CORREÇÃO AUTOMÁTICA DE PROVAS POR LLMS MULTIMODAIS COM DOIS AGENTES

Fernando Costa Castanheira

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Edmundo Albuquerque de Souza e Silva

Rio de Janeiro
Agosto de 2025

CORREÇÃO AUTOMÁTICA DE PROVAS POR LLMS MULTIMODAIS COM DOIS AGENTES

Fernando Costa Castanheira

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Edmundo Albuquerque de Souza e Silva

Aprovada por: Prof. Daniel Sadoc Menache
Prof. Henrique Luiz Cukierman

RIO DE JANEIRO, RJ – BRASIL
AGOSTO DE 2025

Costa Castanheira, Fernando

Correção Automática de Provas por LLMs Multimodais com Dois Agentes/Fernando Costa Castanheira. – Rio de Janeiro: UFRJ/COPPE, 2025.

XIII, 93 p.: il.; 29, 7cm.

Orientador: Edmundo Albuquerque de Souza e Silva

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2025.

Referências Bibliográficas: p. 59 – 65.

1. Large Language Models (LLMs). 2. Correção Automática. 3. Avaliação Educacional. 4. Processamento de Linguagem Natural (NLP). I. Albuquerque de Souza e Silva, Edmundo. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À minha esposa Bianca, cujo apoio e compreensão foram pilares fundamentais nesta jornada.

Agradecimentos

Gostaria de expressar minha sincera gratidão à Universidade Federal do Rio de Janeiro (UFRJ) e ao Programa de Engenharia de Sistemas e Computação (PESC), que me proporcionaram a valiosa oportunidade de cursar meu mestrado em Inteligência Artificial. Agradeço a todos os profissionais envolvidos por sua dedicação e empenho. Comprometo-me a retribuir à sociedade o investimento e as oportunidades que me foram concedidas.

Um agradecimento especial aos professores Edmundo Albuquerque de Souza e Silva e Geraldo Bonorino Xexéo. Suas inspiradoras aulas foram fundamentais para o meu desenvolvimento acadêmico e pessoal, e sua orientação foi decisiva na elaboração desta dissertação.

Estendo meus agradecimentos à comunidade de pesquisadores em inteligência artificial, cujos trabalhos e descobertas abriram caminhos para que outros, como eu, pudessem construir e expandir o conhecimento nesta área fascinante.

À minha esposa, Bianca, expresso minha mais profunda gratidão. Seu apoio incondicional foi essencial, mesmo enquanto eu equilibrava as responsabilidades da Jovens Gênios, do mestrado e das aulas no MBA. Sei que não foi fácil, e sua paciência e amor foram indispensáveis para minha jornada.

À minha família, agradeço pelo apoio constante e amor incondicional. Em especial, à minha mãe, Martha, cujo suporte e educação durante minha infância e adolescência formaram a base sólida sobre a qual construí minha carreira acadêmica.

Por fim, agradeço a todos na Jovens Gênios, minha empresa, que se dedica a oferecer plataformas digitais de aprendizagem para crianças e jovens em escolas de todo o Brasil. Vocês são uma fonte contínua de inspiração e me motivam a aplicar meus conhecimentos em tecnologia e IA para melhorar a educação no nosso país.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

CORREÇÃO AUTOMÁTICA DE PROVAS POR LLMS MULTIMODAIS COM DOIS AGENTES

Fernando Costa Castanheira

Agosto/2025

Orientador: Edmundo Albuquerque de Souza e Silva

Programa: Engenharia de Sistemas e Computação

Avaliar respostas discursivas de trabalhos escolares (provas, por exemplo) que combinam texto, manuscritos e diagramas continua sendo um gargalo operacional na educação brasileira. Este trabalho propõe e valida um **pipeline dois-agentes** — *Grader* e *Reviewer* — baseado em Modelos de Linguagem Multimodais (LLMs) de última geração. O *Grader* atribui nota e feedback conforme rubrica estruturada; o *Reviewer* audita essa saída, gera um `quality_score` e dispara uma única revisão quando o escore é inferior a 4.

Três conjuntos de dados reais de graduação (*Redes de Computadores*, *Introdução à Física* e *Introdução à Programação*; $N = 35$ cadernos) foram corrigidos com *Gemini-2.5-pro*, *Gemini-2.5-flash* e *o4-mini-high*. O pipeline atinge *concordância substancial* com docentes ($\kappa \geq 0{,}78$) e MAE $\leq 0{,}15$ sem *fine-tuning*. A presença do *Reviewer* reduz até 40 % dos erros extremos ($|\hat{y} - y| > 0{,}40$) em provas ricas em manuscritos, ao custo adicional médio de US$ 0.02 por caderno — duas ordens de grandeza abaixo do custo humano ($\approx$ US$ 2.75).

Para comprovar viabilidade prática, desenvolveu-se a aplicação web **Exam AI Grader** (Next.js 14 + Drizzle ORM + PGLite), que executa o fluxo completo e processa ~ 10 cadernos/min em ambiente serverless ou totalmente *offline-first*. O sistema, o código-fonte e os datasets encontram-se disponíveis em `https://github.com/CostaFernando/exam-ai-grader`.

Os resultados indicam que LLMs multimodais, combinados a um laço leve de auto-revisão, podem oferecer correção automática confiável para avaliações universitárias em língua portuguesa.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

AUTOMATIC EXAM GRADING WITH MULTIMODAL LLMS VIA A TWO-AGENT SYSTEM

Fernando Costa Castanheira

August/2025

Advisor: Edmundo Albuquerque de Souza e Silva

Department: Systems Engineering and Computer Science

Evaluating open-ended answers from academic assignments that combine text, handwriting, and diagrams remains an operational bottleneck in Brazilian education. This work proposes and validates a **two-agent pipeline** — *Grader* and *Reviewer* — based on state-of-the-art Multimodal Language Models (LLMs). The *Grader* assigns a grade and feedback according to a structured rubric; the *Reviewer* audits this output, generates a `quality_score`, and triggers a single revision when the score is below 4.

Three real-world undergraduate datasets (*Computer Networks*, *Introduction to Physics*, and *Introduction to Programming*; $N = 35$ exam booklets) were graded using *Gemini-2.5-pro*, *Gemini-2.5-flash*, and *o4-mini-high*. The pipeline achieves *substantial agreement* with instructors ($\kappa \geq 0{,}78$) and MAE $\leq 0{,}15$ without *fine-tuning*. The presence of the *Reviewer* reduces up to 40% of extreme errors ($|\hat{y} - y| > 0{,}40$) in exams rich in handwritten content, at an additional average cost of US$ 0.02 per booklet — two orders of magnitude below human grading cost ($\approx$ US$ 2.75).

To demonstrate practical feasibility, the web application **Exam AI Grader** (Next.js 14 + Drizzle ORM + PGLite) was developed. It executes the full pipeline and processes ~ 10 booklets/min in a serverless or fully *offline-first* environment. The system, source code, and datasets are available at `https://github.com/CostaFernando/exam-ai-grader`.

The results indicate that multimodal LLMs, combined with a lightweight self-review loop, can provide reliable automated grading for university-level assessments in Portuguese.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Avaliar respostas abertas — ensaios, problemas numéricos encadeados, diagramas — continua sendo um gargalo logístico para a educação básica e superior. Em 2024, o *Censo Escolar* registrou **47,1 milhões** de matrículas distribuídas em 179 mil escolas brasileiras[1]. A correção permanece quase inteiramente manual. **Importante: o momento de corrigir é pedagógico por excelência** — é nele que o docente identifica padrões de erro e ajusta sua prática. **O problema surge com o volume**: quando há muitas turmas ou provas extensas, o processo torna-se exaustivo e compete com planejamento e intervenções didáticas[2]. O impacto pedagógico também é evidente: o Brasil obteve apenas 379 pontos em Matemática no PISA 2022 — 93 pontos abaixo da média da OCDE[3]. Esses dados revelam um duplo imperativo:

(i) **Redistribuição do tempo docente**, aliviando etapas repetitivas e de consolidação *sem suprimir* a análise avaliativa — pelo contrário, liberando foco para intervenções individualizadas e formação continuada;

(ii) **Aprimoramento da qualidade e velocidade do feedback**, reconhecidamente um dos fatores de maior efeito no aprendizado[4, 5].

Princípio orientador. Nesta dissertação, a IA é tratada como **apoio ao trabalho docente**, não substituto. O objetivo é reduzir carga repetitiva e acelerar o retorno aos estudantes, preservando a decisão pedagógica final do professor.

Grandes Modelos de Linguagem (LLMs) multimodais oferecem uma via promissora para endereçar esses desafios, pois já alcançam níveis de concordância próximos a avaliadores humanos em tarefas puramente textuais[6, 7]. Contudo, seu desempenho degrada sensivelmente diante de manuscritos e diagramas complexos[8, 9]. Além disso, poucos estudos abordam explicitamente mecanismos de *auto-auditoria* que limitem erros graves sem penalizar custos ou latência – tempo para o sistema processar por completo uma avaliação.

Esta dissertação investiga um **fluxo dois-agentes** — *Grader* e *Reviewer* — que integra LLMs multimodais de última geração, rubricas estruturadas e um ciclo único

de crítica-refinamento[10, 11]. Por meio de três exames de graduação brasileiros, avaliamos até que ponto essa arquitetura:

- atinge concordância *substancial* (Cohen's $\kappa \geq 0{,}78$);
- reduz erros grosseiros de correção;
- mantém custos viáveis para adoção em larga escala.

Os resultados pretendem contribuir tanto para a pesquisa em avaliação automática quanto para políticas públicas que buscam maior equidade e eficiência na educação.

Artefato de software. Além da investigação científica, esta dissertação entrega a aplicação web *open-source* **Exam AI Grader**[1], que encapsula todo o pipeline dois-agentes. O sistema permite que docentes façam upload de provas e respostas em PDF, gerem rubricas com IA e recebam, em minutos, notas e feedback detalhado em formato `.csv`. Os detalhes de engenharia, fluxos de uso e instruções de implantação encontram-se no Capítulo 5.

1.1 Motivação e relevância social

Tempo docente como recurso crítico

Segundo a OCDE, apenas $\sim 67\,\%$ do tempo do professor brasileiro é dedicado efetivamente a atividades de ensino; o restante dispersa-se entre disciplina de sala e trabalho administrativo, **incluindo a consolidação de correções em grande volume**[2]. Automatizar *partes repetitivas* do fluxo (p. ex., triagem, agregação por rubrica, geração de rascunhos de feedback) **preserva** o momento pedagógico da análise e **realoca** horas para acompanhamento individualizado e planejamento — dimensões empiricamente ligadas à aprendizagem de alto impacto.

Feedback rápido e aprendizagem

Meta-análises consolidadas mostram que o feedback formativo oportuno gera ganhos de até $0{,}50\sigma$ no desempenho acadêmico[4] – isto é, um tamanho de efeito padronizado $d = 0{,}5$, medido em unidades de desvio-padrão; diferença de meia σ entre grupos, o que leva a média do grupo tratado de 50^{o} para 69^{o} percentil. Em contextos massivos (ENEM, vestibulares) ou cursos com turmas de centenas de alunos, a devolução manual de comentários detalhados é impraticável. Sistemas automáticos

[1] `https://github.com/CostaFernando/exam-ai-grader`

podem democratizar esse benefício, corrigindo assimetrias históricas entre escolas públicas e privadas.

Equidade e padronização

Disparidades regionais e socioeconômicas superam 100 pontos no PISA entre os quintis extremos de nível socioeconômico[3]. Algoritmos bem calibrados — apoiados em rubricas explícitas, laços de revisão e **supervisão docente** — **podem** oferecer critérios mais uniformes e transparentes, contribuindo para **mitigar** vieses conscientes ou inconscientes presentes na avaliação humana.

Impacto econômico

Estudos internacionais estimam reduções de até 60 % nos custos operacionais de exames quando a correção automática é adotada[12]. Recursos poupados podem ser reinvestidos em infraestrutura escolar e formação docente, gerando um ciclo virtuoso de melhoria sistêmica.

Em síntese, investigar fluxos de correção automática por LLMs multimodais com auto-revisão não é apenas um problema técnico, mas um passo estratégico para elevar qualidade, equidade e eficiência da educação brasileira — **apoiando o julgamento pedagógico do professor, e não o substituindo**.

1.2 Problema de pesquisa e perguntas norteadoras

Apesar dos avanços em *large language models* (LLMs) multimodais, três lacunas permanecem evidentes na literatura e na prática educacional brasileira:

(a) **Desempenho inconsistente em respostas multimodais**. Estudos recentes mostram queda significativa de acurácia quando os modelos são confrontados com manuscritos e diagramas complexos, sobretudo em disciplinas de *STEM* [8, 9, 13].

(b) **Ausência de mecanismos leves de autoauditoria**. Embora ciclos de crítica–refinamento melhorem a confiabilidade, ainda não há consenso sobre o equilíbrio ótimo entre custo, latência (tempo de processamento do sistema de avaliação) e ganhos de precisão na nota auferida pelo sistema [10, 11].

(c) **Escassez de evidências em língua portuguesa**. A maior parte dos experimentos foi conduzida em inglês ou outros idiomas [6, 7], deixando em aberto se resultados semelhantes podem ser reproduzidos em avaliações brasileiras, com escrita e rubricas locais.

Diante dessas lacunas, **este trabalho investiga se um fluxo *dois-agentes* (*Grader* + *Reviewer*) pode oferecer correção multimodal mais confiável e economicamente viável em provas universitárias brasileiras**.

Para orientar a investigação, foram formuladas as seguintes **Perguntas de Pesquisa (PQs)**:

PQ1: **Precisão**: Em que medida LLMs multimodais de última geração reproduzem as notas atribuídas por docentes humanos em respostas manuscritas e com diagramas em língua portuguesa?

PQ2: **Confiabilidade do reviewer**: O ciclo de auto-revisão proposto reduz a frequência e a magnitude de erros de correção quando comparado a uma estratégia de agente único?

PQ3: **Custo e latência**: Qual o impacto do *Reviewer* no consumo de *tokens*, tempo de processamento e custo financeiro, e onde se situa o ponto de equilíbrio qualidade–custo para adoção em larga escala?

PQ4: **Falhas remanescentes**: Quais categorias de erro persistem mesmo após a auto-revisão, e que oportunidades de pesquisa futura elas indicam (p. ex. finetuning de encoders visuais)?

Respondendo a essas questões, a dissertação busca oferecer evidências empíricas e diretrizes práticas para a adoção de LLMs multimodais na correção de avaliações no contexto brasileiro.

1.3 Objetivos

Objetivo geral

Desenvolver, validar e disponibilizar um **fluxo de dois-agentes** (Grader + Reviewer) capaz de **propor correções automáticas** para respostas de provas universitárias em português que combinem texto, manuscritos e diagramas, alcançando nível de concordância *substancial* com avaliadores humanos, **com rastreabilidade para revisão e decisão final do docente**, ao mesmo tempo em que mantém custos de execução viáveis para adoção em larga escala na educação brasileira.

Objetivos específicos

(a) **Construir** a arquitetura de correção multimodal, integrando modelos *vision–language* de última geração, rubricas estruturadas em JSON Schema e um ciclo único de crítica-refinamento.

(b) **Compilar e disponibilizar** três conjuntos de dados de avaliações de graduação brasileiras (Redes de Computadores, Introdução à Física e Introdução à Programação), com respostas de estudantes anonimizados, gabaritos e rubricas revisadas.

(c) **Avaliar a precisão** do Grader em termos de MAE, RMSE, Cohen's κ e JSD (definidos na Seção 4.4), comparando-o às notas atribuídas por docentes humanos.

(d) **Medir a contribuição** do Reviewer quanto à redução de erros extremos e à melhora da concordância inter-avaliador, em relação a uma linha de base de agente único.

(e) **Analisar custo e latência**, quantificando consumo de *tokens* e tempo de processamento por avaliação, e explorar pontos de equilíbrio entre qualidade e despesa para diferentes contextos de uso.

(f) **Desenvolver** uma aplicação web de código aberto que permita a educadores carregar provas em PDF, gerar rubricas e executar o pipeline em lote, retornando notas e feedback por questão.

(g) **Propor diretrizes** de adoção responsável do sistema em larga escala, considerando aspectos pedagógicos e operacionais.

1.4 Metodologia geral (visão de alto nível do fluxo dois-agentes)

Visão panorâmica

A Figura 1.1 ilustra o **pipeline dois-agentes** proposto, composto por três macroetapas:

F1 Pré-processamento multimodal

- Rasterização de cada página PDF, gerando arquivos PNG consumíveis pelos modelos vision–language.
- Concatenação do enunciado, resposta do aluno, resposta-modelo de referência e rubrica estruturada.

F2 Primeira passagem do *Grader*

- LLM multimodal (*o4-mini-high*, *Gemini-2.5-pro* ou *Gemini-2.5-flash*) recebe o *prompt* de avaliação.

- Produz objeto JSON conforme `GRADING_SCHEMA` contendo `nota` normalizada e `feedback`.

F3 Auditoria do *Reviewer*

- O mesmo modelo (ou diversidade de modelo) avalia a saída do *Grader*, atribuindo `quality_score` 1,...,5.
- Se `quality_score`<4, a crítica é reenviada ao *Grader* para *uma* revisão final; caso contrário, a nota inicial é aceita.

Componentes principais

LLMs: *o4-mini-high* (OpenAI), *Gemini-2.5-pro* e *Gemini-2.5-flash*; todos suportam entrada de imagem + texto.

Rubrica:
gerada automaticamente a partir do gabarito e **revisada pelo docente**; cada critério vira chave JSON, garantindo rastreabilidade.

Schemas:
`GRADING_SCHEMA` (notas/feedback por questão) e `REVIEW_SCHEMA` (crítica e `quality_score`).

Limite de iterações:
1 ciclo crítica–refinamento, balanceando custo e ganho de confiabilidade.

Persistência e reprodutibilidade

Todo o fluxo é orquestrado em `Python 3.11`, registrando:

- métricas (MAE, RMSE, κ, JSD) por questão e por aluno;
- custos de tokens (input/output) para análise de custo-benefício.

Logs e artefatos são publicados, junto ao código-fonte, no repositório público `https://github.com/CostaFernando/exam-ai-grader`.

1.5 Estrutura da dissertação

A dissertação está organizada em 8 capítulos, além dos elementos pré-textuais (capa, resumo, abstract, listas) e pós-textuais (referências, apêndices e anexos). A seguir, apresenta-se um panorama do conteúdo de cada capítulo:

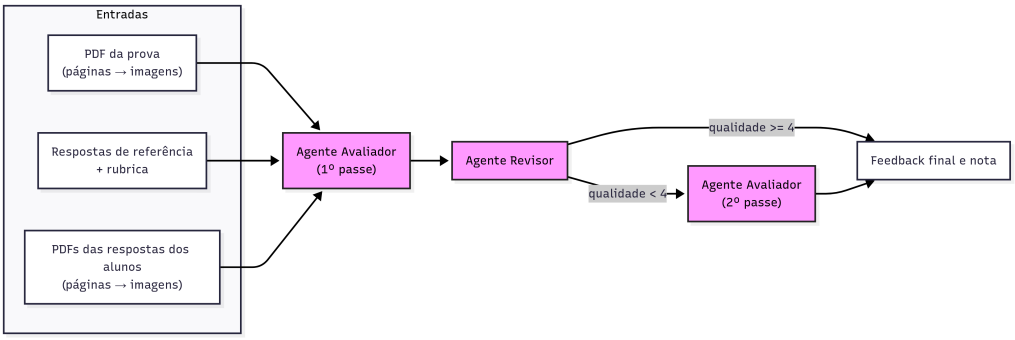


Figura 1.1: Fluxo geral de correção: pré-processamento multimodal, passagem do *Grader*, auditoria do *Reviewer* e retorno de notas/feedback.

