UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA
CURSO BACHARELADO INTERDISCIPLINAR EM TECNOLOGIA DA INFORMAÇÃO

HEITOR CLAUDINO DANTAS

UMA PLATAFORMA WEB VOLTADA AO INCENTIVO À LEITURA EM SALA DE AULA APOIADA POR INTELIGÊNCIA ARTIFICIAL GENERATIVA

PAU DOS FERROS
2026

HEITOR CLAUDINO DANTAS

UMA PLATAFORMA WEB VOLTADA AO INCENTIVO À LEITURA EM SALA DE AULA APOIADA POR INTELIGÊNCIA ARTIFICIAL GENERATIVA

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Interdisciplinar em Tecnologia da Informação.

Orientador: Walber José Adriano Silva, Prof. Dr.

PAU DOS FERROS

2026

D167p Dantas, Heitor Claudino.
Uma plataforma web voltada ao incentivo à leitura em sala de aula apoiada por inteligência artificial generativa / Heitor Claudino Dantas. - 2026.
84 f. : il.

Orientador: Walber José Adriano Silva.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Tecnologia da Informação, 2026.

1. leitura. 2. educação. 3. plataforma web. 4. inteligência artificial generativa. 5. tecnologia educacional. I. Silva, Walber José Adriano, orient. II. Título.

Ficha catalográfica elaborada por sistema gerador automático em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Pau dos Ferros
Bibliotecária: Erlanda Maria Lopes da Silva
CRB: 15/966

HEITOR CLAUDINO DANTAS

UMA PLATAFORMA WEB VOLTADA AO INCENTIVO À LEITURA EM SALA DE AULA APOIADA POR INTELIGÊNCIA ARTIFICIAL GENERATIVA

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Interdisciplinar em Tecnologia da Informação.

Defendida em: 25/06/2026

BANCA EXAMINADORA

Walber José Adriano Silva, Prof. Dr. (UFERSA)
Presidente

Reudismam Rolim de Sousa, Prof. Dr. (UFERSA)
Membro Examinador

Rosana Cibely Batista Rego, Profa. Dra. (UFERSA)
Membro Examinador

DEDICATÓRIA

Aos meus pais, Gaspar e Maricleide, cujo o apoio foram o alicerce de toda esta caminhada. Nos momentos mais difíceis, a fé de vocês em mim superou as minhas próprias dúvidas, tornando este sonho realidade. Esta conquista é, acima de tudo, nossa. (*presentes*)

AGRADECIMENTOS

Expresso minha mais profunda gratidão aos meus pais, Gaspar e Maricleide. Vocês me depositaram em mim a esperança que eu poderia ir longe. Obrigado por todo o apoio e incentido ao longo dessa caminhada. Mesmo nos momentos em que tudo parecia difícil, vocês estiveram ao meu lado, me dando forças para continuar e nunca desistir dos meus sonhos. Obrigado, do fundo do meu coração, por acreditarem em mim até quando eu mesmo duvidei. Esta conquista também é de vocês.

Minha eterna gratidão à Maria Helena. Você também faz parte desta conquista e foi, sem dúvida, quem mais conheceu os bastidores de toda a minha trajetória acadêmica. Este trabalho nasceu das nossas conversas, das suas palavras de incentivo, das suas sugestões e da sua capacidade de me fazer enxergar possibilidades quando eu só via dificuldades. Obrigado por acreditar em mim mais do que eu mesmo fui capaz de acreditar. Sua confiança me fortalece e me inspira a continuar perseguindo sonhos que, um dia, pareciam distantes demais.

Aos meus amigos Jhoan, Tomaz, Augusto e Carlos "Gabas" Gabriel, que caminham comigo desde o início do curso. Vocês são a família que a faculdade me deu e se mostram pessoas verdadeiramente diferenciadas. Obrigado pela parceria de sempre e por todo o conhecimento, estudos e resenhas que tivemos. Se vi mais longe é porque estive sobre ombros de gigantes.

Ao meu orientador, Prof. Walber José, agradeço imensamente por compartilhar seu vasto conhecimento nas disciplinas e por ser a pessoa que me fez enxergar novos horizontes ao longo do curso. Sou profundamente grato por despertar em mim o interesse pelas novas tecnologias e pela dedicação e paciência no tempo investido na orientação deste trabalho.

EPÍGRAFE

"I must not fear. Fear is the mind-killer. Fear is the little-death that brings total obliteration. I will face my fear. I will permit it to pass over me and through me. And when it has gone past I will turn the inner eye to see its path. Where the fear has gone there will be nothing. Only I will remain."

(Frank Herbert, Dune)

RESUMO

Tendo em vista o baixo índice de leitura entre estudantes brasileiros, este trabalho apresenta o LEIA+, uma plataforma *web* educacional desenvolvida para incentivar a prática da leitura em sala de aula. A solução disponibiliza textos organizados em trechos sequenciais com questões interpretativas de múltipla escolha e é assistida pela inteligência artificial generativa do Google Gemini 2.5 Flash para a geração automática de questões e a produção de *feedback* personalizado ao aluno. A plataforma é dividida em dois módulos: o módulo do docente conta com um painel de acompanhamento em tempo real da participação dos discentes, ao passo que o módulo do aluno é acessado por QR *code* ou *link*, sem necessidade de cadastro. A plataforma foi desenvolvida com tecnologias *web* modernas, hospedada em infraestrutura *serverless* na AWS e validada por meio de testes *end-to-end* com Playwright, com aprovação integral dos quarenta e um cenários automatizados. Os resultados evidenciam a viabilidade técnica da solução e seu potencial pedagógico para o fortalecimento das práticas de leitura no ambiente escolar.

Palavras-chave: leitura; educação; plataforma web; inteligência artificial generativa; tecnologia educacional

ABSTRACT

Given the low reading rates among Brazilian students, this work presentes LEIA+, an educational *web* platform developed to encourage reading practice in the classroom. The solution provides texts organized into sequential excerpts with multiple-choice interpretive questions and is assisted by Google Gemini 2.5 Flash's generative artificial intelligence for the automatic generation of questions and the production of personalized *feedback* for the student. The platform is divided into two modules: the teacher's module features a real-time dashboard for monitoring student participation, while the student's module is accessed via QR *code* or *link*, with no registration required. The platform was built using modern *web* technologies, hosted on a *serverless* infrastructure on AWS, and validated through *end-to-end* tests with Playwright, with all forty-one automated scenarios passing. The results demonstrate the technical feasibility of the solution and its pedagogical potential for strengthening Reading practices in the school environment.

Keywords: reading; education; web platform; generative artificial intelligence; educational technology

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
E2E	*End-to-End*
GenIA	Inteligência Artificial Generativa
GPT	*Generative Pre-trained Transformer*
HTTP	*Hypertext Transfer Protocol*
JSON	*JavaScript Object Notation*
UML	*Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

Além de espaço de comunicação, o meio digital apresenta significativo potencial pedagógico, podendo ser utilizado como recurso didático no contexto escolar (Pereira, 2020). Quando integrada de forma planejada ao processo de ensino-aprendizagem, a tecnologia amplia as estratégias metodológicas e favorece maior participação dos estudantes, deixando de ser apenas um suporte técnico para tornar-se elemento estruturante das práticas pedagógicas.

Entretanto, apesar das possibilidades oferecidas pelos recursos digitais, um desafio significativo persiste no cenário educacional brasileiro: o baixo nível de leitura e de compreensão crítica entre os estudantes. Estudos apontam que 66% dos alunos brasileiros não realizam a leitura de textos com mais de dez páginas (Figueiredo, 2023). Esse dado evidencia que o simples acesso à informação não garante o desenvolvimento de competências leitoras, tornando necessária a adoção de estratégias que articulem tecnologia e incentivo à leitura de forma estruturada.

Em paralelo a este contraste, o surgimento do ChatGPT, *chatbot* lançado no final de 2022 e baseado no modelo *Generative Pre-trained Transformer* (GPT), assim como de outras Inteligências Artificiais (IAs) Generativas, marcou o início de uma significativa mudança de paradigmas nas práticas pedagógicas (Baidoo-Anu; Ansah, 2023). Ao integrarem o ChatGPT em suas atividades de pesquisa, ensino e avaliação, educadores destacam que a automação de tarefas manuais e a criação acelerada de questões abertas alinhadas aos objetivos de aprendizagem os possibitam a dedicar mais tempo para outras atividades importantes (Baidoo-Anu; Ansah, 2023), viabilizando uma maior atenção direta aos alunos. Diante disso, torna-se promissora a premissa de utilizar a capacidade dessas *Large Language Models* (LLMs) para a geração de conteúdo inédito e personalizado, abrindo novos caminhos para mitigar as barreiras do letramento tradicional.

Nesse cenário, torna-se pertinente o desenvolvimento de um sistema *web* voltado ao incentivo à leitura em sala de aula, em que a premissa é ter uma plataforma estruturada para disponibilizar textos curtos e diversificados, organizados por níveis de complexidade e gêneros textuais, permitindo que os estudantes realizem leituras, respondam a questionamentos e desenvolvam interpretações críticas. Ao professor, é disponibilizado um módulo baseado em Inteligência Artificial Generativa (GenIA) como suporte à elaboração automatizada de questões interpretativas e à geração de *feedback* personalizado, além de um painel de acompanhamento em tempo real da participação dos discentes, ampliando as possibilidades de intervenção pedagógica.

Visto que as tecnologias abrem novas possibilidades para a educação e para as práticas

pedagógicas (Oliveira; Moura, 2015), o desenvolvimento da plataforma fundamenta-se na utilização de tecnologias voltadas à construção de aplicações *web*, priorizando a interatividade e a organização clara da interface. Dessa forma, o projeto busca contribuir para a integração efetiva entre tecnologia e educação, apresentando uma solução alinhada às demandas contemporâneas do ensino e ao fortalecimento das práticas de leitura no ambiente escolar.

1.1 Problemática

A forma como as pessoas acessam a informação tem mudado nos últimos anos, um cenário que se reflete de maneira expressiva na realidade brasileira. Estudos recentes lançados pelo COMITÊ GESTOR DA INTERNET NO BRASIL e Núcleo de Informação e Coordenação do Ponto BR (2026) apontam que as plataformas de internet consolidaram-se como o principal meio de consumo de conteúdos e notícias no país, superando veículos tradicionais de comunicação em massa como o rádio e a televisão.

Apesar da ampla disponibilidade de recursos tecnológicos capazes de expandir as possibilidades pedagógicas e contribuir para a construção do conhecimento e para a renovação das práticas de ensino por meio das mídias digitais (Pereira, 2020), os índices de leitura no Brasil revelam um cenário preocupante. Dados recentes indicam que cerca de 53% da população não leu nem parte de um livro nos três meses anteriores ao levantamento, além de uma redução de aproximadamente 6,7 milhões de leitores nos últimos quatro anos, configurando o maior percentual de não leitores desde o início da série histórica (Santos, 2024). Esse contraste evidencia que, embora existam recursos tecnológicos disponíveis, o incentivo à leitura ainda se apresenta como um desafio significativo no contexto brasileiro.

Diante desse cenário, observa-se que a integração de ferramentas digitais ao ambiente escolar nem sempre se traduz em estratégias efetivas de promoção da leitura. Assim, questiona-se: de que forma um sistema *web* com suporte de GenIA, utilizada como recurso pedagógico em sala de aula, pode influenciar o interesse e os hábitos de leitura dos alunos?

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver e validar funcionalmente uma plataforma *web* educacional integrada a serviços de GenIA, projetada com requisitos e funcionalidades voltados ao incentivo à leitura e

ao engajamento de discentes com conteúdos literários digitais.

1.2.2 Objetivos Específicos

- Analisar o uso de tecnologias digitais e modelos de linguagem no contexto educacional como base teórica para o *design* de ferramentas de apoio à leitura.
- Modelar o sistema por meio de diagramas de casos de uso e diagramas de sequência, representando as funcionalidades, o fluxo de dados e as interações do ecossistema *web*.
- Projetar a interface da plataforma organizada em módulos distintos para o docente e o discente, adequando a apresentação às especificidades e aos fluxos de uso de cada perfil.
- Implementar a solução tecnológica utilizando arquitetura *serverless* e integração com a *Application Programming Interface* (API) de GenIA, contemplando as funcionalidades de gerenciamento de turmas, criação de tarefas e geração automatizada de questionários.
- Validar o sistema desenvolvido por meio de testes de sistema automatizados de ponta a ponta, ou *End-to-End* (E2E), aferindo sua conformidade funcional, estabilidade operacional e resiliência de integração.

1.3 Justificativa

A proposta da plataforma justifica-se pela necessidade de oferecer um espaço digital que favoreça a reflexão crítica, o debate e a troca de ideias no contexto educacional. O foco principal consiste em incentivar o hábito da leitura de forma gradual, sem sobrecarregar os estudantes, ao mesmo tempo em que se promove a interação social mediada por tecnologias digitais. Espera-se, assim, contribuir para a formação de leitores mais críticos, participativos e engajados, alinhando-se às demandas contemporâneas da educação e às práticas pedagógicas que valorizam o pensamento crítico e a participação ativa dos alunos.

1.4 Estrutura do documento

O presente trabalho está organizado em cinco capítulos, incluindo esta introdução. O Capítulo 2 apresenta a fundamentação teórica que sustenta a pesquisa, abordando o papel da leitura na formação do indivíduo, a inserção das tecnologias digitais no contexto educacional, as tecnologias empregadas no desenvolvimento da aplicação, além de discutir trabalhos correlatos que estabelecem um panorama de soluções similares e destacam as contribuições deste estudo. O

Capítulo 3 descreve a metodologia de pesquisa e o processo de desenvolvimento, com ênfase nas etapas de modelagem, arquitetura e na estratégia de validação funcional. O Capítulo 4 apresenta os resultados obtidos a partir do desenvolvimento do artefato e da validação técnica do sistema via testes automatizados. Por fim, o Capítulo 5 sintetiza as principais conclusões da pesquisa e aponta perspectivas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

Este capítulo aborda os fundamentos teóricos que sustentam o desenvolvimento deste trabalho, discutindo o papel da leitura, a inserção das tecnologias digitais no contexto educacional e o uso da GenIA na educação. Na sequência, são descritas as principais tecnologias que viabilizaram o desenvolvimento e a validação da plataforma *web*, concluindo com uma análise comparativa dos trabalhos relacionados na literatura.

2.1 A importância da leitura

Na perspectiva de Carvalho e França (2021), a leitura constitui-se como uma das práticas mais relevantes para o desenvolvimento humano, atuando tanto na dimensão cognitiva quanto na dimensão social do sujeito. Para o autor, mais do que decodificar signos linguísticos, o ato de ler envolve a construção de sentidos, a ampliação do repertório cultural e o fortalecimento da capacidade interpretativa do indivíduo. Nesse contexto, o pesquisador compreende a leitura como uma prática emancipatória, na qual o leitor amplia sua autonomia intelectual e desenvolve o pensamento crítico necessário para refletir sobre o mundo que o cerca.

Ao estabelecer contato com diferentes perspectivas, narrativas e produções de conhecimento, o leitor passa a explorar contextos e realidades que ultrapassam sua vivência imediata. Esse processo desenvolve habilidades interpretativas que tornam o indivíduo mais preparado para refletir, questionar e posicionar-se de maneira ativa diante das múltiplas demandas sociais (Carvalho; França, 2021). Dessa forma, a leitura ultrapassa a dimensão meramente instrumental e assume um papel formativo, contribuindo para a constituição de sujeitos mais conscientes, participativos e preparados para atuar na sociedade contemporânea.

Reforçando essa perspectiva, Belo *et al.* (2024) destacam que a leitura desempenha papel central no desenvolvimento integral do indivíduo, influenciando não apenas o desempenho escolar, mas também a formação cidadã e a capacidade de inserção crítica na sociedade. Os autores enfatizam que o contato sistemático com diferentes gêneros textuais favorece a construção de competências linguísticas, cognitivas e socioemocionais, evidenciando a leitura como prática multidimensional.

Apesar dessa relevância, o cenário brasileiro revela um descompasso entre a importância atribuída à leitura e os hábitos efetivamente praticados pela população. Dados recentes apontam que cerca de 53% da população brasileira não leu nem parte de um livro nos três meses anteriores

ao levantamento, configurando o maior percentual de não leitores desde o início da série histórica (Santos, 2024). No contexto escolar, o quadro também é preocupante: 66% dos estudantes brasileiros não realizam a leitura de textos com mais de dez páginas (Figueiredo, 2023). Esses indicadores reforçam a necessidade de estratégias pedagógicas capazes de aproximar o estudante da prática leitora de forma gradual e significativa.

2.2 Tecnologias digitais na educação

Com o avanço das tecnologias digitais, o universo da leitura tornou-se mais acessível, especialmente por meio da internet (Pereira, 2020). Esse movimento impulsiona a educação a se reinventar, adaptando-se ao desenvolvimento tecnológico e promovendo a superação gradual de abordagens estritamente tradicionais de ensino (Sousa, 2020). Nesse processo, o meio digital deixa de ser apenas um espaço de comunicação para assumir o papel de recurso pedagógico, capaz de mediar e potencializar o ensino-aprendizagem.

A integração das Tecnologias da Informação e Comunicação (TICs) ao ambiente escolar configura-se como um caminho promissor para a renovação das práticas pedagógicas. Conforme apontam Oliveira e Moura (2015), a utilização das TICs em sala de aula amplia as possibilidades metodológicas do professor, favorece a adoção de personalização do ensino e estimula a participação ativa do estudante. Por consequência, a tecnologia deixa de ser apenas um suporte técnico e passa a constituir-se como elemento estruturante das práticas educativas.

No campo específico da leitura, a internet disponibiliza uma ampla gama de recursos que podem ser explorados pelos educadores com o objetivo de potencializar o ensino e aprimorar as práticas leitoras (Pereira, 2020). Esses recursos favorecem novas formas de interação com os textos, especialmente com os gêneros inseridos no ambiente digital, contribuindo para a construção do conhecimento e para a consolidação de uma perspectiva inovadora acerca do processo de ensino-aprendizagem. A leitura mediada por dispositivos digitais permite, por exemplo, a inclusão de elementos multimídia, hipertextos e atividades interativas que aumentam o engajamento dos estudantes.

Entretanto, a presença das tecnologias no ambiente escolar não garante, por si só, a melhoria das práticas pedagógicas. Sua efetividade depende de uma integração consciente e planejada ao processo de ensino-aprendizagem, na qual o recurso digital seja articulado a objetivos pedagógicos claros (Sousa, 2020). Assim, a proposição de aplicações digitais que articulem leitura, interação e mediação docente configura-se como uma estratégia alinhada às

demandas contemporâneas da educação.

2.3 GenIA no contexto educacional

A GenIA refere-se a uma classe de sistemas computacionais capazes de produzir novos conteúdos (como textos, imagens, áudios e códigos) a partir de padrões aprendidos em grandes volumes de dados. No núcleo dessas aplicações encontram-se os *Large Language Models* (LLMs), ou modelos de linguagem de grande escala, treinados para compreender e gerar texto em linguagem natural com elevado grau de coerência e contextualização.

No contexto educacional, a GenIA vem ganhando espaço como ferramenta pedagógica, especialmente em atividades que envolvem materiais didáticos, elaboração de exercícios e geração de *feedback* de desempenho em tempo real para os professores (Mattos *et al.*, 2026). Esses modelos são capazes de lidar com grandes volumes de dados, o que oferece aos professores uma visão aprofundada do progresso de cada aluno, permitindo a identificação de padrões de aprendizagem e a realização de intervenções imediatas (Mattos *et al.*, 2026). Em vista disso, a GenIA transcende seu propósito como recurso de automação, tornando-se um instrumento de apoio à tomada de decisão pedagógica, com potencial para aliviar a carga de trabalho dos professores e permitindo que estes dediquem mais tempo ao planejamento de aulas criativas e à resolução de problemas complexos em sala de aula.

Por fim, é importante destacar que a adoção de GenIA na educação requer cautela, uma vez que esses sistemas podem produzir respostas imprecisas ou enviesadas, fenômeno conhecido como *alucinação*. Por isso, o papel do docente como mediador permanece central: cabe ao professor validar o conteúdo gerado, contextualizá-lo pedagogicamente e orientar o uso crítico da ferramenta pelos estudantes. Neste trabalho, a GenIA é empregada por meio da API do Google Gemini, modelo que possibilita a geração automatizada de questões interpretativas e a elaboração de *feedback* personalizado a partir de parâmetros definidos pelo docente.

2.4 Trabalhos Relacionados

O objetivo desta seção é apresentar uma análise entre a plataforma desenvolvida neste trabalho e outras encontradas na literatura que possuem propostas alinhadas a este projeto, relacionando as similaridades e diferenças entre os sistemas com base em seu contexto de aplicação.

2.4.1 *TeachHelp - Sistema para Auxílio do Professor*

Pereira, Rosa e Martins (2021), motivados pelas dificuldades enfrentadas por docentes na gestão de suas rotinas e pela ausência de ferramentas digitais adequadas, apresentam o *TeacHelp*: uma plataforma *web* que possibilita ao professor cadastrar turmas e alunos, registrar notas e frequências, registrar observações individuais sobre cada estudante e visualizar relatórios gerais de desempenho por turma.

