vaga, um claramente inadequado e um no limite. Cada uma foi avaliada em duas execuções idênticas, os dois extremos mantiveram a mesma nota nas duas rodadas, enquanto o caso no limite variou. A causa deste problema estava no critério de formação acadêmica, que não tinha descrição: sem saber se uma graduação ainda em andamento atendia à vaga, o modelo decidia isso por conta própria, e a nota do critério mudava entre as rodadas. A correção não veio de uma regra fixa na rubrica, já que o que conta como atendido muda de uma vaga para outra. A exigência fica a cargo do recrutador, que a declara na descrição do critério ou, quando ele é obrigatório, no requisito

conferido pelo *gate*, onde indica se aceita formação em andamento. Sem essa declaração, o critério continua interpretado pelo modelo, sujeito à oscilação observada.

Nessa mesma etapa, surgiu outra restrição que passou para o sistema, descoberta quando duas candidaturas foram avaliadas em sequência, dentro de um mesmo contexto. Os candidatos deixaram de ser julgados contra a vaga e passaram a ser comparados entre si, de modo que a nota de um candidato dependia de quem havia sido avaliado antes. Isso contraria o propósito do *score*, e a solução foi isolar cada avaliação. Cada chamada ao modelo leva os dados de um único candidato, sem histórico de conversa, e por isso não há outro candidato com quem comparar. A rubrica mantém, ainda assim, a instrução explícita de não comparar candidatos, como salvaguarda de baixo custo caso uma mudança futura volte a reunir avaliações em um mesmo contexto.

A metodologia de gerenciamento do desenvolvimento do TrIA foi o Kanban, com versionamento em Git e padronização de código automatizada por ESLint e Prettier. Os testes usam Jest, com requisições de integração verificadas por *supertest* e o banco substituído por uma instância em memória, o que permite executar a suíte sem depender de infraestrutura externa.

3.4. Tratamento de dados pessoais

O sistema guarda dados de pessoas que se candidatam a vagas, e por isso o projeto adotou a Lei Geral de Proteção de Dados Pessoais (LGPD) como referência. Dois princípios da lei guiaram o projeto: usar apenas os dados necessários para a finalidade e protegê-los contra o acesso de quem não tem permissão (Brasil, 2018). Três decisões do sistema respondem a esses princípios.

A primeira decisão controla o que sai do servidor: o modelo de linguagem recebe, do candidato, apenas o perfil profissional e as respostas ao questionário. Nome, e-mail, telefone, LinkedIn e cidade aparecem para o recrutador, que precisa deles para falar com o candidato, mas não são enviados ao modelo, porque não ajudam a avaliar se o candidato atende à vaga.

A segunda decisão é o controle de acesso: com exceção do cadastro e do login, toda rota da API exige que o usuário esteja autenticado e confere se o papel dele tem permissão para aquela operação. A terceira limita quem vê o resultado da triagem, porque a nota e a justificativa são retiradas antes de a resposta chegar ao recrutador, e só o papel de suporte tem acesso a elas.

A LGPD também garante ao candidato um direito específico. Quando uma decisão que o afeta, inclusive uma que defina seu perfil profissional, é tomada só por processamento automatizado, ele pode pedir que ela seja revista e saber quais critérios e procedimentos foram usados, ressalvados os segredos comercial e industrial (Brasil, 2018). No sistema, a avaliação automatizada não reprova ninguém: ela apenas promove à fila de análise as candidaturas que alcançam o limiar, e as demais continuam entre as inscrições, disponíveis para o recrutador. Para Miziara (2024), que inclui a triagem de currículos por ATS entre os usos da IA na seleção, esse dever de explicar pode, em muitos casos, ser cumprido sem revelar o funcionamento interno do modelo. Aplicada ao TrIA, essa leitura, que é deste trabalho e não da fonte, encontra apoio no fato de que critérios, pesos, limiar e fórmula da nota estão no código do projeto, e a evidência gravada para cada critério serve de base para a explicação.

A lei não exige que a decisão final fique com o recrutador, isso foi uma escolha do projeto. Mesmo assim, o projeto não usa essa escolha como argumento para dispensar o direito de revisão, porque não ser promovido à fila também pode prejudicar o candidato. Há ainda um limite nessa revisão, porque quem examina essas candidaturas não vê a evidência produzida pelo modelo.

Apesar dessas medidas, o sistema apresenta cinco limitações conhecidas no tratamento de dados. Não há um aviso que informe ao candidato a finalidade do tratamento dos seus dados. A exclusão da conta apaga o currículo e retira do cadastro os dados de identificação e de contato, mas mantém as candidaturas e as respostas ao questionário, ligadas a uma conta que já não identifica a pessoa. Os dados ficam no MongoDB Atlas, serviço de banco de dados mantido por terceiro (Subseção 5.3), e a região de armazenamento e as condições do contrato com o fornecedor não foram avaliadas. Além disso, o perfil e as respostas, mesmo sem nome e contato, são processados pelo provedor do modelo, e a lei só admite a transferência de dados pessoais para outro país em hipóteses que ela enumera, como a de o país de destino oferecer proteção adequada ou a de o controlador comprovar garantias contratuais (Brasil, 2018). O regulamento da ANPD atribui ao controlador a tarefa de conferir se a operação configura essa transferência e se ela tem base legal e um mecanismo admitido, como as cláusulas-padrão contratuais (Brasil, 2024), conferência que não foi feita. A última limitação vem do uso da cota gratuita da Gemini API. Nessa modalidade, segundo os termos do serviço, o Google usa o conteúdo enviado e as respostas para aprimorar seus produtos, revisores humanos podem lê-los, e o próprio texto pede que não se envie informação pessoal (Google, 2026a). Como perfil e respostas continuam sendo dados pessoais mesmo sem identificação direta, o uso atual só é compatível com dados fictícios, como os dos testes deste trabalho, e a operação com candidatos reais exigiria a modalidade paga, na qual os termos excluem esse uso e remetem a um adendo de tratamento de dados.

4. Desenvolvimento

Esta seção descreve como o sistema foi construído: a ordem do trabalho, a arquitetura e a configuração da vaga pelo recrutador.

4.1. Etapas da construção

A construção ocupou de março a outubro de 2026, em dois repositórios, um para a API e outro para a interface, com cada entrega integrada por um ramo próprio. A ordem foi da plataforma para o motor: usuários, vagas, currículo, candidaturas, questionários e controle de acesso vieram primeiro, e a avaliação automatizada só entrou em agosto, porque depende de vaga, perfil e respostas já existirem para ter o que julgar. Duas regras de negócio nasceram dessa dependência: a vaga passou a exigir questionário para ser ativada, já que as respostas fazem parte da entrada da avaliação, e o papel de suporte foi criado antes do motor, por ser quem ajusta os parâmetros da triagem.

O *score* foi a parte que mais mudou ao longo do caminho. Existia nas primeiras versões e foi retirado em abril, para que a integração com o modelo começasse pela forma mais simples e o funcionamento da chamada pudesse ser entendido antes de qualquer cálculo agregado. Voltou em agosto com outra função, a de registro de

auditoria oculto do recrutador e visível só ao suporte, e a listagem passou a seguir a ordem de inscrição. Nessa volta, a nota final ainda era dada pelo próprio modelo, mas os testes mostraram que ela mudava pouco quando o recrutador alterava os pesos dos critérios, e numa das configurações o modelo promoveu uma candidatura que, pela conta dos pesos, ficaria abaixo do limiar. Por isso, em outubro, o cálculo passou a ser feito pelo sistema, como descrito na Subseção 3.2, e o modelo ficou responsável apenas pela nota de cada critério.