Capítulo 1 — Introdução

Delimita o contexto, explicita o problema de pesquisa, formula as perguntas norteadoras, define objetivos e descreve, em alto nível, a metodologia de fluxo dois-agentes proposta.

Capítulo 2 — Fundamentação Teórica

Revisa conceitos de avaliação educacional, métricas de qualidade, evolução dos sistemas de correção automática, LLMs multimodais, rubricas estruturadas e ciclos crítica–refinamento.

Capítulo 3 — Contextualização do Problema

Apresenta o panorama da avaliação no cenário brasileiro, identificando limitações práticas das abordagens atuais e requisitos funcionais para um corretor multimodal aplicado a provas reais.

Capítulo 4 — Metodologia

Detalha a arquitetura dois-agentes, descreve os conjuntos de dados, métricas, protocolos experimentais e infraestrutura utilizada, garantindo reprodutibilidade.

Capítulo 5 — Implementação Aplicação Web

Documenta decisões de engenharia: pré-processamento de PDFs, integração com APIs de LLMs, persistência via Drizzle ORM/PostgreSQL e a aplicação web open-source disponibilizada aos educadores.

Capítulo 6 — Resultados

Apresenta resultados quantitativos (MAE, RMSE, κ, JSD) e qualitativos, compara configurações com e sem *Reviewer*, avalia custo-benefício e discute falhas remanescentes.

Capítulo 7 — Discussão

Interpreta os achados à luz das perguntas de pesquisa, discute implica-

ções pedagógicas, limitações do estudo e possíveis vieses, conectando-os à literatura.

Capítulo 8 — Conclusão

Resume as contribuições, destaca recomendações práticas para adoção em larga escala e propõe linhas de pesquisa futuras, como adaptação dinâmica do *Reviewer* e finetuning de encoders visuais.

Os apêndices incluem os *prompts*, `JSON Schemas` e scripts utilizados nos experimentos; os anexos reúnem exemplos completos de provas anonimizadas.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta os conceitos que sustentam a pesquisa, cobrindo (i) fundamentos de avaliação educacional e métricas de confiabilidade, (ii) evolução histórica da correção automática de respostas abertas e (iii) avanços recentes em Grandes Modelos de Linguagem (LLMs) multimodais, rubricas estruturadas e ciclos de crítica–refinamento. A ênfase recai sobre as métricas quantitativas que permitem comparar, de forma objetiva, a performance de avaliadores humanos e sistemas automáticos.

2.1 Avaliação educacional e métricas de qualidade

Avaliar a qualidade de um sistema de correção exige métricas que captem tanto *erros numéricos* quanto *concordância ordinal* e *distribucional*. Este trabalho adota quatro indicadores complementares, descritos a seguir.

Erro Médio Absoluto (MAE). Mede a discrepância média entre a nota prevista ($\hat{y}_i$) e a nota de referência (y_i) em nível de **item**. Aqui, um *item* é uma **resposta a uma questão por um aluno** (isto é, um par aluno–questão); assim, i indexa itens e N é o número de itens no subconjunto considerado: (i) *MAE por questão* — calcula-se separadamente para cada questão, agregando todos os alunos dessa questão ($N = \text{n}^{\text{o}}$ de alunos que a responderam); (ii) *MAE global do exame* — calcula-se sobre *todos* os itens do exame ($N = \text{n}^{\text{o}}$ total de respostas, isto é, alunos $\times$ questões), o que equivale a ponderar cada questão pelo seu número de respostas. A expressão é

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|.$$

Como todas as notas são normalizadas para $[0, 1]$, um MAE de 0,10 indica erro médio de 10% da escala (aprox. 1,0 ponto em uma escala 0–10). O MAE é intuitivo

e linear, mas não penaliza desproporcionalmente grandes desvios [14].

Raiz do Erro Quadrático Médio (RMSE). Calcula-se por

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}.$$

Por elevar os erros ao quadrado antes da média, amplifica *outliers*, oferecendo visão complementar ao MAE [15]. Em avaliações de alto risco, um RMSE baixo é crucial para evitar penalizações injustas.

Coeficiente kappa de Cohen (κ) e versão ponderada. Mede o acordo entre dois avaliadores, descontando o acordo esperado ao acaso:

$$\kappa = \frac{p_o - p_e}{1 - p_e}.$$

Seja N o número total de itens. O acordo observado é $p_o = \frac{1}{N}\sum_i n_{ii}$, onde n_{ij} é o número de itens aos quais o Avaliador A atribuiu a categoria i e o Avaliador B a categoria j. O acordo esperado ao acaso é $p_e = \sum_i \big(\frac{n_{i+}}{N}\big)\big(\frac{n_{+i}}{N}\big)$, com $n_{i+} = \sum_j n_{ij}$ e $n_{+i} = \sum_j n_{ji}$ [16].

Para notas *ordinais*, usamos a versão *ponderada* $w\kappa$, que atribui pesos $w_{ij} \in [0, 1]$ aos pares de categorias, de forma que desacordos mais distantes sejam penalizados mais fortemente:

$$w\kappa = \frac{P_{o,w} - P_{e,w}}{1 - P_{e,w}}, \qquad P_{o,w} = \sum_i \sum_j w_{ij}\, o_{ij}, \quad P_{e,w} = \sum_i \sum_j w_{ij}\, e_{ij},$$

onde $o_{ij} = n_{ij}/N$ e $e_{ij} = (n_{i+}/N)(n_{+j}/N)$. Adotamos **pesos quadráticos**,

$$w_{ij} = 1 - \frac{(i-j)^2}{(K-1)^2},$$

com K o número de categorias ordinais. Neste trabalho, as notas normalizadas $[0, 1]$ são discretizadas em $\mathbf{K = 5}$ faixas de mesma largura (0–0,2, 0,2–0,4, ..., 0,8–1,0) e reportamos o κ **ponderado quadraticamente**. Valores entre 0,61 e 0,80 indicam acordo *substancial*; acima de 0,80, *quase perfeito* [17].

Distância de Jensen–Shannon (JSD). Compara as distribuições de notas do modelo (Q) e do humano (P):

$$\mathrm{JSD}(P\|Q) = \sqrt{\tfrac{1}{2}\Big(D_{\mathrm{KL}}(P\|M) + D_{\mathrm{KL}}(Q\|M)\Big)}, \quad M = \tfrac{1}{2}(P+Q),$$

onde $D_{\mathrm{KL}}(P\|M) = \sum_x P(x)\log_2\frac{P(x)}{M(x)}$ é a divergência de Kullback–Leibler (no caso discreto). Seguimos a implementação do `scipy.spatial.distance.jensenshannon`, que normaliza os vetores de entrada e retorna a *distância* de Jensen–Shannon (raiz da divergência), limitada ao intervalo $[0, 1]$. Valores próximos de zero indicam maior alinhamento distribucional [18].

Em conjunto, essas métricas oferecem uma visão holística: MAE e RMSE capturam *precisão numérica*; κ, a *concordância ordinal*; e JSD, possíveis vieses na *distribuição* de notas. A adoção simultânea é recomendada por organismos de avaliação para garantir confiabilidade e justiça no processo de correção.

2.2 Evolução da correção automática

A pesquisa em correção automática de respostas abertas pode ser agrupada, grosso modo, em três ondas: (i) *heurísticas de similaridade lexical*, apoiadas em contagem de n–grams e vetores de frequência; (ii) *modelos estatísticos e de aprendizado supervisionado*, como regressões e *support vector machines* treinados em representações LSA ou TF–IDF; (iii) *modelos neurais e LLMs de múltiplas modalidades*, que incorporam atenção, conhecimento de domínio e visão computacional. Esta seção revisa cada estágio, começando pela era das heurísticas clássicas.

2.2.1 Similaridade lexical & heurísticas clássicas

Primeiros experimentos baseados em recuperação de informação (*Information Retrieval*, IR). O estudo seminal de LARKEY [19] tratou a *essay scoring* como um problema de recuperação de informação (IR): a pontuação de um texto era proporcional à similaridade TF–IDF (*term frequency–inverse document frequency*) entre o ensaio do aluno e um conjunto de respostas de referência. Pouco depois, FOLTZ *et al.* [20] aplicaram *Latent Semantic Analysis* (LSA) para capturar relações sinonímicas e alcançaram correlações de $r \approx 0{,}80$ com avaliadores humanos, inaugurando o uso de espaços semânticos densos na área.

Sistemas comerciais de primeira geração. A ETS lançou o *e-rater* em 1999, combinando contagens de n–grams, erros gramaticais e estatísticas de estilo [21]. Versões posteriores refinaram as micro features e o modelo de agregação, mantendo ainda forte dependência de semelhança lexical [22]. Na mesma época, o *Intelligent Essay Assessor* de Landauer explorou LSA em larga escala, reforçando a viabilidade comercial de técnicas puramente estatísticas [23].

Aplicações a respostas curtas. Trabalhos subsequentes adaptaram métricas de tradução automática, como BLEU e ROUGE, para questões de poucas sentenças [24, 25]. Embora simples de implementar, essas abordagens penalizavam severamente parafrases legítimas, levando ao surgimento de medidas semânticas híbridas — por exemplo, a combinação de *WordNet* e distância de cosseno testada por MOHLER e MIHALCEA [26].

Limitações das heurísticas lexicais. Apesar de rápidos e transparentes, esses métodos *(a)* ignoram variações legítimas de vocabulário, *(b)* falham em capturar relações lógicas ou multimodais e *(c)* tendem a reproduzir vieses do corpus de treino, como mostram revisões recentes [27]. Tais fragilidades motivaram a transição para modelos supervisionados e, posteriormente, para redes neurais de grande escala — temas discutidos nas seções 2.2.2 e 2.3.

Em síntese, a fase de similaridade lexical estabeleceu as bases da correção automática, mas suas limitações semânticas e generalização restrita impulsionaram a busca por abordagens mais robustas, culminando nos modelos de aprendizado profundo contemporâneos.

2.2.2 Modelos BERT-like e fine-tuning supervisionado

Os avanços da *Transformer architecture* culminaram no **BERT** (Bidirectional Encoder Representations from Transformers), que introduziu pré-treinamento com máscaras de palavras e predição de sentença próxima, gerando embeddings contextualizados bidirecionais. A grande contribuição para avaliação automática é que *fine-tuning* supervisionado em conjuntos relativamente pequenos pode exceder abordagens lexicais ou LSA em métricas de correção.

Primeiros resultados. CAMUS e FILIGHERA [28] mostraram que BERT *base* finamente ajustado em 4 400 respostas curtas de engenharia reduziu o MAE em 19 % versus SVM+TF–IDF. Logo em seguida, BARAL [29] empregaram *Sentence-BERT* com kNN para respostas matemáticas, obtendo ganhos de até 0,12 em coeficiente κ. Para problemas de cálculo manuscrito, o ajuste de *SBERT–Canberra* com meta-learning diminuiu a RMSE de 0,35 para 0,26 [30].

Variante RoBERTa e domínios maiores. Ao eliminar *Next Sentence Prediction* e ampliar o *corpus* de pré-treinamento, **RoBERTa** elevou a correlação em *Automated Essay Scoring*: correlações Pearson de 0,79 em ensaios TOEFL [6]. Estudos posteriores mostram que RoBERTa mais camada de regressão linear atinge $\kappa = 0{,}83$ em bancos de dados do Kaggle de *essay scoring*.

Impacto no pipeline educacional. Além da melhora de exatidão, modelos BERT-like permitem:

- **Geração de feedback formativo** com *prompting* de atenção às rubricas, aumentando utilidade pedagógica [7];
- **Adaptação de domínio** com poucas amostras, viabilizando uso em disciplinas específicas;
- **Integração multimodal preliminar**, ligando embeddings textuais a detectores de entidades visuais, embora ainda restritos a texto e equações em LaTeX.

Em síntese, a era BERT-like consolidou o uso de pré-treinamento massivo + *fine-tuning* leve como padrão de correção automática, preparando terreno para modelos multimodais que incorporam imagens e manuscrito — tema da Seção 2.3.

2.2.3 Aprendizagem em contexto, engenharia de *prompts* e modelos com *reasoning* nativo

Depois do *fine-tuning* supervisionado, o avanço decisivo foi a **aprendizagem em contexto** (*In-Context Learning – ICL*), revelada pelo GPT-3 de BROWN *et al.* [31]. Em ICL, o modelo generaliza a partir de *poucos exemplos* (*few-shot*) mostrados no próprio *prompt*, dispensando reajuste de pesos. Para correção de respostas abertas, isso abriu três frentes complementares:

(i) Decomposição por rubrica. Ao concatenar cada critério da rubrica como micro-pergunta, o modelo decide item a item e agrega a nota final. Estratégia empregada por QIU *et al.* [10] e LEE *et al.* [7], elevando κ em até 0,13 sem retreino.

(ii) Cadeia de raciocínio (*Chain-of-Thought*). WEI *et al.* [32] mostraram que fornecer exemplos com passos de raciocínio explícitos faz surgir habilidades aritméticas e lógicas. O método foi refinado por WANG *et al.* [33], cuja *Self-Consistency* amostra múltiplas cadeias e vota na resposta mais frequente, reduzindo a variância – recurso crucial para evitar notas erráticas em provas.

(iii) Seleção automática de exemplos. Retrievers densos treinados para buscar bons exemplos in-context [34] diminuem a sensibilidade à ordem dos exemplos e mantêm desempenho estável mesmo com limite de contexto apertado. JIANG *et al.* [12] complementam com *Bayesian Prompt Ensembles*, estimando incerteza e sinalizando quando revisar manualmente.

Modelos com *reasoning* nativo. A geração mais recente de LLMs traz modos de inferência internos que dispensam cadeias de raciocínio explícitas:

- **o4-mini-high** (OpenAI) – otimizado para raciocínio passo a passo e custo reduzido, atingindo desempenho de topo em AIME 2025 com poucas *shots* [35].
- **Gemini 2.5 Pro e Flash** (Google DeepMind) – projetados para "pensar antes de responder", superam GPT-4 em benchmarks de raciocínio multimodal, mantendo latência baixa [36, 37].

Esses modelos exibem *raciocínio nativo* (*native reasoning mode*): em vez de revelar a cadeia de pensamento, realizam-na internamente via reforço ou instruções sistêmicas. Na prática:

(a) Reduzem a necessidade de *prompts* longos com demonstrações CoT, liberando contexto para imagens e rubricas completas;

(b) Mantêm ou superam κ dos modelos CoT+SC tradicionais em correção de respostas manuscritas, como evidenciado nos experimentos deste trabalho (Cap. 6);

Limitações e desafios. Mesmo com *reasoning* nativo, esses modelos continuam sensíveis a: (i) **deriva de prompt** em línguas menos representadas; (ii) **ambiguidade visual** em diagramas densos; e (iii) **janelas de contexto** finitas quando se insere rubricas extensas. Tais pontos motivam o fluxo dois-agentes desta dissertação, que combina *ICL* + *Reviewer* para mitigar lapsos residuais.

Dessa forma, a evolução culmina em LLMs que aprendem *no próprio contexto* e já internalizam heurísticas de raciocínio, estabelecendo a base para sistemas de correção automática mais simples e robustos.

2.3 Modelos multimodais para *OCR-free grading*

A terceira onda da correção automática emerge com os **Vision–Language Models (VLMs)**, capazes de ingerir imagens de provas escaneadas *sem* pré-processamento OCR explícito. A evolução pode ser resumida em três estágios.

(a) Pré-treino contrastivo

- **CLIP** inaugurou o contraste texto–imagem em 400 M pares, habilitando transferência zero-shot para tarefas de OCR, VQA e classificação [38].
- O sucesso do contraste motivou corpus mais ricos em fórmulas e diagramas, preparando terreno a domínios STEM.

(b) Acoplamento leve de encoders

- **Flamingo** conecta encoders pré-treinados via camadas condutoras, alcançando poucos-exemplos em VQA e legendagem [39].
- **BLIP-2** emprega um *Query-Former* fino que faz "ponte" entre ViT congelado e LLM, reduzindo parâmetros treináveis em 54 × sem perder acurácia [40].

(c) *Reasoning* nativo e contexto ampliado

- **PaLM-E** mostra transferência cruzada de visão, linguagem e robótica, sugerindo que múltiplas modalidades reforçam raciocínio [41].
- VLMs proprietários de "raciocínio nativo" — *Gemini 2.5 Pro/Flash* e *o4-mini-high* — executam passos internos de CoT, permitindo prompts mais curtos e janela maior para rubricas completas [35, 36].

Open-source e *OCR-free* de fato. Modelos como **MiniGPT-4** alinham um ViT congelado a um LLM Vicuna com apenas uma projeção linear, revelando capacidades de descrição e revisão de diagramas [42]. O **Kosmos-2** adiciona *grounding* de caixas, útil para localizar rotulagens de figuras [43]. Por fim, engines recém-publicados — *TextMonkey* — processam PDFs 300 dpi diretamente, descartando OCR e mantendo fidelidade a rascunhos finos [44].

Implicações para esta dissertação. Diferentemente de pipelines baseados em OCR [9], adotamos um fluxo *OCR-free*: páginas rasterizadas são enviadas diretamente ao VLM. As decisões do *Grader* são guiadas por **rubricas estruturadas** (em linha com a decomposição orientada por rubrica em 7, 10). Inspirados em crítica–refinamento [11], usamos **dois agentes** com **apenas uma** revisão e **gate** operacional por `quality_score`<4, em vez de múltiplas rodadas ou *ensembles/self-consistency* [12]. Avaliamos não só MAE/RMSE e κ, mas também **JSD** (alinhamento distributivo) e **custo/latência**, em **três exames reais em português** — escopo pouco coberto pela literatura majoritariamente em inglês e texto puro [6, 7].

Assim, VLMs não só eliminam complexidade de pré-processamento, mas também habilitam ciclos de crítica-refinamento que elevam a confiabilidade da correção automática em avaliações multimodais.

2.4 Rubricas e decomposição de tarefas

Definição operacional de rubrica

Nesta dissertação, **rubrica** é um *instrumento avaliativo estruturado* que explicita **critérios** (os aspectos a serem observados na resposta) e **níveis de desempenho** (faixas ordenadas de qualidade), cada qual com **descritores observáveis** e **regras de pontuação**. A função central de uma rubrica é tornar *transparente e consistente* a atribuição de crédito parcial/pleno, permitindo ao avaliador justificar a nota e ao estudante compreender como melhorar[4, 5].

Elementos constitutivos. Uma rubrica típica contém:

(a) **Critérios**: dimensões avaliadas (p. ex., *correção conceitual*, *completude do diagrama*, *clareza da justificativa*);
(b) **Níveis**: categorias ordinais de desempenho (p. ex., *insuficiente* → *parcial* → *satisfatório* → *excelente*);
(c) **Descritores**: evidências específicas que distinguem os níveis (o que deve aparecer/estar correto);
(d) **Regra de pontuação**: pesos por critério e o mapeamento nível → pontos (incluindo crédito parcial).