O TeacHelp difere da solução proposta neste trabalho por direcionar-se exclusivamente à gestão administrativa da rotina docente, ao passo que esta plataforma foca no processo de aprendizagem por meio da disponibilização de tarefas aos alunos. Além disso, o fluxo operacional do TeacHelp é inteiramente manual, sem o emprego de recursos de inteligência artificial para a automatização da gestão.

Ademais, apesar de suas contribuições ao âmbito da gestão pedagógica, o sistema também não contempla interação entre professor e aluno. Dessa forma, a plataforma carece de funcionalidades voltadas à realização de leituras e atividades pelos estudantes, recursos que poderiam enriquecer o processo avaliativo e ampliar o engajamento discente.

2.4.2 Revisa+ - Plataforma Web de Revisão de Conteúdos Escolares Por Meio de Histórias em Quadrinhos

O Revisa+ (Silva *et al.*, 2024) é uma a aplicação *web* com a finalidade de facilitar e aprimorar a revisão de conteúdos escolares da educação básica e é estruturada em três pilares: o primeiro é a história em quadrinhos em primeira pessoa, núcleo da proposta, que situa o aluno em situações do cotidiano para facilitar a assimilação do conteúdo; o segundo é o resumo teórico, que retoma os principais conceitos da disciplina estudada; e o terceiro é o *quiz* animado, composto por cinco questões contextuais, no qual o aluno obtém ao final o número de acertos e a identificação das respostas corretas e incorretas, promovendo assim um retorno imediato sobre seu desempenho. A plataforma abrange as cinco disciplinas da educação básica (História, Português, Matemática, Geografia e Inglês), permitindo que o aluno navegue pelos conteúdos de forma autônoma e interativa.

Embora tanto a plataforma proposta quanto o Revisa+ visem à aprendizagem, este último adota uma abordagem distinta ao concentrar-se na revisão de conteúdos escolares por meio de histórias em quadrinhos e *quizzes* contextuais. Ademais, todo o fluxo de criação das histórias no

Revisa+ é manual, sem o uso de recursos de IA para a automatização da geração de conteúdo. Essa limitação, reconhecida pelos próprios autores, torna o processo artesanal e dificulta o crescimento exponencial dos conteúdos disponibilizados na plataforma.

2.4.3 Leia Rapidinho: Uma aplicação para exercitar a leitura de palavras da língua portuguesa

Braga (2021), diante dos índices preocupantes de alfabetização no Brasil e da pouca atratividade dos exercícios tradicionais de leitura, apresenta o Leia Rapidinho: uma aplicação *web* gamificada que busca aprimorar a fluência na leitura de palavras em estudantes em processo de alfabetização. A solução exibe palavras da língua portuguesa na tela e desafia o usuário a reproduzi-las oralmente, empregando uma *API* de reconhecimento de voz disponível em alguns navegadores para validar cada leitura. Recursos de gamificação, como restrições de tempo, conquistas e gráficos de desempenho, mantêm o engajamento do aluno, enquanto os responsáveis (pais ou professores) podem acompanhar seu progresso.

O Leia Rapidinho foca no exercício da fluência de leitura de palavras isoladas durante o processo de alfabetização, enquanto a solução proposta neste trabalho tem como público-alvo os professores e, sobretudo, os estudantes do ensino médio, dos quais se pressupõe que já sejam alfabetizados (leitores), de modo a realizarem as tarefas de leitura propostas pelos docentes. Além disso, a solução proposta vai além ao automatizar, por meio da GenIA, a geração dessas tarefas para o professor.

2.4.4 Análise comparativa

O Quadro 1 sintetiza as principais características dos trabalhos analisados em relação ao sistema proposto, destacando os aspectos centrais de cada solução no contexto educacional digital.

Quadro 1 – Comparativo entre trabalhos relacionados e o sistema proposto

Solução	**Objetivo Geral**	**Tópico de Estudo**	**Público-alvo**	**Metodologia**
TeacHelp	Gestão do trabalho docente	Organização pedagógica	Professores	Plataforma *web*
Revisa+	Revisão de conteúdos escolares por meio de histórias em quadrinhos	Gamificação educacional	Alunos do ensino fundamental	Plataforma *web* com *quiz* animado
Leia Rapidinho	Exercitar a fluência na leitura de palavras	Alfabetização e leitura	Alunos em alfabetização	Plataforma *web* gamificada com reconhecimento de voz
Trabalho Proposto	Incentivo à leitura em sala de aula apoiado por GenIA	Leitura e IA na educação	Professores e alunos do ensino básico	Plataforma *web* com GenIA

Fonte: Autoria própria (2026).

A partir da análise comparativa sintetizada no Quadro 1, percebe-se que as soluções operam de forma fragmentada, embora ofereçam contribuições relevantes para a gestão docente ou para o estímulo à aprendizagem lúdica. Os sistemas focados no gerenciamento pedagógico não possuem o módulo para o estudanto, enquanto plataformas voltadas ao engajamento do discente carecem de ferramentas de controle em tempo real para os professores. Ademais, nota-se a dependência de processos puramente manuais para a criação de conteúdos didáticos, o que limita a escalabilidade dessas aplicações.

O trabalho proposto diferencia-se dos demais ao unificar a experiência de professores e alunos do ensino básico em um ambiente integrado de incentivo à leitura. O grande diferencial competitivo e científico desta pesquisa reside na incorporação da GenIA. Essa tecnologia é empregada não apenas para mitigar o gargalo da produção manual de atividades evidenciado na literatura, mas também para possibilitar a automação pedagógica e a personalização do aprendizado em larga escala. Dessa forma, as seções subsequentes deste documento apresentarão a metodologia e o desenvolvimento da solução proposta, detalhando como tais lacunas foram superadas.

3 METODOLOGIA

Este capítulo apresenta a metodologia adotada para o desenvolvimento e a validação funcional da plataforma proposta. Inicialmente, caracteriza-se a natureza da pesquisa quanto à abordagem e aos objetivos. Em seguida, descrevem-se as etapas que estruturam o processo de construção da aplicação, abrangendo desde a modelagem do sistema, baseada na *Unified Modeling Language* (UML), até a implementação da arquitetura *serverless*. Por fim, detalham-se os procedimentos de validação do *software* por meio de testes automatizados de ponta a ponta (E2E), utilizados como critério de verificação da consistência e da robustez técnica do artefato desenvolvido.

3.1 Caracterização da pesquisa

Este estudo caracteriza-se como uma pesquisa de natureza aplicada e abordagem quantitativa, segundo a classificação de Gerhardt e Silveira (2009). É aplicada por voltar-se à criação de uma solução tecnológica real para o ambiente de ensino, materializada no desenvolvimento de uma ferramenta de *software* (Gerhardt; Silveira, 2009, p. 35). É quantitativa porque a verificação do sistema se apoia na medição de indicadores de engenharia de *software*, como a taxa de êxito nos testes automatizados, com a meta de 100% de sucesso nos cenários avaliados (Gerhardt; Silveira, 2009, p. 33).

Quanto aos objetivos, a pesquisa é exploratória e descritiva, conforme Gil (2002). O caráter exploratório reflete a familiarização com um problema ainda pouco consolidado, referente ao desenvolvimento de uma plataforma *web* apoiada por GenIA para o contexto escolar (Gil, 2002, p. 41); o descritivo evidencia-se no detalhamento das características da solução, como a implementação da arquitetura e a organização do código (Gil, 2002, p. 42). Como procedimento técnico, recorre-se à pesquisa tecnológica, associada à engenharia de testes de *software*, para confirmar empiricamente a robustez e a adequação funcional do artefato.

3.2 Etapas do desenvolvimento

O processo de desenvolvimento da plataforma abrange quatro etapas sequenciais, com revisões pontuais sempre que ajustes se mostram necessários. Essa organização é essencial para garantir a coerência entre a modelagem, o projeto da interface, a implementação na infraestrutura em nuvem e a posterior homologação via testes automatizados.

3.2.1 Modelagem do sistema

No contexto da Engenharia de Software, a modelagem de sistemas constitui uma etapa fundamental, pois permite o desenvolvimento de representações abstratas e estruturadas das funcionalidades, dos componentes e das interações que podem ocorrer no sistema antes de sua implementação (Sommerville, 2018). Essa atividade contribui para a comunicação entre os envolvidos no projeto, a identificação precoce de inconsistências e a documentação técnica da solução.

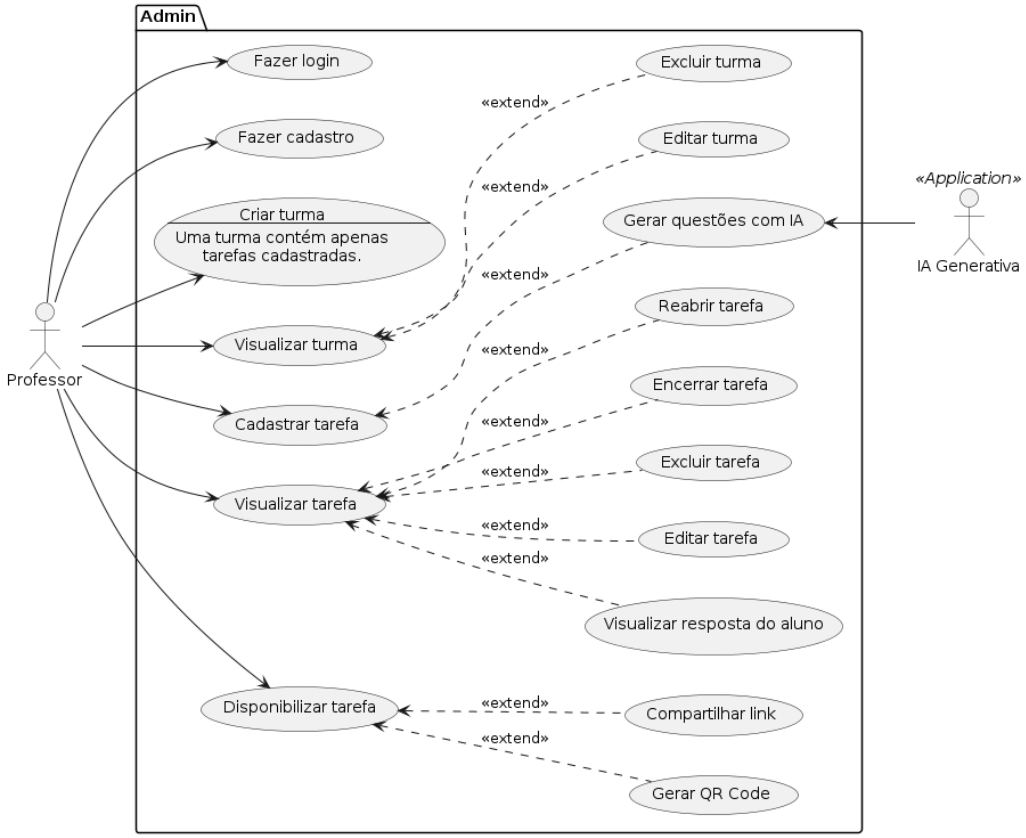
Neste trabalho, adotam-se o Diagrama de Casos de Uso, como instrumento para representar a interação dos usuários com o sistema, e o Diagrama de Sequência, utilizado para modelar a ordem das interações realizadas durante a execução de um determinado caso de uso. Ambos os diagramas são baseados na UML, linguagem padronizada amplamente empregada para especificação, visualização e documentação de sistemas de *software*.

3.2.1.1 Diagramas de casos de uso

O Diagrama de Casos de Uso elaborado contempla dois atores humanos, o docente e o discente, além de um ator de sistema correspondente à GenIA, responsável pela geração automatizada de questões interpretativas. Por concentrar as funcionalidades de gestão da plataforma, o módulo administrativo, destinado ao docente, reúne o maior conjunto de casos de uso, detalhados a seguir; os casos de uso do discente são apresentados na sequência.

A Figura 1 apresenta os casos de uso associados ao docente. Após autenticar-se na plataforma ou realizar seu cadastro, o professor gerencia as turmas sob sua responsabilidade, podendo criar, visualizar, editar e excluir turmas. De modo análogo, administra as tarefas de leitura: a partir de um texto dividido em trechos, cadastra a tarefa e, opcionalmente, recorre à GenIA para gerar questões interpretativas de forma automatizada, caso de uso que estende o cadastro da tarefa, uma vez que as questões também podem ser elaboradas manualmente. As demais operações sobre a tarefa (visualizar, editar, excluir, encerrar e reabrir), assim como a visualização das respostas submetidas pelos discentes, ocorrem a partir da tela de detalhamento da tarefa. Por fim, o docente disponibiliza a tarefa aos alunos, compartilhando-a por meio de um *link* ou de um *QR Code*.

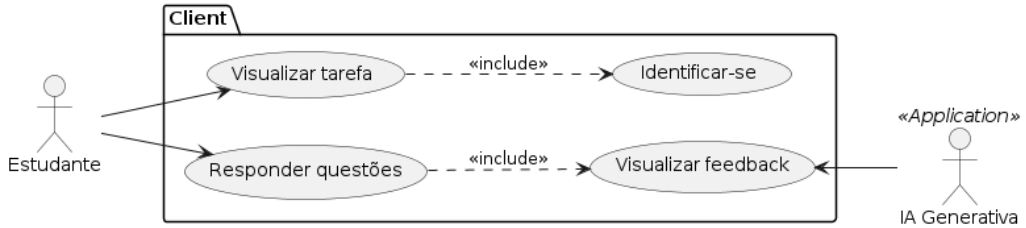
Figura 1 – Diagrama de casos de uso do docente.



Fonte: Autoria própria (2026).

A Figura 2 apresenta os casos de uso associados ao discente, reunidos no módulo cliente. O aluno acessa a tarefa por meio do *link* ou *QR Code* disponibilizado pelo docente e visualiza suas informações iniciais, como título, turma, quantidade de trechos e questões e tempo estimado de leitura. Para iniciar a atividade, identifica-se informando o nome, etapa que integra a visualização da tarefa. Em seguida, realiza a leitura do texto, organizado em trechos sucessivos, e responde às questões interpretativas correspondentes. Ao concluir, o discente visualiza o *feedback* de seu desempenho, gerado pela GenIA a partir das respostas submetidas, caso de uso necessariamente associado ao ato de responder.

Figura 2 – Diagrama de casos de uso do discente.



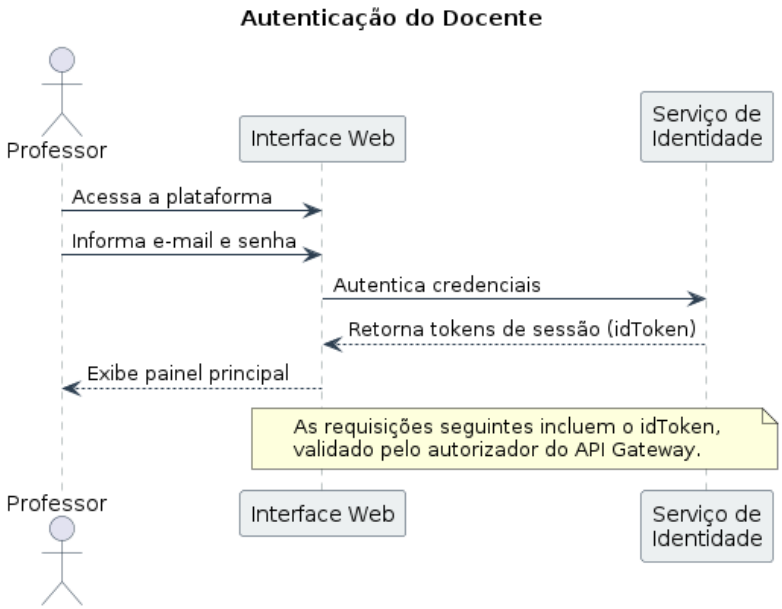
Fonte: Autoria própria (2026).

3.2.1.2 Diagramas de sequência

Para complementar a visão estática dos casos de uso, são elaborados Diagramas de Sequência que detalham a ordem temporal das interações entre os participantes do sistema em fluxos considerados centrais. O fluxo do docente é apresentado em cinco etapas sucessivas: autenticação, criação de turma, cadastro de tarefa com apoio da GenIA, disponibilização da tarefa e visualização das respostas dos discentes, cada uma ilustrada separadamente a seguir.

A Figura 3 ilustra a etapa de autenticação do docente. O professor acessa a plataforma e informa suas credenciais (e-mail e senha) à interface *web*, que as encaminha ao Amazon Cognito. Após validar as credenciais, o serviço retorna os *tokens* de sessão (em especial o *idToken*), que passam a acompanhar, no cabeçalho `Authorization`, todas as requisições subsequentes à API. Concluída a autenticação, o painel principal é exibido ao docente.

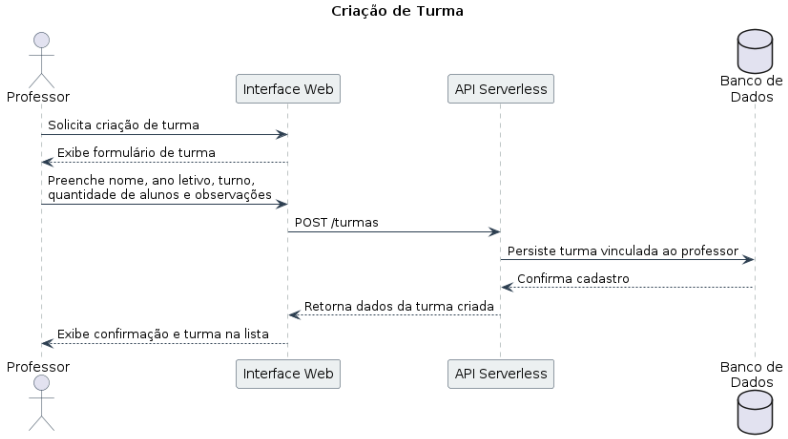
Figura 3 – Diagrama de sequência — autenticação do docente.



Fonte: Autoria própria (2026).

A Figura 4 apresenta a criação de uma turma. A partir do painel principal, o docente solicita a criação de uma nova turma e preenche o formulário com informações como nome, ano letivo, turno, quantidade de alunos e observações. A interface envia esses dados à API *Serverless* (rota `POST /turmas`), que os persiste no banco de dados vinculados ao identificador do docente. Concluído o cadastro, a nova turma é exibida na listagem.

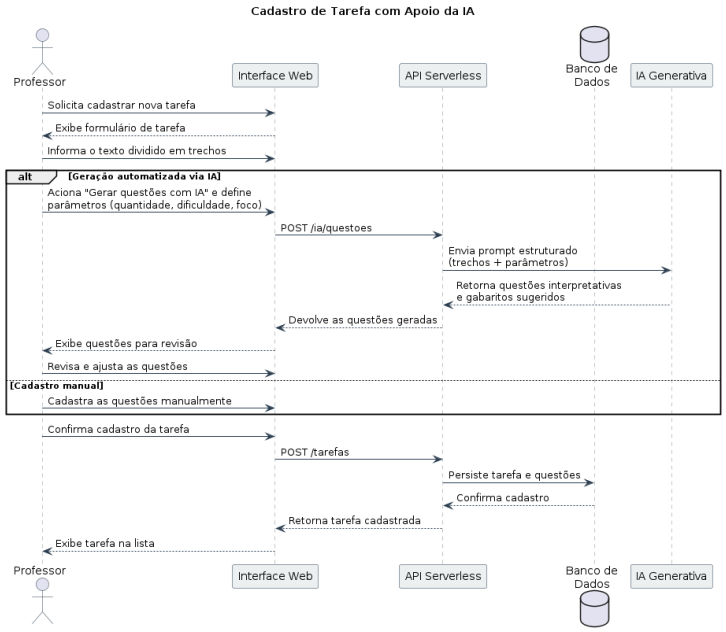
Figura 4 – Diagrama de sequência — criação de turma pelo docente.



Fonte: Autoria própria (2026).

A Figura 5 detalha o cadastro de uma tarefa de leitura, contemplando dois caminhos alternativos. No primeiro, o docente aciona a geração automatizada de questões pela GenIA: fornece o texto dividido em trechos e define parâmetros de geração (quantidade de questões, nível de dificuldade e tipo de foco); a API encaminha um *prompt* estruturado ao modelo generativo, que retorna questões interpretativas e gabaritos sugeridos para revisão e ajuste pelo docente. No segundo caminho, o docente elabora as questões manualmente. Independentemente da abordagem adotada, o docente confirma o cadastro e a tarefa, juntamente com suas questões, é persistida no banco de dados via `POST /tarefas`.