Outubro concentrou ainda as duas decisões que tornaram a triagem verificável. O *gate* passou a conferir os critérios obrigatórios contra o catálogo, pelo motivo exposto na Subseção 3.2, e o perfil do candidato passou a ser preenchido a partir dele. A nota de cada critério, que o modelo já devolvia mas o sistema descartava depois de calcular o *score*, passou a ser gravada com a evidência, para que o registro de auditoria mostre por que a candidatura ficou acima ou abaixo do limiar.

4.2. A arquitetura em níveis

O sistema atende a diferentes tipos de usuário e, como mostra a Figura 1, três papéis o usam: o recrutador configura a vaga e analisa a fila, o candidato cadastra o currículo e responde ao questionário, e o suporte calibra a triagem e é o único que vê o *score*. Há ainda o papel de administrador, que cadastra usuários e é o único autorizado a incluir e alterar itens do catálogo. Essa manutenção é feita por requisições à API, sem tela própria, e os itens iniciais do catálogo são carregados por um *script* de povoamento do banco. Como não tem função própria na triagem, o administrador fica fora do diagrama. Os sistemas externos são dois: o Google Gemini, modelo de IA que avalia currículo e respostas, e o MongoDB Atlas, serviço gerenciado em que fica o banco de dados (Subseção 5.3). A autenticação não aparece como serviço externo porque roda dentro da própria API e guarda a sessão nesse banco.

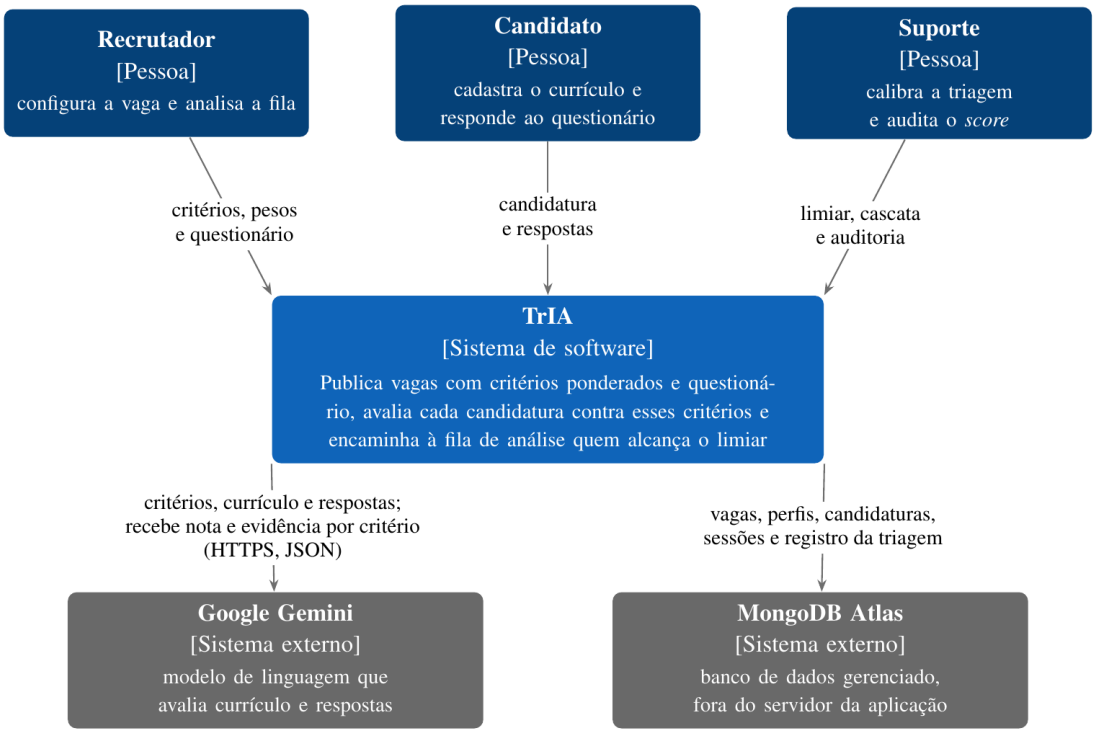


Figura 1. Nível de contexto: os três papéis atendidos pelo sistema e os dois sistemas externos que ele consome.

Fonte: Elaborado pelo autor (2026).

A Figura 2 acompanha uma candidatura do fim do questionário até o registro do resultado. A avaliação é disparada quando o candidato conclui o questionário, mas corre em segundo plano, e a resposta ao candidato não espera por ela, já que a chamada ao modelo depende de um serviço externo. Um único serviço da API, o orquestrador da avaliação, conduz os passos seguintes, sempre na mesma ordem. Antes do primeiro, ele lê a configuração da triagem definida pelo suporte, que traz o limiar, a cascata de modelos e um interruptor com o qual o próprio suporte suspende a avaliação sem tirar o sistema do ar (Figura 8). Enquanto ele está desligado, as candidaturas concluídas ficam pendentes, sem nota, até serem reprocessadas.

No primeiro passo, os dados são reunidos em uma estrutura única: os critérios da vaga, o perfil do candidato e as respostas, com as questões objetivas já corrigidas contra o gabarito. O segundo passo é o *gate* dos critérios obrigatórios, e a candidatura que não comprova algum deles contra o catálogo é registrada como incompatível, com o motivo por extenso e o fluxo encerrado ali, sem chamada ao modelo. As demais seguem ao terceiro passo, em que o Gemini, acessado pela cascata de modelos, devolve a nota e a evidência de cada critério, e o sistema combina essas notas na nota final, ponderada pelos pesos. Se o provedor falhar, nada é gravado, e a candidatura fica pendente para ser avaliada de novo.

No quarto passo, a nota final é comparada ao limiar. A candidatura que o alcança é promovida à fila de análise, mas só se ainda estiver entre as inscrições: como a avaliação corre em segundo plano, o recrutador pode tê-la movido nesse intervalo, e a decisão dele prevalece sobre a promoção automática. A que não o alcança continua onde estava. O quinto passo grava o resultado de uma só vez: a nota final, a nota e a evidência de cada critério, o limiar aplicado, o modelo que de fato respondeu e a justificativa. Cada critério é gravado com o nome e o peso que tinha naquele momento, porque o recrutador pode editar a vaga depois, e o registro precisa mostrar o que valia quando a nota foi dada. Como tudo é gravado só no fim, uma falha no meio do caminho não deixa na candidatura um resultado incompleto. Esses dados ficam ocultos do recrutador e servem ao suporte para auditar e calibrar a triagem.