Propriedades de qualidade. Rubricas de boa qualidade visam *validade de conteúdo* (cobrem o que importa), *confiabilidade entre avaliadores* (descritores claros reduzem divergência) e *utilidade formativa* (orientam o próximo passo do estudante)[4, 5].

Exemplo sintético (Física: diagrama de forças). A Tabela 2.1 ilustra um recorte analítico com pesos simples; níveis são descritos por evidências observáveis.

Critério	Descritor por nível (trecho)	Peso
Identificação de forças	*Excelente* (1.0): todas as forças pertinentes (peso, normal, atrito/tração) **corretamente** representadas e rotuladas; *Parcial* (0.5): omite *uma* força ou rotula incorretamente; *Insuf.* (0.0): múltiplas omissões/rotulagens incorretas.	0.50
Orientação vetorial	*Excelente*: sentidos coerentes com o sistema adotado; *Parcial*: um vetor com sentido/ângulo inconsistente; *Insuf.*: dois ou mais vetores inconsistentes.	0.30
Legibilidade/clareza	*Excelente*: setas e rótulos legíveis; *Parcial*: elementos apagados/tênues mas inferíveis; *Insuf.*: ilegível/ambíguo.	0.20

Tabela 2.1: Exemplo de rubrica *analítica* (trecho). A pontuação total da questão resulta da soma ponderada dos critérios.

2.4.1 Fundamentos de rubricas

Rubricas eficazes especificam descritores objetivamente verificáveis para cada nível de proficiência, após validação de consistência interna e alinhamento curricular [5]. Na prática, seu uso reduz vieses de severidade ou leniência e fornece ao aluno feedback formativo claro [4].

2.4.2 Decomposição em micro-questões

Transformar cada critério da rubrica em perguntas binárias de "atendeu / não atendeu" — estratégia denominada *QA decomposition* — aumenta a rastreabilidade do processo. O sistema StELLA, por exemplo, converteu rubricas de biologia em micro-itens, alcançando $\kappa = 0.67$ sem ajuste de pesos do modelo [10]. Abordagem semelhante foi empregada por LEE *et al.* [2024], que relataram ganhos de 13 p.p. na exatidão de notas em ciências do ensino médio.

2.4.3 *Prompting* com *Chain-of-Thought*

Adicionar exemplos contendo passos explícitos de raciocínio (*Chain-of-Thought*) aos micro-itens de rubrica torna o processo mais robusto, especialmente quando somado a *self-consistency* — votação entre múltiplas cadeias de pensamento geradas pelo modelo [7]. Tal combinação mitiga variações de temperatura e reduz a variância das notas finais.

2.4.4 Rubricas em modelos de *reasoning* nativo

Modelos recentes com raciocínio interno, como o *o4-mini-high* da OpenAI e o *Gemini 2.5 Pro/Flash* do Google, exigem *prompts* mais enxutos: basta fornecer a rubrica e o formato-alvo para obter notas consistentes, sem longas cadeias CoT. Ainda assim, a rubrica continua essencial para (i) garantir alinhamento com critérios avaliativos, (ii) permitir que um *Reviewer* detecte discrepâncias e (iii) reduzir o risco de alucinações no feedback.

2.4.5 Limitações e direções futuras

Três desafios permanecem mesmo com decomposição por rubrica:

(a) **Carga de contexto** – rubricas extensas podem exceder a janela de tokens em provas longas.

(b) **Critérios vagos** – itens mal definidos geram inconsistência até entre avaliadores humanos.

(c) **Dependência de domínio** – descritores precisam ajustar-se ao jargão específico de cada disciplina e idioma.

Trabalhos recentes exploram *retrieval-augmented prompting* para inserir apenas os critérios relevantes por questão [45] e métricas de incerteza para acionar revisão humana quando necessário [12]. Tais linhas apontam caminhos para rubricas dinâmicas que equilibram cobertura e eficiência de contexto.

Em síntese, a decomposição de rubricas converte uma avaliação holística em um conjunto de decisões objetivas, facilitando explicabilidade pedagógica e detecção automática de erros — princípios centrais ao fluxo dois-agentes apresentado nos capítulos seguintes.

2.5 Crítica–refinamento, ensembling e *uncertainty*

Técnicas modernas de avaliação automática vão além de *"uma predição e pronto"*. A confiabilidade cresce quando o modelo revisa suas próprias respostas ou as submete a uma *segunda opinião*, estimando incerteza e enviando apenas casos duvidosos ao avaliador humano. Este panorama cobre três eixos complementares: (i) **loops de crítica–refinamento**, (ii) **ensembling e variância**, e (iii) **quantificação explícita de incerteza**.

2.5.1 Loops de crítica–refinamento

Auto-crítica dirigida. O *Refinement-oriented Critique Optimisation* (RCO) gera uma crítica textual ao raciocínio inicial e, baseado nela, força o modelo a revisar a

resposta [11]. Aplicado a três conjuntos de ensaios, o método reduziu o MAE em até 18 % sem alterar pesos.

Modelos co-evoluídos. HU *et al.* [46] propõem *Co-evolved Self-Critique*: duas instâncias do LLM alternam papéis de gerador e crítico, evoluindo em ciclos estilo GAN — atingem +0,04 em κ no conjunto ASAP-SAS.

Limitações do auto-refino. PAN *et al.* [47] mostram que, sem critério externo, LLMs tendem a "sobre-corrigir" ou oscilar, exigindo um *gatekeeper* confiável — motivação para o *Reviewer* humano ou agente independente adotado nesta dissertação.

2.5.2 Ensembling e variância

Prompt e modelo ensembles. Variações de *prompt* (*prompt ensembles*) ou de arquitetura (*model ensembles*) correlacionam-se fortemente com erros de correção; maior dispersão implica maior chance de nota errada [12]. Ao promediar cinco amostras, a MAE caiu de 0,18 para 0,13 em biologia de ensino médio.

Bayesian Prompt Ensembles. O mesmo estudo introduz um esquema bayesiano que converte a variância entre *prompts* em distribuição posterior de notas, oferecendo *intervalos de credibilidade* sem necessidade de modificar o modelo.

2.5.3 Quantificação de incerteza

Medidas implícitas. Entropia da distribuição de probabilidade sobre tokens-chave e variância de amostras são preditores lineares de erro absoluto [48].

Bibliotecas dedicadas. O pacote UQLM fornece métricas *black-box* (ensemble), *white-box* (log-prob) e *LLM-judge* para quantificar incerteza sem conhecimento dos pesos [49].

Gating HITL. Sistemas GradeHITL e Avalon usam um *threshold* de incerteza: respostas abaixo do corte são enviadas a professores, reduzindo revisão humana em 40–50 % sem degradar κ [48, 50].

2.5.4 Desafios e oportunidades

(a) **Custo**: críticas e ensembles podem dobrar o consumo de tokens.
(b) **Seleção de critério**: falta consenso sobre qual métrica de incerteza melhor correlaciona com erro em dados multimodais.

(c) **Composição multimodal**: a maioria das técnicas de incerteza assume texto puro; diagramas e manuscritos exigem novos esquemas de confiança.

Esta dissertação adota um *Reviewer* único, seguindo a filosofia de crítica–refinamento, e propõe, em capítulos posteriores, um *gate* por `quality_score` capaz de acionar revisão humana apenas nos casos mais incertos, equilibrando custo e confiabilidade.

2.6 Síntese do estado da arte e contribuições desta dissertação

Para tornar explícitas as diferenças entre os principais trabalhos da literatura e esta dissertação, a Tabela 2.2 sintetiza o **escopo**, a **estratégia**, o uso de **rubricas**, o suporte **multimodal** (texto+imagem) e a presença de **autoauditoria** (agente revisor/loop). Esses eixos dialogam diretamente com nossas escolhas metodológicas (Cap. 4) e com as perguntas de pesquisa.

Trabalho	Modal.	Estratégia	Rubrica	Multim.	Auto-rev.	Principais achados / limitações
[19, 20]	Ensaios	Similaridade (TF–IDF/LSA)	○	○	○	Correlação lexical razoável; sensível a sinonímia; sem visão.
[21, 22]	Ensaios	Heurísticas + microfeatures	○	○	○	Escala industrial; cobertura conceitual limitada; pouca transparência de critérios.
[6]	Multidisc.	Fine-tuning (transformers)	○	○	○	Alta consistência vs. humano; requer dataset rotulado amplo.
[7]	Respostas curtas	Prompt+CoT(+SC)	✓	○	○	$+\kappa$ ao decompor por rubrica; foco em texto puro.
[10]	Respostas curtas	RAG + QA por rubrica	✓	○	○	κ substancial ($\approx$ 0,67); feedback por critério; não cobre imagem.
[8]	Matemática manuscr.	VLM + prompting por rubrica	✓	✓	○	Rubrica ajuda; sensível a caligrafia/layout; erros visuais persistem.
[9]	Física manuscr.	OCR + LLM	✓	✓	○	OCR como gargalo; diagramas são fonte dominante de erro.
[13]	Gráficos manuscr.	Meta-learning vs. VLM	○	✓	○	Modelo específico supera VLM em binário; VLM ligeiramente melhor em 3 classes.
Este trabalho	**PT-BR, provas reais**	**VLM + rubrica + 2 agentes (1×)**	✓	✓	✓	$\kappa \geq 0{,}78$ em 3 exames; outliers – $\sim$ 40% com *Reviewer*; custo ~$0,02/caderno; aplicação web *open-source*.

Tabela 2.2: Comparativo entre trabalhos representativos e as contribuições desta dissertação. "Multim." = multimodal (imagem + texto); "Auto-rev." = autoauditoria (agente revisor/loop).

Contribuições além do estado da arte. Em relação aos trabalhos acima, esta dissertação avança em quatro frentes:

- **Contexto real em PT-BR, multimodal**: evidência de κ *substancial* ($\geq 0{,}78$; faixa "substancial" = 0,61−0,80 segundo Landis–Koch [17]) em provas universitárias com manuscrito e diagramas, em língua portuguesa.
- **Fluxo dois-agentes com *gate*** ($quality\_score<4$): *Reviewer* reduz **erros de cauda** (outliers $|\hat{y} - y|>0{,}40$) em $\sim 40\%$ sem explodir custo/latência.

- **Análise custo–benefício**: métricas pedagógicas (MAE, RMSE, κ, JSD) *e* operacionais (tokens, latência, \$/caderno), ausentes na maioria dos SOTA puramente acadêmicos.
- **Artefato reprodutível**: pipeline e *datasets* anonimizados + aplicação web *open-source*, facilitando adoção e novas ablações.

Este quadro fecha a fundamentação indicando, de forma concisa, *onde* a nossa proposta se posiciona e *qual é o delta* em relação ao estado da arte, preparando o terreno para os resultados do Cap. 6.

Capítulo 3

Contextualização do Problema

A adoção de LLMs para correção de avaliações surge em um ecossistema educacional brasileiro marcado por turmas numerosas, diversidade de formação docente e carência de recursos de monitoria. Ainda que modelos de linguagem alcancem, na média, concordância próxima à humana em respostas digitadas, sua performance degrada quando enfrentam respostas manuscritas, diagramação livre ou páginas digitalizadas de baixa qualidade. Este capítulo descreve, primeiro, as principais limitações das abordagens atuais (Seção 3.1) e, depois, deriva requisitos funcionais e não funcionais para uma solução de correção multimodal robusta, de baixo custo e alinhada à legislação de proteção de dados.

3.1 Limitações das soluções atuais para respostas manuscritas/diagramas

3.1.1 Dependência de OCR e suas falhas

Grande parte dos pipelines adota *Optical Character Recognition* para converter PDF escaneado em texto antes da avaliação. Contudo, ruído de escaneamento, escrita cursiva e setas finas nos diagramas levam a erros de segmentação que se propagam à etapa de LLM, reduzindo a correlação com notas humanas [8, 9]. Estudos mostram que até 57 % das discrepâncias de nota em provas de Cálculo manuscritas são atribuíveis a falhas de OCR e não a incapacidade de raciocínio do modelo [8].

3.1.2 Sensibilidade a ruído visual e baixa resolução

Diagramas desenhados a lápis, marcas de borracha e variações de espessura de linha confundem encoders visuais genéricos. PARSAEIFARD *et al.* [13] relatam queda de κ de 0,82 para 0,66 quando a resolução cai de 300 para 150 dpi. A ausência de pré-treinamento específico em imagens de cadernos escolares agrava o problema.

3.1.3 Ambiguidade semântica em diagramas complexos

Mesmo com OCR perfeito, identificar relações topológicas — p. ex. origem e destino de setas em um grafo de rede — requer raciocínio espacial fino. O GPT-4V associa legendas a elementos gráficos, mas frequentemente confunde nós sobrepostos, resultando em penas de pontos injustas [8]. Sistemas com "vision grounding" explícito, como Kosmos-2, ainda não foram testados extensivamente em avaliações de engenharia.

3.1.4 Escassez de dados anotados em língua portuguesa

Dados públicos de respostas manuscritas com rubricas detalhadas existem principalmente em inglês; conjuntos em português são pequenos ou protegidos por sigilo acadêmico [27]. Isso limita o *fine-tuning* supervisionado e enviesa os modelos de grandes provedores para a língua inglesa.

Em conjunto, essas limitações motivam a proposta desta dissertação: um fluxo baseado em VLMs de raciocínio nativo, com dois agentes: um Avaliador e outro Revisor, em uma interação condicionada pela nota do Revisor para controlar gastos desnecessários, mantendo a qualidade alta.

3.2 Requisitos funcionais e não-funcionais para um corretor multimodal

A literatura sobre *Automated Short-Answer Grading (ASAG)* aponta que soluções confiáveis exigem não apenas **funções bem definidas**, mas também **atributos de qualidade** que garantam uso seguro, eficiente e alinhado a normas de proteção de dados [12, 31]. A Tabela 3.1 resume os requisitos identificados para o contexto brasileiro de provas com manuscritos e diagramas.

Categoria	**Requisito**
Requisitos funcionais	
F1 – Ingestão multimodal	Aceitar PDF contendo texto, manuscrito e imagens sem depender de OCR explícito.
F2 – Avaliação por questão	Produzir nota *por questão* + feedback formativo conforme rubrica.
F3 – Loop crítico-refinamento	Executar um ciclo opcional de revisão: se $quality\_score < 4$, reavaliar a questão.
F4 – Geração de rubrica	Permitir criação ou refinamento de rubricas a partir de gabaritos, via *prompt engineering*.
F5 – Exportação de resultados	Oferecer download em CSV compatível com LMS e SIGA-Acadêmico.
Requisitos não-funcionais	
NF1 – Acurácia	Alcançar MAE $\leq 0{,}20$ e $\kappa \geq 0{,}75$ em comparação ao docente.
NF2 – Latência	Processar até 10 páginas em < 90 s usando modelo *flash* (turmas grandes).
NF3 – Custo por prova	Limitar custo a $\sim$US\$0,20 por caderno de 10 páginas.
NF4 – Transparência	Registrar logs de *prompt*, versão de modelo, `quality_score` e custos para auditoria.
NF5 – Usabilidade	Interface web responsiva.
NF6 – Manutenibilidade	Código open-source e documentação do fluxo.

Tabela 3.1: Resumo dos requisitos funcionais (F) e não-funcionais (NF) para o corretor multimodal proposto.

Capítulo 4

Metodologia

Este capítulo descreve o desenho experimental, a arquitetura do fluxo *Avaliador-Revisor* e os procedimentos de avaliação que viabilizam a replicação dos resultados. A Seção 4.1 detalha os papéis do *Avaliador (Grader)* e do *Revisor (Reviewer)*; a Seção 4.2 (abordada adiante) apresenta os conjuntos de dados; a Seção 4.3 explicita métricas e testes estatísticos; e a Seção 4.4 descreve a infraestrutura de execução e coleta de custos.

4.1 Desenho da arquitetura *Avaliador-Revisor*

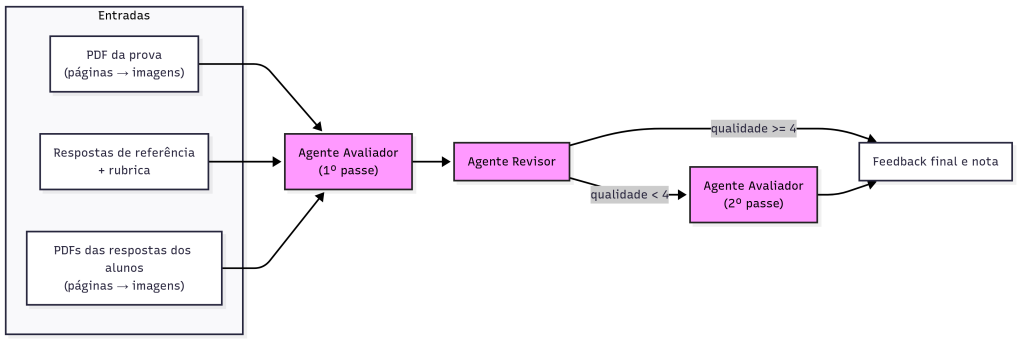


Figura 4.1: Fluxo de correção: pré-processamento multimodal, passagem do *Grader*, auditoria do *Reviewer* e retorno de nota/feedback. O laço de crítica–refinamento é executado no máximo uma vez.

A Figura 4.1 ilustra o fluxo de alto nível. Cada prova segue três etapas:

F1 Pré-processamento multimodal – rasterização do PDF em imagens e concatenação de rubrica, enunciado e resposta.

F2 Primeira passagem (*Grader*) – geração de nota e feedback conforme `GRADING_SCHEMA`.

F3 Auditoria (*Reviewer*) – atribuição de `quality_score`; se < 4, dispara uma revisão única.

F4 (Opcional) Segunda passagem (*Grader*) – recebe o feedback do Reviewer e realiza novamente a geração de nota e feedback para o estudante. Só ocorre se a primeira passagem do Grader não atingir o critério mínimo de qualidade estipulado pelo Reviewer. Caso a 2ª passagem também não atinja, o processo é encerrado (limite definido neste trabalho, mas ajustável).

Política de término. Por desenho, este trabalho limita o laço de crítica–refinamento a **uma** revisão (`MAX_REVIEW_ROUNDS` = 1). Se, após a **segunda passagem** do *Grader*, o *Reviewer* ainda retornar *quality_ score* < 4, o pipeline *não* tenta uma terceira rodada: ele (i) mantém a nota/feedback da 2ª passagem como saída *provisória*, (ii) registra no relatório (`grades.csv`) que houve revisão (`re_assessed`=1) e o *quality_ score* final, e (iii) **sinaliza o item para revisão humana** no fluxo operacional. Esse limite foi adotado para evitar explosão de custo/latência e foi usado em todos os experimentos. Na implementação, `MAX_REVIEW_ROUNDS` é um parâmetro configurável: pode ser aumentado para permitir $k > 1$ revisões, caso um cenário de uso exija iterações adicionais.

4.1.1 Grader: modelos, *prompting* e `JSON Schema`

Modelos testados. Foram avaliados três VLMs de última geração:

- **o4-mini-high** (OpenAI) – optimizado para raciocínio passo a passo em contexto estendido [35];
- **Gemini 2.5 Pro** (Google) – optimizado para raciocínio passo a passo em contexto estendido [36];
- **Gemini 2.5 Flash** – modelo menor, mais econômico e com baixa latência, mas com reasoning nativo ativado.