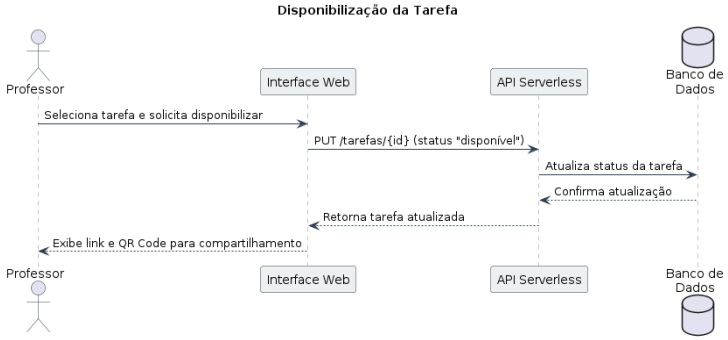
Figura 5 – Diagrama de sequência — cadastro de tarefa com apoio da GenIA.



Fonte: Autoria própria (2026).

A Figura 6 corresponde à disponibilização da tarefa aos discentes. O docente seleciona a tarefa desejada e solicita sua disponibilização; a interface envia uma requisição de atualização de *status* à API (rota `PUT /tarefas/{id}`), que persiste a alteração no banco de dados. Em resposta, a interface exibe ao docente o *link* de acesso e o *QR Code* gerado para compartilhamento com a turma.

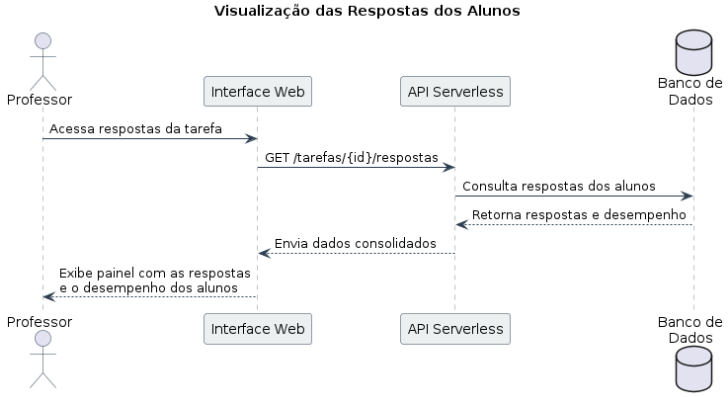
Figura 6 – Diagrama de sequência — disponibilização da tarefa.



Fonte: Autoria própria (2026).

A Figura 7 descreve o acompanhamento das respostas submetidas pelos discentes. O docente acessa a área de respostas de uma tarefa; a interface consulta a API (rota `GET /tarefas/{id}/respostas`), que recupera do banco de dados as respostas e os dados de desempenho consolidados. Os resultados são então exibidos em um painel que permite ao docente acompanhar a participação e o desempenho da turma.

Figura 7 – Diagrama de sequência — visualização das respostas dos discentes.

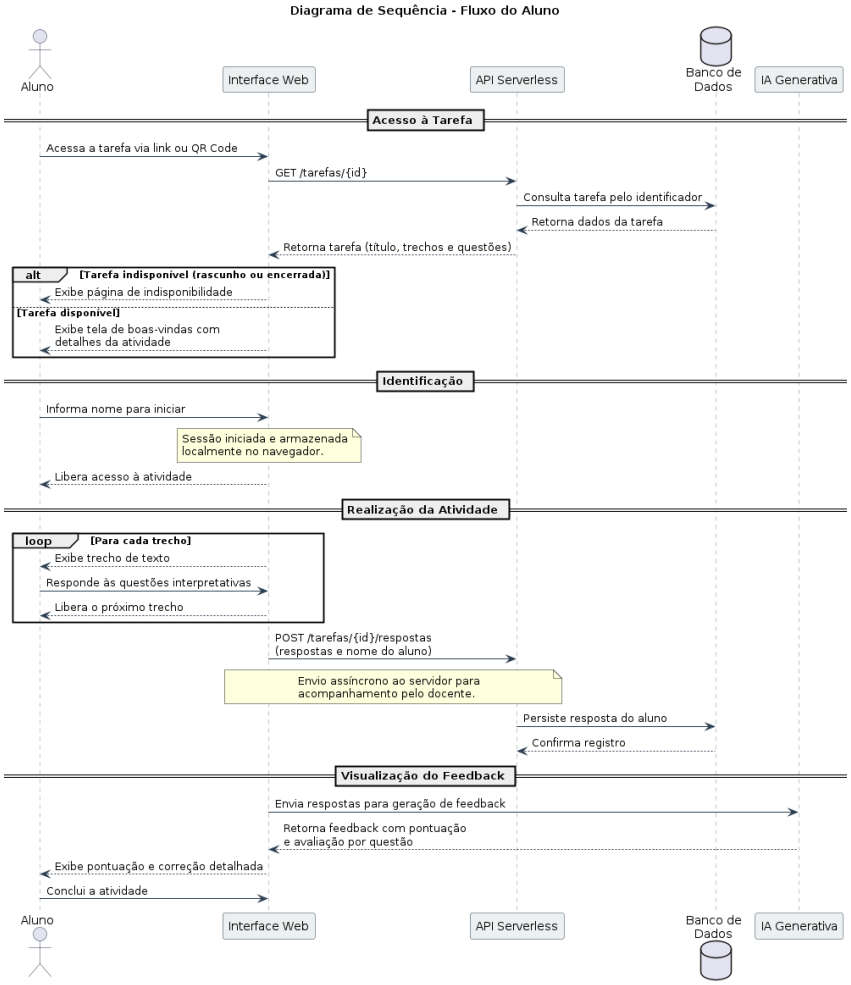


Fonte: Autoria própria (2026).

Por fim, a Figura 8 apresenta o fluxo do discente. Ao acessar a tarefa por meio de um *link* ou *QR Code*, a interface recupera os dados da atividade junto à API, verificando o *status* da tarefa: caso esteja indisponível, o discente é direcionado a uma página de aviso; caso contrário,

visualiza a tela de boas-vindas com os detalhes da atividade. Para iniciar, o discente informa seu nome, cuja sessão é armazenada localmente pelo navegador, sem envolvimento do servidor. A atividade é realizada em trechos sucessivos, isto é, o discente lê cada trecho e responde às questões interpretativas antes de avançar. Ao concluir, as respostas são enviadas de forma assíncrona à API, a fim de que o docente possa acompanhar o desempenho da turma, sem que esse envio bloqueie a navegação do discente. Por fim, a GenIA processa as respostas e retorna o *feedback* com a pontuação e a avaliação detalhada por questão.

Figura 8 – Diagrama de sequência do fluxo do discente.



Fonte: Autoria própria (2026).

3.2.2 Projeto da interface

O projeto da interface pautou-se por diretrizes gerais de clareza e de consistência visual, considerando a diversidade de níveis de familiaridade dos usuários com tecnologias digitais (Silva, 2025). A interface é organizada em dois módulos distintos: um módulo administrativo, destinado

ao docente, voltado à criação e ao acompanhamento de turmas e tarefas; e um módulo cliente, destinado ao discente, focado em uma experiência de leitura centrada no texto. A separação entre os módulos permite ajustar a tipografia, o espaçamento e a densidade de informação às necessidades específicas de cada público, favorecendo o engajamento na atividade leitora. Cabe ressaltar que essas escolhas constituíram decisões de projeto: no escopo deste trabalho, não foram aplicados métodos formais de avaliação de usabilidade, como testes com usuários ou inspeções heurísticas, cuja realização é apontada entre os trabalhos futuros no Capítulo 5.

3.2.3 Implementação

A implementação da plataforma adota a arquitetura *serverless*, modelo no qual a infraestrutura de execução é abstraída pelo provedor de nuvem e as funcionalidades do *back-end* são organizadas em funções executadas sob demanda. Essa decisão favorece a escalabilidade, a redução de custos operacionais e a simplificação da manutenção, mostrando-se adequada ao caráter pontual da utilização da aplicação em ambiente escolar.

A organização do código-fonte é estruturada em um *monorepo* gerenciado pela ferramenta Turborepo, modelo no qual os módulos administrativo, cliente e *back-end* compartilham um único repositório, juntamente com pacotes utilitários comuns. Para a construção das interfaces, opta-se pelo *framework* Vite em conjunto com a biblioteca React, e pelo Tailwind CSS para a camada de estilização. A persistência dos dados é apoiada por um banco de dados não relacional gerenciado em nuvem, e a autenticação dos docentes é mediada por um serviço dedicado de identidade. A integração com a GenIA é realizada por meio da API do Google Gemini, responsável pela geração das questões interpretativas e pela produção do *feedback* apresentado aos discentes.

A infraestrutura do *back-end* é provisionada na *Amazon Web Services* (AWS) e organizada em torno de três serviços gerenciados: o Amazon API *Gateway*, responsável por concentrar e rotear todas as requisições *Hypertext Transfer Protocol* (HTTP); o AWS *Lambda*, no qual residem as funções de negócio executadas sob demanda; e o Amazon DynamoDB, banco de dados não relacional utilizado para a persistência das entidades da aplicação. Essa composição evita a necessidade de servidores dedicados e delega ao provedor a responsabilidade pelo escalonamento automático e pela alta disponibilidade da plataforma.

A autenticação dos docentes é mediada pelo Amazon Cognito, serviço que gerencia o ciclo completo de identidade do usuário. Esse gerenciamente inclui o cadastro, a verificação de endereço de e-mail, recuperação de senha e a autorização de acesso aos recursos da plataforma

com a emissão de *tokens* JWT. Uma vez autenticado, o *token* gerado deve ser enviado no cabeçalho *Authorization* para validar as requisições do usuário junto às rotas protegidas da API.

O Amazon API *Gateway* expõe uma única REST API que agrupa os recursos da aplicação em quatro caminhos raiz: `/turmas`, `/tarefas`, `/respostas` e `/ia`. Um autorizador é vinculado às rotas sensíveis à identidade do docente, de modo que o *Gateway* valida o *token* JWT diretamente junto ao Cognito antes de encaminhar a requisição à função correspondente. As rotas de consulta destinadas ao discente (`GET /tarefas`, `GET /tarefas/{id}` e `GET /respostas`) permanecem públicas, dispensando autenticação, uma vez que o aluno acessa as atividades por meio de um *link* ou *QR Code* sem necessidade de cadastro.

As funções *Lambda* da arquitetura foram organizadas em quatro grupos funcionais de acordo com seus respectivos recursos de API. Os dois primeiros grupos, vinculados aos recursos `/turmas` e `/tarefas`, implementam operações CRUD (criação, leitura, atualização e exclusão) utilizando o ambiente de execução Node.js. Essas funções operam diretamente nas tabelas homônimas do DynamoDB (`turmas` e `tarefas`). O terceiro grupo, associado ao recurso `/respostas`, gerencia as interações dos discentes por meio das funções `create-response` (registro de respostas) e `get-responses` (consulta ao histórico de submissões).

Por fim, o recurso `/ia` engloba os serviços de GenIA do sistema, compostos por duas funções desenvolvidas em Python. Devido à latência intrínseca dos modelos generativos, ambas foram configuradas com *timeout* de 180 segundos e 512 MB de memória RAM. A primeira delas, `generate-questions` (`POST /ia/questoes`), permite ao docente gerar novas tarefas a partir de textos fragmentados e parâmetros personalizados enviados à API do Google Gemini via HTTPS; a segunda, `generate-feedback` (`POST /ia/feedback`), é acionada pelo discente ao término da atividade para a geração de uma avaliação de desempenho personalizada. Em ambos os fluxos de GenIA, os dados retornados são validados antes da persistência ou exibição no *front-end*.

3.2.4 Gemini 2.5 Flash

O modelo de GenIA adotado neste trabalho para a criação das questões e dos *feedbacks* foi o **Gemini 2.5 Flash**, desenvolvido pela Google. A seleção desta ferramenta fundamentou-se em sua eficiência computacional e capacidade de processamento ágil diante da grande quantidade de textos de entrada (os quais foram divididos em trechos) e da complexidade estrutural das saídas demandadas (enunciados, alternativas e indicação da resposta correta).

Este modelo apresenta uma janela de contexto de até 1.000.000 (um milhão) de *tokens* de entrada (*input*) e capacidade de geração de até 65.536 (sessenta e cinco mil) *tokens* de saída (*output*) por requisição (GOOGLE, 2025a). Essa amplitude de contexto garante a retenção e a consistência das informações ao longo do processamento de entradas extensas.

No que tange aos hiperparâmetros de configuração da API, todos foram mantidos em seus valores predefinidos (GOOGLE, 2025a): a temperatura de geração foi estabelecida em 1,0, configuração que equilibra a precisão técnica e a diversidade criativa nas respostas. Adicionalmente, o parâmetro *Top-P* foi mantido em 0,95, delimitando o escopo probabilístico de seleção do vocabulário para assegurar a coesão textual.

Apesar de esta pesquisa ter sido conduzida por meio do plano gratuito de desenvolvimento (*Free Tier*) disponibilizado na plataforma Google AI Studio, o ambiente mostrou-se plenamente viável e robusto para o escopo experimental proposto, não gerando custos financeiros diretos para a execução do projeto.

3.2.5 Engenharia de *prompt*

A qualidade e a consistência das respostas de um modelo de linguagem estão diretamente ligadas à forma como as instruções são escritas. Seguindo as boas práticas de elaboração de *prompts* recomendadas por GOOGLE (2025b), que orientam a definir com clareza o papel do modelo, a especificar o formato de saída desejado e a delimitar restrições explícitas, optou-se por organizar cada requisição enviada ao Gemini em duas mensagens complementares. A primeira é a instrução de sistema (*system prompt*), de caráter fixo, que estabelece o papel assumido pelo modelo, as regras de geração e o formato de saída esperado. A segunda é a instrução de usuário (*user prompt*), de caráter dinâmico, construída em tempo de execução a partir dos dados específicos de cada solicitação. Essa separação foi adotada nas duas funções de GenIA da plataforma, a `generate-questions` e a `generate-feedback`. Em ambos os casos, a resposta do modelo foi restringida a um objeto *JavaScript Object Notation* (JSON) validado por um *schema* definido na configuração da chamada, o que garante a estrutura consistente exigida pelo *back-end* antes de os dados serem persistidos ou exibidos.

Na geração de questões, o *system prompt* define que o modelo deve atuar como um especialista em pedagogia e na elaboração de questões de interpretação de texto para o ensino básico brasileiro. A partir desse papel, a instrução reúne as regras que o modelo precisa seguir. Ele deve gerar exatamente a quantidade de questões pedida para cada trecho, e cada questão

deve trazer um enunciado claro, quatro alternativas com apenas uma correta, a resposta indicada somente pela letra correspondente e uma breve explicação apoiada no trecho. A instrução também associa os três níveis de dificuldade (`fácil`, `médio` e `difícil`) ao grau de inferência esperado e delimita os quatro focos de análise possíveis, que são a compreensão, o vocabulário, a gramática e a distinção entre informação explícita e implícita (`explicito_implicito`). Por fim, o *prompt* orienta o modelo a não inventar informações que não estejam no trecho, a não repetir perguntas e a evitar alternativas incorretas pouco plausíveis e de pedir que cada conjunto de questões seja vinculado ao seu trecho de origem por um identificador (`segmentId`) mantido sem alterações.

Já o *user prompt* dessa função é um objeto JSON montado em tempo de execução com os dados fornecidos pelo docente. Esse objeto reúne a lista de trechos (`segments`), em que cada trecho aparece com um identificador (`id`) e o texto correspondente (`text`), a quantidade de questões desejada por trecho (`questionCount`), o nível de dificuldade (`difficulty`) e a lista de focos de análise a serem contemplados (`focus`). Com essa organização, a instrução de sistema permanece a mesma em todas as requisições, enquanto a instrução de usuário carrega apenas os parâmetros que variam a cada tarefa cadastrada.

Na geração de *feedback*, o *system prompt* define que o modelo deve assumir o papel de um tutor de Língua Portuguesa que conversa diretamente com o estudante, na segunda pessoa, depois que ele conclui a atividade. A instrução pede um comentário curto para cada questão, sempre apoiado no trecho correspondente e ajustado ao desempenho. Quando o estudante acerta, o modelo reforça o raciocínio que ele empregou. Quando erra, explica com acolhimento o caminho até a resposta certa, sem qualquer tom de punição. E quando a questão fica sem resposta, convida o estudante a reler o trecho com calma. A instrução também pede uma mensagem geral (`overall`), que começa com um incentivo coerente com o desempenho obtido e traz uma ou duas sugestões de estudo ligadas às dificuldades observadas. Assim como na função de questões, cada comentário deve ser vinculado ao identificador da questão (`questionId`) sem alterações, e todo o texto deve seguir a ortografia vigente e manter um tom humano e encorajador.

Por sua vez, o *user prompt* do *feedback* é um objeto JSON que resume o desempenho do estudante ao final da atividade. Ele traz o nome do estudante (`studentName`), a pontuação percentual (`scorePct`) e a relação entre acertos e total de questões (`correctCount` e `total`), além da lista de questões respondidas (`questions`). Cada questão dessa lista informa o seu identificador (`questionId`), o trecho de origem (`segmentText`), o enunciado (`questionText`),

as alternativas (`alternatives`), a resposta correta (`correctAnswer`), a alternativa que o discente escolheu (`selectedAlternative`) e se ela estava certa (`isCorrect`). Com todas essas informações, o modelo tem o contexto necessário para produzir um retorno individualizado e coerente com o desempenho que o estudante realmente alcançou.

3.2.6 Testes automatizados e validação funcional

Os testes de *software* integram o processo de verificação e validação de *software* (V&V) e são realizados durante o ciclo de desenvolvimento, com o propósito de assegurar que o sistema atenda às especificações definidas e corresponda às expectativas de seus usuários (Sommerville, 2018). Essas práticas permitem identificar defeitos antes da disponibilização do sistema para o usuário final, evitando assim um maior custo associado à sua correção.

Na literatura da Engenharia de Software, o teste de desenvolvimento é executado pela própria equipe responsável pelo sistema e organizado em três estágios, conforme a granularidade abordada (Sommerville, 2018). No teste de unidade, verificam-se componentes de programa ou classes isoladamente; o teste de componentes concentra-se em verificar as interfaces que promovem acesso às suas funções; e o teste de sistema integra os componentes e examina o comportamento da aplicação como um todo, concentrando-se nas interações entre eles. Os testes de ponta a ponta (E2E) inserem-se nesse último estágio, na medida em que simulam fluxos completos de uso da aplicação de modo a validar o sistema sob uma perspectiva próxima à experiência real do usuário.

Para este trabalho, adota-se o teste de sistema como nível de validação, materializado pela execução de testes automatizados de ponta a ponta por meio da ferramenta Playwright. Essa ferramenta possibilita a validação do funcionamento da aplicação em diferentes navegadores e cenários de uso, contribuindo para a identificação precoce de defeitos e para a manutenção da confiabilidade do sistema ao longo de seu ciclo de desenvolvimento. Por envolverem a interface do usuário e suas dependências externas, foram utilizados *mocks*, que substituem, de forma simulada, partes do sistema com as quais o usuário interage, como serviços externos ou APIs. Essa prática foi adotada para tornar os cenários de teste mais determinísticos e reduzir falhas intermitentes não relacionadas ao código sob avaliação.

Para garantir a repetibilidade consistente dos cenários e suprimir a volatilidade do ambiente de servidor, as respostas da API são interceptadas e substituídas por dados predefinidos (*mocks*). A autenticação dos professores ocorre de maneira mista: o fluxo de acesso (*login*) com

credenciais válidas é integrado ao *Amazon Cognito* em ambiente real, verificando-se a emissão de *tokens* de sessão, ao passo que os processos de registro, confirmação de e-mail e encerramento de sessão (*logout*) são emulados, haja vista a complexidade temporal dessas operações. No escopo das funcionalidades do usuário, a resposta gerada pela GenIA também é simulada, o que viabiliza a validação tanto do fluxo ideal de execução quanto do comportamento do sistema diante de eventuais falhas no serviço.

A trajetória completa das entidades da plataforma é mapeada em cenários críticos de teste. No fluxo administrativo do docente, as rotinas de automação confirmam a autenticação integrada ao *Amazon Cognito*, as operações de criação, edição e listagem de turmas e tarefas, bem como o tratamento de erros de formulário e de estados vazios (*empty states*). No módulo do discente, a execução automatizada aborda o ingresso na atividade mediante parâmetros de rota, a renderização progressiva dos trechos de leitura, o envio de respostas e o recebimento do *feedback*. Adicionalmente, avalia-se o comportamento da interface frente à indisponibilidade da tarefa ou à falha no serviço de GenIA.

Cada cenário de teste inspeciona as respostas HTTP do *back-end*, assegurando a conformidade dos códigos de estado (*status codes* 200, 201 e erros tratados), o isolamento de escopo por *token* JWT e a resiliência do *front-end* diante do tempo de execução das funções AWS *Lambda*. A execução sistemática dessa suíte de testes fornece as métricas quantitativas de taxas de sucesso por cenário e de tempo de execução total necessárias para atestar a estabilidade operacional e a conformidade funcional do sistema.