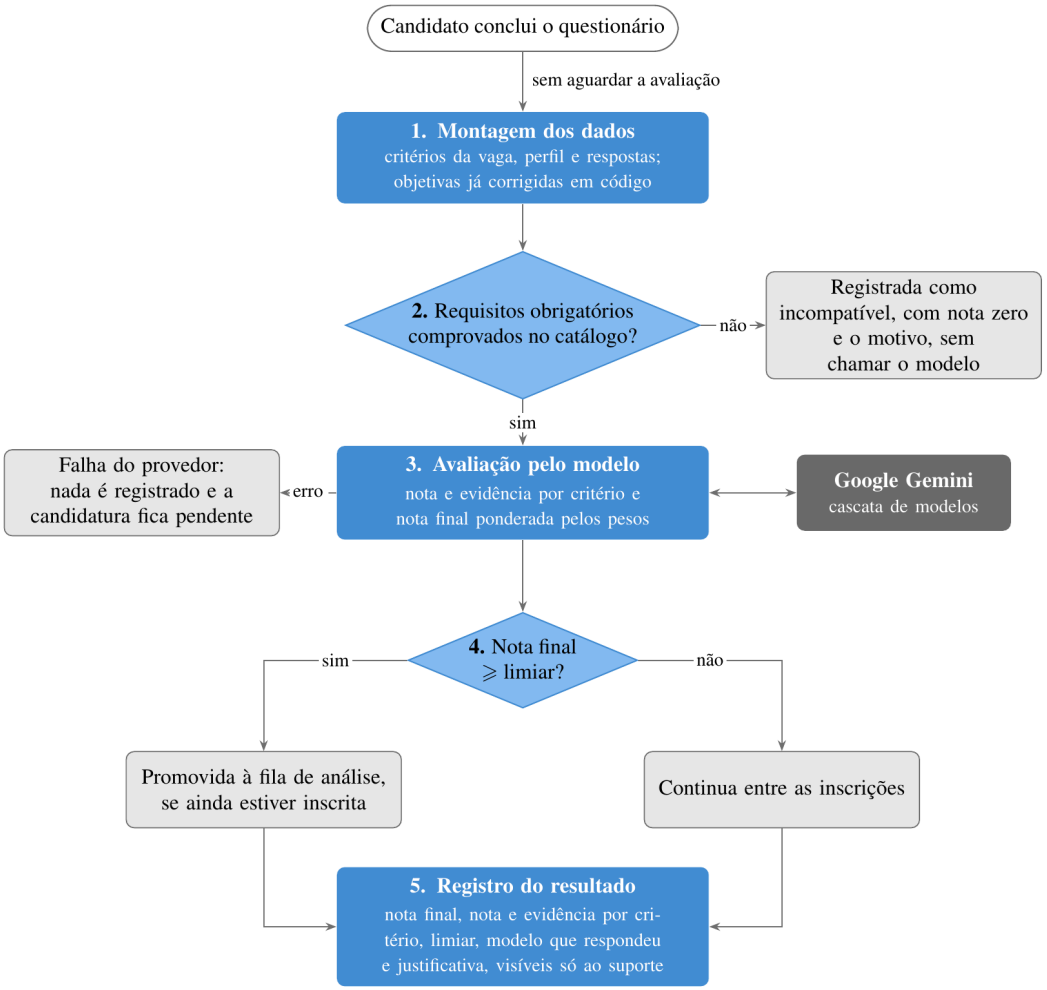


Figura 2. Fluxo da avaliação de uma candidatura: os cinco passos conduzidos pelo orquestrador e as saídas antecipadas pelo *gate* e pela falha do provedor

Fonte: Elaborado pelo autor (2026).

4.3. A configuração da vaga pelo recrutador

A configurabilidade do critério, uma das premissas do trabalho, está no cadastro da vaga. Para cada critério, o recrutador define nome, tipo (habilidade técnica, formação, experiência ou certificação), peso percentual, se é obrigatório e uma descrição livre de até trezentos caracteres. A Figura 3 mostra essa tela com os três critérios de uma vaga de exemplo.

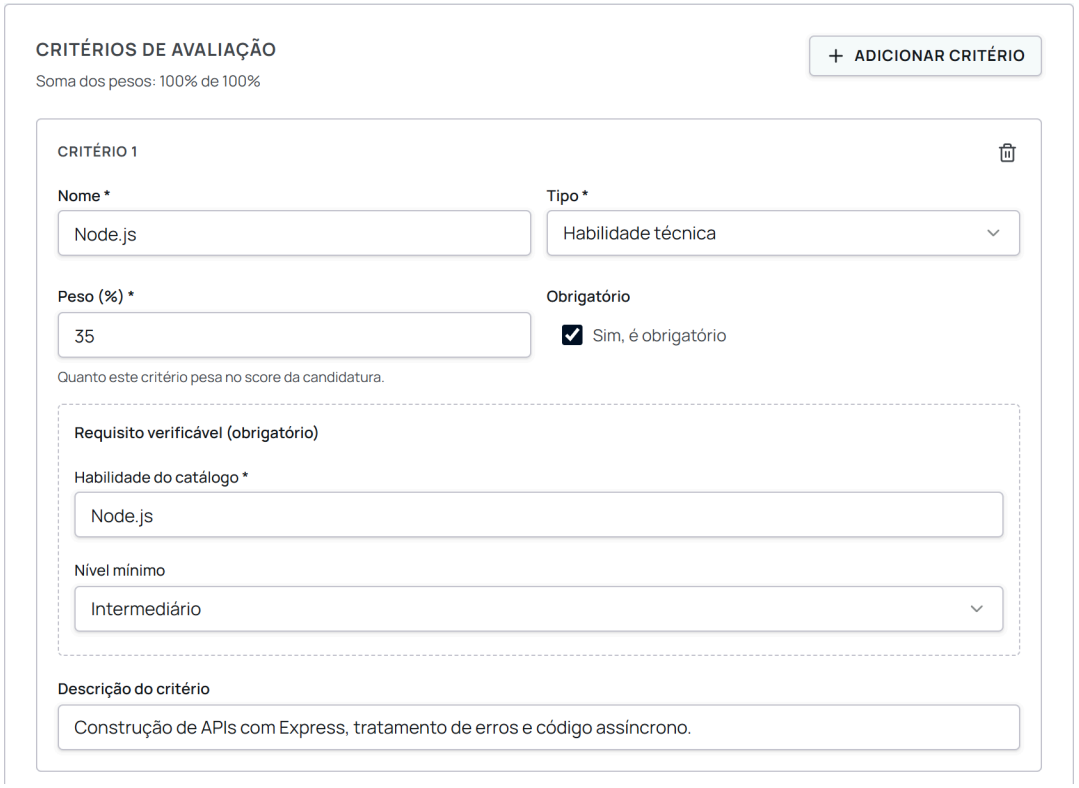


Figura 3. Cadastro dos critérios de avaliação da vaga, com nome, tipo, peso percentual, obrigatoriedade e descrição definidos pelo recrutador

Fonte: Elaborado pelo autor (2026).

Cada peso é um inteiro de 1 a 100, e a soma dos pesos de uma vaga não pode passar de 100. A regra é verificada em três lugares, cada um por um motivo. A interface mostra no alto da tela a soma e o quanto ainda resta, porque repartir os pontos é decisão que se toma olhando o conjunto. O validador da API devolve uma mensagem que nomeia o critério em falta, e o esquema do banco impede que outro caminho de escrita grave uma vaga inválida. Um critério novo recebe o que sobra, até dez pontos, e por isso acrescentar critério não invalida a vaga enquanto houver espaço.

O critério obrigatório pede ainda o requisito que o *gate* confere, escolhido no catálogo descrito na Subseção 3.2. Na habilidade, é o item e, se o recrutador quiser, o nível mínimo; na certificação, só o item; na formação, o grau mínimo, a área e a aceitação do curso em andamento; na experiência, a área e os meses mínimos. A vaga com critério obrigatório sem requisito é recusada ao salvar, e a descrição livre continua existindo, porque orienta o modelo no julgamento dos critérios.

O questionário é o segundo instrumento configurável: o recrutador redige as perguntas, escolhe o tipo de cada uma (múltipla escolha, dissertativa ou verdadeiro ou falso) e, nas objetivas, marca a alternativa correta, que o sistema usa para corrigi-las em código. A Figura 4 mostra essa tela com uma pergunta de múltipla escolha e a alternativa correta marcada.