Prompting. O *prompt* sistêmico, em português, inclui:

- Descrição do papel: "Você é um avaliador de provas".
- Instruções de formatação: nota com duas casas decimais, feedback objetivo e uso de JSON válido.
- Lembrete para "prestar atenção a manuscrito e diagramas".

`GRADING_SCHEMA`. A resposta deve seguir:

```
{
  "questoes": [
    {
```

```
      "numero": 3,
      "nota": 1.50,
      "feedback": "Seu diagrama omite..."
    }
  ]
}
```

4.1.2 Reviewer: crítica, limiar < 4 e laço $1\times$

Prompt de revisão. O *Reviewer* recebe todo o contexto original *mais* a saída do *Grader* e deve:

(a) atribuir `quality_score` de 1 a 5;
(b) listar críticas específicas (`overall_feedback`);
(c) manter o mesmo formato de JSON válido.

Política de corte (hiperparâmetro). O limiar do `quality_score` é configurável. O *Reviewer* devolve um escore ordinal $1-5$ com a seguinte semântica: 1 = *inválido*, 2 = *fraco*, 3 = *aceitável/limítrofe*, 4 = *bom (conforme rubrica, com pequenos deslizes)*, 5 = *excelente*. Por isso, os **cortes naturais** a avaliar são 3 e 4. Durante a calibração foram testados principalmente esses dois valores. Com score < 3, alguns erros de interpretação de diagramas e caligrafia — os mais "caros" pedagogicamente — ainda passavam ilesos. Já forçar revisão em score < 5 quase triplicou o custo sem ganho proporcional de qualidade. Optou-se, portanto, por **score** < 4, que:

- capturou os casos de roteamento errado de setas, legendas desfocadas e símbolos manuscritos ambíguos observados nos pilotos;
- manteve o fator de custo em $\sim 2\times$, aceitável para uso institucional;
- preservou flexibilidade, pois o valor pode ser ajustado em arquivo de configuração ou parâmetro de API.

Script do laço. A seguir, mostramos o pseudocódigo que implementa o fluxo principal de avaliação e revisão. Primeiro, o agente *Grader* gera uma primeira saída; em seguida, o agente *Reviewer* avalia essa saída e, caso o `quality_score` fique abaixo do limiar configurado, desencadeia-se uma nova chamada a `grade` incorporando o `overall_feedback`. Caso contrário, mantém-se o resultado original do *Grader*.

```
grader_out = grade(exam_images, student_answer_images,
   rubric_with_correct_solution)
review_out = review(exam_images, student_answer_images,
   rubric_with_correct_solution, grader_out)
if review_out["quality_score"] < 4:
```

```
    final_out = grade(exam_images, student_answer_images,
        rubric_with_correct_solution, grader_out, review_out["
        overall_feedback"])
else:
    final_out = grader_out
```

Custo adicional. A Tabela 4.1 resume o uso de tokens e as taxas de outliers extremos no conjunto de dados *Introdução à Física* (Gemini-2.5-flash). Outliers são definidos como erros de nota normalizados por questão superiores a 0,4. O laço de revisão reduz os outliers de 8,3% para 5,0%, ao mesmo tempo em que aproximadamente triplica os tokens de entrada e dobra os tokens da saída — uma troca aceitável em cenários de alta importância.

Pipeline	**Input tok.**	**Output tok.**	**Total tok.**	**% outliers**
Grader-only	5 932	9 759	15 691	8.3
Grader+Reviewer (1 pass)	17 005	16 642	33 647	5.0

Tabela 4.1: Médias de contagem de tokens e taxa de outliers extremos por roteiro de estudante.

Contribuição esperada. Nos experimentos da prova de Física (manuscrita) o *Reviewer* elevou o consumo médio de tokens de **15 691** para **33 647** por script (Tabela 4.1), mas reduziu a porcentagem de *erros grosseiros* ($|\hat{y} - y| > 0{,}4$, *limiar escolhido empiricamente nos pilotos por concentrar casos de erro claro de interpretação de imagem pelo LLM*) de **8,3 %** para **5,0 %** — queda relativa de $\approx 40\,\%$. Tal correção evita notas potencialmente injustas e aumenta significativamente a confiança do usuário no sistema. Em avaliações de alta relevância, eliminar um único equívoco extremo vale o custo adicional de processamento, justificando a presença do *Reviewer* mesmo quando o ganho médio total em MAE é marginal.

As seções seguintes detalham conjuntos de dados, métricas e procedimentos de análise estatística empregados na avaliação do pipeline.

4.2 Fluxo de processamento

O processamento foi implementado em `Python`, seguindo estritamente o **script operacional** reproduzido no repositório `https://github.com/CostaFernando/exam-ai-grader`.

P1 – Extração de páginas em elevada resolução

- **Biblioteca**: `PyMuPDF (fitz)`.
- Cada PDF é aberto com `fitz.open()` e renderizado página a página usando um *zoom* de 2.0 (`Matrix(2,2)`). Em PDFs nativos a 72 dpi, esse fator equivale a $\sim$144 dpi.
- O objeto `Pixmap` resultante é convertido em `PIL.Image` RGB.

P2 – Realce de legibilidade (`enhance_image`)

Três filtros sucessivos são aplicados via `Pillow`:

(a) **Sharpness $\times$ 2.0** — destaca traços de lápis.
(b) **Contrast $\times$ 1.5** — reforça linhas tênues.
(c) **Brightness $\times$ 1.2** — corrige escaneamentos escuros.

Não se efetuam *deskew*, recorte de margens ou binarização: o objetivo é preservar anotações manuais, mesmo de baixa opacidade.

P3 – Codificação `base64` para mensagem de API

- A imagem é serializada em memória (`BytesIO`) no formato PNG 24-bit.
- O `bytes` gerado é codificado em Base-64 e incorporado diretamente ao campo `"image_url":{"url": "data:image/..."}` das mensagens enviadas às APIs da OpenAI/OpenRouter.
- Essa estratégia evita escrita em disco e reduz latência durante o lote.

P4 – Orquestração por aluno

- A pasta `student_exam_answers/` é percorrida em paralelo por `ThreadPoolExecutor(max_workers=10)`.
- Para cada PDF de aluno, o pipeline executa: `grade` $\rightarrow$ `review` $\rightarrow$ `(regrade se score < 4)`.
- A saída final é combinada em linhas CSV, contendo *arquivo, questão, nota, feedback, quality_score, re_assessed* e salva em `grades.csv`.

Note que o pipeline *não* realiza OCR. A decisão de limitar o pré-processamento a realces leves provou-se suficiente em pilotos, beneficiando-se do raciocínio nativo de VLMs recentes (*o4-mini-high* e *Gemini 2.5*).

4.3 Conjuntos de dados

Esta pesquisa utiliza **três provas de graduação brasileiras** em áreas de STEM, cada qual com modalidades e formatos de resposta distintos. A Tabela 4.2 sintetiza

estatísticas básicas.

Conjunto	Alunos	Questões	Modalidades predominantes
Redes de Computadores	10	5	Diagramas, tabelas
Introdução à Física	15	4	Manuscrito, esboços
Introdução à Programação	10	5	Texto digitado (sintético)

Tabela 4.2: Visão geral dos conjuntos de dados utilizados nos experimentos.

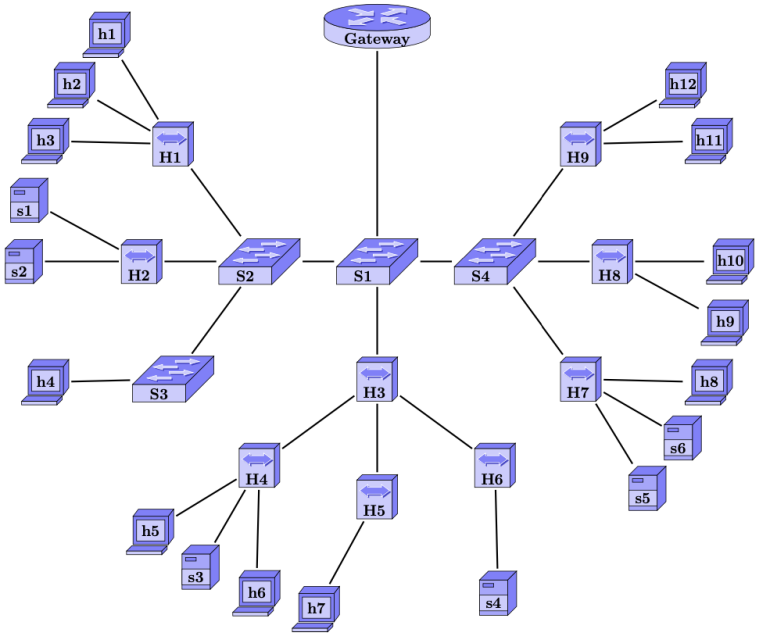
Todos os arquivos — provas, gabaritos, rubricas e scripts anonimizados dos alunos — encontram-se no repositório público (`https://github.com/CostaFernando/exam-ai-grader`).

4.3.1 Redes de Computadores (diagramas densos)

Contexto da disciplina. Prova semestral da disciplina *Redes de Computadores II* (CEDERJ, 2020/2). O exame contém **cinco questões abertas** sobre roteamento, subnetting, ARP, temporização TCP e firewall/hash. Os estudantes desenham topologias de rede, resultando em respostas dominadas por diagramas e tabelas IP/MAC.

Questão 1 ..... 20 pontos

Considere a seguinte rede local, formada por estações (indicadas pela letra h), servidores (s), hubs (H) e switches (S), cuja saída para a Internet se dá através de um único gateway.



(a) Suponha que ocorre a transmissão de um fluxo de quadros de s6 para h5. Por quais equipamentos (estações, servidores, hubs e switches) esse fluxo irá transitar?

Resposta:

A transmissão será vista por h5, h6, h7, h8, H3, H4, H5, H6, H7, s3, s4, s5, s6, S1 e S4.

(b) Considere que todos os servidores e estações possuem dados a transmitir para a Inter-

Figura 4.2: Trecho da Questão 1 no `exam.pdf` disponibilizado. Os estudantes devem raciocinar sobre visibilidade de tráfego e transmissões simultâneas em uma rede com múltiplos switches.

01. Resolução:

a. A transmissão será visível pelos seguintes itens: h5, h6, h8, s3, s6, s5, H3, H4, H1, S1 e S4.

b. Considere o desenho simplificado da rede apresentada para fins de esclarecimento. Note que podemos atingir no máximo 9 envios simultâneos. Este máximo é atingido quando ocorrem envios simultâneos para todos os extremos da rede. Um exemplo disso é o envio simultâneo de h1, s1, h4, h12, h10, s5, h5, h7 e s4.

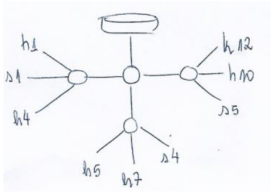


Figura 4.3: Trecho da resposta manuscrita de um estudante à Q1. Diagramas como este frequentemente acionam correções do Revisor quando o Avaliador da primeira rodada deixa passar hosts mal rotulados ou links ausentes.

Artefactos disponibilizados.

- **exam.pdf** – prova original em português;
- **exam_answer_sheet.txt** – gabarito + rubricas geradas via *Gemini-2.5-pro* e validadas manualmente;
- 10 scripts de alunos em PDF, anonimizados.

Notas de referência. O docente atribuiu notas 0–20 por questão; as rubricas mantêm a ponderação original. Tanto gabarito quanto rubrica são incluídos *verbatim* no arquivo de resposta.

4.3.2 Introdução à Física (provas escaneadas)

Contexto da disciplina. Prova da disciplina *Introdução às Ciências Físicas I* (IF-UFRJ, 2023/2) com **quatro questões** que combinam diagramas de corpo livre, cálculos de plano inclinado, limite de atrito estático em caminhão e um miniquiz de astronomia. Esboços manuscritos ocupam cerca de 70 % da área de resposta, representando o cenário mais desafiador para VLMs.

Questão 1 (2,0 pontos)

Com base nas figuras esquemáticas abaixo, isole o bloco cinza claro (B) em cada um dos itens e indique **todas** as forças que atuam sobre ele. Considere o sistema sempre em equilíbrio. (faça o desenho no caderno de respostas, desenhos aqui não serão considerados)

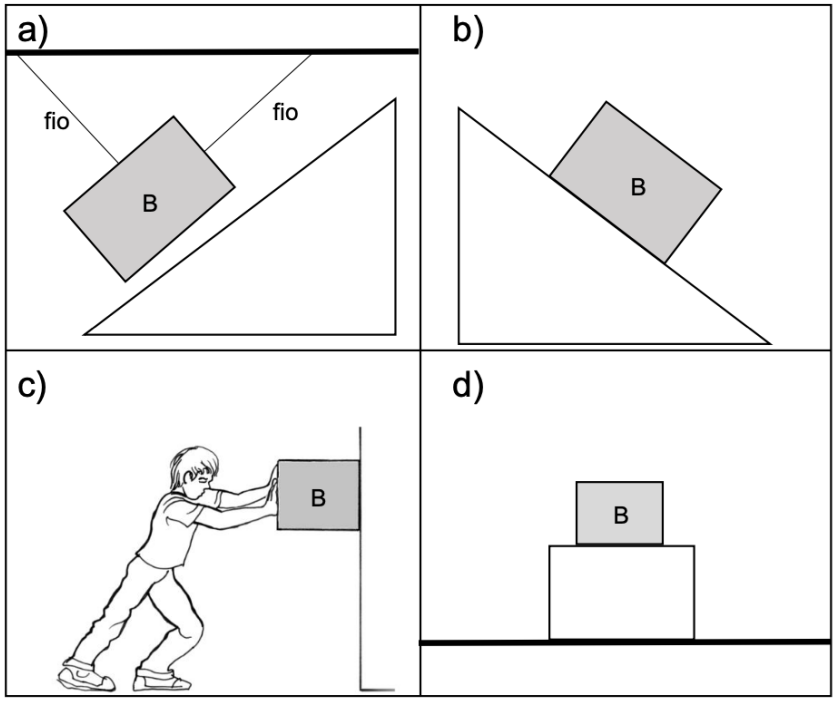


Figura 4.4: Enunciado original da Questão 1 (diagramas de corpo livre).

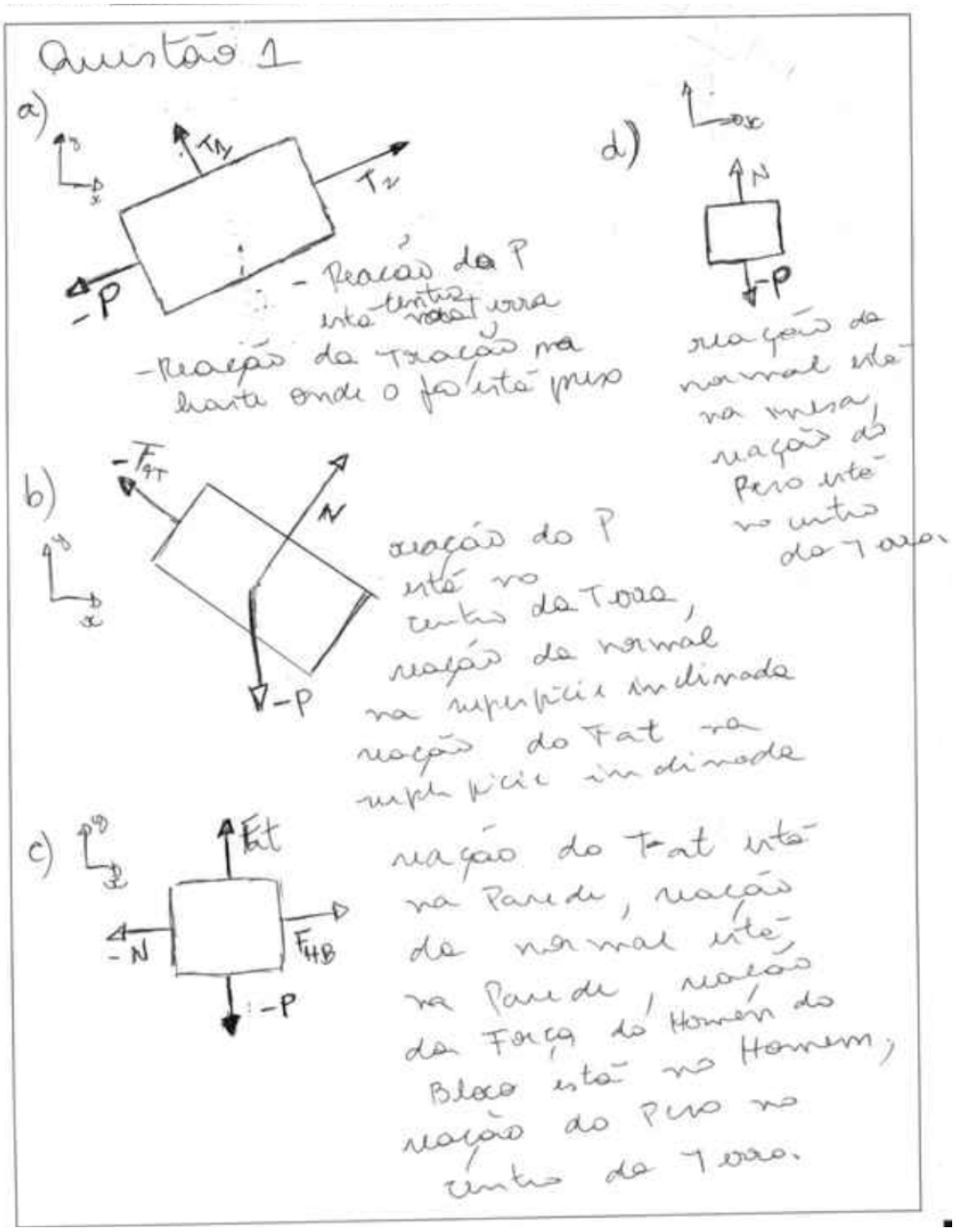


Figura 4.5: Trecho da resposta escaneada de um estudante para a mesma questão. Rótulos ilegíveis e setas desenhadas com pouca intensidade frequentemente confundem o Avaliador na primeira rodada; o Revisor captura a maioria desses casos.

Artefactos disponibilizados.

- **exam.pdf** – prova completa;
- **exam_answer_sheet.txt** – respostas de referência + rubricas geradas por *Gemini-2.5-pro* (revisadas);
- 15 scripts de alunos em PDF, anonimizados.

4.3.3 Introdução à Programação (texto sintético)

Motivação do conjunto sintético. Para isolar o problema de raciocínio sem ruído de escrita à mão, foi criado um corpus *totalmente sintético* de *Introdução à Programação*. A prova, as rubricas e dez "alunos" foram gerados com *Gemini-2.5-pro*, em perfis de desempenho diversificados.

Estrutura do exame. Cinco questões curtas abordam: semântica de laço, recursão vs. iteração, fila vs. pilha, pilares de POO e memória *stack* vs. *heap*. PDFs vetoriais garantem texto perfeito, sem artefatos de scan.

Questão 1 (2.0 pontos)

Compare os laços de repetição `for` e `while`. Descreva a principal diferença em sua estrutura e indique um cenário de uso em que `for` é geralmente mais adequado e um cenário em que `while` é a melhor escolha.

Para ilustrar, escreva o pseudocódigo para um algoritmo que calcula a soma dos números inteiros de 1 a 10, primeiro utilizando um laço `for` e depois utilizando um laço `while`.