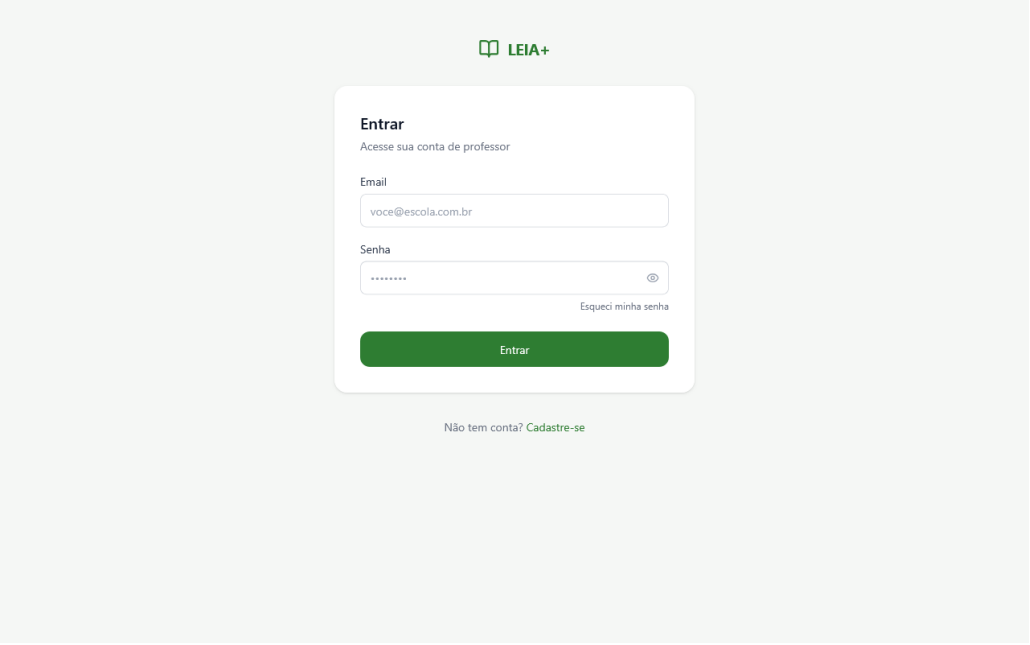
4 RESULTADOS

Este capítulo dedica-se à apresentação e análise dos resultados decorrentes do desenvolvimento do LEIA+. Para tanto, a abordagem foca na descrição da plataforma final e na análise dos dados coletados durante a etapa de testes da solução.

4.1 Aplicação LEIA+

O acesso ao módulo administrativo da plataforma LEIA+ exige a autenticação do usuário por meio de e-mail e senha, conforme ilustrado na Figura 9.

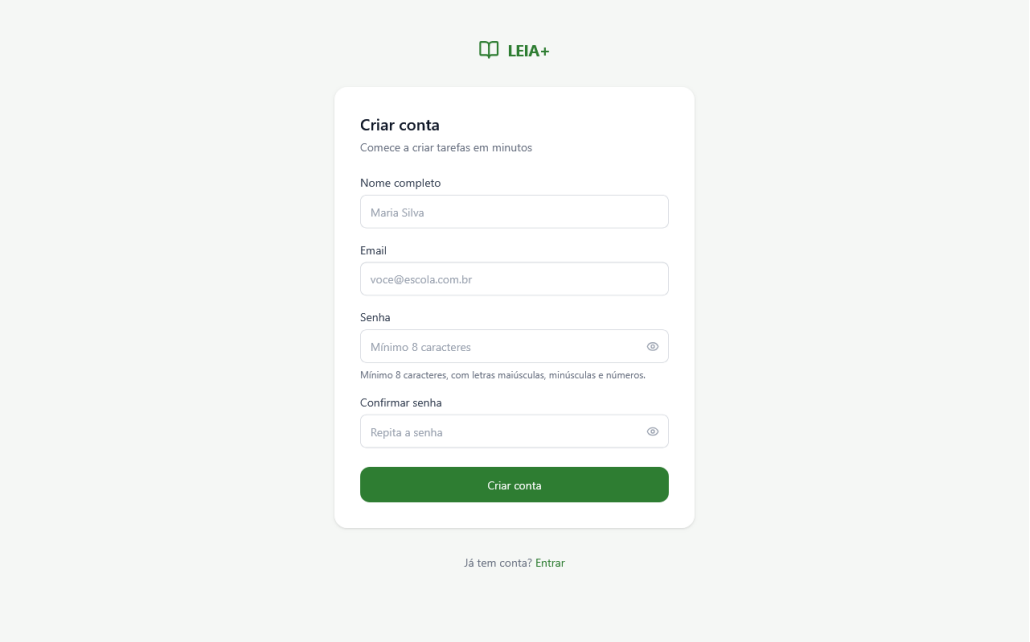
Figura 9 – Tela de autenticação



Fonte: Autoria própria (2026).

Como estabelecido na modelagem apresentada na Figura 1, a implementação permite que o docente, caso não possua cadastro na plataforma, crie uma nova conta inserindo seu nome, endereço de e-mail e senha. A Figura 10 demonstra a respectiva página de cadastro.

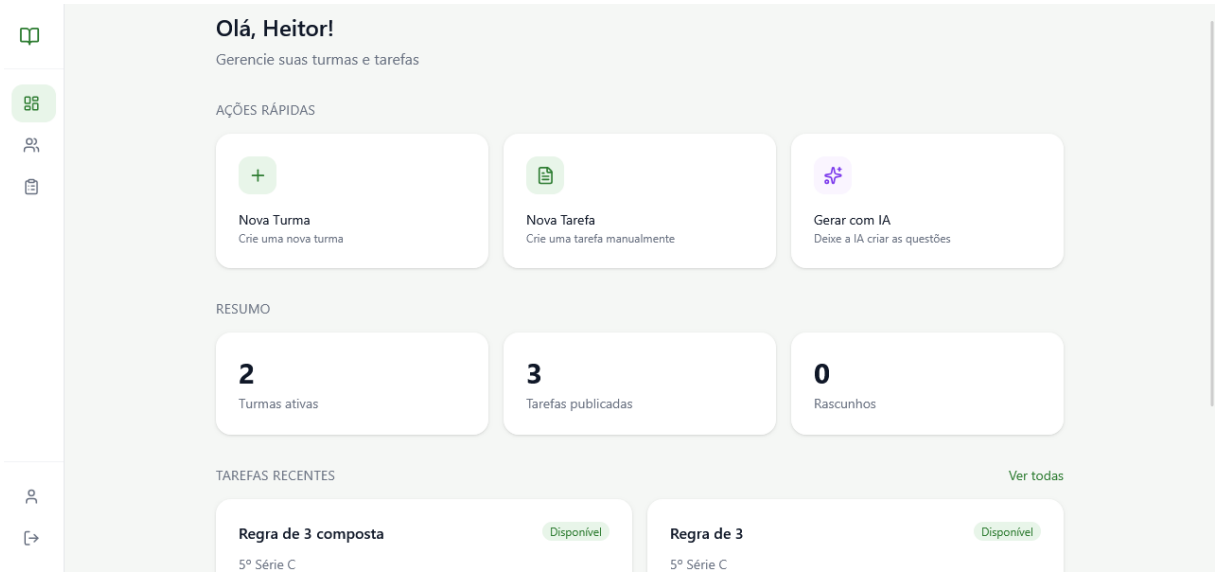
Figura 10 – Tela de cadastro



Fonte: Autoria própria (2026).

Após a autenticação bem-sucedida, o usuário é redirecionado automaticamente para a tela inicial (*dashboard*) da plataforma. Essa interface apresenta uma mensagem de boas-vindas personalizada e os botões de ações rápidas: “Nova Turma”, “Nova Tarefa” e “Gerar com IA”. Ademais, a página exibe um panorama quantitativo com o total de turmas cadastradas, tarefas publicadas e rascunhos salvos, seguido por um *grid* contendo as tarefas criadas recentemente. A estrutura desse ambiente virtual é ilustrada na Figura 11.

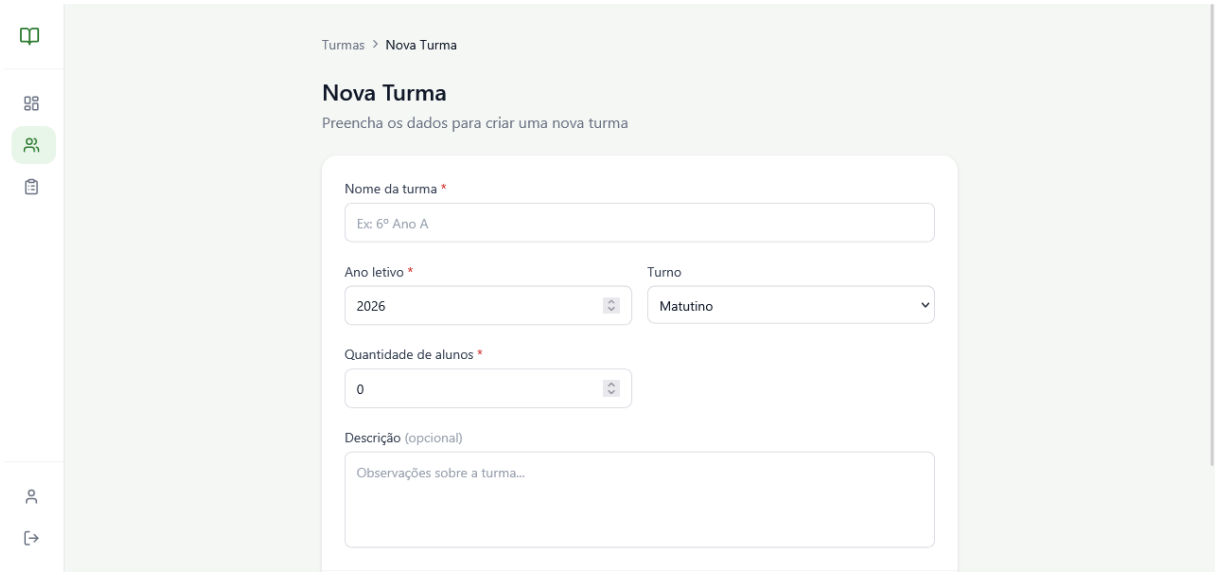
Figura 11 – Tela inicial



Fonte: Autoria própria (2026).

Para cadastrar uma nova turma, o docente pode utilizar a barra lateral de navegação ou acionar o botão de ação rápida "Nova Turma". No formulário de criação, o professor deve informar o nome da turma, o ano letivo, o turno, a quantidade de alunos e, opcionalmente, uma descrição, conforme ilustrado na Figura 12.

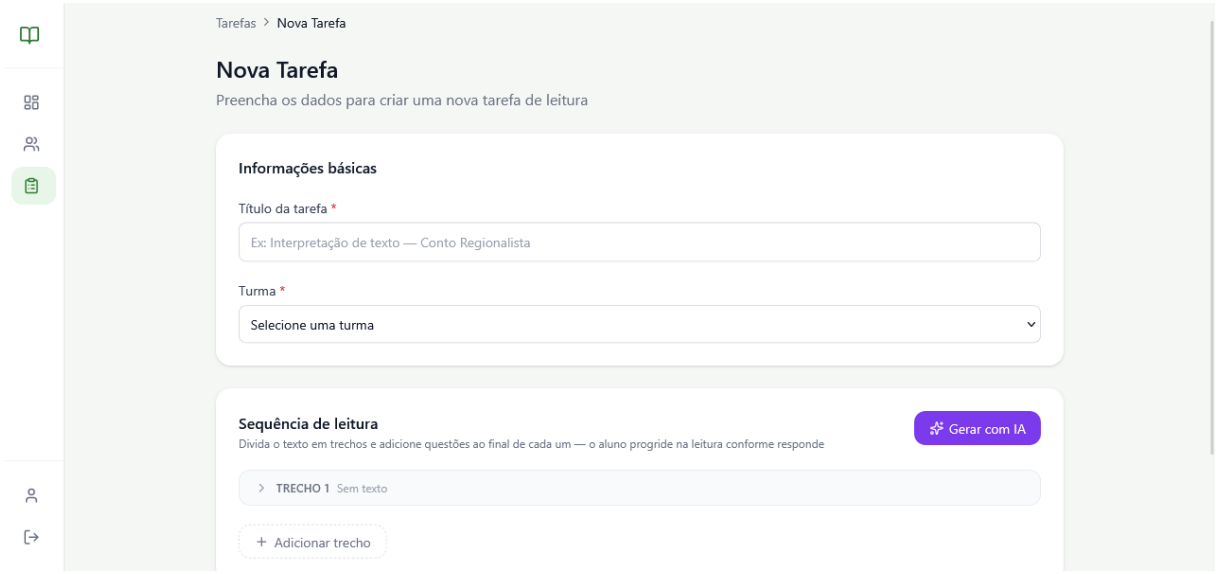
Figura 12 – Tela de criar turma



Fonte: Autoria própria (2026).

Com pelo menos uma turma existente, o docente pode criar uma nova tarefa ao acessar o formulário de criação, disponível tanto pela barra lateral quanto pelo botão de ação rápida na tela inicial. O formulário é organizado em duas seções principais. A primeira, denominada “Informações básicas”, solicita o título da tarefa e a turma destinatária, ambos campos obrigatórios. A segunda seção, "Sequência de leitura", estrutura o conteúdo da tarefa em trechos sequenciais: para cada trecho, o professor insere um fragmento do texto e pode adicionar questões a serem respondidas ao final daquele segmento, de modo que o aluno avança na leitura conforme responde às perguntas. Adicionalmente, essa seção disponibiliza o botão "Gerar com IA", que permite ao docente utilizar GenIA para auxiliar na criação automática dos trechos e das questões. Cabe ressaltar que os trechos e as questões produzidos pela GenIA são inseridos diretamente nos campos editáveis do formulário, de modo que o docente pode revisá-los, ajustá-los ou removê-los antes de salvar a tarefa; nenhuma questão é disponibilizada ao aluno sem essa validação prévia, preservando-se a responsabilidade do educador sobre o conteúdo apresentado à turma. A Figura 13 ilustra essa interface.

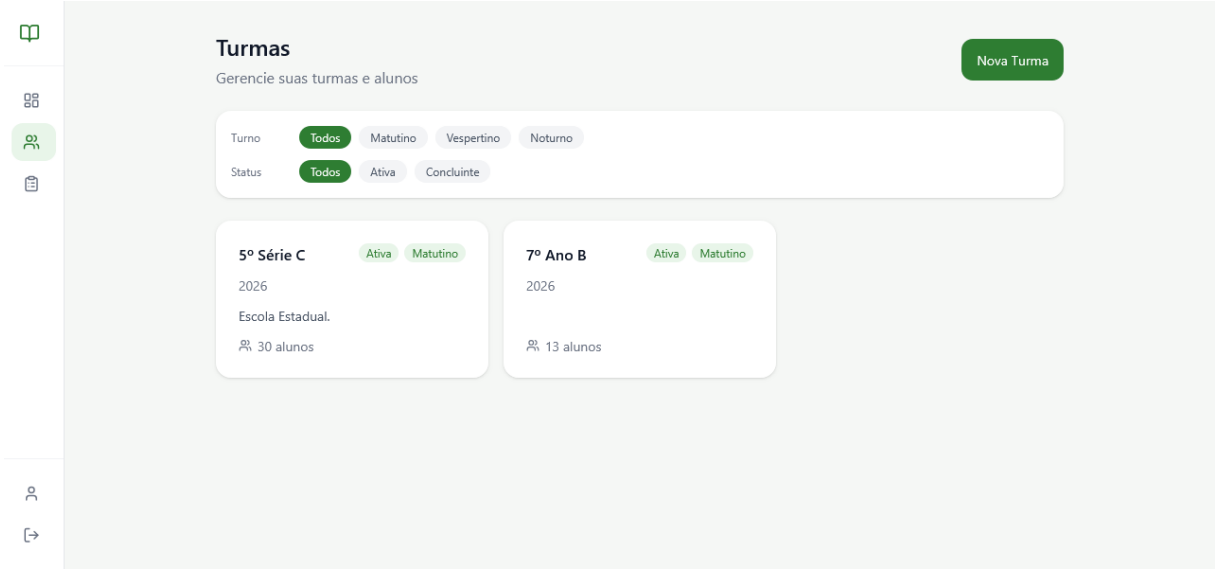
Figura 13 – Tela de criar tarefa



Fonte: Autoria própria (2026).

A listagem de turmas cadastradas é acessível pelo ícone correspondente na barra lateral de navegação. A tela exibe os *cards* de cada turma com informações resumidas, como nome, ano letivo, escola, turno e quantidade de alunos matriculados. Para facilitar a navegação, a interface oferece filtros por turno (Matutino, Vespertino e Noturno) e por status (Ativa e Concluinte), além do botão “Nova Turma” para acesso rápido ao formulário de cadastro. Essa tela é ilustrada na Figura 14.

Figura 14 – Tela de turmas

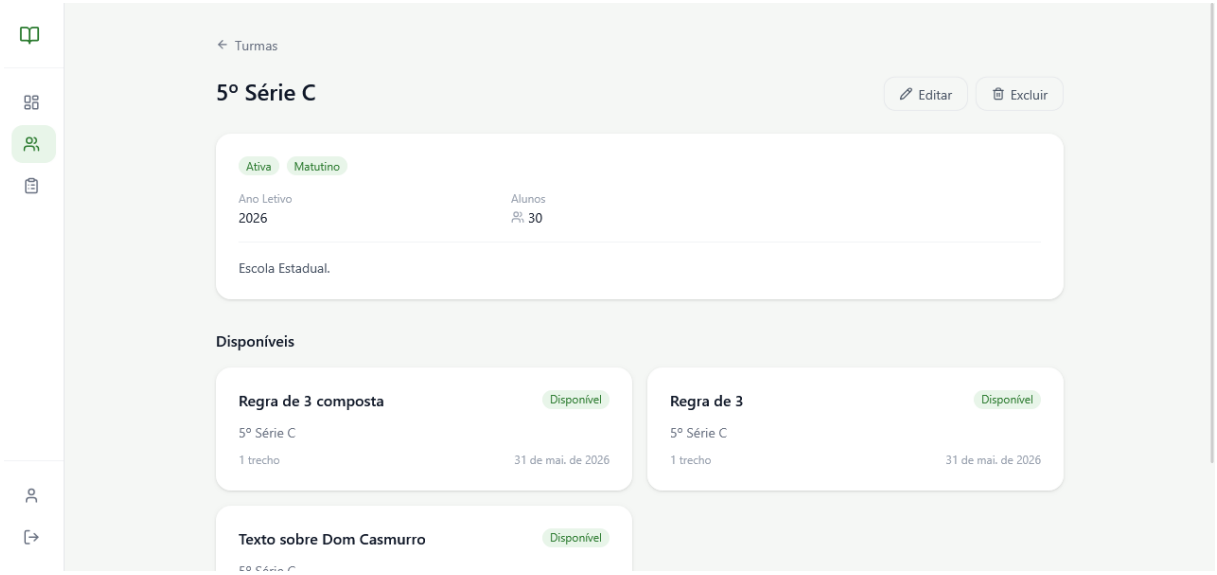


Fonte: Autoria própria (2026).

A página de detalhas da turma mostra o *status*, turno, ano letivo, quantidade de alunos e

escola e as tarefas vinculadas àquela turma, agrupadas por *status*. A tela também disponibiliza os botões “Editar” e “Excluir” para o gerenciamento do registro. Essa visualização é apresentada na Figura 15.

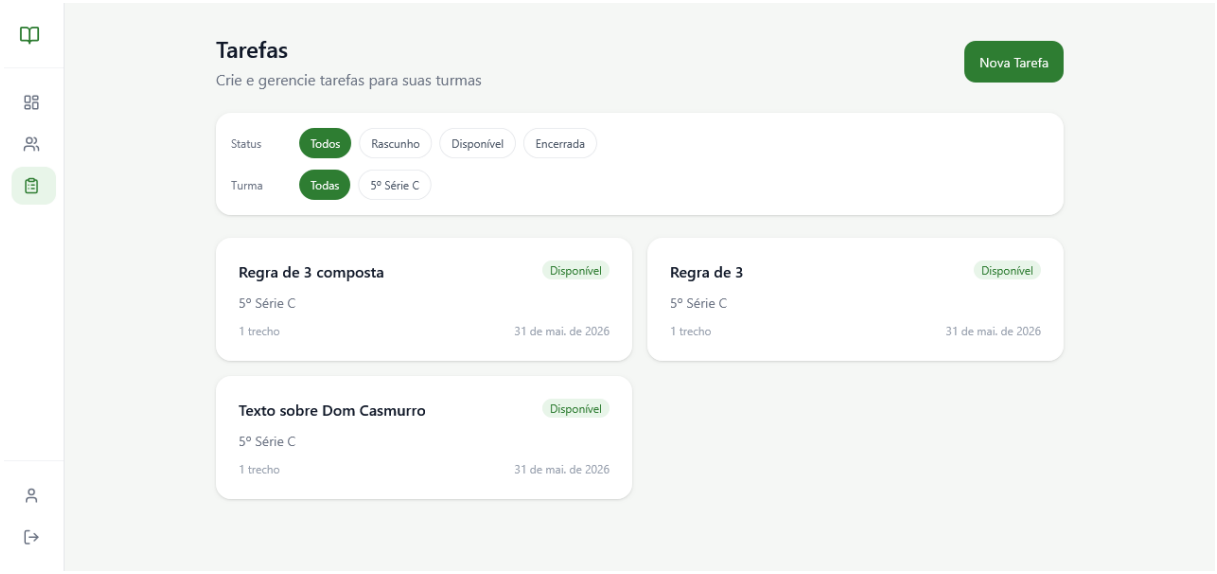
Figura 15 – Tela de detalhes da turma



Fonte: Autoria própria (2026).