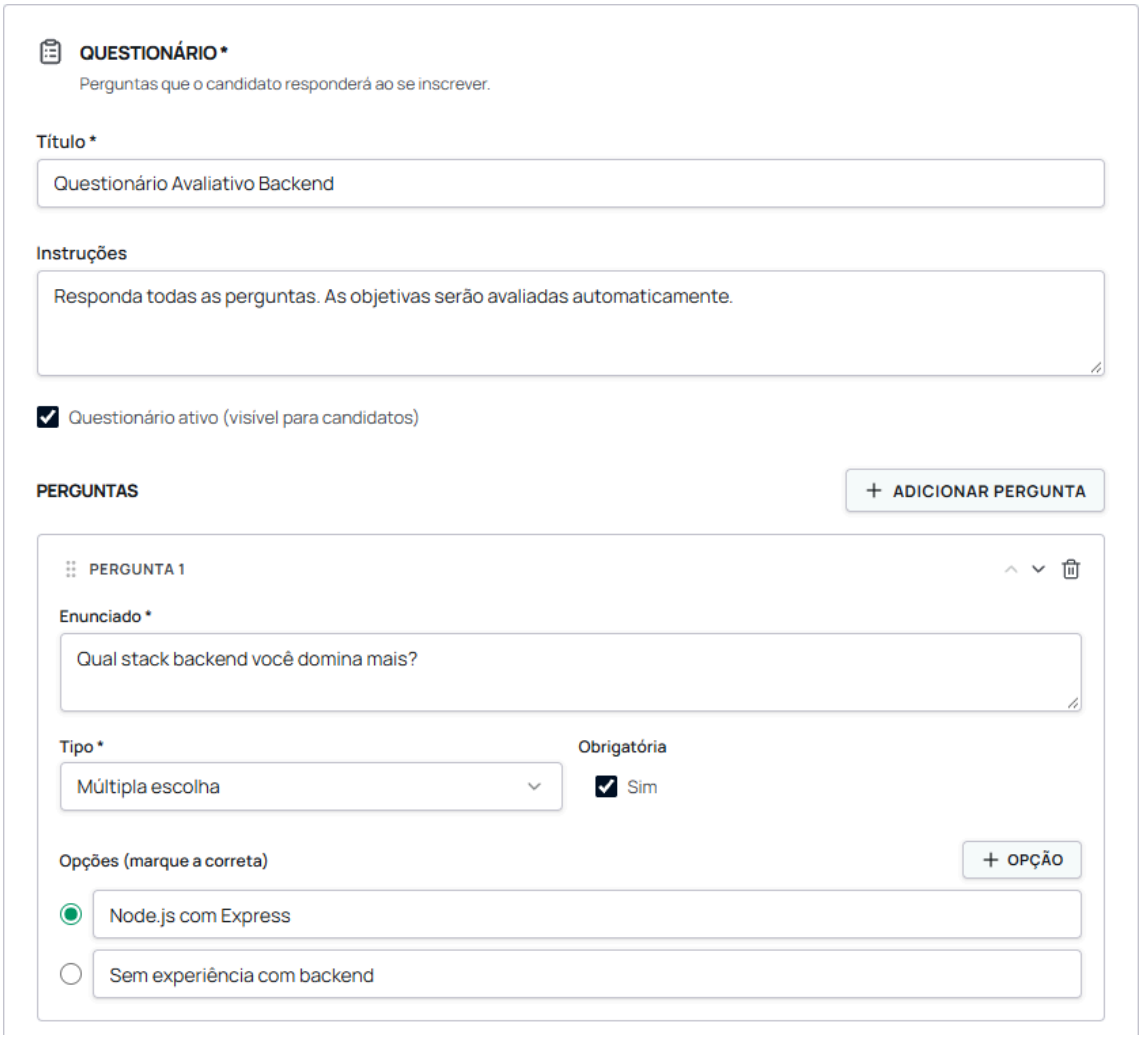


Figura 4. Montagem do questionário pelo recrutador: enunciado, tipo e obrigatoriedade de cada pergunta e, nas objetivas, a alternativa correta

Fonte: Elaborado pelo autor (2026).

A vaga só recebe candidaturas com o questionário ativo, e a Figura 5 mostra como o candidato a vê, com critérios e perguntas na mesma tela e sem a indicação da resposta correta. Ele vê nome, tipo, se é obrigatório e descrição de cada critério, mas não vê os pesos. A omissão é proposital: conhecer o peso levaria o candidato a focar apenas nas questões com nota maior, e o questionário deixaria de coletar evidência útil.

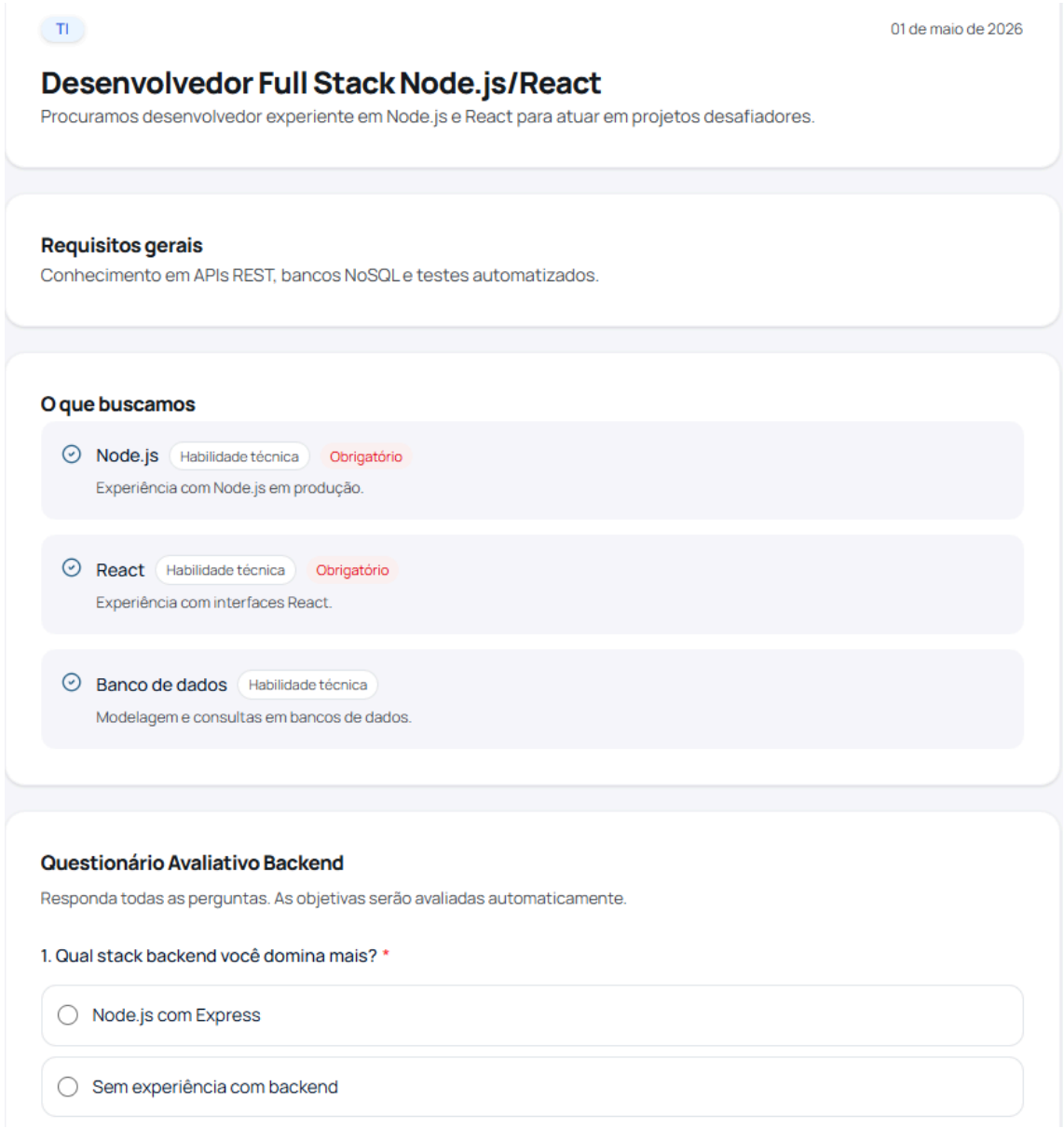


Figura 5. A vaga na visão do candidato: os critérios aparecem com nome, tipo, obrigatoriedade e descrição, sem os pesos, e o questionário traz as perguntas redigidas pelo recrutador. Vaga e dados fictícios, criados para a demonstração

Fonte: Elaborado pelo autor (2026).