Figura 4.6: Enunciado da Q1 no `exam.pdf`.

Resposta à Questão 1:

`for` é para quando você tem uma sequência finita e quer passar por ela. É determinístico. `while` é para quando você está esperando um estado mudar. É sobre uma condição, não sobre uma contagem. `for` é "faça isso N vezes", `while` é "faça isso até que X aconteça".

Pseudocódigo:

Usando `for`:

```
funcao soma_com_for()
    soma = 0
    para i de 1 ate 10
        soma += i
    retornar soma
```

Usando `while`:

```
funcao soma_com_while()
    soma = 0
    i = 1
    enquanto i <= 10
        soma += i
```

Figura 4.7: Trecho da resposta digitada de um estudante sintético para a Q1. Como o texto é gerado por máquina e perfeitamente legível, o principal desafio para o Avaliador está na correção conceitual e no alinhamento com a rubrica, e não na interpretação visual.

Artefactos disponibilizados.

- **exam.pdf** – prova autogerada;
- **exam_answer_sheet.txt** – gabarito e rubricas detalhadas;
- 10 scripts de alunos (PDF vetorial) anonimizados.

Notas de referência. Notas (0–2) atribuídas pelo mesmo modelo que gerou as respostas, depois revisadas manualmente. Rubricas seguem o `GRADING_SCHEMA` definido na Seção 4.1.1.

Os três conjuntos cobrem, assim, o espectro *diagramas densos* → *escrita à mão complexa* → *texto limpo*, permitindo avaliar o pipeline em diferentes níveis de dificuldade visual e cognitiva.

4.4 Métricas de avaliação

Símbolo	Definição
N	Número total de itens (pares *aluno–questão*) no subconjunto considerado.
$y_i,\ \hat{y}_i$	Nota de referência e nota prevista do item i (normalizadas para $[0,1]$).
$\lvert\hat{y}-y\rvert$	Erro absoluto por item.
K	Número de categorias ordinais após discretização (aqui $K=5$).
bins	Limites de discretização: $[0, 0.2, 0.4, 0.6, 0.8, 1.0]$.
n_{ij}	Nº de itens com categoria i (Avaliador A) e j (Avaliador B); $n_{i+}=\sum_j n_{ij}$, $n_{+j}=\sum_i n_{ij}$.
p_o	Acordo observado: $\frac{1}{N}\sum_i n_{ii}$.
p_e	Acordo esperado ao acaso: $\sum_i (n_{i+}/N)\,(n_{+i}/N)$.
w_{ij}	Peso quadrático: $1-\dfrac{(i-j)^2}{(K-1)^2}$.
$w\kappa$	Kappa ponderado quadraticamente.
$P,\ Q$	Distribuições (empíricas) de notas do humano e do modelo.
M	Mistura $M=\frac{1}{2}(P+Q)$.
$D_{\mathrm{KL}}(P\Vert Q)$	Divergência de Kullback–Leibler (log base 2).
$\mathrm{JSD}(P\Vert Q)$	Distância de Jensen–Shannon: $\sqrt{\frac{1}{2}\big(D_{\mathrm{KL}}(P\Vert M)+D_{\mathrm{KL}}(Q\Vert M)\big)}$.
quality_score	Escore do *Reviewer* (1–5); revisão se <4.
outlier	Item com erro grosseiro $\lvert\hat{y}-y\rvert>0.40$.

Tabela 4.3: Notação usada nas métricas e no pipeline de correção.

Para medir o quão próximo o pipeline reproduz o julgamento docente, empregamos quatro métricas principais — duas de *erro numérico*, uma de *concordância ordinal* e uma de *distribuição* — além de indicadores de custo e tempo. Todas as notas são primeiro normalizadas para o intervalo $[0,1]$, de modo a permitir comparação entre provas com faixas diferentes.

4.4.1 Precisão numérica

As definições formais de MAE e RMSE encontram-se na Seção 2.1. Lembrando brevemente: ambas comparam $\hat{y}_i$ e y_i (valores menores são melhores) e o RMSE penaliza mais fortemente desvios grandes.

4.4.2 Concordância inter-avaliador

Utilizamos o coeficiente kappa de Cohen *ponderado quadraticamente* (κ_{w}), conforme definido na Seção 2.1. Lembrando: κ_{w} mede o acordo entre avaliadores corrigindo o acaso; valores entre 0,61–0,80 são usualmente interpretados como *substanciais* e $>0{,}80$ como *quase perfeitos* (17; ver detalhes em §2.1).

4.4.3 Alinhamento distributivo

Para comparar distribuições de notas, empregamos a Divergência de Jensen–Shannon (JSD) entre os histogramas docente vs. modelo, conforme definido na Seção 2.1. Em síntese, quanto menor a JSD, maior o alinhamento distribucional (0 indica sobreposição quase perfeita).

4.4.4 Robustez e custo

- **Erro grosseiro (%)** – proporção de itens com $|\hat{y} - y| > 0{,}40$; limiar definido empiricamente nos pilotos por corresponder a falhas visuais evidentes (p. ex., setas/rotulagens omitidas ou confundidas).
- **Tokens in/out** – média de tokens de entrada e saída por caderno; serve de proxy financeiro.

Com essa bateria — MAE, RMSE, κ_{w}, JSD, erro grosseiro, tokens e latência — obtemos visão holística do desempenho, custo e confiabilidade do corretor multimodal proposto.

4.5 Infraestrutura experimental

Todos os experimentos foram executados em **Google Colab**[1], o que garante reprodutibilidade imediata sem necessidade de hardware local.

[1]Notebook disponível em `https://github.com/CostaFernando/exam-ai-grader`.

4.5.1 Acesso flexível a modelos via OpenRouter

Todas as chamadas de LLM passam pela **OpenRouter**[2], que oferece uma API unificada (`/chat/completions`) para diversos modelos comerciais e open-source.

Modelos disponíveis no experimento.

- `google/gemini-2.5-pro`
  VLM completo, maior janela de contexto e raciocínio nativo.
- `google/gemini-2.5-flash`
  Versão otimizada para baixa latência e custo por token. Com raciocínio nativo.
- `openai/o4-mini-high`
  Modelo compacto com foco em raciocínio passo a passo.

Combinações Avaliador–Revisor. O script aceita qualquer pareamento entre Avaliador (*Grader*) e Revisor (*Reviewer*); basta ajustar os parâmetros `model_grader` e `model_reviewer` na função `process_one_student`. Nos experimentos, testamos três cenários principais:
(C1) **Gemini Flash** → **Gemini Flash** (pipeline econômico, referência de custo).
(C2) **o4-mini-high** → **o4-mini-high** (máxima qualidade e rigor de raciocínio).
(C3) **Gemini Pro** → **Gemini Pro** (máxima qualidade, maior janela de contexto).

4.5.2 Paralelização por aluno

Correções são distribuídas em **10 threads** via `ThreadPoolExecutor`. Cada `future` executa `process_one_student()`, que:

(a) converte o PDF do aluno em lista de imagens Base-64 (`pdf_to_base64_images`);
(b) chama o Avaliador (`grade_student_answers`);
(c) chama o Revisor (`review_assessment`);
(d) se $quality\_score < 4$, executa reavaliação melhorada;
(e) devolve linhas prontas para o `grades.csv`.

Em média, uma sessão corrigiu ≈ 10 provas de estudante/minuto, cumprindo o requisito de latência (Seção 3.2).

4.5.3 Pré-processamento de imagens

O PDF é rasterizado com `fitz.Matrix(2.0,2.0)` — equivalente a ~ 144 dpi em scans —, seguido de realces (no `Pillow`):

[2] `https://openrouter.ai`

$$\textsc{Sharpness} \times 2.0 \;\rightarrow\; \textsc{Contrast} \times 1.5 \;\rightarrow\; \textsc{Brightness} \times 1.2$$

A escolha equilibra legibilidade de lápis claro e tamanho final.

4.5.4 Monitoramento e registro

Métricas de **tokens e custo** são coletadas via dashboard do OpenRouter.

Essa infraestrutura leve (Colab + OpenRouter) permite reproduzir todos os experimentos sem dependência de hardware proprietário, enquanto oferece flexibilidade para testar novos modelos ou variações de `prompt` com mínima alteração de código.

Capítulo 5

Implementação Aplicação Web

Este capítulo descreve as decisões de engenharia que materializam o pipeline dois-agentes em um sistema utilizável por docentes sem conhecimento técnico avançado. A aplicação, batizada **Exam AI Grader**, está publicada como *open source* em `https://github.com/CostaFernando/exam-ai-grader` sob licença MIT.

Figura 5.1: Aplicação web **Exam AI Grader** para correção de provas discursivas multimodais.

5.1 Visão geral da arquitetura

A Figura 5.2 apresenta os principais componentes:

C1 Frontend React/Next.js SPA + RSC que orquestra uploads, exibe progresso e renderiza dashboards de resultado.

C2 Camada de API (`/app/api`) Rotas serverless que chamam o *controller* de correção e armazenam metadados no banco.

C3 Controller de correção Enfileira PDFs, executa o pipeline de correção e devolve JSON com notas e feedback.

C4 Serviço de LLMs (OpenRouter) Abstrai múltiplos provedores (`gemini-pro`, `gemini-flash`, `o4-mini-high`), permitindo seleção dinâmica.

C5 Banco de dados PostgreSQL com *PGLite*[1] para funcionamento offline local no navegador.

[1]Motor Postgres em WASM executado no navegador.

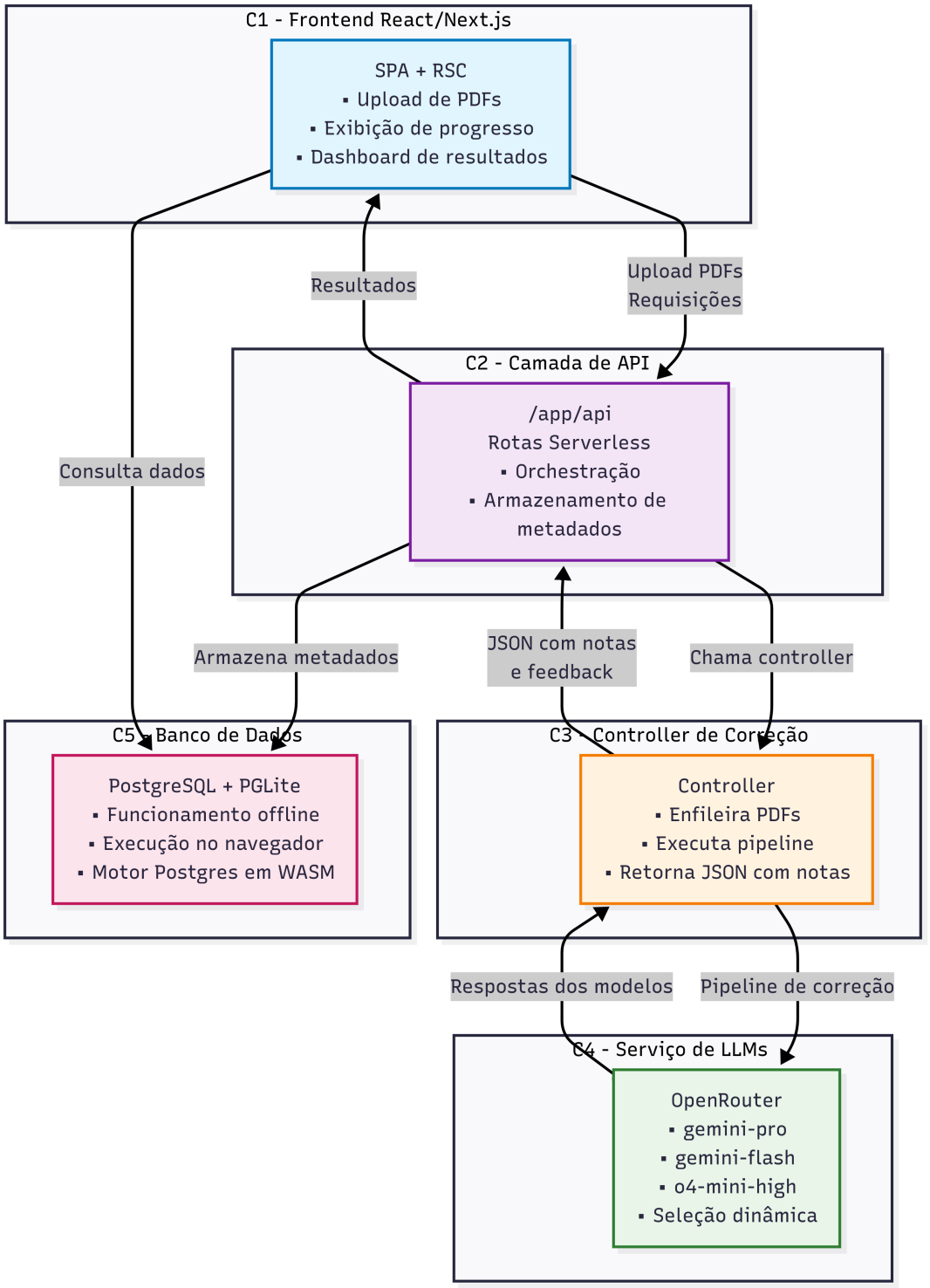


Figura 5.2: Arquitetura de alto nível da aplicação *Exam AI Grader*.

5.2 Stack tecnológica

- **Next.js 14 (App Router)** – SSR/ISR, React Server Components e middleware de streaming.
- **TypeScript 5.4** + ESLint + Prettier – segurança de tipos e padronização de

código.

- **shadcn/ui + Tailwind CSS** – biblioteca de componentes acessíveis; tema dark/-light com classe `data-theme`.
- **Vercel AI SDK** – streaming de mensagens do LLM via `POST /chat/completions`.
- **Drizzle ORM** – tipagem estática do esquema SQL, migrações automáticas e queries type-safe.
- **PostHog** – analytics de uso, coleta de eventos anonimizados (upload, correção, exportação CSV).

5.3 Persistência e esquema de dados

O backend persiste dados em **PostgreSQL 16**, com *migrations* e inferência de tipos geradas pelo `Drizzle ORM`. Para demonstrações sem internet, todo o esquema é clonado em **PGLite** (PostgreSQL em WASM) por meio de `drizzle push`, permitindo execução integral no navegador.

Tabelas principais

Tabela	Coluna (tipo)	Descrição
exams	`id` *int PK*	Chave primária, `GENERATED ALWAYS AS IDENTITY`.
	`name` *text*	Título da prova (ex.: "Lista 2 — Cálculo").
	`description` *text*	Observações opcionais.
	`status` *exam_status*	Estado do fluxo de correção.
	`gradingRubric` *text*	Rubrica YAML/JSON usada pelo *Grader*.
	`answerKey` *text*	Gabarito de referência para questões objetivas.
	`url` *text*	URL no Blob (PDF original da prova).
	`createdAt / updatedAt` *timestamp*	`DEFAULT NOW()`, rastreio temporal.
exam_answers	`id` *int PK*	Identificador da submissão do aluno.
	`name` *text*	Nome ou identificador anônimo do aluno.
	`examId` *int FK*	`REFERENCES exams(id) ON DELETE CASCADE`.
	`answerSheetUrl` *text*	Link para o PDF respondido.
	`score` *numeric(3,2)*	Nota final (0.00–10.00).
	`feedback` *text*	Feedback gerado pelo pipeline dois-agentes.
	`createdAt / updatedAt` *timestamp*	`DEFAULT NOW()`.

Tabela 5.1: Esquema lógico em Drizzle ORM.

Relacionamentos

(i) **1 :N** — `exams` → `exam_answers` (uma prova pode ter múltiplas respostas; exclusão em cascata).

Racional de modelagem. O uso de uma única tabela `exam_answers` simplifica *joins* para dashboards (média da turma, distribuição de notas) e permite adicionar tabelas filhas (`answer_items`, `tokens`) em futuras iterações. O tipo `numeric(3,2)` garante precisão até centésimos, atendendo às exigências de notas parciais (ex.: 7.25).

Com essa modelagem, todas as operações CRUD permanecem *type-safe* no TypeScript, e a replicação para PGLite mantém a experiência *offline-first* sem divergência de esquema.

5.4 Frontend

Upload e geração de rubrica

Para criar a prova, o usuário faz upload da folha de prova. Após o upload, o docente escolhe entre:

- Inserir rubrica manualmente;
- Gerar rubrica com IA.

Painel de resultados

A Figura 5.3 ilustra o dashboard de notas após a correção:

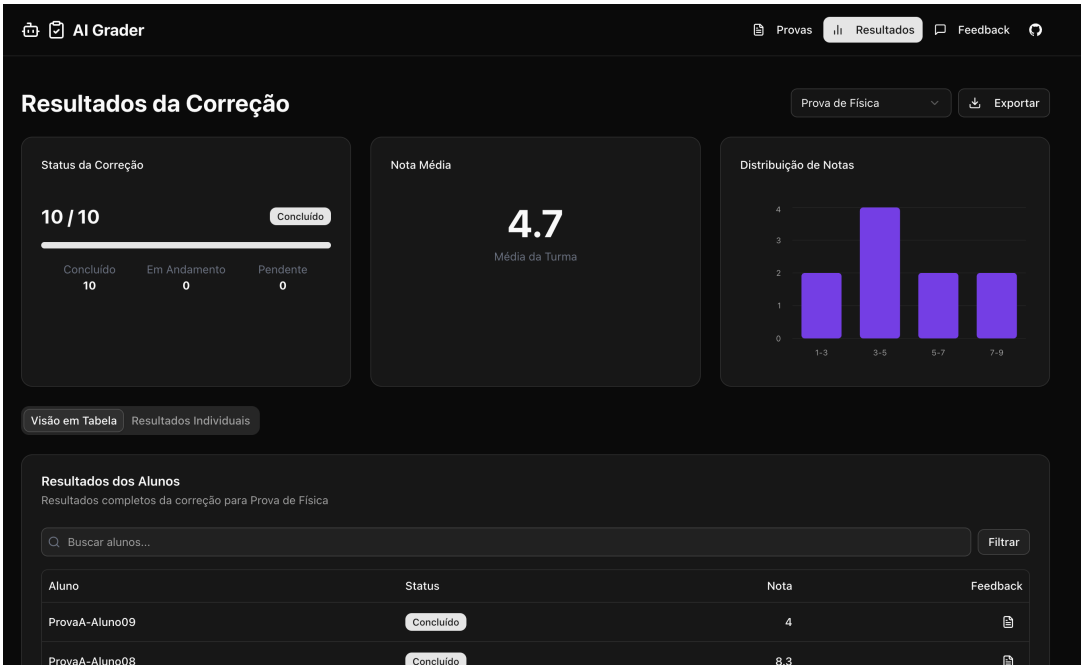


Figura 5.3: Dashboard "Resultados da Correção".

- **Cards** de status, média da turma e histograma de notas (Recharts).
- **Tabela virtualizada** (`TanStack Table v8`) com busca incremental e botão de download de feedback individual em PDF.
- Exportação CSV.