De forma análoga, a tela de tarefas, acessível pela barra lateral, apresenta um *grid* com os *cards* de todas as tarefas criadas pelo docente. Cada *card* exibe o título da tarefa, a turma destinatária, o número de trechos e a data de criação, além de um *badge* indicando o *status* atual (Rascunho, Disponível ou Encerrada). A tela também disponibiliza filtros combinados por *status* e por turma, permitindo ao professor localizar rapidamente uma tarefa específica, bem como o botão “Nova Tarefa” para iniciar a criação de uma nova atividade. A Figura 16 ilustra essa página.

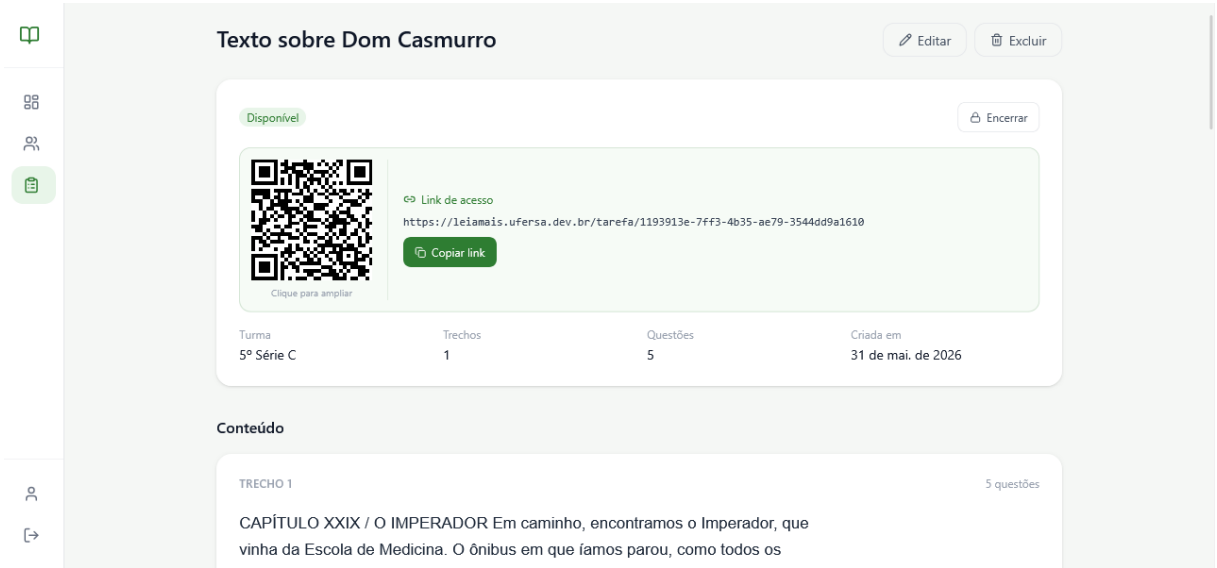
Figura 16 – Tela de tarefas



Fonte: Autoria própria (2026).

Ao acessar uma tarefa específica, o docente visualiza sua página de detalhes, ilustrada na Figura 17. Nessa tela são exibidos o *status* atual da tarefa, um QR *code* e o *link* de acesso direto para compartilhamento com os alunos, além de um resumo quantitativo contendo a turma destinatária, o número de trechos, o total de questões e a data de criação. A seção "Conteúdo" apresenta cada trecho do texto com a respectiva quantidade de questões associadas. O docente pode ainda editar ou excluir a tarefa, bem como encerrá-la quando julgar apropriado.

Figura 17 – Tela de detalhes da tarefa

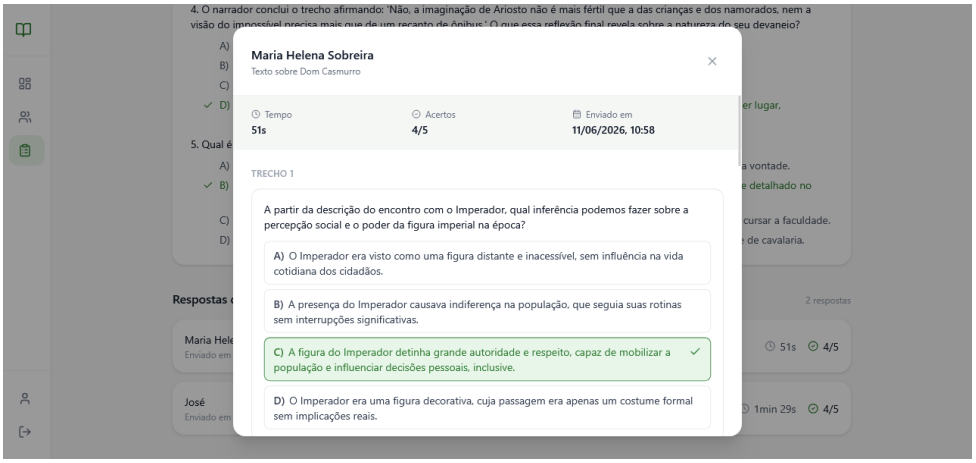


Fonte: Autoria própria (2026).

Na seção de respostas da mesma tela, o professor pode visualizar as submissões de cada

aluno. Ao clicar em uma resposta, um painel modal é aberto exibindo o desempenho individual do estudante: tempo de conclusão, número de acertos e data de envio. O painel também apresenta, para cada questão, as alternativas disponíveis e a opção marcada pelo aluno, com destaque visual para a resposta correta, conforme ilustrado na Figura 18.

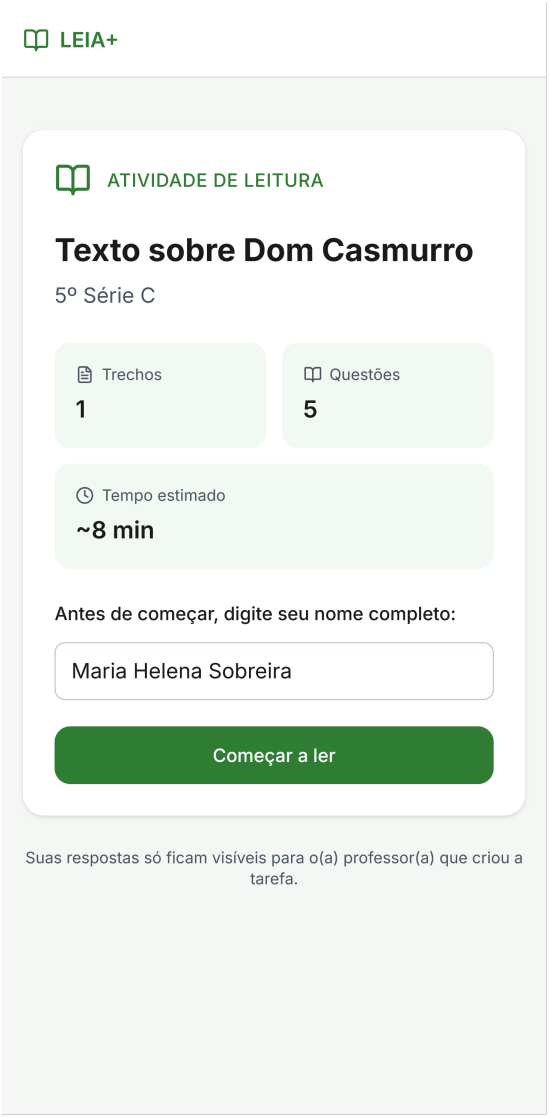
Figura 18 – Modal de resposta do aluno



Fonte: Autoria própria (2026).

Do ponto de vista do estudante, a tarefa é acessada por meio do *link* ou QR *code* disponibilizado pelo professor. A tela de entrada, ilustrada na Figura 19, apresenta as informações gerais da atividade, como título, turma, número de trechos, quantidade de questões e tempo estimado de conclusão, e solicita que o aluno informe seu nome completo antes de iniciar. A interface também esclarece que as respostas serão visíveis apenas para o docente responsável pela tarefa.

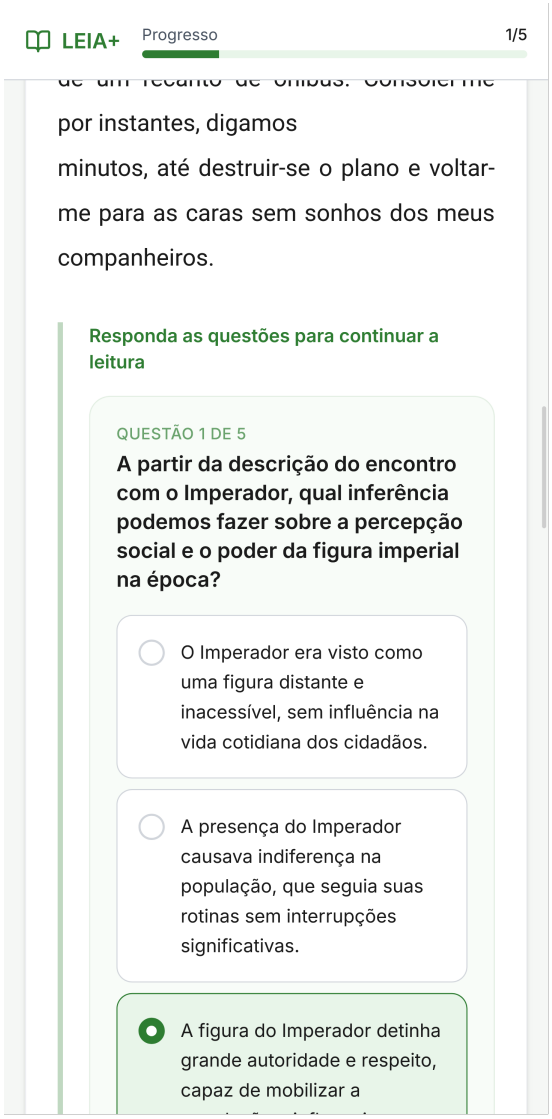
Figura 19 – Tela de entrada da atividade



Fonte: Autoria própria (2026).

Após iniciar a atividade, o aluno é direcionado ao fluxo de leitura sequencial. A interface exibe o texto do trecho atual em destaque e, ao final dele, apresenta as questões de múltipla escolha que devem ser respondidas antes de avançar para o próximo segmento. Uma barra de progresso no topo da tela indica a evolução do aluno ao longo das questões. Essa dinâmica é apresentada na Figura 20.

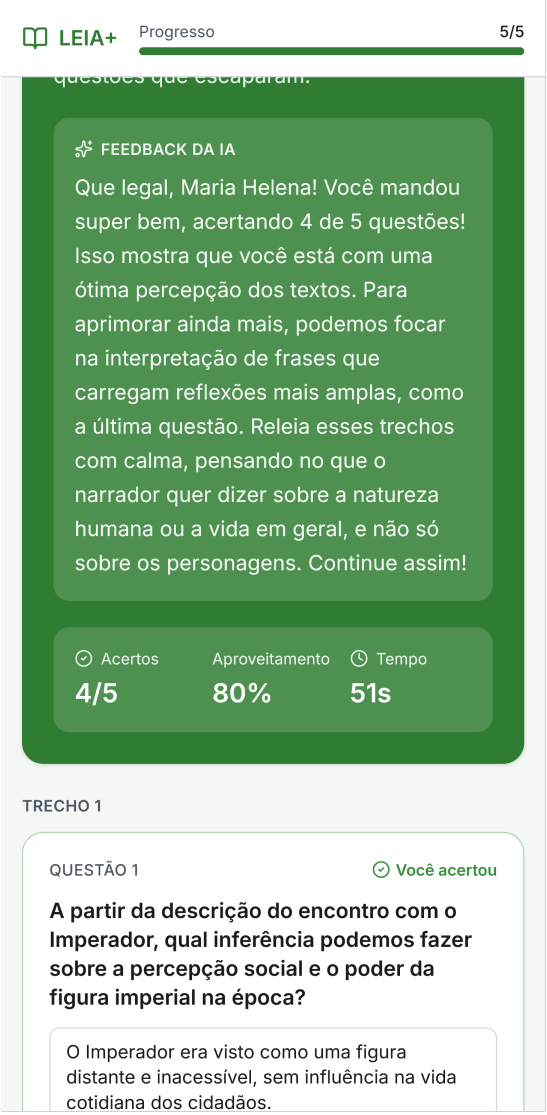
Figura 20 – Tela de leitura com questão



Fonte: Autoria própria (2026).

Ao concluir todas as questões, o aluno é direcionado à tela de resultados, ilustrada na Figura 21. Essa tela exibe um *feedback* personalizado gerado pela GenIA, que comenta o desempenho do estudante e oferece orientações específicas para os pontos de melhoria identificados. Abaixo do *feedback*, são apresentados os indicadores de desempenho (total de acertos, aproveitamento percentual e tempo de conclusão) seguidos pela revisão detalhada de cada questão, com a indicação de acerto ou erro para cada alternativa respondida.

Figura 21 – Tela de resultados com *feedback* da GenIA



Fonte: Autoria própria (2026).

4.2 GenIA: Google Gemini API

A plataforma LEIA+ emprega GenIA em duas frentes complementares, ambas apoiadas na API do Google Gemini, por meio do modelo `gemini-2.5-flash`: a geração assistida de questões, voltada ao docente durante a criação de tarefas, e a produção de *feedback* personalizado, dirigido ao discente ao término da atividade. Cada frente é atendida por uma função *Lambda* dedicada, escrita em Python e exposta sob o recurso `/ia` da API REST — respectivamente, `generate-questions` (`POST /ia/questoes`) e `generate-feedback` (`POST /ia/feedback`).

A função `generate-questions` recebe um ou mais trechos de texto, acompanhados

do número de questões desejado por trecho, do nível de dificuldade (fácil, médio ou difícil) e dos focos de análise pretendidos (compreensão, vocabulário, gramática ou distinção entre o explícito e o implícito). A partir desses parâmetros, retorna, para cada trecho, um conjunto de questões de múltipla escolha contendo enunciado, quatro alternativas, a letra correta e uma breve explicação. Já a função `generate-feedback` é acionada após a submissão das respostas pelo aluno e recebe o desempenho obtido (total de acertos, aproveitamento percentual e o detalhamento de cada questão respondida), devolvendo uma mensagem geral de incentivo e um comentário individualizado para cada questão.

Ambas as funções compartilham a mesma arquitetura interna, organizada em quatro etapas: a validação do *payload* recebido, a construção do *prompt*, a chamada ao modelo e a serialização da resposta. A construção do *prompt* divide-se em duas partes: uma instrução de sistema (*system instruction*), que define o papel assumido pelo modelo e as regras obrigatórias de geração, e um *prompt* de usuário, que injeta dinamicamente os dados da requisição. Adicionalmente, para assegurar respostas estruturadas e diretamente consumíveis pela aplicação, as chamadas configuram a saída em formato JSON (`response_mime_type`) vinculada a um esquema explícito (`response_schema`), o que dispensa a análise de texto livre e reduz a ocorrência de respostas malformadas.

4.3 Testes *End-to-End*

Para simular interações reais através de um navegador automatizado, implementaram-se testes E2E com a ferramenta Playwright sob o motor Chromium. Situada no nível de teste de sistema, essa abordagem validou o comportamento da plataforma por inteiro, cobrindo desde a interface gráfica até as chamadas de rede sob a perspectiva do usuário final. Os cenários de teste dividiram-se conforme os módulos da solução: o administrativo, voltado ao docente, e o do aluno, focado nas atividades de leitura. O código-fonte de ambos os cenários de testes realizados é apresentado no Apêndice A.

4.3.1 Módulo do docente

Os testes do módulo do docente contemplam desde a autenticação e o gerenciamento de conta até a criação e a edição de turmas e tarefas. Para isolar a aplicação de dependências externas durante a execução, as respostas da *API REST* foram interceptadas e substituídas por

dados controlados (*mocks*), técnica que assegura a reprodutibilidade dos cenários e elimina a variabilidade associada ao ambiente de *back-end*. A autenticação, por sua vez, foi exercitada de forma híbrida: o *login* com credenciais válidas integra-se ao Amazon Cognito em ambiente real, validando o fluxo completo de emissão de *tokens* de sessão, ao passo que os fluxos de cadastro, confirmação de *e-mail* e encerramento de sessão foram simulados, dada a natureza assíncrona desses processos.

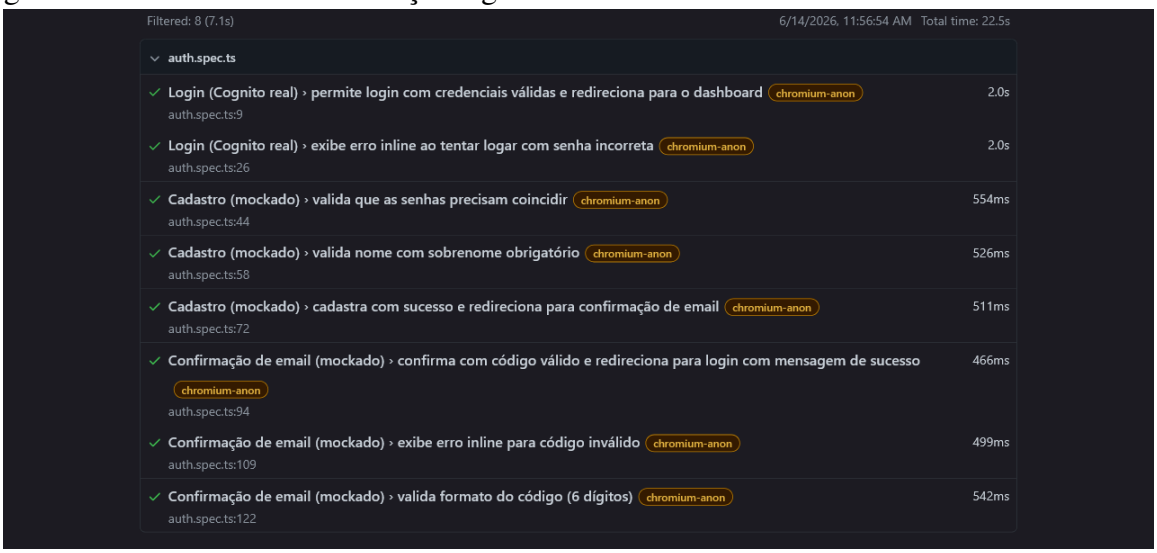
A suíte de autenticação e gerenciamento de conta reúne os cenários que verificam a entrada do docente na plataforma, o cadastro de novos perfis e o controle de acesso. Os casos contemplam tanto os fluxos de sucesso quanto as validações de erro (como a divergência entre senhas, a ausência de sobrenome e a inserção de códigos de confirmação inválidos), conforme sintetizado na Figura 22.

Os testes de *login* foram realizados acessando a página `/login`, preenchendo o campo de e-mail com `E2E_EMAIL` e o de senha com `E2E_PASSWORD`, e acionando o botão “Entrar”. O retorno para a página inicial (`/`) com a exibição da mensagem de boas-vindas validou o sucesso do fluxo. Para testar o cenário de senha incorreta, seguiram-se os mesmos passos, utilizando-se `senha-errada` no campo correspondente, o que permitiu confirmar a exibição da mensagem de erro esperada.

Os fluxos de registro e confirmação foram validados por meio da simulação de requisições ao Amazon Cognito, com respostas predefinidas através do método `page.route()`. Para o cenário de senhas divergentes, digitaram-se as sequências “Abcdef12” e “Outracoisa1A” nos campos da rota `/cadastro`, verificando-se a apresentação da mensagem “As senhas não coincidem”. Ao omitir o sobrenome, preencheu-se o campo de nome apenas com “Maria”, constatando-se a exibição do aviso “Informe nome e sobrenome”.

Para um registro bem-sucedido, simulou-se uma resposta de sucesso do Cognito, confirmando-se o redirecionamento para a rota `/confirmar?email=...`. Na etapa de confirmação de e-mail, também simulada, a inserção do código “123456” redirecionou o fluxo para `/login` com sucesso. O envio do código “000000” resultou em um erro de código inválido e, ao digitar apenas dois dígitos, o usuário permaneceu na mesma página, confirmando a validação do formato exigido.