4.4. Uso de inteligência artificial como ferramenta de desenvolvimento

O motor de avaliação descrito nas subseções anteriores é a inteligência artificial que este trabalho estuda, esta subseção trata de outro uso, o da IA como ferramenta na construção do sistema. Para isso se usou o Claude Code (Anthropic, 2026), um assistente que opera dentro do repositório: abre e modifica arquivos e executa comandos e testes conforme o autor deste trabalho pede. Todo o código gerado pelo Claude Code foi revisado pelo autor, corrigido quando necessário e aprovado antes de ser incorporado ao sistema. Foram utilizados os modelos Claude Opus 5 e Claude Opus 5.5.

No código, o uso foi pontual e variou ao longo dos oito meses, sem um fluxo fixo. Nas tarefas maiores, como a troca do controle de acesso em junho, o Claude Code produziu primeiro um documento de desenho e um plano em passos, aprovados pelo autor antes de o código ser alterado. Esse fluxo se apoiou em *skills*, roteiros de trabalho reutilizáveis que o assistente carrega conforme a tarefa, do pacote Superpowers (Vincent, 2025): *brainstorming* conduz o desenho, *writing-plans* redige o plano e *executing-plans* o executa em etapas. Na interface, a *skill* Impeccable (Bakaus, 2026) orientou ajustes visuais, e o Context7 (Upstash, 2026), servidor ligado ao assistente pelo *Model Context Protocol* (MCP), forneceu a documentação atualizada das bibliotecas. O escopo de cada funcionalidade e as regras de negócio foram definidos pelo autor.

5. Resultados e Discussões

Esta seção apresenta o que o sistema entrega diante dos objetivos específicos declarados na Introdução, os testes a que foi submetido e a forma de implantação, e termina comparando o resultado com os trabalhos discutidos na Subseção 2.3 e com o desenho previsto no início do projeto.

5.1. Funcionalidades entregues

Os quatro objetivos citados na Introdução têm funcionalidade correspondente no sistema, sendo o primeiro a configuração da vaga pelo recrutador, apresentada na Subseção 4.3, com os critérios, seus pesos e o questionário com o gabarito das questões objetivas. A essa configuração se soma o ciclo de vida da vaga, que pode estar ativa, pausada ou arquivada, sendo o arquivamento a única mudança que não pode ser desfeita.

Do lado do candidato, que corresponde ao segundo objetivo, o perfil é cadastrado em quatro grupos (formações, experiências, habilidades e certificações), cada um com inclusão, edição e exclusão, e é esse conteúdo estruturado que entra na avaliação. Habilidades, áreas de formação, áreas de atuação e certificações são escolhidas no catálogo, com sugestões enquanto o candidato digita. O item que não corresponde a nenhum do catálogo, nem pelo nome nem por sinônimo, também é aceito, porque o perfil pertence ao candidato e o modelo ainda o considera na análise, mas aparece com um aviso de que não serve para comprovar critério obrigatório. A resposta ao questionário ocorre em três momentos, o início, o registro de cada resposta e a finalização, e é a finalização que dispara a avaliação. Enquanto a candidatura não recebe uma decisão, o candidato pode cancelá-la.

O terceiro objetivo, o mecanismo de análise, foi entregue com o desenho das Subseções 3.2 e 4.2: o *gate* dos critérios obrigatórios, a cascata de modelos, a nota com evidência por critério e o *score* calculado em código. O quarto, o encaminhamento pelo limiar, completa o percurso, e a candidatura promovida chega à fila de análise do recrutador, mostrada na Figura 6. Daí em diante a decisão é humana, pois nenhuma parte automatizada do sistema aprova ou reprova. Uma decisão já tomada pode ser reaberta e devolvida à análise, para corrigir um engano.

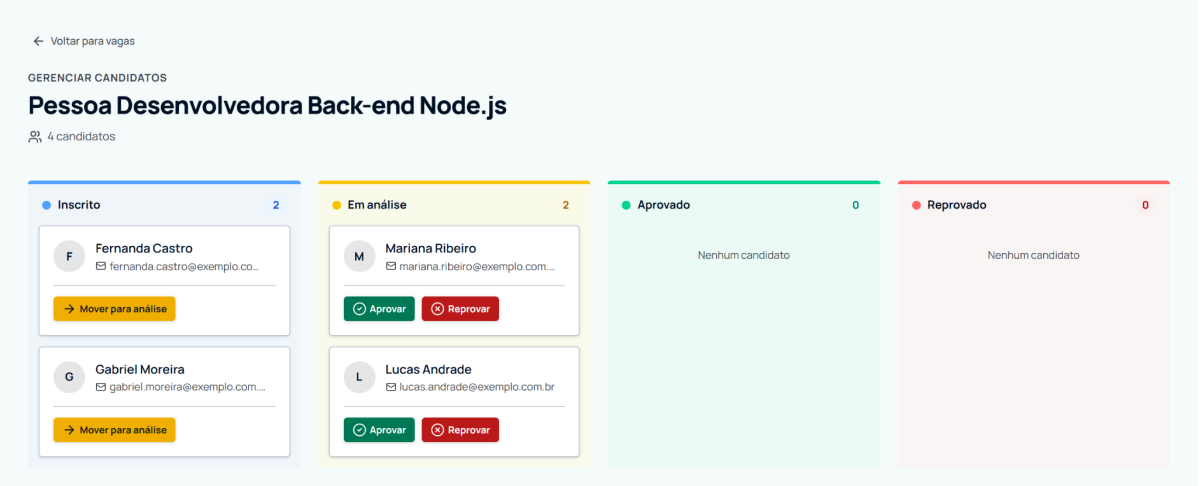


Figura 6. Fila de candidaturas como o recrutador a vê: dividida por status e, em cada coluna, na ordem de inscrição, sem nota e sem *ranking*. Nomes e e-mails fictícios, criados para a demonstração

Fonte: Elaborado pelo autor (2026).

A fila não traz nota, veredito nem justificativa, e a ocultação não é apenas visual. Os dados da triagem são retirados pela própria API antes de a resposta deixar o servidor, porque escondê-los só na interface os manteria visíveis a quem inspecionasse o tráfego do navegador. A mesma regra vale para as outras saídas que o recrutador consulta: a ficha da candidatura, que reúne contato, currículo e respostas para impressão, é montada campo a campo, sem os dados da triagem e sem o gabarito, e os relatórios trazem apenas contagens e datas, sem nenhum dado da triagem. Aplicar a ocultação em todas as saídas, e não só na listagem, é decisão do projeto, porque bastaria uma tela que exibisse o *score* para que ele voltasse a orientar o julgamento.

O *score* continua existindo como registro no painel do papel de suporte, mostrado na Figura 7, com a nota, o limiar aplicado, a justificativa e o modelo que respondeu. É também o suporte quem ajusta a triagem, na tela da Figura 8, e quem identifica as candidaturas que ficaram pendentes, seja por falha do provedor, seja pelo interruptor desligado. Para essas existe a reavaliação manual, aberta também ao recrutador e ao administrador, que só é aceita enquanto a candidatura está entre as inscrições: depois disso ela já passou pela promoção automática ou por decisão humana, e uma nota nova deixaria de corresponder à fila em que ela se encontra.