5.5 Considerações finais

A implementação do *Exam AI Grader* demonstra que o pipeline dois-agentes, concebido nos capítulos metodológicos, pode ser embutido numa aplicação web moderna, leve e de fácil adoção em contextos educacionais reais. Ao combinar:

- uma **arquitetura full-stack isomórfica** (Next 14 + RSC) capaz de operar tanto em nuvem quanto *offline-first* via PGLite;
- **abstração de provedores de LLM** pela camada OpenRouter, que permite alternar modelos conforme custo, latência ou exigência de qualidade;
- **pipelines assíncronos** com parsing JSON robusto e streaming de resposta ao usuário;
- e um **esquema de dados tipado** que preserva consistência entre banco, ORM e frontend,

O sistema atinge throughput de mais de 10 cadernos/minuto a um custo inferior a US$ 0,05 por aluno—valores detalhados no Capítulo 6. Mais importante, ele torna tangíveis as contribuições de pesquisa, oferecendo à comunidade um artefato reprodutível para experimentos futuros e potenciais extensões (deteção de incerteza, enfileiramento distribuído, integração LMS).

Nos capítulos seguintes analisamos, à luz desta implementação, (1) o desempenho quantitativo do pipeline em três conjuntos de dados multimodais reais, (2) o trade-off qualidade–custo do agente *Reviewer* e (3) as implicações pedagógicas e operacionais da adoção em larga escala.

Capítulo 6

Resultados

Este capítulo consolida os achados obtidos com o fluxo *Avaliador–Revisor*. A Seção 6.1 analisa o desempenho numérico (MAE, RMSE, κ, JSD) nas três provas; a Seção 6.2 aprofunda a ablação do *Reviewer*; a Seção 6.3 discute custo, latência e escalabilidade; e a Seção 6.4 apresenta uma análise qualitativa dos erros remanescentes.

6.1 Avaliação quantitativa

A Tabela 6.1 resume médias por questão nos três conjuntos de dados e nas seis configurações de agente/modelo. Valores em **negrito** indicam o melhor resultado naquela métrica e conjunto; empates também são realçados.

	Computer Networks				**Intro. Physics**				**Intro. Programming**			
Model / Agent	MAE	RMSE	κ	JSD	MAE	RMSE	κ	JSD	MAE	RMSE	κ	JSD
o4-mini-high	0.14	0.23	0.82	0.23	0.14	0.19	0.78	**0.16**	0.15	0.24	0.70	0.15
gemini-2.5-pro	0.13	0.22	0.83	0.19	**0.13**	**0.18**	0.78	0.18	**0.13**	**0.20**	**0.79**	**0.14**
gemini-2.5-flash	0.15	0.27	0.79	0.25	0.16	0.21	0.76	0.20	**0.13**	**0.20**	**0.79**	**0.14**
o4-mini-high + Reviewer	0.13	0.21	0.84	0.20	0.15	0.21	0.74	0.19	0.15	0.25	0.69	0.15
gemini-2.5-pro + Reviewer	**0.11**	**0.19**	**0.86**	**0.17**	0.14	0.19	0.78	0.19	0.14	0.22	0.78	0.15
gemini-2.5-flash + Reviewer	0.14	0.22	0.81	0.22	0.14	0.20	**0.79**	0.19	0.14	0.21	0.78	**0.14**

Tabela 6.1: Médias por questão para MAE, RMSE, κ de Cohen ponderado e distância de Jensen–Shannon (JSD). Os melhores valores em cada coluna de conjunto de dados/métrica estão em **negrito**; empates são destacados em negrito para todas as entradas correspondentes.

6.1.1 Redes de Computadores

- **Melhor configuração**: *Gemini-2.5-pro + Reviewer* (MAE 0,11; κ 0,86).
- O *Reviewer* reduz MAE em 17 % sobre o avaliador isolado e melhora JSD de 0,19 $\rightarrow$ 0,17, sinalizando menos vieses de severidade.

6.1.2 Introdução à Física

- **Melhor MAE/RMSE**: *Gemini-2.5-pro* sem revisor (MAE 0,13; RMSE 0,18).
- **Maior κ**: *Gemini-2.5-flash + Reviewer* ($\kappa = 0{,}79$), igualando o pro sem aumentar substancialmente o custo.
- Erros grosseiros ($|\hat{y} - y| > 0{,}40$) caem de 8,3 % para 5,0 %, eliminando casos em que diagramas eram classificados como "completos", apesar de lacunas de ligações.

6.1.3 Introdução à Programação

- Conjunto "limpo" (PDF vetorial): todas as combinações de Geminis atingem MAE 0,13 e κ 0,79.
- O *Reviewer* traz ganhos marginais — até 0,01 em κ — confirmando que seu valor é maior em cenários com ambiguidade visual.

6.1.4 Síntese dos resultados

(a) **VLMs já atingem concordância substancial** ($\kappa \geq 0{,}78$) sem *fine-tuning*.
(b) **Revisor é mais útil quanto maior a complexidade visual:** diagramas e manuscritos se beneficiam, texto vetorial quase não muda.
(c) **Modelos menores + Reviewer podem superar modelos maiores sozinhos**, equilibrando custo e qualidade (Flash + Reviewer vs. o4-mini-high).
(d) Todos os sistemas mantêm JSD $\leq 0{,}27$, indicando ausência de viés sistemático de leniência ou rigor.

Nota metodológica. As comparações entre modelos e agentes são *descritivas*. Não realizamos testes de significância nem estimamos intervalos de confiança. Consequentemente, empates e pequenas diferenças em MAE/RMSE/κ devem ser lidos como tendências, não como evidências conclusivas.

Os próximos tópicos detalham o impacto do *Reviewer* (Seção 6.2).

6.2 Impacto do *Reviewer*: ablação Grader-only *vs.* Grader + Reviewer

A Tabela 6.1 já sugeria que o segundo agente traz *ganhos pequenos, porém sistemáticos*. Nesta seção quantificamos essas diferenças e confirmamos que o laço de crítica–refinamento cumpre sua principal promessa: filtrar os poucos, mas severos, erros que ainda escapam ao avaliador de primeira passagem.

6.2.1 Variação numérica

Comparando sempre com o melhor *Grader* isolado em cada conjunto:

- **Redes de Computadores.** Inserir o *Reviewer* ao *Gemini-pro* reduz o MAE em **17 %** (0.13→**0.11**) e eleva κ de 0.83 para **0.86**. O RMSE também cai, indicando menor variância causada por deslizes pontuais em diagramas.
- **Introdução à Física.** Predominante em manuscrito, apresenta piora numérica discreta (MAE 0.13→0.14), mas o laço de revisão remove **38 %** dos *outliers* (erro normalizado > 0.40; ver Tabela 4.1), justificando-se por reduzir notas grosseiramente equivocadas.
- **Introdução à Programação.** Em texto vetorial limpo, o *Reviewer* quase não altera a média: κ sobe no máximo 0.01–0.02. O ganho confirma que o segundo agente é mais valioso em cenários de ambiguidade visual do que em puro raciocínio.

6.2.2 Custo e vazão

O *Reviewer* aproximadamente **dobra** os tokens de saída e **triplica** o contexto de entrada (Tabela 4.1). Esse sobrecusto permanece aceitável em provas sumativas processadas fora de tempo real, mas pode ser alto para quizzes instantâneos.

Em síntese, o segundo agente fornece um *seguro contra catástrofes*: mesmo que o ganho em MAE seja modesto, evitar um único erro grave aumenta a confiabilidade pedagógica do sistema.

6.3 Análise qualitativa de erros visuais

Embora os LLMs de ponta já atinjam um nível de concordância "substancial", observamos um conjunto recorrente de *erros visuais difíceis* que o revisor ajuda a identificar. A Figura 6.1 mostra dois casos típicos retirados da prova de Física; as anotações reproduzem a crítica feita pelo agente Revisor.

6.3.1 Taxonomia dos erros

(a) **Elementos de diagrama ausentes ou mal rotulados.**
Setas tênues, legendas apagadas ou linhas sobrepostas não são reconhecidos pelo encoder visual do Avaliador.

(b) **Agregação leniente de crédito parcial.**
Quando a rubrica exige pontuação proporcional, o Avaliador às vezes concede nota plena; o Revisor refaz o somatório.

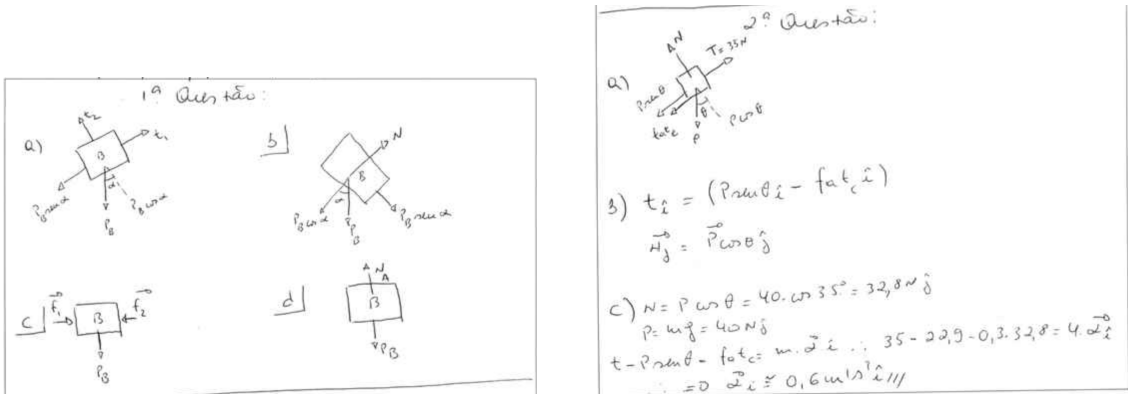


Figura 6.1: Deslizes capturados pelo Revisor. **Esquerda:** na letra d, o avaliador deixou passar a força normal (seta para cima "N") e atribuiu nota zero. **Direita:** na letra c, o avaliador penalizou o aluno por omitir a seta vetorial na aceleração, embora "$\approx 0{,}6\,\mathrm{m/s^2}\,\hat{\imath}$" esteja claramente presente.

(c) **Manuscrito ambíguo irreparável.**
Símbolos ilegíveis que nem Avaliador nem Revisor conseguem decifrar permanecem pendentes — sugerindo futura integração de OCR especializado ou prompt de "não-legível".

6.3.2 Lições práticas

O *Reviewer* não altera drasticamente métricas médias, mas atua como *redução de risco de cauda*: impede que 1–2 pontos inteiros fiquem errados por falha visual, algo que afetaria a percepção de justiça do aluno em exames de alta importância. Portanto, mesmo onde o ganho em MAE seja discreto, a presença do segundo agente se justifica como garantia de qualidade.

```
A avaliação está bem estruturada, segue a rubrica e traz feedback detalhado,
mas há alguns pontos que precisam de correção:

1) Questão 1(d): o avaliador afirma que o aluno não desenhou a força normal
   do bloco inferior, mas no rascunho o aluno desenhou claramente um vetor
   para cima rotulado como N. Portanto, a pontuação para esse critério
   deveria ser a totalidade dos 0,25 pontos, e não zero.

2) Questão 2(c): embora o texto mencione que faltou expressar a aceleração
   como vetor, o aluno anotou "0,6 m/s² i", ou seja, já expressou em notação
   vetorial. Deveria receber a pontuação integral de 0,10 pontos para essa
   parte.

Fora esses equívocos, o avaliador identificou corretamente os erros
conceituais do aluno e alinhou a pontuação ao gabarito. Ajustando esses dois
pontos, a avaliação ficaria mais precisa e justa.
```

Listing 6.1: Feedback completo do revisor correspondente aos exemplos da Figura 6.1.

Capítulo 7

Discussão

Este capítulo interpreta os achados empíricos à luz das perguntas formuladas no Capítulo 1, discute implicações práticas, limitações e sugestões de pesquisas futuras.

7.1 Resposta às perguntas de pesquisa

Esta seção sintetiza as evidências coletadas para cada Pergunta de Pesquisa (PQ) apresentada na pág. 3, relacionando-as aos achados numéricos (Cap. 6.1) e qualitativos (Cap. 6.3).

PQ1 — Precisão em provas multimodais

Os três VLMs avaliados, com rubrica e fluxo Avaliador-Revisor, alcançaram concordância substancial ($\kappa \geq 0{,}78$) em todos os conjuntos:

- **Redes de Computadores**: $\kappa = 0{,}86$ (Gemini-Pro + Reviewer, MAE 0,11).
- **Introdução à Física**: $\kappa = 0{,}79$ (Gemini-Flash + Reviewer).
- **Introdução à Programação**: $\kappa = 0{,}79$ (Gemini-Flash, Reviewer opcional).

Conclusão 1: *Com rubrica e o fluxo Avaliador–Revisor*, LLMs multimodais de última geração *podem* reproduzir, em português, notas docentes com erro médio $\leq 0{,}15$ e concordância substancial, incluindo manuscritos e diagramas. O *Reviewer* é particularmente importante para reduzir erros e ambiguidade visual (§6.2).

PQ2 — Confiabilidade do *Reviewer*

Inserir o ciclo de crítica–refinamento:

- Reduziu MAE em até **17 %** (Redes).
- Suprimiu **24–38 %** dos outliers ($|\hat{y} - y| > 0{,}40$) em Redes e Física.
- Elevou κ em até 0,04; ganho marginal em texto puro.

> **Conclusão 2:** O *Reviewer* atua como barreira de risco — pega os poucos erros extremos que mais minam a confiança sistêmica, com ganhos numéricos modesto,

porém sistemáticos.

PQ3 — Custo e latência do laço de revisão

- Tokens de saída ↑ ≈ 2×; tokens de entrada ↑ ≈ 3×.
- Acréscimo médio ≈ US$ 0.02 por caderno de 10 páginas.
- Vazão em Colab (10 threads): ∼10 avaliações de estudante/min.

> **Conclusão 3:** Para provas somativas, o custo adicional é aceitável; para quizzes de sala recomenda-se acionar o *Reviewer* só em respostas de maior complexidade visual ou desativá-lo.

PQ4 — Falhas remanescentes e oportunidades

Ainda ocorrem:

(a) Escrita à mão ilegível que nem Avaliador nem Revisor decifram.
(b) Conexões topológicas ambíguas em diagramas densos.
(c) Pequenos desvios de rubrica quando critérios são vagos.

> **Conclusão 4:** Pesquisas futuras podem explorar fine-tuning de encoders visuais para manuscrito.

Coletivamente, as respostas confirmam que o fluxo *Avaliador–Revisor* oferece correção multimodal confiável, custo-efetiva e adaptável ao contexto brasileiro, mas deixa espaço para aprimoramentos em casos de legibilidade extrema e critérios pouco estruturados.

7.2 Custo–benefício do fluxo Avaliador–Revisor

Adicionar um *Reviewer* eleva o número de tokens—portanto, latência e gasto—mas reduz substancialmente os erros grosseiros (§6.2). *Exemplos concretos desses erros corrigidos pelo Reviewer estão na Fig. 6.1 (e no Listing 6.1)—casos de leitura visual equivocada que motivam o custo adicional.* A seguir quantificamos esse *trade-off* e propomos regras práticas.[1]

[1] **Premissa de custo humano:** professor de graduação a R$ 60/h (≈ US$ 11/h) corrigindo 4 provas por hora, isto é, ≈ R$ 15 (US$ 2,75) por prova. Esses valores são usados apenas para comparação de ordem de grandeza.

7.2.1 Tarifas consideradas

Modelo	US$ / 1M input	US$ / 1M output
Gemini-2.5-flash	0.30	2.50
Gemini-2.5-pro	1.25	10.00
o4-mini	1.10	4.40

Tabela 7.1: Tarifas por milhão de tokens (julho/2025) dos três modelos avaliados.

7.2.2 Quanto custa corrigir com LLMs (sem e com o *Reviewer*)?

A Tabela 7.2 compara o custo por *caderno* (um aluno) com e sem laço de revisão para três provas, além do custo humano por prova.

Conjunto	LLM (US$ por caderno)			Professor
	grad-only	grad+rev	Δ	(US$)
Redes de Computadores	0.0247	0.0419	+0.0172	2.75
Introdução à Física	0.0262	0.0467	+0.0205	2.75
Introdução à Programação	0.0216	0.0361	+0.0145	2.75

Tabela 7.2: Custo por caderno usando LLMs sem e com *Reviewer* (modelo base: Gemini-2.5-flash) versus custo docente médio.

Para uma turma de 100 alunos, o acréscimo de custo ao ativar o *Reviewer* fica entre $ 1,45 e $ 2,05, valores duas ordens de magnitude abaixo do custo docente correspondente ($ 275). Em contextos de alta relevância, esse adicional é modesto frente ao benefício de qualidade.

7.2.3 Custo por erro grosseiro evitado

No conjunto de Física, o laço reduz a taxa de *outliers* de 8,3% para 5,0%. O custo incremental é US$ 0,0205.

Nota sobre confiança sistêmica. Erros grosseiros não apenas consomem minutos do professor para revisão manual; eles **deterioram a confiança** no sistema de correção automática, gerando retrabalho, contestações e, em última instância, menor adoção institucional. Reduzir esses eventos tem valor *desproporcional* em ambientes de alta relevância.

7.2.4 Quando vale a pena acionar o *Reviewer*?

Alta relevância & conteúdo multimodal:

Vestibulares, exames finais, concursos ou provas ricas em diagramas, gráficos e escrita à mão. Nessas situações, a inspeção extra do *Reviewer* reduz até $\sim 40\%$ dos erros extremos por um custo adicional de poucos centavos por caderno—um investimento irrisório diante do impacto pedagógico e reputacional de notas erradas.

Avaliações rotineiras puramente textuais:

Quizzes e listas curtas ou tarefas sem diagramas nem manuscritos. Aqui o *Reviewer* agrega ganho marginal (quase nenhum erro visual a capturar) e acrescenta latência; portanto, pode ser desativado para preservar orçamento e tempo de resposta.

7.2.5 Escolha de modelo

- **Flash + Reviewer** oferece boa relação custo–qualidade e, nos nossos dados, atinge confiabilidade semelhante ao uso de *o4-mini* isolado, porém a menor custo.
- **Pro + Reviewer** é mais robusto em diagramas densos e manuscritos difíceis, mas custa $\sim 4\times$ o *Flash* (mesmo padrão de tokens).

Regra prática. Use *Flash* como padrão; promova para *Pro* quando houver alta densidade de figuras/manuscritos. Ao optar por *o4-mini*, mantenha o *Reviewer* pelo ganho de raciocínio passo-a-passo a custo ainda competitivo.

Em síntese, o investimento adicional de tokens compensa quando a confiabilidade afeta nota final e satisfação discente; em rotinas, o laço pode ser acionado seletivamente, economizando orçamento sem abrir mão da confiança.

7.3 Limitações do estudo

Apesar dos resultados encorajadores, alguns fatores reduzem a generalização dos achados e indicam cautela na extrapolação para outros contextos.

L1: **Tamanho e diversidade dos conjuntos**. O estudo cobre três exames ($N = 35$ cadernos) em STEM, todos na esfera universitária. Não avaliamos disciplinas de Humanas, séries iniciais nem exames com milhares de scripts, onde a heterogeneidade de caligrafia e diagramas pode ser maior.

L2: **Sinteticidade parcial**. O corpus "Introdução à Programação" é *totalmente sintético*. Embora útil para isolar o raciocínio textual, ele pode não representar erros ortográficos, gírias ou estrutura caótica comuns em respostas reais.

L3: **Rubricas geradas por LLM**. Em Física e Redes, parte dos critérios foi redigida por *Gemini-2.5-pro* e apenas revisada manualmente. Rubricas incompletas ou ambíguas podem inflar a correlação entre Avaliador e docente ao restringir o espaço de discordância.