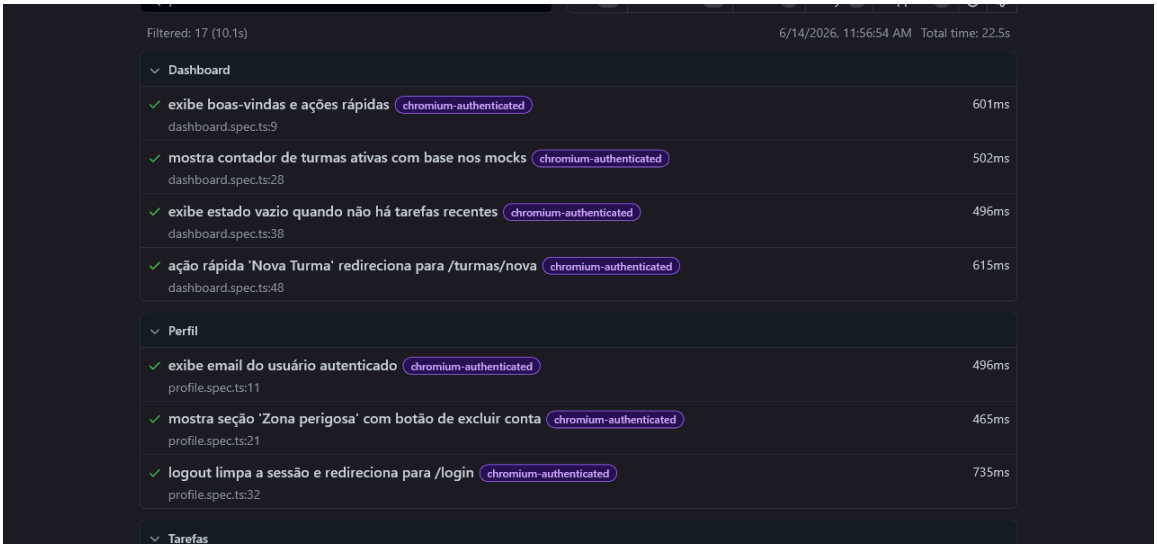
Figura 22 – Cenários de autenticação e gerenciamento de conta



Fonte: Autoria própria (2026).

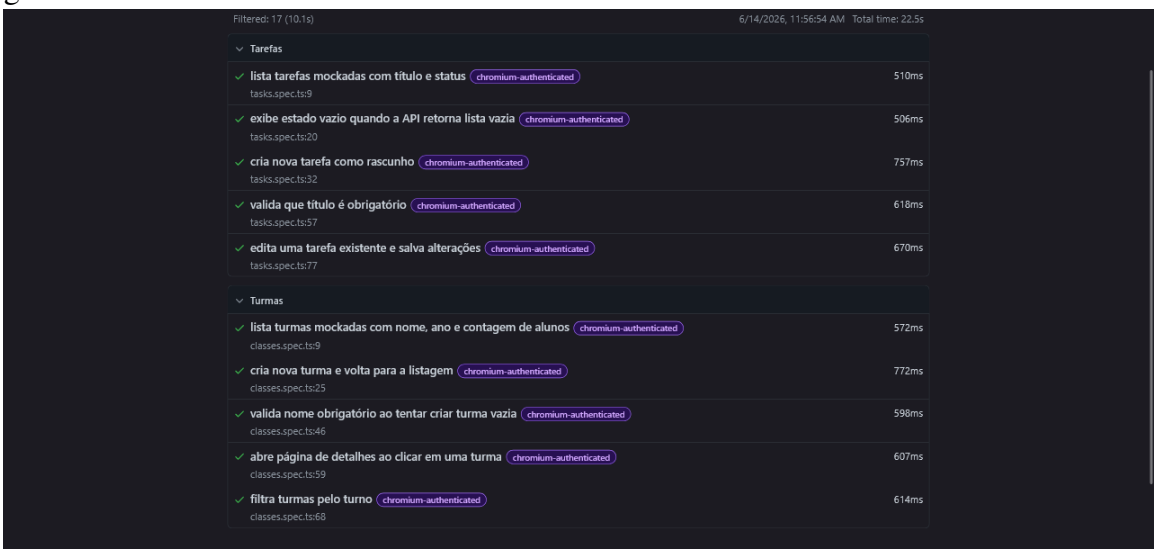
A segunda suíte concentra-se nas funcionalidades centrais do painel do docente, abrangendo a tela inicial, a gestão de turmas, a gestão de tarefas e o perfil do usuário. Os cenários verificam a correta renderização das informações, a navegação entre as telas, a aplicação de filtros, o tratamento de estados vazios e as validações dos formulários de cadastro, além das operações de criação e edição de registros. As Figura 23 e Figura 24 apresentam a relação completa desses casos.

Figura 23 – Cenários de dashboard e perfil



Fonte: Autoria própria (2026).

Figura 24 – Cenários de tarefas e turmas



Fonte: Autoria própria (2026).

Antes de cada teste, o auxiliar `mockAdminApi` foi ativado para interceptar as chamadas à API REST e substituí-las por respostas simuladas, contendo turmas e tarefas fictícias definidas diretamente no código dos testes. No *dashboard*, navegou-se à rota raiz (`/`) e verificou-se o cabeçalho de saudação personalizada, os três botões de ação rápida (“Nova Turma”, “Nova Tarefa” e “Gerar com IA”) e o contador de turmas ativas, calculado como a quantidade de registros do *mock* com o campo `active` verdadeiro; para o estado vazio, sobrescreveu-se o *mock* de tarefas com lista vazia e confirmou-se a mensagem “Nenhuma tarefa ainda”.

Para os cenários de turmas, percorreu-se o fluxo completo: listagem em `/turmas` com verificação do nome e da contagem de alunos do primeiro item do *mock*; criação com preenchimento de nome “Turma E2E Teste”, ano “2026”, turno “*morning*” e quantidade de alunos “25”, seguida de confirmação do retorno à listagem em `/turmas`; validação de nome em branco inserindo-se uma cadeia de espaços e verificando a mensagem de erro correspondente; navegação à página de detalhes pelo clique no *card* da primeira turma; e filtragem por turno “Matutino” com conferência da contagem de *cards* resultante comparada ao total de turmas do *mock* com `shift === ‘‘morning’’`.

Nos testes envolvendo as tarefas, realizou-se a listagem com verificação do título do primeiro item do *mock*; a criação de rascunho com título “Tarefa E2E — Interpretação” e trecho de texto livre; a validação de título obrigatório pela tentativa de submissão sem preenchimento do campo; e a edição de tarefa existente navegando-se a `/tarefas/{id}/editar`, alterando o título com a sufixo “(editado)” e confirmando o redirecionamento à página de detalhes.

Por último, no perfil navegou-se a rota `/perfil` e verificou-se a exibição do e-mail da conta autenticada (proveniente de `E2E_EMAIL`) e a presença da seção "Zona perigosa" com o botão de exclusão de conta; o *logout* foi exercitado com a chamada ao Amazon Cognito interceptada, acionando o botão "Sair" no *dashboard* e confirmando o redirecionamento para `/login` e a incapacidade de retornar à área autenticada sem nova autenticação.

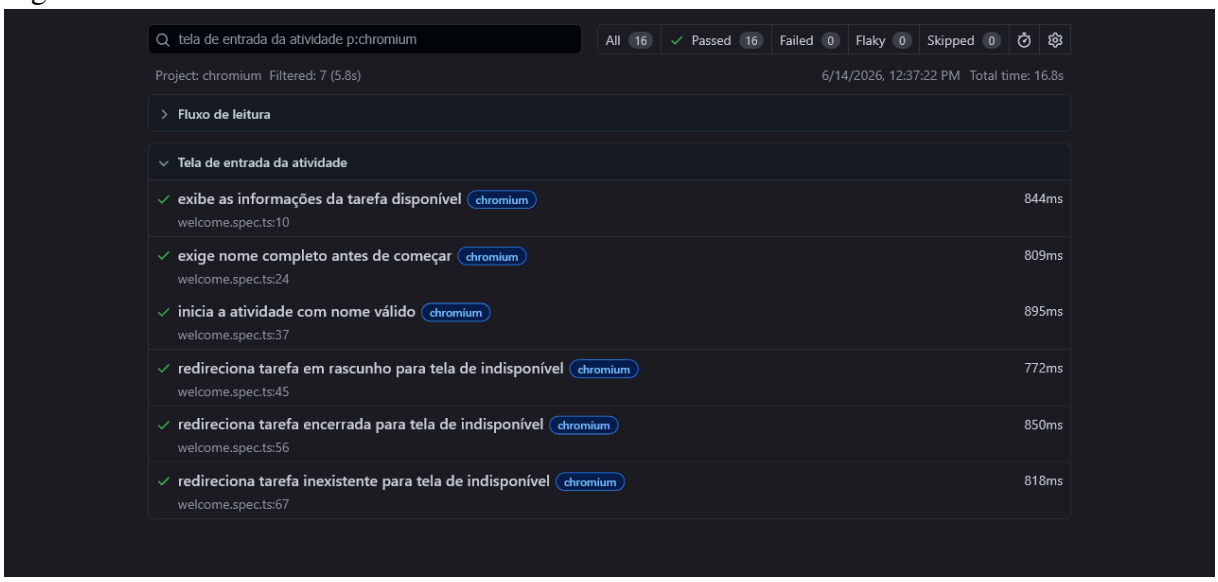
4.3.2 Módulo do discente

Os testes do módulo do discente foram orientados desde o acesso à atividade por meio do *link* compartilhado pelo docente até o recebimento do *feedback* ao final da leitura. À semelhança do módulo do docente, as respostas da API REST foram interceptadas e substituídas por dados controlados (*mocks*), garantindo a reprodutibilidade dos cenários independentemente do *back-end*. O *feedback* gerado pela GenIA, originalmente provido pela API do Google Gemini, também foi simulado, o que permitiu exercitar tanto o fluxo de sucesso quanto o comportamento de contingência diante de falhas no serviço de GenIA.

As tarefas simuladas foram organizadas com identificadores fixos: `t1` (disponível, dois trechos, três questões), `t3` (encerrada), `t5` (rascunho) e `t7` (disponível, um trecho, uma questão). Para compor os cenários de leitura, dois auxiliares foram utilizados: `startTask` acessa a tarefa pelo identificador, preenche o nome do aluno com "Maria de Souza" e inicia a atividade; `answer` seleciona a alternativa indicada em uma determinada questão. Para descrever de forma compacta as interações de resposta ao longo desta seção, adota-se a notação `tX-qY=Z`, em que `tX` identifica a tarefa, `qY` a questão e `Z` a alternativa selecionada; assim, `t1-q1=B` indica a seleção da alternativa B para a primeira questão da tarefa `t1`.

A primeira suíte, mostrada na Figura 25, concentra-se na tela de entrada da atividade responsável por apresentar as informações gerais da tarefa e por identificar o aluno antes do início da leitura. Os cenários verificam a exibição correta dos metadados da atividade, a validação do nome completo e o tratamento de tarefas indisponíveis (seja por estarem em rascunho, encerradas ou inexistentes).

Figura 25 – Cenários de tela de entrada da atividade



Fonte: Autoria própria (2026).

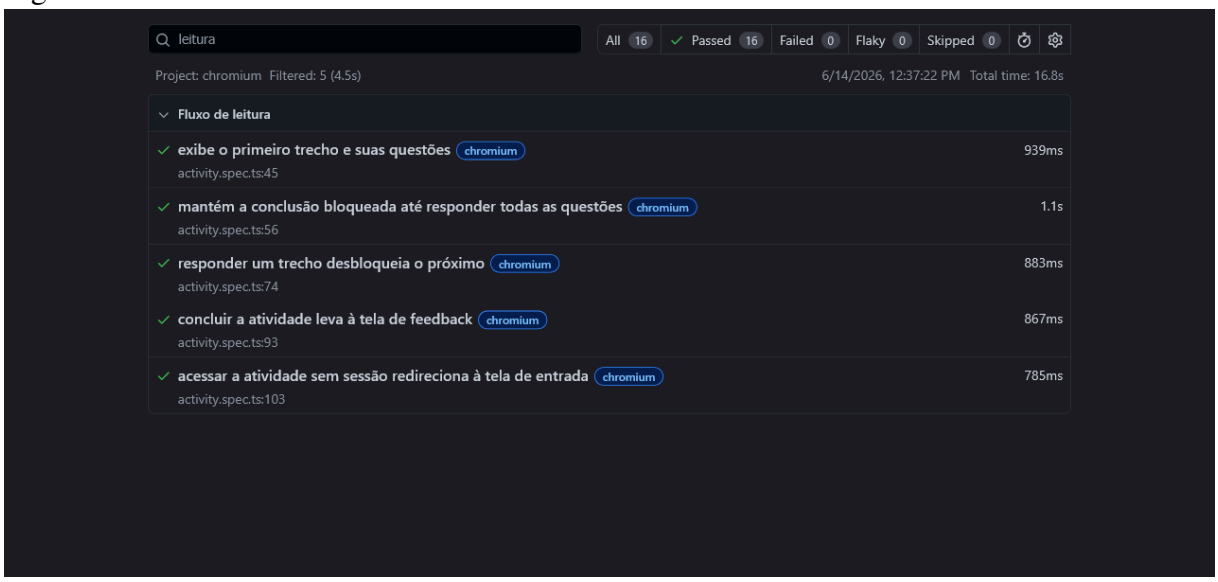
Para atingir cada um dos cenários, acessou-se diretamente às rotas das tarefas com códigos já estabelecidos. A checagem dos detalhes da tarefa em questão foi feita ao navegar para `/tarefa/t1`, onde se confirmou a existência de título, turma, quantidade de trechos e perguntas e o tempo previsto.

A validação do nome foi efetuada ao digitar a sequência ‘‘`a `’’, que, embora atenda ao requisito mínimo de dois caracteres do HTML, falha na regra da aplicação após a exclusão de espaços. Ao clicar em “Começar a Ler”, a mensagem de erro esperada foi exibida; no entanto, com o nome “Maria de Souza”, o direcionamento para `/tarefa/t1/atividade` validou o fluxo correto.

Em relação às tarefas não disponíveis, acessaram-se as rotas `/tarefa/t5` para rascunho, `/tarefa/t3` para concluída e `/tarefa/inexistente`, confirmando-se, em cada situação, a rota de redirecionamento com o parâmetro `reason` apropriado (`rascunho`, `encerrada` e `not-found`), bem como o cabeçalho exibido na tela de indisponibilidade.

A segunda suíte cobre o fluxo de leitura propriamente dito, no qual o conteúdo é apresentado em trechos sequenciais e o avanço depende da resposta às questões de cada segmento. Os cenários validam a renderização progressiva dos trechos, o bloqueio da conclusão enquanto houver questões pendentes, o desbloqueio do trecho seguinte após a resposta e o controle de acesso que impede o ingresso na atividade sem identificação prévia. A Figura 26 lista todos os cenários para o fluxo de leitura.

Figura 26 – Cenários de fluxo de leitura



Fonte: Autoria própria (2026).

O ponto de partida para a validação de todos os cenários acima foi o auxiliar `startTask` para estabelecer a sessão do aluno com o nome "Maria de Souza". O bloqueio de conclusão foi verificado em três etapas sobre a tarefa `t1`: confirmação de que o botão "Concluir atividade" estava desabilitado ao início; resposta às questões `t1-q1=B` e `t1-q2=C` com nova verificação de que o botão permanecia desabilitado; e resposta a `t1-q3=C` com verificação de que o botão foi habilitado.

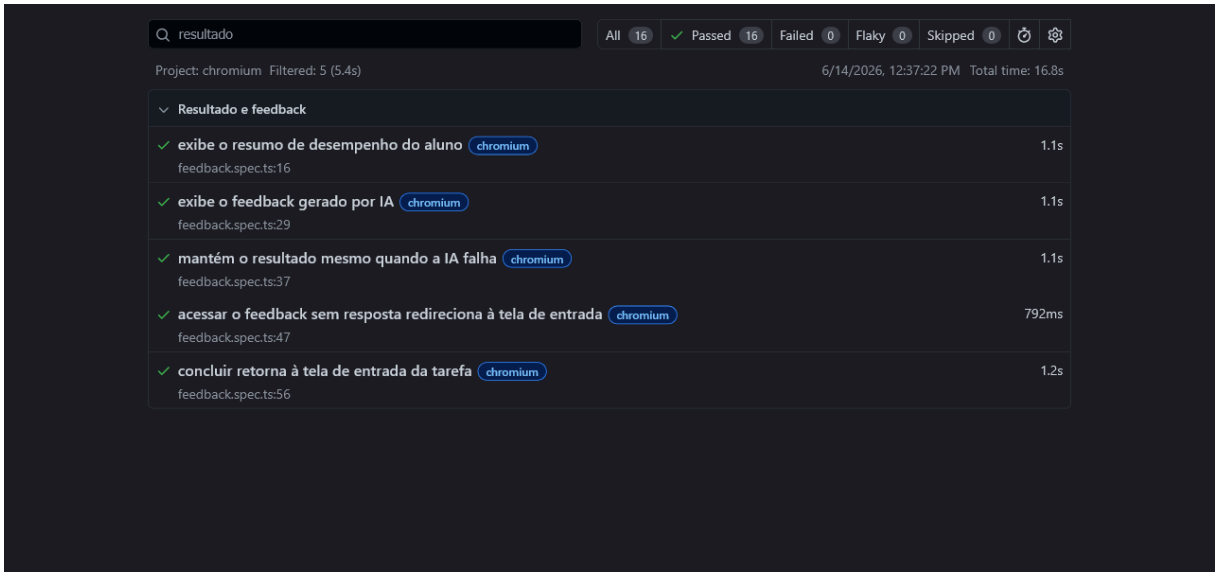
O desbloqueio sequencial foi exercitado com uma tarefa de dois trechos definida localmente no arquivo de especificação (`multiSegmentTask`), com um segmento e uma questão cada; o segundo trecho permanecia oculto até que a questão do primeiro fosse respondida (`seg1-q1=A`) e o botão "Continuar leitura" fosse acionado, momento em que o segundo segmento tornou-se visível.

A conclusão com redirecionamento foi exercitada na tarefa `t7` (um trecho, uma questão), respondendo `t7-q1=B` e acionando "Concluir atividade". O controle de sessão foi verificado navegando diretamente para `/tarefa/t1/atividade` sem passar pela tela de entrada, o que resultou no redirecionamento do usuário para `/tarefa/t1`.

A terceira suíte verifica a tela de resultado, que consolida o desempenho do aluno e exibe o *feedback* ao final da atividade. Os cenários contemplam a apresentação do resumo quantitativo (acertos, aproveitamento e tempo de conclusão), a exibição do *feedback* gerado pela GenIA e, sobretudo, a resiliência da interface: ainda que o serviço de GenIA falhe, o resultado determinístico permanece visível, sem prejuízo à experiência do estudante. A Figura 27 ilustra

essa abordagem.

Figura 27 – Cenários de resultado e feedback



Fonte: Autoria própria (2026).

Em cada cenário presente, utilizou-se a função de apoio `completeTask1`, que inicia `startTask` para a tarefa `t1`. Em seguida, são respondidas as três perguntas com as opções corretas (`t1-q1=B`, `t1-q2=C` e `t1-q3=C`). Após essa etapa, aciona-se o botão “Concluir atividade”, direcionando para o endereço `/tarefa/t1/feedback`.

Nesta página, verificou-se o nome do estudante no topo (indicando “Resultado de Maria”), as métricas “3/3” para acertos e “100%” de desempenho, além da área de *feedback* com a resposta simulada da GenIA. No teste de resiliência, configurou-se o *mock* com `failAiFeedback: true`, fazendo com que a resposta da GenIA na rota de *feedback* apresentasse um erro de servidor. Constatou-se que os indicadores fixos continuaram visíveis, enquanto a seção de *feedback* da GenIA foi ocultada.

A segurança de acesso foi testada ao tentar acessar `/tarefa/t1/feedback` sem sessão prévia, o que resultou no redirecionamento do usuário para `/tarefa/t1`. O retorno à tela inicial foi confirmado ao clicar em “Concluir” na tela de resultados e observar a URL final.

4.4 Discussão

Os resultados apresentados nesta seção permitem retomar as lacunas identificadas na análise comparativa da seção 2.4 e avaliar em que medida o LEIA+ as superou no que diz respeito

à interação professor-aluno, à automatização de tarefas e às práticas pedagógicas.

A primeira limitação observada nos trabalhos relacionados é a divisão da experiência entre professores e alunos. O TeacHelp (Pereira; Rosa; Martins, 2021), por exemplo, foca apenas na gestão administrativa do docente, sem apresentar um módulo para o estudante. Em contrapartida, o Revisa+ (Silva *et al.*, 2024) e o Leia Rapidinho (Braga, 2021) atendem apenas ao aluno, sem oferecer ferramentas de controle ou acompanhamento em tempo real para o professor. O LEIA+ resolve esse problema ao integrar ambos os perfis em um único ambiente.

A segunda lacuna encontrada é a dependência de processos manuais para a criação de conteúdo educacional. O Revisa+ reconhece explicitamente que a produção artesanal de histórias em quadrinhos limita o crescimento do conteúdo disponível (Silva *et al.*, 2024), enquanto o TeacHelp opera inteiramente sem nenhuma automação (Pereira; Rosa; Martins, 2021). O LEIA+, por outro lado, aborda esse problema com a função `generate-questions`. Esse recurso utiliza a API do Google Gemini para criar automaticamente perguntas de múltipla escolha a partir de textos inseridos pelo professor, que pode controlar o nível de dificuldade e os focos de análise desejados. Essa automação reduz significativamente o esforço docente na elaboração de atividades, tornando viável a criação de tarefas em larga escala.