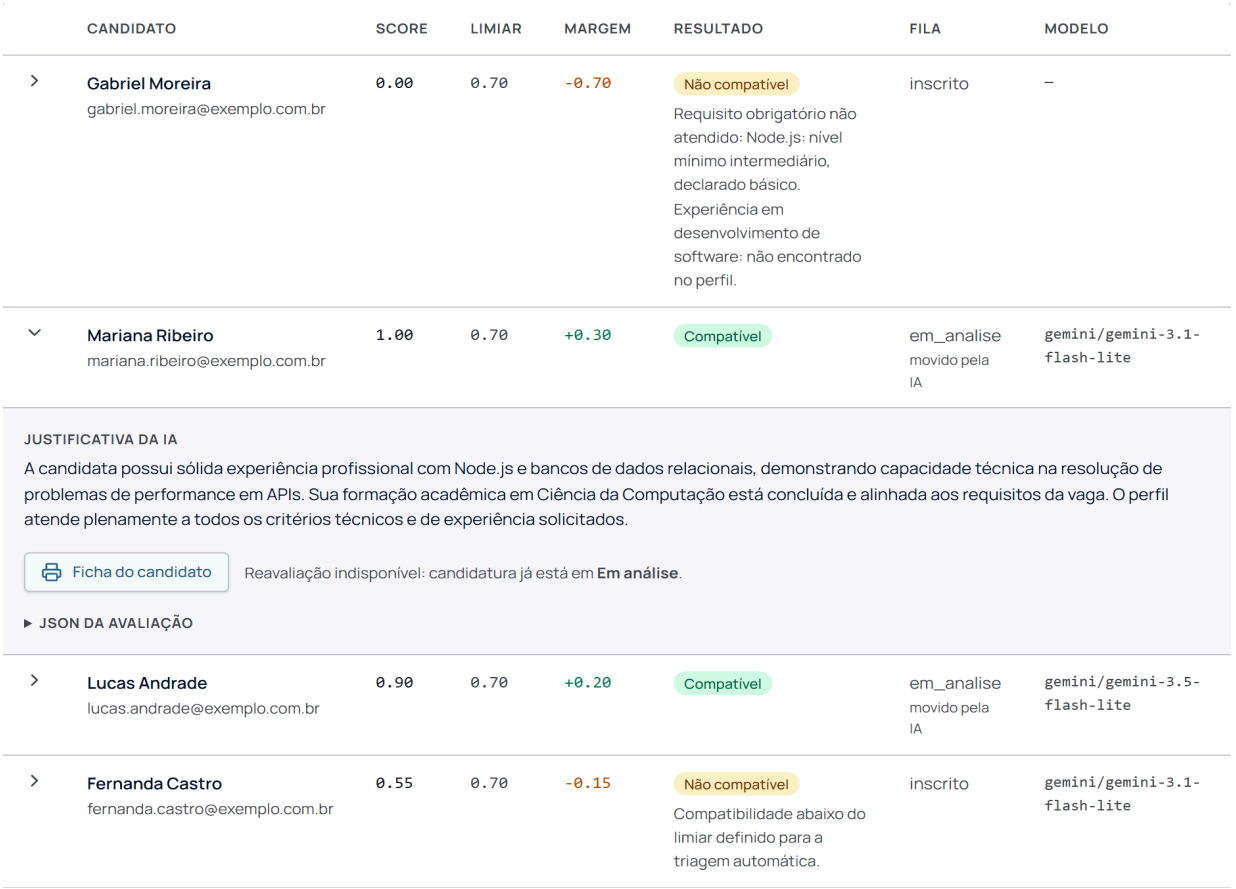


Figura 7. Painel do papel de suporte, com o *score*, o limiar aplicado, a justificativa e o modelo que respondeu. Nomes e e-mails fictícios. A nota 0,00 da primeira linha não foi dada pelo modelo: a candidatura foi barrada no *gate* por não atender a requisitos obrigatórios, e por isso a coluna do modelo está vazia

Fonte: Elaborado pelo autor (2026).



Figura 8. Configuração da triagem pelo suporte: limiar de compatibilidade, provedor, cascata de modelos e interruptor de desligamento.

Fonte: Elaborado pelo autor (2026).

5.2. Testes realizados

O código do sistema foi verificado por testes automatizados, que comparam o resultado obtido com um valor esperado. O modelo de linguagem foi examinado em rodadas manuais, porque a mesma entrada pode gerar notas diferentes e não existe uma nota certa com que comparar a resposta.

A suíte da API reúne 1.961 testes, todos aprovados: 1.425 unitários, 146 de integração e 390 de ponta a ponta, que percorrem o caminho do cadastro da vaga à promoção da candidatura, e cobre 98,9% das instruções, 95,8% dos ramos e 99,8% das funções do código da API, disponível em https://gitlab.fslab.dev/match-de-curriculos/match-de-curriculos-api. Na interface, 180 testes com o Cypress, ferramenta que opera o navegador como um usuário, cobrem o acesso e as telas de cada ator, e também passaram. O código da interface está em https://gitlab.fslab.dev/match-de-curriculos/match-de-curriculos-front.

Esses testes automatizados não se aplicam às notas que o modelo de linguagem entrega e, para avaliá-las, foram feitas rodadas manuais, registradas nas folhas descritas na Subseção 3.3 e resumidas no Quadro 2.

Período	**Pergunta**	**Resultado e consequência**
ago. 2026	Qual provedor e qual temperatura usar?	O Groq, com o Llama 3.3 70B, mudou o conteúdo da resposta conforme o estilo pedido e não garantia JSON: foi retirado. Temperatura fixada em zero
ago. 2026	A escala separa perfis forte, fraco e limítrofe?	Extremos estáveis; o limítrofe variou por causa de um critério sem descrição (Subseção 3.3)
ago. 2026	Como o sistema reage às falhas do provedor?	Cinco falhas provocadas; o tempo esgotado era tratado como desistência e passou a gerar nova tentativa
out. 2026	O *score* do modelo acompanha os pesos?	Não: o *score* do modelo variou pouco com os pesos e, numa configuração, promoveria uma candidatura que a conta não promoveria. O cálculo do *score* passou a ser feito por código
out. 2026	As notas se mantêm nos dois modelos da cascata?	Mesmas notas em 108 chamadas, com o gemini-3.1-flash-lite e o gemini-3.5-flash-lite; limiar de 0,7 mantido
out. 2026	O sistema integrado promove quem deve?	Sim: numa vaga com quatro candidatos fictícios, as notas do modelo promoveram corretamente os dois que superaram o limiar, o que ficou abaixo permaneceu entre as inscrições e o que não cumpria um critério obrigatório foi barrado no *gate*

Quadro 2. Rodadas manuais de teste do modelo de linguagem durante o desenvolvimento

Fonte: Elaborado pelo autor (2026).

Na rodada dos pesos, que motivou a mudança relatada na Subseção 4.1, o modelo ainda calculava o *score* da candidatura, e esse valor nem sempre respeitava os pesos definidos na vaga. Num dos testes, a candidata tirou a menor nota justamente no critério mais importante, com 80% do peso. O *score* correto seria 0,60, abaixo do limiar de 0,7, mas o modelo devolveu 0,70, e a candidata seria promovida. Por isso, o modelo passou a dar apenas a nota de cada critério, e o *score* passou a ser calculado por código.