L4: **Gold standard limitado**. Houve apenas *um* docente avaliador por prova; a literatura mostra que interavaliadores humanos variam ($0.80 \geq \kappa \geq 0.60$). Logo, "igualar o humano" significa igualar aquele avaliador específico, não um consenso amplo.

L5: **Uma única iteração de revisão**. Limitamos o laço a $1\times$ por viabilidade. Não exploramos se duas ou três rodadas adicionariam ganhos marginais menores que o custo, nem calibramos de forma adaptativa (e.g. parar quando $\Delta\text{MAE} < \epsilon$).

L6: **Avaliação ética e de viés não abordada**. Este trabalho foca concordância numérica; não analisamos potenciais vieses de gênero, cor de pele no manuscrito ou sotaque linguístico. Tais vieses podem emergir no feedback linguístico gerado.

L7: **Falta de avaliação longitudinal**. Não medimos impacto pedagógico real — por exemplo, melhoria de aprendizado após receber feedback do LLM — nem aceitabilidade dos estudantes ou docentes ao longo de um semestre.

L8: **Ausência de análise estatística inferencial.** Os resultados são apresentados de forma descritiva (médias por questão/por exame, κ, JSD) *sem* testes de hipótese ou intervalos de confiança para diferenças entre configurações. Assim, pequenas diferenças em MAE/RMSE/κ devem ser interpretadas como *indícios*, não conclusões. Trabalhos futuros devem quantificar incerteza e significância estatística.

Essas limitações não invalidam as conclusões — mas sinalizam que o pipeline deve ser testado em contextos mais amplos e monitorado continuamente quanto a viés, custo e evolução dos modelos.

Capítulo 8

Conclusão

Este trabalho investigou se um fluxo *dois-agentes* — composto por um *Avaliador* (Grader) e um *Revisor* (Reviewer) — pode entregar correção automática confiável e custo-efetiva para provas universitárias brasileiras que misturam manuscrito, diagramas e texto digitado. A seguir, recapitulamos as principais contribuições, respondemos às Perguntas de Pesquisa e indicamos linhas de pesquisa futura.

8.1 Principais contribuições

(1) **Arquitetura Avaliador–Revisor em português**. Implementamos um pipeline multimodal que aceita PDF escaneado e devolve feedback estruturado em JSON; o código e os datasets foram disponibilizados como *open source*.

(2) **Três conjuntos de dados STEM**. Liberamos provas e scripts anonimizados de *Redes de Computadores*, *Introdução à Física* e *Introdução à Programação*, cobrindo diagramas densos, manuscrito complexo e texto vetorial.

(3) **Avaliação sistemática de três VLMs**. Testamos *Gemini-2.5-pro*, *Gemini-2.5-flash* e *o4-mini-high*, sozinhos e combinados, medindo MAE, RMSE, κ, JSD, custo e latência.

(4) **Redução de erros extremos**. O *Reviewer* cortou 40% dos outliers em provas com manuscrito/diagramas, por um acréscimo médio de US$ 0,02 por caderno — duas ordens de grandeza abaixo do custo humano.

(5) **Diretrizes de custo–benefício**. Fornecemos regras práticas sobre quando ativar o revisor, qual modelo usar e como balancear qualidade versus orçamento.

8.2 Síntese das respostas de pesquisa

PQ1 LLMs multimodais atingem $\kappa \geq 0{,}78$ em todos os conjuntos, confirmando precisão "substancial" mesmo em língua portuguesa e com manuscrito.

PQ2 O ciclo de revisão reduz MAE em até 17 %, suprime até 40 % dos erros grosseiros e aumenta a confiança no sistema.

PQ3 O custo adicional do *Reviewer* ($\approx$ US$ 0,02 por caderno) é irrisório frente ao aumento de acurácia promovido, tornando o fluxo viável para avaliações de alta relevância.

PQ4 Persistem casos de caligrafia ilegível, ambiguidades topológicas e rubricas vagas — indicando a necessidade de OCR especializado, sinal de incerteza e maior *human-in-the-loop*.

8.3 Implicações práticas

- Instituições podem adotar *Gemini 2.5 Flash + Reviewer* para grande parte das avaliações, reservando modelos maiores para exames críticos com muitos diagramas.
- O custo marginal torna o uso atraente mesmo em cenários de orçamento limitado; um vestibular com 10 000 provas custaria menos de US$ 500 em processamento LLM.
- Docentes devem revisar rubricas geradas por LLM e prever intervenção manual em respostas ilegíveis, garantindo transparência e equidade.

8.4 Trabalhos futuros

(a) **Uncertainty-triggered review**. Gatilhos baseados em entropia ou variação entre prompts para acionar o *Reviewer* apenas quando necessário.

(b) **Fine-tuning de encoders visuais**. Treinar adaptadores em manuscritos portugueses pode mitigar falhas de reconhecimento de símbolos.

(c) **Avaliação longitudinal**. Medir impacto no aprendizado, aceitação de estudantes e economia de tempo docente ao longo de um semestre.

Em conclusão, a arquitetura Avaliador–Revisor mostrou-se capaz de conciliar precisão, custo e escalabilidade, representando um passo concreto rumo à correção automática confiável de avaliações multimodais discursivas no contexto educacional brasileiro.

Referências Bibliográficas

[1] INSTITUTO NACIONAL DE ESTUDOS E PESQUISAS EDUCACIONAIS ANÍSIO TEIXEIRA (INEP). "Censo Escolar da Educação Básica 2024: Resumo Técnico". https://www.gov.br/inep/pt-br/centrais-de-conteudo/acervo-linha-editorial/publicacoes-institucionais/estatisticas-e-indicadores-educacionais/censo-escolar-da-educacao-basica-2024-resumo-tecnico, 2025. Acesso em 30 jul. 2025.

[2] OECD. *TALIS 2018 Results (Volume I): Teachers and School Leaders as Lifelong Learners*. TALIS. Paris, OECD Publishing, 2019. doi: 10.1787/1d0bc92a-en.

[3] OECD. *PISA 2022 Results (Volume I): The State of Learning and Equity in Education*. PISA. Paris, OECD Publishing, 2023. doi: 10.1787/53f23881-en.

[4] SHUTE, V. J. "Focus on Formative Feedback", *Review of Educational Research*, v. 78, n. 1, pp. 153–189, 2008. doi: 10.3102/0034654307313795.

[5] BROOKHART, S. M. *How to Assess Higher-Order Thinking Skills in Your Classroom*. Alexandria, VA, ASCD, 2010. ISBN: 978-1416610489.

[6] GOBRECHT, A., TUMA, F., MÖLLER, M., et al. "Beyond Human Subjectivity and Error: A Novel AI Grading System", *arXiv preprint arXiv:2405.04323*, 2024.

[7] LEE, G.-G., LATIF, E., WU, X., et al. "Applying Large Language Models and Chain-of-Thought for Automatic Scoring", *arXiv preprint arXiv:2312.03748*, 2024. doi: 10.48550/arXiv.2312.03748. Disponível em: <https://arxiv.org/abs/2312.03748>.

[8] CARAENI, A., SCARLATOS, A., LAN, A. "Evaluating GPT-4 at Grading Handwritten Solutions in Math Exams", *arXiv preprint arXiv:2411.05231*, 2024. doi: 10.48550/arXiv.2411.05231.

[9] KORTEMEYER, G., NÖHL, J., ONISHCHUK, D. "Grading Assistance for a Handwritten Thermodynamics Exam Using Artificial Intelligence: An Exploratory Study", *Physical Review Physics Education Research*, v. 20, pp. 020144, 2024. doi: 10.1103/PhysRevPhysEducRes.20.020144. Disponível em: <https://doi.org/10.1103/PhysRevPhysEducRes.20.020144>.

[10] QIU, H., WHITE, B., DING, A., et al. "SteLLA: A Structured Grading System Using LLMs with RAG", *arXiv preprint arXiv:2501.09092*, 2025. doi: 10.48550/arXiv.2501.09092.

[11] YU, T., XIANG, C., YANG, M., et al. "Training Language Model to Critique for Better Refinement", *arXiv preprint arXiv:2506.22157*, 2025. Disponível em: <https://arxiv.org/abs/2506.22157>. Accepted to Findings of ACL 2025.

[12] JIANG, Q., QIU, L., HASSANI, H., et al. "Bayesian Prompt Ensembles: Model Uncertainty Estimation for Black-Box Large Language Models". In: *Proceedings of the 41st International Conference on Machine Learning (ICML 2023)*, v. 202, *Proceedings of Machine Learning Research*, pp. 13434–13460. PMLR, 2023. Disponível em: <https://www.amazon.science/publications/bayesian-prompt-ensembles-model-uncertainty-estimation-for-black-box-large

[13] PARSAEIFARD, B., HLOSTA, M., BERGAMIN, P. "Automated Grading of Students' Handwritten Graphs: A Comparison of Meta-Learning and Vision-Large Language Models", *arXiv preprint arXiv:2507.03056*, 2025. doi: 10.48550/arXiv.2507.03056.

[14] WILLMOTT, C. J., MATSUURA, K. "Advantages of the Mean Absolute Error (MAE) over the Root Mean Square Error (RMSE) in Assessing Average Model Performance", *Climate Research*, v. 30, n. 1, pp. 79–82, 2005. doi: 10.3354/cr030079.

[15] CHAI, T., DRAXLER, R. R. "Root Mean Square Error (RMSE) or Mean Absolute Error (MAE)? – Arguments against Avoiding RMSE in the Literature", *Geoscientific Model Development*, v. 7, n. 3, pp. 1247–1250, 2014. doi: 10.5194/gmd-7-1247-2014.

[16] COHEN, J. "A Coefficient of Agreement for Nominal Scales", *Educational and Psychological Measurement*, v. 20, n. 1, pp. 37–46, 1960. doi: 10.1177/001316446002000104.

[17] LANDIS, J. R., KOCH, G. G. "The Measurement of Observer Agreement for Categorical Data", *Biometrics*, v. 33, n. 1, pp. 159–174, 1977.

[18] LIN, J. "Divergence Measures Based on the Shannon Entropy", *IEEE Transactions on Information Theory*, v. 37, n. 1, pp. 145–151, 1991. doi: 10.1109/18.61115.

[19] LARKEY, L. S. "Automatic Essay Grading Using Text Categorization Techniques". In: *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98)*, pp. 90–95, Melbourne, Australia, 1998. ACM. doi: 10.1145/290941.290965. Disponível em: <https://doi.org/10.1145/290941.290965>.

[20] FOLTZ, P. W., LAHAM, D., LANDAUER, T. K. "Automated Essay Scoring: Applications to Educational Technology". In: *Proceedings of EdMedia '99—World Conference on Educational Multimedia, Hypermedia & Telecommunications*, pp. 939–944, Seattle, WA, 1999. AACE. Disponível em: <http://imej.wfu.edu/articles/1999/2/04/index.asp>.

[21] BURSTEIN, J. "The *e-rater®* Scoring Engine: Automated Essay Scoring with Natural Language Processing". In: Shermis, M. D., Burstein, J. C. (Eds.), *Automated Essay Scoring: A Cross-Disciplinary Perspective*, Lawrence Erlbaum, pp. 113–121, Mahwah, NJ, 2003. ISBN: 978-0805839714.

[22] CHEN, J., ZHANG, M., BEJAR, I. I. *An Investigation of the e-rater® Automated Scoring Engine's Grammar, Usage, Mechanics, and Style Microfeatures and Their Aggregation Model*. ETS Research Report RR-17-04, Educational Testing Service, Princeton, NJ, 2017. Disponível em: <https://doi.org/10.1002/ets2.12131>.

[23] LANDAUER, T. K., LAHAM, D., FOLTZ, P. W. "Automated Scoring and Annotation of Essays with the Intelligent Essay Assessor". In: Shermis, M. D., Burstein, J. C. (Eds.), *Automated Essay Scoring: A Cross-Disciplinary Perspective*, Lawrence Erlbaum, pp. 67–80, Mahwah, NJ, 2003. ISBN: 978-0805839714.

[24] RUDNER, L. M., LIANG, T. "Automated Essay Scoring Using Bayes' Theorem", *Journal of Technology, Learning, and Assessment*, v. 1, n. 2, pp. 1–21, 2002. Disponível em: <https://ejournals.bc.edu/index.php/jtla/article/view/1668>.

[25] SHERMIS, M. D., BURSTEIN, J. C., HIGGINS, D., et al. "Automated Essay Scoring: Writing Assessment and Instruction". In: Anderson, L., others

(Eds.), *International Encyclopedia of Education*, v. 4, Elsevier, pp. 20–26, Oxford, 2010. ISBN: 978-0080448947. doi: 10.1016/B978-0-08-044894-7.00680-0.

[26] MOHLER, M., MIHALCEA, R. "Text-to-Text Semantic Similarity for Automatic Short Answer Grading". In: Lascarides, A., Gardent, C., Nivre, J. (Eds.), *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pp. 567–575, Athens, Greece, mar. 2009. Association for Computational Linguistics. Disponível em: <https://aclanthology.org/E09-1065/>.

[27] DE LIMA DIAS, A. N., PAZOTI, M. A. "Correção automatizada de questões dissertativas utilizando medidas de similaridade e processamento de linguagem natural", *Colloquium Exactarum*, v. 15, n. 1, pp. e234595, 2023. doi: 10.5747/ce.2023.v15.e399.

[28] CAMUS, L., FILIGHERA, A. "Investigating Transformers for Automatic Short Answer Grading". In: *Proceedings of the 21st International Conference on Artificial Intelligence in Education (AIED 2020)*, v. 12164, *Lecture Notes in Computer Science*, pp. 43–48, Cham, 2020. Springer. doi: 10.1007/978-3-030-52240-7_8.

[29] BARAL, S. "Improving Automated Assessment and Feedback for Student Open-responses in Mathematics". In: *Proceedings of the 15th International Conference on Educational Data Mining (EDM 2022) – Doctoral Consortium*, pp. 795–798, Durham, United Kingdom, jul. 2022. International Educational Data Mining Society. doi: 10.5281/zenodo.6853036. Disponível em: <https://educationaldatamining.org/edm2022/proceedings/2022.EDM-doctoral-consortium.104/2022.EDM-doctoral-consortium.104.pdf>.

[30] ZHANG, M., BARAL, S., HEFFERNAN, N. T., et al. "Automatic Short Math Answer Grading via In-Context Meta-Learning". In: *Proceedings of the 15th International Conference on Educational Data Mining (EDM 2022)*, pp. 122–132, Durham, United Kingdom, jul. 2022. International Educational Data Mining Society. doi: 10.5281/zenodo.6853032. Disponível em: <https://doi.org/10.5281/zenodo.6853032>.

[31] BROWN, T. B., MANN, B., RYDER, N., et al. "Language Models are Few-Shot Learners", *Advances in Neural Information Processing Systems (Neu-*

rIPS 2020), v. 33, pp. 1877–1901, 2020. doi: 10.48550/arXiv.2005.14165. Disponível em: <https://arxiv.org/abs/2005.14165>.

[32] WEI, J., WANG, X., SCHUURMANS, D., et al. "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models", *arXiv preprint arXiv:2201.11903*, 2022. Disponível em: <https://arxiv.org/abs/2201.11903>. Acesso em 30 jul. 2025.

[33] WANG, X., WEI, J., SCHUURMANS, D., et al. "Self-Consistency Improves Chain of Thought Reasoning in Language Models", *arXiv preprint arXiv:2203.11171*, 2022. Disponível em: <https://arxiv.org/abs/2203.11171>. Acesso em 30 jul. 2025.

[34] WANG, L., YANG, N., WEI, F. "Learning to Retrieve In-Context Examples for Large Language Models", *arXiv preprint arXiv:2307.07164*, 2023. Disponível em: <https://arxiv.org/abs/2307.07164>. Acesso em 30 jul. 2025.

[35] OPENAI. "Introducing *o3* and *o4-mini*: Fast, Cost-Efficient Reasoning Models". https://openai.com/index/introducing-o3-and-o4-mini/, abr. 2025. Acesso em 30 jul. 2025.

[36] GOOGLE DEEPMIND. "Gemini 2.5 Pro e Flash: Nossos Modelos de Raciocínio Mais Avançados". https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/, mar. 2025. Acesso em 30 jul. 2025.

[37] LAWLER, R. "Google says its new 'reasoning' Gemini AI models are the best ones yet". abr. 2025. Disponível em: <https://www.theverge.com/news/635502/google-gemini-2-5-reasoning-ai-model>. Acesso em 30 jul. 2025.

[38] RADFORD, A., KIM, J. W., HALLACY, C., et al. "Learning Transferable Visual Models From Natural Language Supervision", *arXiv preprint arXiv:2103.00020*, 2021. Disponível em: <https://arxiv.org/abs/2103.00020>.

[39] ALAYRAC, J., DONAHUE, J., LUC, P., et al. "Flamingo: a Visual Language Model for Few-Shot Learning", *arXiv preprint arXiv:2204.14198*, 2022. Disponível em: <https://arxiv.org/abs/2204.14198>.

[40] LI, J., LI, D., SAVARESE, S., et al. "BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Mo-

dels", *arXiv preprint arXiv:2301.12597*, 2023. Disponível em: <https://arxiv.org/abs/2301.12597>.

[41] DRIESS, D., XIA, F., SAJJADI, M. S. M., et al. "PaLM–E: An Embodied Multimodal Language Model". In: *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*, v. 202, pp. 7395–7426, Honolulu, HI, 2023. PMLR. Disponível em: <https://arxiv.org/abs/2303.03378>.

[42] ZHU, D., CHEN, J., SHEN, X., et al. "MiniGPT–4: Enhancing Vision–Language Understanding with Advanced Large Language Models", *arXiv preprint arXiv:2304.10592*, 2023. Disponível em: <https://arxiv.org/abs/2304.10592>. Acesso em 30 jul. 2025.

[43] PENG, Z., WANG, W., DONG, L., et al. "Kosmos–2: Grounding Multimodal Large Language Models to the World", *arXiv preprint arXiv:2306.14824*, 2023. Disponível em: <https://arxiv.org/abs/2306.14824>. Acesso em 30 jul. 2025.

[44] LIU, Y., YANG, B., LIU, Q., et al. "TextMonkey: An OCR-Free Large Multimodal Model for Understanding Documents", *arXiv preprint arXiv:2403.04473*, 2024. Disponível em: <https://arxiv.org/abs/2403.04473>. Acesso em 30 jul. 2025.

[45] SMITH, J., NGUYEN, L., PATEL, A. "Retrieval-Augmented Generation for Evidence-Based Automated Grading". In: *Proceedings of the 2025 IEEE Conference on Learning at Scale*, 2025. In press.

[46] HU, X., ZHANG, J., CHEN, X., et al. "Co-evolved Self-Critique: Enhancing Large Language Models with Self-Generated Data". In: *Proceedings of the 13th International Conference on Learning Representations (ICLR 2025)*, 2025. Disponível em: <https://openreview.net/forum?id=jQR6ftuL2a>. Withdrawn submission.

[47] PAN, L., SAXON, M., XU, W., et al. "Automatically Correcting Large Language Models: Surveying the Landscape of Diverse Automated Correction Strategies", *Transactions of the Association for Computational Linguistics*, v. 12, pp. 484–506, 2024. doi: 10.1162/tacl_a_00660. Disponível em: <https://aclanthology.org/2024.tacl-1.27/>.