Outro ponto a destacar é a ausência de um retorno personalizado ao aluno após a realização das atividades. Nenhum dos trabalhos analisados oferece esse tipo de suporte: o Revisa+ mostra apenas a contagem de acertos ao final do *quiz* (Silva *et al.*, 2024), enquanto o Leia Rapidinho exibe gráficos de desempenho que dizem como o estudante foi, mas não explicam como ele pode melhorar (Braga, 2021). O LEIA+ diferencia-se ao gerar, por meio da função `generate-feedback`, uma mensagem individual para cada aluno, comentando suas respostas e trazendo dicas específicas para os pontos de melhoria. É importante esclarecer que na geração de questões o professor define a dificuldade e os focos de análise. Já no *feedback*, o conteúdo é montado automaticamente a partir do desempenho de cada aluno, com base nos acertos e erros de cada questão, sem que o docente ajuste parâmetros na versão atual da plataforma. Em outras palavras, a personalização acontece para cada estudante, mas o professor ainda não consegue configurá-la, ponto que é retomado como limitação e possibilidade de evolução no Capítulo 5. Além disso, os testes descritos na Figura 27 demonstraram que, mesmo se o serviço de GenIA falhar, o aluno ainda consegue ver suas informações básicas de desempenho, como o número de acertos, a porcentagem de aproveitamento e o tempo de conclusão. Isso garante que a experiência do usuário não seja prejudicada por instabilidades externas da API do Google Gemini.

Por fim, observa-se uma distinção relevante em relação ao público e ao nível de leitura abordado. O Leia Rapidinho foca na fluência de palavras isoladas durante o processo de alfabetização (Braga, 2021), atendendo estudantes em fase inicial de aprendizagem da leitura. O LEIA+, por outro lado, pressupõe alunos já alfabetizados e dirige-se ao desenvolvimento da compreensão leitora, habilidade de ordem superior que envolve interpretação, inferência e análise crítica do texto.

Em síntese, os resultados mostram que o LEIA+ representa um passo à frente em relação às soluções existentes. A plataforma une a experiência de professores e alunos, gera conteúdo pedagógico de forma automatizada com o apoio da GenIA e oferece *feedback* personalizado aos estudantes, abordando limitações que os trabalhos relacionados não conseguiram endereçar. Com essa abordagem, o sistema traz inovação tanto na criação de atividades quanto no acompanhamento do aprendizado, servindo como uma base sólida para futuras validações em cenários reais de ensino.

5 CONCLUSÕES E TRABALHOS FUTUROS

A plataforma LEIA+ consiste em uma solução *web* educacional voltada ao incentivo à leitura em sala de aula. Sua estrutura divide-se em dois escopos principais: o módulo do docente, que viabiliza o gerenciamento de turmas virtuais e a criação de atividades baseadas em fragmentos textuais e questões interpretativas; e o módulo do discente, destinado ao acesso e à resolução das tarefas propostas. Conforme apresentado no Capítulo 4, o professor detém o controle sobre a gestão das turmas, estando apto a elaborar e a disponibilizar os exercícios para os respectivos estudantes.

A partir do percurso descrito, é possível retomar os objetivos específicos estabelecidos na 1 e evidenciar seu cumprimento. O primeiro objetivo, de analisar o uso de tecnologias digitais e modelos de linguagem no contexto educacional, foi contemplado na fundamentação teórica e na análise dos trabalhos relacionados (Capítulo 2), que embasaram as decisões de projeto da plataforma.

O segundo objetivo, voltado à modelagem do sistema, concretizou-se nos diagramas de casos de uso do docente e do discente (Figura 1 e Figura 2) e nos diagramas de sequência dos fluxos centrais (Figura 3 a Figura 8), que representaram as funcionalidades, o fluxo de dados e as interações do ecossistema.

O terceiro objetivo, de projetar a interface organizada em módulos distintos para docentes e discentes, materializou-se na separação entre os módulos administrativo e cliente e nas telas apresentadas no Capítulo 4, que evidenciam o tratamento diferenciado para cada perfil de usuário. O quarto objetivo, de implementar a solução, foi cumprido por meio da arquitetura *serverless* sobre os serviços da AWS e da integração com a API do Google Gemini, responsável pela geração automatizada de questões e pela produção de *feedback*, conforme detalhado na Figura 13 e na seção de GenIA.

Por último, o quinto objetivo, de validar o sistema, foi atendido pela suíte de testes de sistema automatizados de ponta a ponta (E2E) descrita no Capítulo 4, que cobriu os fluxos críticos dos módulos do docente e do discente, atingindo a meta de 100% de aprovação nos cenários avaliados e atestando a conformidade funcional e a estabilidade operacional da plataforma.

A criação de tarefas constitui a funcionalidade central do sistema, permitindo o cadastramento de trechos que fundamentam os problemas interpretativos propostos pelo educador, a quem cabe a mediação pedagógica do conteúdo trabalhado. A integração do modelo de LLM Google Gemini para a geração automatizada de questões demonstrou-se eficaz. Essa automa-

ção visa mitigar a carga de trabalho docente associada à elaboração manual de questionários, auxiliando o profissional na formulação de problemas para cada texto inserido no sistema.

O mecanismo de disponibilização das atividades também se mostrou operacional e cumpre os requisitos estabelecidos. O acesso dos alunos às tarefas restringe-se ao período em que estas permanecem com o *status* aberto, funcionando como um limitador de prazo de entrega. Para a distribuição do conteúdo, foram implementadas duas estratégias: o uso de códigos QR, idealizado para cenários presenciais nos quais os estudantes utilizam dispositivos móveis para evitar a digitação manual da URL no navegador; e o compartilhamento de *hiperlinks*, direcionado ao envio por meio de canais de comunicação digital (como correio eletrônico ou aplicações de mensagens instantâneas).

No que tange aos discentes, os cenários simulados indicaram a viabilidade do fluxo planejado para a execução das tarefas em um ambiente real. Após a identificação nominal, o estudante visualiza a atividade compartilhada pelo professor, procedendo à leitura e à resolução das questões interpretativas. Ao término do processo, o sistema fornece um *feedback* de desempenho, quantificando os acertos e erros obtidos.

Por fim, a interface de tarefas permite ao professor visualizar as respostas dos alunos detalhadamente. Por meio desse recurso, torna-se possível monitorar o tempo despendido por cada estudante para a conclusão da atividade, o índice de acertos e as alternativas selecionadas em cada pergunta, as quais contam com destaque visual. Essa funcionalidade evidencia o potencial da tecnologia digital na educação: ao prover uma visão minuciosa do progresso individual, o sistema auxilia na identificação de dificuldades específicas e subsidia ações pedagógicas direcionadas, ampliando as opções metodológicas do docente em sala de aula.

Apesar das conquistas alcançadas, o sistema ainda apresenta limitações na experiência do usuário. Na tela inicial (*dashboard*), por exemplo, os botões “Nova Tarefa” e “Gerar com IA” direcionam ao mesmo formulário, o que configura uma redundância na navegação. Além disso, para acessar as respostas dos alunos, o professor precisa navegar até a tarefa e rolar a página até o final; em tarefas com trechos extensos, esse fluxo se torna trabalhoso devido à ausência de um atalho dedicado.

Adicionalmente, nota-se uma restrição atrelada ao uso do modelo `gemini-2.5-flash` do Google. Por utilizar o plano gratuito da API, em alguns cenários de geração de questões ocorreram erros por alta demanda do modelo ou pelo atingimento do limite de requisições, comprometendo pontualmente uma das principais funcionalidades da plataforma.

Outra limitação relevante diz respeito ao *feedback* gerado pela GenIA ao término das atividades. Diferentemente das questões, cujo conteúdo é revisado e validado pelo docente antes da publicação, o *feedback* é produzido automaticamente no momento da conclusão da tarefa e exibido diretamente ao aluno, sem etapa de configuração ou supervisão por parte do professor. Embora essa automação assegure um retorno imediato ao estudante, ela reduz o controle docente sobre o conteúdo pedagógico efetivamente entregue, uma vez que a responsabilidade sobre as orientações apresentadas ao discente cabe ao educador.

Ademais, devido a fatores externos ao escopo deste trabalho, não foi possível aplicar a ferramenta em um ambiente real de sala de aula, o que inviabilizou a coleta de *feedbacks* práticos de professores e alunos sobre o LEIA+. Essa validação seria essencial para determinar com maior precisão o comportamento da solução no cotidiano escolar, mapear melhorias necessárias e verificar o alinhamento com os objetivos inicialmente planejados.

Em conformidade com as diretrizes de integridade científica e boas práticas acadêmicas recomendadas pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), os autores declaram que utilizaram ferramentas de inteligência artificial, incluindo Google Gemini 2.5 Flash, exclusivamente para auxiliar na melhoria da linguagem e legibilidade deste manuscrito. O uso dessas ferramentas foi realizado sob supervisão humana, e os autores revisaram e editaram cuidadosamente o conteúdo, assumindo total responsabilidade pelo texto final publicado.

5.1 Trabalhos futuros

Os obstáculos identificados e as possibilidades observadas durante o desenvolvimento apontam para diversas direções de evolução da plataforma. Esta seção descreve as principais melhorias e extensões que podem ser incorporadas em versões futuras do LEIA+.

Um aprimoramento importante previsto refere-se ao controle do docente sobre o *feedback* gerado pela GenIA. Essa autonomia consistiria em permitir que o professor habilite ou desabilite o *feedback* automático em cada tarefa, defina diretrizes pedagógicas, como tom e foco da mensagem, que orientem sua geração, e visualize integralmente as orientações entregues a cada aluno. Tais mecanismos restabeleceriam a responsabilidade do educador sobre o conteúdo apresentado ao discente, sem abrir mão dos benefícios do retorno automatizado e imediato.

Uma das funcionalidades mais relevantes seria a implementação de contas para os discentes. Atualmente, o aluno acessa a tarefa apenas informando seu nome, sem qualquer vínculo permanente no sistema. Com a criação de perfis de estudante, o professor poderia

cadastrar os usuários diretamente em sua turma, garantindo maior controle sobre o acesso às atividades e possibilitando o acompanhamento do histórico individual ao longo do tempo.

Além disso, planeja-se o desenvolvimento de um painel de desempenho analítico, o qual representará um avanço significativo na análise de dados do ecossistema. Com essa ferramenta, os educadores poderão visualizar métricas fundamentais sobre as atividades, tais como a média de acertos, o tempo médio de resolução das tarefas e a taxa de engajamento. Essa funcionalidade proporcionará aos docentes uma visão geral do progresso coletivo, auxiliando-os na identificação de áreas que demandam maior suporte pedagógico. Ademais, o painel analítico otimizará a navegação, centralizando o fluxo de respostas em uma interface unificada e de fácil utilização.

Outra melhoria desejável consiste na migração para um plano pago da API do Google Gemini ou no estabelecimento de limites de requisições (*rate limiting*) na aplicação. Essa mudança mitigará as falhas intermitentes observadas durante o processo de geração automatizada de questões, as quais decorrem das restrições de cota impostas pela modalidade gratuita. Com essa adequação, a estabilidade e a confiabilidade dos recursos de GenIA serão ampliadas, sobretudo em cenários de alta concorrência e acessos simultâneos. Adicionalmente, o suporte a outros modelos de linguagem poderá ser explorado, reduzindo a dependência de um único provedor.

Por fim, a próxima etapa da pesquisa consiste na validação do sistema em ambiente educacional real. A aplicação prática com docentes e discentes permitirá analisar a usabilidade do *software*, identificar limitações não previstas e mensurar o impacto real da ferramenta no engajamento dos alunos com a leitura. Esse procedimento será fundamental para avançar na questão norteadora deste trabalho.

REFERÊNCIAS

BAIDOO-ANU, David; ANSAH, Leticia. Education in the era of generative artificial intelligence (ai): Understanding the potential benefits of chatgpt in promoting teaching and learning. **Journal of AI**, v. 7, p. 52–62, 12 2023.

BELO, Elisabeth Mendes; PEREIRA, Sandra Maria Jerônimo; SILVA, Bruno Henrique Fernandes da; MALTA, Daniela Paula de Lima Nunes; FILHO, Marcos Antonio Soares de Andrade. A importância da leitura na formação do indivíduo. **Revista Ibero-Americana de Humanidades, Ciências e Educação – REASE**, São Paulo, v. 10, n. 5, p. 3942–3959, maio 2024. ISSN 2675-3375.

BRAGA, Pedro Guedes. **Leia Rapidinho: uma aplicação para exercitar a leitura de palavras da língua portuguesa**. Monografia (Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação)) — Universidade Federal de Campina Grande, Campina Grande, 2021.

CARVALHO, Ricardo Gondim de; FRANÇA, Aurenia Pereira de. A leitura como ação formadora do desenvolvimento pessoal e profissional do indivíduo. **Id on Line, Revista de Psicologia**, 2021.

COMITÊ GESTOR DA INTERNET NO BRASIL; Núcleo de Informação e Coordenação do Ponto BR. Release de Imprensa, **Plataformas da internet são o principal meio de acesso à informação entre brasileiros e superam rádio e TV, aponta nova pesquisa**. 2026. Disponível em: https://nic.br/noticia/releases/plataformas-da-internet-sao-o-principal-meio-de-acesso-a-informacao-entre-brasileiros-e-superam-radio-e-tv-aponta-nova-pesquisa/. Acesso em: 9 jun. 2026.

FIGUEIREDO, Carolina. **66% dos alunos brasileiros não leem textos com mais de dez páginas, diz estudo**. 2023. Disponível em: https://www.cnnbrasil.com.br/nacional/66-dos-alunos-brasileiros-nao-leem-textos-com-mais-de-dez-paginas-diz-estudo/. Acesso em: 18 set. 2025.

GERHARDT, Tatiana Engel; SILVEIRA, Denise Tolfo (Org.). **Métodos de pesquisa**. Porto Alegre: Editora da UFRGS, 2009. (Série Educação a Distância).

GIL, Antônio Carlos. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Atlas, 2002.

GOOGLE. **Gemini 2.5 Flash: Models and Hyperparameters**. 2025. Disponível em: https://ai.google.dev/gemini-api/docs/models/gemini-2.5-flash. Acessado em: 1 jul. 2026.

GOOGLE. **Prompt design strategies**. 2025. Disponível em: https://ai.google.dev/gemini-api/docs/prompting-strategies. Acessado em: 2 jul. 2026.

MATTOS, Adenilson Mariotti *et al.* A inteligência artificial como ferramenta pedagógica no ensino fundamental: Uma revisão sistemática temática. **Revista multidisciplinar integrada - REMI**, v. 2, n. 05, p. 1–18, mar. 2026. Disponível em: https://revistas.unipacto.com.br/index.php/multidisciplinar/article/view/346.

OLIVEIRA, Cláudio de; MOURA, Samuel Predosa. **Tic's na educação: a utilização das tecnologias da informação e comunicação na aprendizagem do aluno**. 2015. Disponível em: https://periodicos.pucminas.br/pedagogiacao/article/view/11019. Acesso em: 24 dez. 2025.

PEREIRA, Gabrielle Costa. As mídias digitais na prática docente do professor de língua portuguesa. **Língu@ Nostr@**, v. 8, n. 1, p. 295–310, nov. 2020. Disponível em: https://periodicos2.uesb.br/lnostra/article/view/16625.

PEREIRA, Lilian Inês Caballero Mauricio; ROSA, Matheus dos Santos; MARTINS, William D'Abruzzo. **TeacHelp: sistema para auxílio do professor**. Monografia (Trabalho de Conclusão de Curso (Técnico em Desenvolvimento de Sistemas Integrado ao Ensino Médio)) — Etec Philadelpho Gouvea Netto, Centro Paula Souza, São José do Rio Preto, 2021.

SANTOS, Emily. **O Brasil que lê menos: pesquisa aponta perda de quase 7 milhões de leitores em 4 anos; veja raio X**. 2024. Disponível em: https://g1.globo.com/educacao/noticia/2024/11/19/o-brasil-que-le-menos-pesquisa-aponta-que-pais-perdeu-quase-7-milhoes-de-leitores-em-4-anos-veja-raio-x.ghtml. Acesso em: 24 dez. 2025.

SILVA, Luiz Ricardo Mantovani da. **Interação Humano-Computador**. 1. ed. Rio de Janeiro: Freitas Bastos, 2025. E-book. Disponível em: https://plataforma.bvirtual.com.br.

SILVA, Mikael Almondes da; JUNIOR, Warley M Valente; VIEIRA, Alex de Souza; ALVEZ, Elton Rafael. Revisa+: plataforma web de revisão de conteúdos escolares por meio de histórias em quadrinhos. **Informática na Educação: teoria & prática**, v. 26, n. 2, p. 80–92, 2024.

SOMMERVILLE, Ian. **Engenharia de software**. 10. ed. São Paulo, SP: Pearson, 2018.

SOUSA, Lorena Ramalho de. O uso da internet nas práticas didáticas do ensino de língua portuguesa. **Universidade Federal do Tocantins**, 2020.

APÊNDICE A – TESTES AUTOMATIZADOS *END-TO-END*

Neste apêndice, apresenta-se o código-fonte dos testes ponta a ponta (E2E) desenvolvidos com o *framework* Playwright, cujos resultados foram discutidos na Seção 4. A organização dos arquivos estruturou-se com base nos dois módulos principais da solução: o administrativo, direcionado aos docentes, e o do cliente, voltado aos discentes. Cada arquivo de especificação (`.spec.ts`) reflete diretamente uma das suítes de testes detalhadas no referido capítulo. Visando à concisão do documento, foram omitidos os módulos auxiliares de infraestrutura de suporte, tais como a interceptação de chamadas à API (*mocks*), a reprodução dos fluxos de autenticação do Amazon Cognito e a gestão do processo de leitura, uma vez que não configuram cenários de validação propriamente ditos.