As rodadas envolveram poucas vagas e não contaram com um recrutador real avaliando as mesmas candidaturas para comparar com as notas do modelo, por isso não servem para medir acurácia. O que elas permitem é conferir se o sistema se comporta como foi projetado.

5.3. Implantação

O sistema foi implantado em uma máquina virtual na infraestrutura do Instituto Federal de Rondônia (IFRO), Campus Vilhena, criada pela plataforma https://cloud-light.fslab.dev. A máquina é virtualizada em Proxmox e tem 2 processadores virtuais, 4 GB de memória e 40 GB de disco, com Ubuntu 24.04 e o Docker já instalado, o que bastou para executar os contêineres em que o sistema já era empacotado.

Na máquina, o Docker Compose executa dois contêineres, um com a API e outro com a interface, servida pelo nginx, que também encaminha à API as requisições destinadas a ela. A conexão segura (HTTPS) fica a cargo de um *proxy* da infraestrutura do IFRO, à frente da máquina virtual, e o banco de dados está no MongoDB Atlas, fora da máquina virtual. O endereço do banco e a chave de acesso ao Gemini ficam em variáveis de ambiente do servidor, e não no código. O sistema está disponível em https://matchcd.app.fslab.dev.

5.4. Comparação com outras soluções e com o desenho inicial

Esta subseção compara o sistema com os trabalhos apresentados na Subseção 2.3 e com o que o projeto previa no início. A comparação com os trabalhos é de desenho, e não de desempenho, porque nenhum deles foi executado sobre as mesmas candidaturas. O que se compara é o modo como cada um avalia, a partir do que os próprios autores descrevem.

A principal diferença está em quem decide o que conta na avaliação: no TrIA, essa decisão é do recrutador, e isso aparece em três pontos. O primeiro é o peso de cada critério: o recrutador distribui até 100 pontos entre os critérios da vaga, como mostra a Figura 3, e o sistema combina as notas por média ponderada, recurso que nenhum dos sistemas de triagem consultados descreve. O ATS examinado por Puccini et al. (2022) também elimina candidatos, mas por perguntas de resposta objetiva, no TrIA, esse papel cabe ao *gate* de obrigatórios, em código, e a avaliação se estende às respostas dissertativas, que aquele filtro não conseguia tratar. O segundo é o questionário, cujas perguntas são escritas pelo recrutador para aquela vaga, enquanto no trabalho de Stuart, Kaneko e Baldwin (2026), que entrevista o candidato, elas são geradas pelo modelo durante a conversa. O terceiro é a forma do resultado. Os outros trabalhos devolvem uma nota global para cada currículo (Padmaja et al., 2023; Gan; Zhang; Mori, 2024) ou uma lista dos candidatos em ordem (Yuksel et al., 2026); no TrIA, o recrutador recebe a candidatura já colocada na fila de análise, sem nota nem *ranking*, enquanto a nota de cada critério e a evidência que a justifica ficam registradas para a auditoria do suporte.

Essa escolha tem um custo, na leitura do autor deste trabalho: como é o recrutador quem define critérios e pesos, a qualidade da avaliação passa a depender também do cuidado com que a vaga é configurada. O problema não é exclusivo deste sistema: Yuksel et al. (2026) também reconhecem que o resultado piora quando os dados de entrada, como o anúncio da vaga, são mal escritos.

Em relação ao planejamento inicial, a maior mudança foi na saída do sistema: o *canvas* do projeto, documento de planejamento que reunia o problema, os objetivos e o cronograma, previa uma pré-seleção dos candidatos em ordem de pontuação, e o sistema entregou a promoção por limiar, em que a candidatura que alcança a nota mínima vai para a fila de análise e o *score* fica escondido do recrutador. O motivo, apresentado na Introdução, é evitar que o número influencie o julgamento humano, mas

isso é uma premissa do projeto que não foi medida. Cada um dos quatro objetivos específicos tem a funcionalidade correspondente descrita na Subseção 5.1. O objetivo geral, apoiar equipes de recursos humanos, não foi testado com recrutadores, porque a validação com profissionais da área ficou fora do escopo deste trabalho, e por isso esta seção mostra que o sistema faz o que foi projetado para fazer, mas não que melhora a triagem de uma equipe real.

A rodada dos pesos, descrita na Subseção 5.2, reforça a primeira diferença apontada nesta subseção, o peso de cada critério: quando o próprio modelo devolvia o *score*, o peso definido na vaga nem sempre era seguido, e uma candidata seria promovida sem atingir o limiar. Com a conta feita por código, o modelo fica responsável apenas pela nota e pela evidência de cada critério, e o peso passa a valer como foi configurado.

6. Conclusão

Este trabalho desenvolveu o TrIA, um sistema de triagem de candidatos em que o recrutador define o que conta na avaliação de cada vaga: os critérios, o peso de cada um, quais são obrigatórios e as perguntas do questionário. A partir dessa configuração, o sistema confere os requisitos obrigatórios contra um catálogo, o modelo de linguagem atribui a cada critério uma nota acompanhada da evidência que a justifica, e o próprio sistema combina essas notas pelos pesos definidos. A candidatura que alcança o limiar é encaminhada à fila de análise, enquanto a nota fica oculta do recrutador e registrada para a auditoria do suporte. Os quatro objetivos específicos têm funcionalidade correspondente, como mostrou a Subseção 5.1, e o sistema está publicado na infraestrutura do Instituto Federal.

A principal contribuição está em levar à triagem automatizada o critério ponderado definido por quem conhece a vaga, inclusive para avaliar respostas abertas, algo que os sistemas de triagem comparados na Subseção 5.4 não descrevem e que um filtro por perguntas objetivas não alcança. A outra contribuição é entregar o resultado como uma ação sobre a fila, e não como um número que orienta o julgamento. O desenvolvimento trouxe também uma lição prática: enquanto o modelo dava a nota final, os testes mostraram que o peso definido pelo recrutador perdia parte do efeito, e passar a ponderação para o código foi o que garantiu que a configuração dele fosse de fato respeitada.

Entre as limitações, a principal é que o sistema não foi validado com profissionais da área, etapa que ficou fora do escopo deste trabalho, e por isso o que se pode afirmar é que ele faz o que foi projetado para fazer, não que melhora a triagem de uma equipe real. O nível de cada habilidade é informado pelo próprio candidato, e o *gate* confere apenas o que foi declarado. O catálogo de itens só pode ser alterado por requisições do administrador à API, porque ainda não há tela para isso, a avaliação depende de um provedor externo, e a adequação à LGPD tem as lacunas apontadas na Subseção 3.4, como a falta de um aviso de finalidade ao candidato.

Como trabalhos futuros, a continuação mais importante é testar o sistema com recrutadores em vagas reais, comparando as candidaturas promovidas com a escolha feita por eles, o que permitiria também calibrar o limiar com mais vagas e candidaturas. Outras melhorias são confirmar as habilidades declaradas por meio do questionário ou

de certificações, criar a tela de administração do catálogo e fechar as lacunas da LGPD apontadas na Subseção 3.4.

Referências

AN, Jiafu et al. Measuring gender and racial biases in large language models: intersectional evidence from automated resume evaluation. *PNAS Nexus*, v. 4, n. 3, art. pgaf089, 2025. DOI: 10.1093/pnasnexus/pgaf089.

ANTHROPIC. *Claude Code: overview*. [S. l.], 2026. Disponível em: https://code.claude.com/docs/en/overview. Acesso em: 12 nov. 2026.

BAKAUS, Paul. *Impeccable*. GitHub, 2026. Disponível em: https://github.com/pbakaus/impeccable. Acesso em: 12 nov. 2026.