Código-fonte 1 – Suíte de autenticação e gerenciamento de conta (`auth.spec.ts`)

```
import { test, expect } from "@playwright/test";
import { E2E_CREDENTIALS } from "./helpers/auth-state";
import {
  mockCognitoConfirmSignUp,
  mockCognitoSignUp,
} from "./helpers/cognito-mock";

test.describe("Login (Cognito real)", () => {
  test("permite login com credenciais válidas e redireciona para o
   dashboard", async ({
    page,
  }) => {
    await page.goto("/login");
    await page
      .getByRole("textbox", { name: "Email" })
      .fill(E2E_CREDENTIALS.email);
    await page.locator("#password").fill(E2E_CREDENTIALS.password);
    await page.getByRole("button", { name: /entrar/i }).click();

    await page.waitForURL("/", { timeout: 30_000 });
```

```
20      await expect(page).toHaveURL("/");
21      await expect(
22        page.getByRole("heading", { name: /olá,/i }),
23      ).toBeVisible();
24    });
25
26    test("exibe erro inline ao tentar logar com senha incorreta", async
        ({
27      page,
28    }) => {
29      await page.goto("/login");
30      await page
31        .getByRole("textbox", { name: "Email" })
32        .fill(E2E_CREDENTIALS.email);
33      await page.locator("#password").fill("senha-errada-Aa1!");
34      await page.getByRole("button", { name: /entrar/i }).click();
35
36      await expect(
37        page.locator("p").filter({ hasText: /senha|incorret|inválid/i }),
38      ).toBeVisible({ timeout: 15_000 });
39      await expect(page).toHaveURL(/\/login/);
40    });
41  });
42
43  test.describe("Cadastro (mockado)", () => {
44    test("valida que as senhas precisam coincidir", async ({ page }) => {
45      await page.goto("/cadastro");
46
47      await page.locator("#name").fill("Maria Silva");
48      await page.locator("#email").fill("teste@e2e.local");
49      await page.locator("#password").fill("Abcdef12");
50      await page.locator("#confirmPassword").fill("Outracoisa1A");
```

```
51      await page.getByRole("button", { name: /criar conta/i }).click();
52
53      await expect(
54        page.getByText("As senhas não coincidem.", { exact: false }),
55      ).toBeVisible();
56    });
57
58    test("valida nome com sobrenome obrigatório", async ({ page }) => {
59      await page.goto("/cadastro");
60
61      await page.locator("#name").fill("Maria");
62      await page.locator("#email").fill("teste@e2e.local");
63      await page.locator("#password").fill("Abcdef12");
64      await page.locator("#confirmPassword").fill("Abcdef12");
65      await page.getByRole("button", { name: /criar conta/i }).click();
66
67      await expect(
68        page.getByText("Informe nome e sobrenome.", { exact: false }),
69      ).toBeVisible();
70    });
71
72    test("cadastra com sucesso e redireciona para confirmação de email",
       async ({
73      page,
74    }) => {
75      await mockCognitoSignUp(page);
76
77      await page.goto("/cadastro");
78
79      await page.locator("#name").fill("Maria Silva");
80      await page.locator("#email").fill("nova-prof@e2e.local");
81      await page.locator("#password").fill("Abcdef12");
```

```
82      await page.locator("#confirmPassword").fill("Abcdef12");
83      await page.getByRole("button", { name: /criar conta/i }).click();
84
85      await page.waitForURL(/\/confirmar/, { timeout: 15_000 });
86      await expect(page).toHaveURL(/email=nova-prof%40e2e\.local/);
87      await expect(
88        page.getByRole("heading", { name: /confirmar email/i }),
89      ).toBeVisible();
90    });
91  });
92
93  test.describe("Confirmação de email (mockado)", () => {
94    test("confirma com código válido e redireciona para login com
        mensagem de sucesso", async ({
95      page,
96    }) => {
97      await mockCognitoConfirmSignUp(page);
98
99      await page.goto("/confirmar?email=nova-prof%40e2e.local");
100     await page.locator("#code").fill("123456");
101     await page.getByRole("button", { name: /confirmar email/i
        }).click();
102
103     await page.waitForURL(/\/login/, { timeout: 15_000 });
104     await expect(
105       page.getByText("Email confirmado!", { exact: false }),
106     ).toBeVisible();
107   });
108
109   test("exibe erro inline para código inválido", async ({ page }) => {
110     await mockCognitoConfirmSignUp(page, { fail: true });
111
```

```
    await page.goto("/confirmar?email=nova-prof%40e2e.local");
    await page.locator("#code").fill("000000");
    await page.getByRole("button", { name: /confirmar email/i
  }).click();

    await expect(
      page.getByText(/código inválido|verification code/i),
    ).toBeVisible({ timeout: 10_000 });
    await expect(page).toHaveURL(/\/confirmar/);
  });

  test("valida formato do código (6 dígitos)", async ({ page }) => {
    await page.goto("/confirmar?email=nova-prof%40e2e.local");
    await page.locator("#code").fill("12");

    const submit = page.getByRole("button", { name: /confirmar email/i
  });
    await submit.click();

    await expect(page).toHaveURL(/\/confirmar/);
  });
});
```

Código-fonte 2 – Suíte da tela inicial (`dashboard.spec.ts`)

```
import { test, expect } from "@playwright/test";
import { mockClasses, mockAdminApi } from "./helpers/api-mock";

test.describe("Dashboard", () => {
  test.beforeEach(async ({ page }) => {
    await mockAdminApi(page);
  });
```

```
8
9   test("exibe boas-vindas e ações rápidas", async ({ page }) => {
10    await page.goto("/");
11
12    await expect(
13      page.getByRole("heading", { name: /olá,/i }),
14    ).toBeVisible();
15    await expect(page.getByText("Gerencie suas turmas e
      tarefas")).toBeVisible();
16
17    await expect(
18      page.getByRole("link", { name: /nova turma/i }).first(),
19    ).toBeVisible();
20    await expect(
21      page.getByRole("link", { name: /nova tarefa/i }).first(),
22    ).toBeVisible();
23    await expect(
24      page.getByRole("link", { name: /gerar com ia/i }),
25    ).toBeVisible();
26  });
27
28  test("mostra contador de turmas ativas com base nos mocks", async ({
29    page,
30  }) => {
31    await page.goto("/");
32
33    const activeCount = mockClasses.filter((c) => c.active).length;
34    const card = page.getByText("Turmas ativas").locator("..");
35    await expect(card).toContainText(String(activeCount));
36  });
37
38  test("exibe estado vazio quando não há tarefas recentes", async ({
```

```
    page }) => {
39    await mockAdminApi(page, { tasks: [] });
40    await page.goto("/");
41
42    await expect(page.getByText("Nenhuma tarefa ainda")).toBeVisible();
43    await expect(
44      page.getByText("Crie sua primeira tarefa para começar"),
45    ).toBeVisible();
46  });
47
48  test("ação rápida 'Nova Turma' redireciona para /turmas/nova", async
      ({
49    page,
50  }) => {
51    await page.goto("/");
52    await page.getByRole("link", { name: /nova turma/i
    }).first().click();
53    await expect(page).toHaveURL(/\/turmas\/nova/);
54  });
55});
```

Código-fonte 3 – Suíte de gestão de turmas (`classes.spec.ts`)

```
1 import { test, expect } from "@playwright/test";
2 import { mockAdminApi, mockClasses } from "./helpers/api-mock";
3
4 test.describe("Turmas", () => {
5   test.beforeEach(async ({ page }) => {
6     await mockAdminApi(page);
7   });
8
9   test("lista turmas mockadas com nome, ano e contagem de alunos",
```

```
  async ({
10  page,
11 }) => {
12   await page.goto("/turmas");
13
14   await expect(
15     page.getByRole("heading", { name: "Turmas", exact: true }),
16   ).toBeVisible();
17
18   const first = mockClasses[0];
19   await expect(page.getByRole("heading", { name: first.name
     })).toBeVisible();
20   await expect(
21     page.getByText(`${first.studentCount} alunos`).first(),
22   ).toBeVisible();
23 });
24
25 test("cria nova turma e volta para a listagem", async ({ page }) => {
26   await page.goto("/turmas");
27   await page.getByRole("link", { name: /nova turma/i }).click();
28
29   await expect(page).toHaveURL(/\/turmas\/nova/);
30   await expect(
31     page.getByRole("heading", { name: "Nova Turma", exact: true }),
32   ).toBeVisible();
33
34   await page.getByLabel(/nome da turma/i).fill("Turma E2E Teste");
35   await page.getByLabel(/ano letivo/i).fill("2026");
36   await page.getByLabel(/turno/i).selectOption("morning");
37   await page.getByLabel(/quantidade de alunos/i).fill("25");
38   await page
39     .getByLabel(/descrição/i)
```

```
40        .fill("Turma criada via Playwright para verificar fluxo de
      cadastro.");
41
42      await page.getByRole("button", { name: /criar turma/i }).click();
43      await page.waitForURL(/\/turmas$/, { timeout: 10_000 });
44    });
45
46    test("valida nome obrigatório ao tentar criar turma vazia", async ({
47      page,
48    }) => {
49      await page.goto("/turmas/nova");
50
51      await page.getByLabel(/nome da turma/i).fill("   ");
52      await page.getByRole("button", { name: /criar turma/i }).click();
53
54      await expect(
55        page.getByText("O nome da turma é obrigatório.", { exact: false
      }),
56      ).toBeVisible();
57    });
58
59    test("abre página de detalhes ao clicar em uma turma", async ({ page
      }) => {
60      await page.goto("/turmas");
61
62      const first = mockClasses[0];
63      await page.getByRole("link", { name: new RegExp(first.name, "i")
      }).click();
64
65      await expect(page).toHaveURL(new RegExp(`/turmas/${first.id}$`));
66    });
67
```

```
  test("filtra turmas pelo turno", async ({ page }) => {
    await page.goto("/turmas");

    await page.getByRole("button", { name: /^matutino$/i }).click();
    const morningCount = mockClasses.filter((c) => c.shift ===
   "morning").length;
    const cards =
   page.locator("a[href^='/turmas/']:not([href='/turmas/nova'])");
    await expect(cards).toHaveCount(morningCount);
  });
});
```

Código-fonte 4 – Suíte de gestão de tarefas (`tasks.spec.ts`)

```
import { test, expect } from "@playwright/test";
import { mockAdminApi, mockTasks } from "./helpers/api-mock";

test.describe("Tarefas", () => {
  test.beforeEach(async ({ page }) => {
    await mockAdminApi(page);
  });

  test("lista tarefas mockadas com título e status", async ({ page })
   => {
    await page.goto("/tarefas");

    await expect(
      page.getByRole("heading", { name: "Tarefas", exact: true }),
    ).toBeVisible();

    const first = mockTasks[0];
    await expect(page.getByText(first.title, { exact: false
```

```
  })).toBeVisible();
});

test("exibe estado vazio quando a API retorna lista vazia", async ({
  page,
}) => {
  await mockAdminApi(page, { tasks: [] });
  await page.goto("/tarefas");

  await expect(page.getByText("Nenhuma tarefa ainda")).toBeVisible();
  await expect(
    page.getByText("Crie sua primeira tarefa para começar"),
  ).toBeVisible();
});

test("cria nova tarefa como rascunho", async ({ page }) => {
  await page.goto("/tarefas");
  await page.getByRole("link", { name: /nova tarefa/i
 }).first().click();

  await expect(page).toHaveURL(/\/tarefas\/nova/);

  await page
    .getByLabel(/título da tarefa/i)
    .fill("Tarefa E2E - Interpretação");

  await page
    .getByLabel(/turma/i)
    .selectOption({ index: 1 });

  await page
    .getByPlaceholder(/cole ou escreva aqui este trecho/i)
```

```
48        .first()
49        .fill(
50          "Este é um trecho de teste para verificar o fluxo de criação
     de tarefa via Playwright.",
51        );
52
53      await page.getByRole("button", { name: /salvar como rascunho/i
     }).click();
54      await page.waitForURL(/\/tarefas$/, { timeout: 10_000 });
55    });
56
57    test("valida que título é obrigatório", async ({ page }) => {
58      await page.goto("/tarefas/nova");
59
60      await page
61        .getByLabel(/turma/i)
62        .selectOption({ index: 1 });
63      await page
64        .getByPlaceholder(/cole ou escreva aqui este trecho/i)
65        .first()
66        .fill("Texto qualquer");
67
68      await page.getByRole("button", { name: /salvar como rascunho/i
     }).click();
69
70      const titleInput = page.getByLabel(/título da tarefa/i);
71      const isInvalid = await titleInput.evaluate(
72        (el) => (el as HTMLInputElement).validity.valueMissing,
73      );
74      expect(isInvalid).toBe(true);
75    });
76
```

```
77    test("edita uma tarefa existente e salva alterações", async ({ page
       }) => {
78      const target = mockTasks[0];
79      await page.goto(`/tarefas/${target.id}/editar`);
80
81      await expect(
82        page.getByRole("heading", { name: /editar tarefa/i }),
83      ).toBeVisible();
84
85      const titleInput = page.getByLabel(/título da tarefa/i);
86      await expect(titleInput).toHaveValue(target.title);
87
88      await titleInput.fill(`${target.title} (editado)`);
89      await page.getByRole("button", { name: /salvar alterações/i
       }).click();
90
91      await page.waitForURL(new RegExp(`/tarefas/${target.id}$`), {
92        timeout: 10_000,
93      });
94    });
95  });
```

Código-fonte 5 – Suíte de perfil do usuário (`profile.spec.ts`)

```
1   import { test, expect } from "@playwright/test";
2   import { mockAdminApi } from "./helpers/api-mock";
3   import { mockCognitoSignOut } from "./helpers/cognito-mock";
4   import { E2E_CREDENTIALS } from "./helpers/auth-state";
5
6   test.describe("Perfil", () => {
7     test.beforeEach(async ({ page }) => {
8       await mockAdminApi(page);
```

```
9   });
10
11  test("exibe email do usuário autenticado", async ({ page }) => {
12    await page.goto("/perfil");
13
14    await expect(
15      page.getByRole("heading", { name: "Perfil", exact: true }),
16    ).toBeVisible();
17
18    await expect(page.getByText(E2E_CREDENTIALS.email)).toBeVisible();
19  });
20
21  test("mostra seção 'Zona perigosa' com botão de excluir conta",
     async ({
22    page,
23  }) => {
24    await page.goto("/perfil");
25
26    await expect(page.getByText("Zona perigosa")).toBeVisible();
27    await expect(
28      page.getByRole("button", { name: /excluir conta/i }),
29    ).toBeVisible();
30  });
31
32  test("logout limpa a sessão e redireciona para /login", async ({
     page }) => {
33    await mockCognitoSignOut(page);
34    await page.goto("/");
35
36    await page.getByRole("button", { name: "Sair" }).click();
37    await page.waitForURL(/\/login/, { timeout: 15_000 });
38
```

```
    await page.goto("/");
    await page.waitForURL(/\/login/, { timeout: 10_000 });
    await expect(page).toHaveURL(/\/login/);
  });
});
```

Código-fonte 6 – Suíte da tela de entrada da atividade (`welcome.spec.ts`)

```
import { test, expect } from "@playwright/test";
import { mockClientApi } from "./helpers/api-mock";

test.describe("Tela de entrada da atividade", () => {
  test.beforeEach(async ({ page }) => {
    await mockClientApi(page);
  });

  test("exibe as informações da tarefa disponível", async ({ page })
   => {
    await page.goto("/tarefa/t1");

    await expect(
      page.getByRole("heading", {
        name: /Interpretação de texto - A Chegada do Outono/i,
      }),
    ).toBeVisible();
    await expect(page.getByText("6º Ano A")).toBeVisible();
    await expect(page.getByText("Trechos")).toBeVisible();
    await expect(page.getByText("Questões")).toBeVisible();
    await expect(page.getByText("Tempo estimado")).toBeVisible();
  });

  test("exige nome completo antes de começar", async ({ page }) => {
```

```
24      await page.goto("/tarefa/t1");
25
26      await page.getByLabel(/digite seu nome completo/i).fill("a ");
27      await page.getByRole("button", { name: /começar a ler/i }).click();
28
29      await expect(
30        page.getByText("Digite seu nome completo para começar."),
31      ).toBeVisible();
32      await expect(page).toHaveURL(/\/tarefa\/t1$/);
33    });
34
35    test("inicia a atividade com nome válido", async ({ page }) => {
36      await page.goto("/tarefa/t1");
37      await page.getByLabel(/digite seu nome completo/i).fill("Maria de
       Souza");
38      await page.getByRole("button", { name: /começar a ler/i }).click();
39
40      await expect(page).toHaveURL(/\/tarefa\/t1\/atividade$/);
41    });
42
43    test("redireciona tarefa em rascunho para tela de indisponível",
       async ({
44      page,
45    }) => {
46      await page.goto("/tarefa/t5");
47
48      await expect(page).toHaveURL(/indisponivel\?reason=rascunho/);
49      await expect(
50        page.getByRole("heading", { name: /em preparação/i }),
51      ).toBeVisible();
52    });
53
```

```
54   test("redireciona tarefa encerrada para tela de indisponível", async
       ({
55     page,
56   }) => {
57     await page.goto("/tarefa/t3");
58
59     await expect(page).toHaveURL(/indisponivel\?reason=encerrada/);
60     await expect(
61       page.getByRole("heading", { name: /tarefa encerrada/i }),
62     ).toBeVisible();
63   });
64
65   test("redireciona tarefa inexistente para tela de indisponível",
       async ({
66     page,
67   }) => {
68     await page.goto("/tarefa/inexistente");
69
70     await expect(page).toHaveURL(/indisponivel\?reason=not-found/);
71     await expect(
72       page.getByRole("heading", { name: /não encontrada/i }),
73     ).toBeVisible();
74   });
75 });
```

Código-fonte 7 – Suíte do fluxo de leitura (`activity.spec.ts`)

```
1  import { test, expect } from "@playwright/test";
2  import type { Task } from "@leiamais/types";
3  import { mockClientApi } from "./helpers/api-mock";
4  import { answer, startTask } from "./helpers/task-flow";
5
```

```
const multiSegmentTask: Task = {
  id: "e2e-multi",
  title: "Atividade de múltiplos trechos",
  status: "disponivel",
  classId: "1",
  className: "Turma E2E",
  createdAt: new Date("2026-03-01"),
  segments: [
    {
      id: "seg1",
      text: "Primeiro trecho da leitura para o teste automatizado.",
      questions: [
        {
          id: "seg1-q1",
          text: "Qual é a ordem deste trecho?",
          alternatives: ["A) Primeiro", "B) Segundo"],
          correctAnswer: "A",
          explanation: "É o trecho inicial.",
        },
      ],
    },
    {
      id: "seg2",
      text: "Segundo trecho da leitura para o teste automatizado.",
      questions: [
        {
          id: "seg2-q1",
          text: "Qual é a ordem deste trecho?",
          alternatives: ["A) Primeiro", "B) Segundo"],
          correctAnswer: "B",
          explanation: "É o trecho final.",
        },
```

```
38        ],
39      },
40    ],
41  };
42
43  test.describe("Fluxo de leitura", () => {
44    test("exibe o primeiro trecho e suas questões", async ({ page }) => {
45      await mockClientApi(page);
46      await startTask(page, "t1");
47
48      await expect(page.getByText(/O outono chegou trazendo
      folhas/)).toBeVisible();
49      await expect(
50        page.getByText(/Qual é a ideia principal do trecho/i),
51      ).toBeVisible();
52      await expect(page.getByText(/Olá, Maria/)).toBeVisible();
53    });
54
55    test("mantém a conclusão bloqueada até responder todas as questões",
      async ({
56      page,
57    }) => {
58      await mockClientApi(page);
59      await startTask(page, "t1");
60
61      const concluir = page.getByRole("button", { name: /concluir
      atividade/i });
62      await expect(concluir).toBeDisabled();
63
64      await answer(page, "t1-q1", "B");
65      await answer(page, "t1-q2", "C");
66      await expect(concluir).toBeDisabled();
```

```
67
68      await answer(page, "t1-q3", "C");
69      await expect(concluir).toBeEnabled();
70    });
71
72    test("responder um trecho desbloqueia o próximo", async ({ page })
       => {
73      await mockClientApi(page, { tasks: [multiSegmentTask] });
74      await startTask(page, "e2e-multi");
75
76      await expect(
77        page.getByText("Primeiro trecho da leitura para o teste
       automatizado."),
78      ).toBeVisible();
79      await expect(
80        page.getByText("Segundo trecho da leitura para o teste
       automatizado."),
81      ).toBeHidden();
82
83      await answer(page, "seg1-q1", "A");
84      await page.getByRole("button", { name: /continuar leitura/i
       }).click();
85
86      await expect(
87        page.getByText("Segundo trecho da leitura para o teste
       automatizado."),
88      ).toBeVisible();
89    });
90
91    test("concluir a atividade leva à tela de feedback", async ({ page
       }) => {
92      await mockClientApi(page);
```

```
93      await startTask(page, "t7");
94
95      await answer(page, "t7-q1", "B");
96      await page.getByRole("button", { name: /concluir atividade/i
        }).click();
97
98      await expect(page).toHaveURL(/\/tarefa\/t7\/feedback$/);
99    });
100
101   test("acessar a atividade sem sessão redireciona à tela de entrada",
        async ({
102     page,
103   }) => {
104     await mockClientApi(page);
105     await page.goto("/tarefa/t1/atividade");
106
107     await expect(page).toHaveURL(/\/tarefa\/t1$/);
108   });
109 });
```

Código-fonte 8 – Suíte de resultado e *feedback* (`feedback.spec.ts`)

```
1  import { test, expect } from "@playwright/test";
2  import { mockClientApi } from "./helpers/api-mock";
3  import { answer, startTask } from "./helpers/task-flow";
4
5  async function completeTask1(page: import("@playwright/test").Page) {
6    await startTask(page, "t1");
7    await answer(page, "t1-q1", "B");
8    await answer(page, "t1-q2", "C");
9    await answer(page, "t1-q3", "C");
10   await page.getByRole("button", { name: /concluir atividade/i
```

```
    }).click();
  await page.waitForURL("**/tarefa/t1/feedback");
}

test.describe("Resultado e feedback", () => {
  test("exibe o resumo de desempenho do aluno", async ({ page }) => {
    await mockClientApi(page);
    await completeTask1(page);

    const main = page.getByRole("main");
    await expect(main.getByText(/Resultado de Maria/i)).toBeVisible();
    await expect(main.getByText("Acertos")).toBeVisible();
    await expect(main.getByText("3/3")).toBeVisible();
    await expect(main.getByText("Aproveitamento")).toBeVisible();
    await expect(main.getByText("100%")).toBeVisible();
  });

  test("exibe o feedback gerado por IA", async ({ page }) => {
    await mockClientApi(page);
    await completeTask1(page);

    await expect(page.getByText("Feedback da IA")).toBeVisible();
    await expect(page.getByText(/Bom trabalho, Maria/)).toBeVisible();
  });

  test("mantém o resultado mesmo quando a IA falha", async ({ page })
   => {
    await mockClientApi(page, { failAiFeedback: true });
    await completeTask1(page);

    await expect(page.getByText(/Resultado de Maria/i)).toBeVisible();
    await expect(page.getByText("100%")).toBeVisible();
```

```
41      await expect(page.getByText("Feedback da IA")).toBeHidden();
42    });
43
44    test("acessar o feedback sem resposta redireciona à tela de
       entrada", async ({
45      page,
46    }) => {
47      await mockClientApi(page);
48      await page.goto("/tarefa/t1/feedback");
49
50      await expect(page).toHaveURL(/\/tarefa\/t1$/);
51    });
52
53    test("concluir retorna à tela de entrada da tarefa", async ({ page
       }) => {
54      await mockClientApi(page);
55      await completeTask1(page);
56
57      await page.getByRole("button", { name: /concluir/i }).click();
58      await expect(page).toHaveURL(/\/tarefa\/t1$/);
59    });
60  });
```