BARDUCCI, Alessandro et al. An end-to-end framework for information extraction from Italian resumes. *Expert Systems with Applications*, v. 210, art. 118487, 2022. DOI: 10.1016/j.eswa.2022.118487.

BARELLI, Alexandre Antonio; LEME, Ricardo Roberto. Impacto da inteligência artificial na transformação digital das empresas: remodelando processos, negócios e tomadas de decisões. *Brazilian Journal of Technology*, v. 7, n. 4, p. 1–19, 2024. DOI: 10.38152/bjtv7n4-004.

BERRY, Christopher M. et al. Insights from an updated personnel selection meta-analytic matrix: revisiting general mental ability tests' role in the validity-diversity trade-off. *Journal of Applied Psychology*, v. 109, n. 10, p. 1611–1634, 2024. DOI: 10.1037/apl0001203.

BRASIL. Lei nº 13.709, de 14 de agosto de 2018. Lei Geral de Proteção de Dados Pessoais (LGPD). *Diário Oficial da União*: seção 1, Brasília, DF, 15 ago. 2018. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709compilado.htm. Acesso em: 12 nov. 2026.

BRASIL. Autoridade Nacional de Proteção de Dados. Resolução CD/ANPD nº 19, de 23 de agosto de 2024. Aprova o Regulamento de Transferência Internacional de Dados e o conteúdo das cláusulas-padrão contratuais. *Diário Oficial da União*: seção 1, Brasília, DF, n. 163, p. 123, 23 ago. 2024. Disponível em: https://www.in.gov.br/en/web/dou/-/resolucao-cd/anpd-n-19-de-23-de-agosto-de-2024-580095396. Acesso em: 13 nov. 2026.

COGGO, Beatriz Loffredo. *Uso da inteligência artificial no processo de recrutamento e seleção*. Orientador: Marcio Shoiti Kuniyoshi. 2025. Trabalho de Conclusão de Curso (Graduação em Administração) – Faculdade de Economia, Administração, Contábeis e Atuária, Pontifícia Universidade Católica de São Paulo, São Paulo, 2025. Disponível em: https://repositorio.pucsp.br/jspui/handle/handle/45688. Acesso em: 13 nov. 2026.

CORTES, Eduardo G.; VIEIRA, Renata; BARONE, Dante A. C. Perguntas e respostas. In: CASELI, Helena de Medeiros; NUNES, Maria das Graças Volpe (org.). *Processamento de linguagem natural: conceitos, técnicas e aplicações em português*. 2. ed. [S. l.]: BPLN, 2024, cap. 16. Disponível em:

https://brasileiraspln.com/livro-pln/2a-edicao/parte-interacao/cap-qa/cap-qa.html. Acesso em: 12 nov. 2026.

DENG, Haotian et al. *Rubric-conditioned LLM grading: alignment, uncertainty, and robustness*. Preprint arXiv:2601.08843, 2025. Disponível em: https://arxiv.org/abs/2601.08843. Acesso em: 12 nov. 2026.

DRESCH, Aline; LACERDA, Daniel Pacheco; CAUCHICK MIGUEL, Paulo Augusto. Uma análise distintiva entre o estudo de caso, a pesquisa-ação e a design science research. *Revista Brasileira de Gestão de Negócios*, v. 17, n. 56, p. 1116–1133, 2015. DOI: 10.7819/rbgn.v17i56.2069.

DUARTE, Jáder Felipe Silva; SOUZA, Adler Diniz de. AI and semantic modeling for profile-opportunity matching: a systematic review with quantitative synthesis. In: *Proceedings of the Brazilian Symposium on Multimedia and the Web (WebMedia)*. Porto Alegre: SBC, 2026, p. 581–594. DOI: 10.5753/webmedia.2026.30942.

GAN, Chengguang; ZHANG, Qinghao; MORI, Tatsunori. Application of LLM agents in recruitment: a novel framework for automated resume screening. *Journal of Information Processing*, v. 32, p. 881–893, 2024. DOI: 10.2197/ipsjjip.32.881.

GARCIA, Maria Eduarda Pucci Torres; MENDES, Tatiana Dornelas de Oliveira. Transformações no recrutamento e seleção: um estudo sobre a inteligência artificial no processo seletivo. *Revista de Gestão e Secretariado*, v. 16, n. 2, p. 1–20, 2025. DOI: 10.7769/gesec.v16i2.4665.

GOOGLE. *Gemini API additional terms of service*. [S. l.], 2026a. Disponível em: https://aistudio.google.com/docs/terms. Acesso em: 13 nov. 2026.

GOOGLE. *Gemini API: docs*. [S. l.], 2026b. Disponível em: https://aistudio.google.com/docs. Acesso em: 13 nov. 2026.

GU, Jiawei et al. *A survey on LLM-as-a-Judge*. Preprint arXiv:2411.15594v6, 2025. Disponível em: https://arxiv.org/abs/2411.15594v6. Acesso em: 12 nov. 2026.

MIZIARA, Raphael. A proteção contra despedida algorítmica no contexto laboral: consequências pelo descumprimento do direito à explicabilidade previsto no art. 20 da LGPD. *Revista do Tribunal Superior do Trabalho*, v. 90, n. 1, p. 230–249, 2024. Disponível em: https://revista.tst.jus.br/rtst/article/view/44. Acesso em: 13 nov. 2026.

PADMAJA, D. Lakshmi et al. Automated resume screening using natural language processing. *Journal of Emerging Technologies and Innovative Research*, v. 10, n. 3, p. f100–f104, 2023. Disponível em: https://www.jetir.org/papers/JETIR2303510.pdf. Acesso em: 12 nov. 2026.

PUCCINI, Luísa et al. Impactos da utilização da Applicant Tracking System nos processos de recrutamento e seleção de pessoas: estudo em uma organização do segmento de soluções de Recursos Humanos. *Navus: Revista de Gestão e Tecnologia*, v. 12, p. 1–12, 2022. DOI: 10.22279/navus.2022.v12.p01-12.1718.

SIQUEIRA, Rayanna da Silva. Uso da inteligência artificial no processo de recrutamento e seleção: benefícios e desafios. *Revista Tópicos*, v. 3, n. 17, 2025. DOI: 10.5281/zenodo.14625531. Disponível em: https://doi.org/10.5281/zenodo.14625531. Acesso em: 12 nov. 2026.

STUART, Harry; KANEKO, Masahiro; BALDWIN, Timothy. *Beyond the resumé: a rubric-aware automatic interview system for information elicitation*. Preprint arXiv:2603.01775, 2026. Disponível em: https://arxiv.org/abs/2603.01775. Acesso em: 12 nov. 2026.

UPSTASH. *Context7*. GitHub, 2026. Disponível em: https://github.com/upstash/context7. Acesso em: 12 nov. 2026.

VINCENT, Jesse. *Superpowers*. GitHub, 2025. Disponível em: https://github.com/obra/superpowers. Acesso em: 12 nov. 2026.

WILSON, Kyra; CALISKAN, Aylin. Gender, race, and intersectional bias in resume screening via language model retrieval. In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. [S. l.]: AAAI Press, 2024, v. 7, p. 1578–1590. DOI: 10.1609/aies.v7i1.31748.

YU, Xiao et al. *ConFit v3: improving resume-job matching with LLM-based re-ranking*. Preprint arXiv:2605.09760, 2026. Disponível em: https://arxiv.org/abs/2605.09760. Acesso em: 12 nov. 2026.

YUKSEL, Kamer Ali et al. *Agentic AI for human resources: LLM-driven candidate assessment*. Preprint arXiv:2603.26710, 2026. Disponível em: https://arxiv.org/abs/2603.26710. Acesso em: 12 nov. 2026.