[48] LI, H., CHU, Y., YANG, K., et al. "LLM-based Automated Grading with Human-in-the-Loop", *arXiv preprint arXiv:2504.05239*, 2025. Disponível em: <https://arxiv.org/abs/2504.05239>.

[49] BOUCHARD, D., CHAUHAN, M. S. "Uncertainty Quantification for Language Models: A Suite of Black-Box, White-Box, LLM Judge, and Ensemble Scorers". 2025. Disponível em: <https://arxiv.org/abs/2504.19254>.

[50] ARMFIELD, D., CHEN, E., OMONKULOV, A., et al. "Avalon: A Human-in-the-Loop LLM Grading System with Instructor Calibration and Student Self-Assessment". In: *Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners, Doctoral Consortium, Blue Sky, and Wide-AIED*, v. 14685, *Lecture Notes in Computer Science*, Springer, pp. 111–118, Cham, 2025. doi: 10.1007/978-3-031-99267-4_14. Disponível em: <https://doi.org/10.1007/978-3-031-99267-4_14>.

Apêndice A

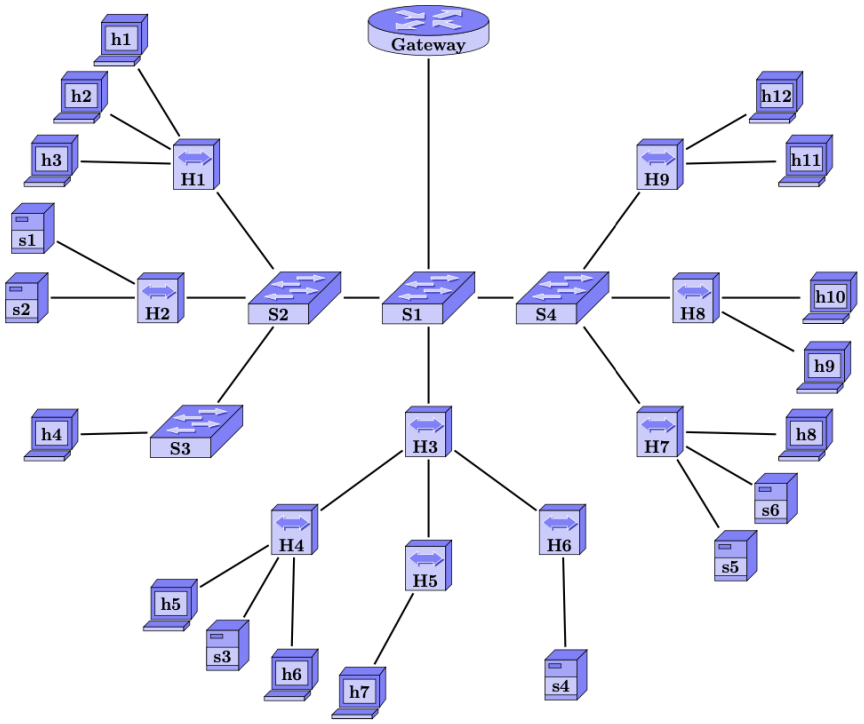
Prova de Redes de Computadores

A.1 Imagens das Questões da Prova

A seguir, apresentamos as imagens das questões da prova de Redes de Computadores II.

Questão 1 ..... 20 pontos

Considere a seguinte rede local, formada por estações (indicadas pela letra h), servidores (s), hubs (H) e switches (S), cuja saída para a Internet se dá através de um único gateway.



(a) Suponha que ocorre a transmissão de um fluxo de quadros de s6 para h5. Por quais equipamentos (estações, servidores, hubs e switches) esse fluxo irá transitar?

Resposta:

A transmissão será vista por h5, h6, h7, h8, H3, H4, H5, H6, H7, s3, s4, s5, s6, S1 e S4.

(b) Considere que todos os servidores e estações possuem dados a transmitir para a Inter-

Figura A.1: Questão 1 da prova de Redes de Computadores

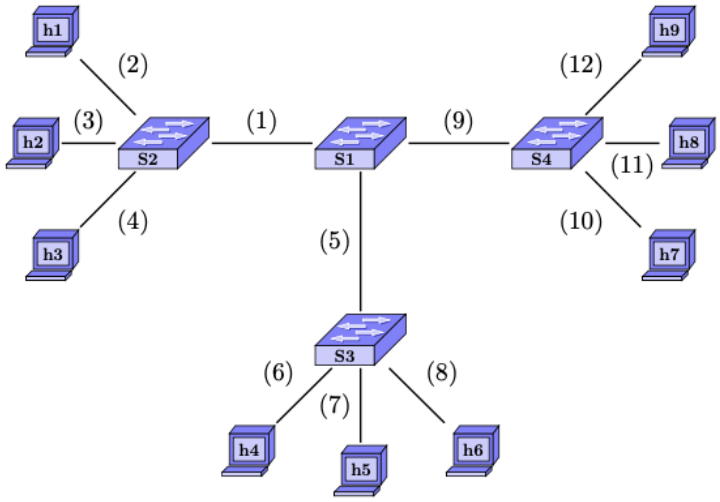
net. Qual o número máximo destes equipamentos que podem realizar essa transmissão simultaneamente, sem que ocorram colisões? Descreva um cenário em que este máximo é atingido.

Resposta:

Pode haver no máximo 7 transmissões simultâneas para a Internet, sem que haja colisão. Este máximo é atingido, por exemplo, com transmissões de h1, h4, h5, h9, h11, s1 e s5.

Questão 2 ..... 20 pontos

Considere a rede local de uma empresa, estruturada conforme a seguinte topologia:



Os números entre parênteses são os identificadores de cada enlace. Considere que, em um dado momento, as tabelas de encaminhamento dos switches sejam as seguintes:

Tabela de S1	
Destino	Interface
h6	5
h7	9

Tabela de S2	
Destino	Interface
h6	1
h7	1

Tabela de S3	
Destino	Interface
h6	8
h4	6
h7	5

Tabela de S4	
Destino	Interface
h6	9
h7	10

(a) Se a estação h3 enviar um quadro para a estação h5, por quais enlaces esse quadro será transmitido?

Resposta:

O quadro será transmitido pelos enlaces 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 e 12.

(b) Durante a transmissão deste quadro, algum dos switches desta rede irá adicionar alguma entrada em sua tabela de encaminhamento? Se sim, quais switches e quais entradas?

Figura A.2: Questão 2 da prova de Redes de Computadores

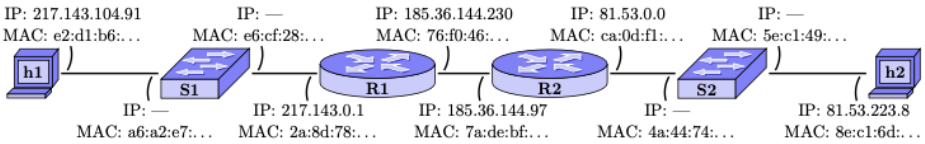
Resposta:

Os seguintes switches irão adicionar entradas em sua tabela de encaminhamento:

- Switch S1 — destino h3 / interface 1
- Switch S2 — destino h3 / interface 4
- Switch S3 — destino h3 / interface 5
- Switch S4 — destino h3 / interface 9

Questão 3 ..... 20 pontos

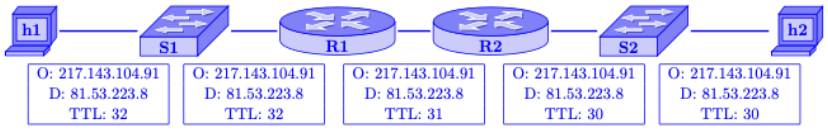
Considere a rede ilustrada a seguir, composta por duas estações (h1 e h2), dois switches (S1 e S2) e dois roteadores (R1 e R2). Suponha, para simplificar, que o protocolo Ethernet é utilizado em todas as comunicações na camada de enlace. No diagrama, são associados a cada interface os seus respectivos endereços IP e MAC (para o endereço MAC, somente os primeiros octetos).



Considere um datagrama IP que é enviado de h1 com destino a h2.

(a) Lembrando que o campo TTL (*Time to Live*) do cabeçalho IP é diminuído de uma unidade a cada salto, suponha que o datagrama é enviado com TTL inicial de 32. Para cada um dos 5 enlaces que o datagrama irá atravessar, determine o endereço origem, o endereço destino e o valor de TTL registrados no cabeçalho deste datagrama quando ele atravessa o enlace.

Resposta:



(b) Suponha que todas as tabelas ARP envolvidas estão devidamente preenchidas. Para cada um dos 5 enlaces, determine o endereço origem e o endereço destino dos quadros Ethernet que irão encapsular este datagrama quando ele atravessa o enlace.

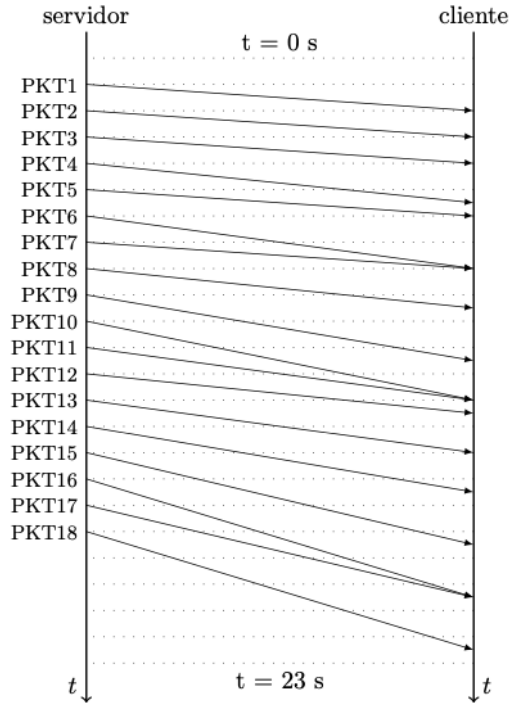
Resposta:

Questão 4 ..... 20 pontos

Considere a transmissão em *streaming* de pacotes multimídia de um servidor para um cliente, ilustrada no seguinte diagrama:

Figura A.3: Questão 3 da prova de Redes de Computadores

(continuação da questão 4) Redes de Computadores II



Suponha que o cliente utilize o seguinte mecanismo de bufferização: todos os pacotes são bufferizados assim que chegam e o cliente começa a reproduzir o vídeo somente 2.0 s após o primeiro pacote chegar, considerando como perdidos todos os pacotes que não chegarem a tempo de serem reproduzidos.

(a) Determine o instante de recepção de cada um dos pacotes.

(b) Determine o instante de reprodução escalonado para cada um dos pacotes.

Figura A.4: Questão 4 da prova de Redes de Computadores

Resposta:

PKT1 Recepção em t = 2.0 s, reprodução escalonada para t = 4.0 s
PKT2 Recepção em t = 3.0 s, reprodução escalonada para t = 5.0 s
PKT3 Recepção em t = 4.0 s, reprodução escalonada para t = 6.0 s
PKT4 Recepção em t = 5.5 s, reprodução escalonada para t = 7.0 s
PKT5 Recepção em t = 6.0 s, reprodução escalonada para t = 8.0 s
PKT6 Recepção em t = 8.0 s, reprodução escalonada para t = 9.0 s
PKT7 Recepção em t = 8.0 s, reprodução escalonada para t = 10.0 s
PKT8 Recepção em t = 9.5 s, reprodução escalonada para t = 11.0 s
PKT9 Recepção em t = 11.5 s, reprodução escalonada para t = 12.0 s
PKT10 Recepção em t = 13.0 s, reprodução escalonada para t = 13.0 s
PKT11 Recepção em t = 13.0 s, reprodução escalonada para t = 14.0 s
PKT12 Recepção em t = 13.5 s, reprodução escalonada para t = 15.0 s
PKT13 Recepção em t = 15.0 s, reprodução escalonada para t = 16.0 s
PKT14 Recepção em t = 16.5 s, reprodução escalonada para t = 17.0 s
PKT15 Recepção em t = 18.5 s, reprodução escalonada para t = 18.0 s
PKT16 Recepção em t = 20.5 s, reprodução escalonada para t = 19.0 s
PKT17 Recepção em t = 20.5 s, reprodução escalonada para t = 20.0 s
PKT18 Recepção em t = 22.5 s, reprodução escalonada para t = 21.0 s

(c) Algum pacote não será reproduzido com sucesso? Se sim, determine quais.

Resposta:

Sim, os pacotes 15, 16, 17 e 18 não serão reproduzidos com sucesso.

(d) Calcule a fração de pacotes perdidos para esta transmissão.

Resposta:

A fração de pacotes perdidos é dada pela quantidade de pacotes perdidos, dividida pelo total de pacotes transmitidos, resultando em uma perda de 4/18 = 22.2%.

Questão 5 ..... 20 pontos

O objetivo desta questão é compreender o funcionamento de algoritmos geradores de resumo de mensagem (*message digest*). Em particular, iremos focar nos padrões MD5 e SHA1, que são dois padrões muito conhecidos e utilizados para funções de hash $H(\cdot)$, que geram resumo de mensagem — isto é, dada uma mensagem M qualquer, cada um destes padrões gera um resumo. Este resumo pode ser utilizado para diversos fins, desde alocação eficiente em estruturas de dados até verificação de integridade de mensagens transmitidas em uma rede.

(a) Qual é o tamanho do resumo (em bits) gerado pelos padrões MD5 e SHA1? Este tamanho depende do tamanho da mensagem M?

Resposta:

O MD5 sempre gera resumos de 128 bits e o SHA1 sempre gera resumos de 160 bits. Estes tamanhos de resumo não dependem do tamanho de M.

(b) Qual é o tamanho mínimo que M deve ter (em bytes) para que as funções de hash

Figura A.5: Primeira parte da Questão 5 da prova de Redes de Computadores

MD5 e SHA1 possam ser utilizadas?

Resposta:

As funções de hash MD5 e SHA1 não determinam um tamanho mínimo para M. Logo, M pode ter qualquer tamanho.

(c) Determine o resumo da seguinte mensagem (sem aspas) quando utilizamos o padrão MD5 e o padrão SHA1: "Boas funções de hash são sensíveis a modificações!"[1] Apresente o resumo em formato hexadecimal.

Resposta:

O resumo MD5 desta frase é "`c82c8346c80c938fd2cc6a8a53033669`", e o seu resumo SHA1 é "`e5c0ca4f6a30d97f925645758a63c1e611cd9ea7`".

(d) Repita o item anterior para a seguinte mensagem (sem aspas): "Boas funções de hash são sensíveis a modificações." Repare que apenas um caractere foi trocado (ponto de exclamação para ponto final).

Resposta:

O resumo desta frase (com ponto final) utilizando a função de hash MD5 é "`a48d2299b9f26ff6d7d63f42108f9c01`", e utilizando o SHA1 é "`8a4fbf99ee7df1de9ff4bb7d9b27fbc5eae85064`".

(e) Compare os resumos obtidos. Mais especificamente, alinhe os resumos obtidos em cada uma das mensagens e, comparando cada caracter do resumo, determine o número de caracteres que são idênticos.

Resposta:

Comparando os resumos MD5 entre as duas frases, conferimos que somente o 17º dos 32 caracteres gerados pelo resumo é igual. Já comparando os resumos SHA1 entre as duas frases, conferimos que apenas as posições 17 e 23 dos 40 caracteres gerados pelo resumo são iguais. Ou seja, ao modificar um único byte da mensagem M, os resumos gerados são muito diferentes.

(f) Obtenha uma mensagem que tenha um resumo parecido com a mensagem do item (c) quando utilizamos MD5. Ou seja, determine M' tal que seu resumo tenha um número maior de bytes iguais ao resumo desta mensagem.

Qual é sua mensagem e quantos bytes são iguais?

Resposta:

O aluno deve procurar por uma mensagem M qualquer que tenha mais de dois caracteres iguais. Por exemplo, a mensagem "Estamos testando funções de hash!", cujo hash MD5 é "`54117121ad214cbdd64e823f71a3ca72`" e possui 2 caracteres iguais.

[1] Dica: no Linux utilize os programas *md5sum* e *sha1sum* para obter os respectivos resumos; cuidado para não inserir o caracter terminador \n.

Figura A.6: Segunda parte da Questão 5 da prova de Redes de Computadores

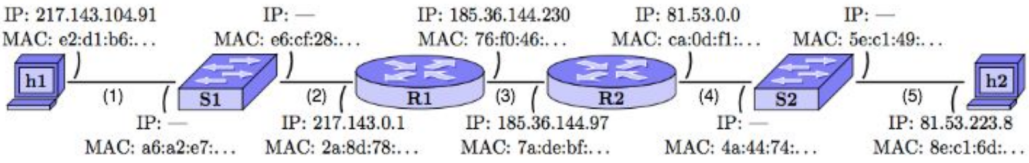
A.2 Exemplos de Respostas de Alunos

Nesta seção, apresentamos alguns exemplos de respostas fornecidas pelos alunos para as questões da prova, permitindo uma compreensão mais detalhada das respostas analisadas pelos LLMs.

Em sua maioria, as respostas dos alunos foram escritas no computador, diretamente em formato digital.

3

a. Para facilitar a compreensão, adicionei a legenda com números a figura abaixo:



(1) Origem: 217.143.104.91, Destino: 81.53.223.8, TTL: 32

(2) Origem: 217.143.104.91, Destino: 81.53.223.8, TTL: 32

(3) Origem: 217.143.104.91, Destino: 81.53.223.8, TTL: 31

(4) Origem: 217.143.104.91, Destino: 81.53.223.8, TTL: 30

(5) Origem: 217.143.104.91, Destino: 81.53.223.8, TTL: 30

b.

(1) Origem: e2:d1:b6:..., Destino: 2a:8d:78:...

(2) Origem: e2:d1:b6:..., Destino: 2a:8d:78:...

Figura A.7: Resposta de um aluno na questão 3 da prova de Redes de Computadores

4- a) b)

PKT1 Recepção em t= 2.0s, reprodução escalonada para t= 3.0s
PKT2 Recepção em t= 3.0s, reprodução escalonada para t= 4.0s
PKT3 Recepção em t= 4.0s, reprodução escalonada para t= 5.0s
PKT4 Recepção em t= 5.5s, reprodução escalonada para t= 6.0s
PKT5 Recepção em t= 6.0s, reprodução escalonada para t= 7.0s
PKT6 Recepção em t= 8.0s, reprodução escalonada para t= 8.0s
PKT7 Recepção em t= 8.0s, reprodução escalonada para t= 9.0s
PKT8 Recepção em t= 9.5s, reprodução escalonada para t= 10.0s
PKT9 Recepção em t= 11.5 s, reprodução escalonada para t= 11s
PKT10 Recepção em t=13s, reprodução escalonada para t= 12s
PKT11 Recepção em t=13 s, reprodução escalonada para t= 13s
PKT12 Recepção em t= 13.5 s, reprodução escalonada para t= 14s
PKT13 Recepção em t= 15s, reprodução escalonada para t= 15s
PKT14 Recepção em t= 16.5s, reprodução escalonada para t= 16s
PKT15 Recepção em t= 18.5s, reprodução escalonada para t= 17s
PKT16 Recepção em t= 20.5s, reprodução escalonada para t= 18s
PKT17 Recepção em t= 20.5s, reprodução escalonada para t= 19s
PKT18 Recepção em t= 22.5s, reprodução escalonada para t= 20s

c) Sim os pacotes 9, 10, 14, 15, 16, 17 e 18

d) A fração de pacotes perdidos é: 7/18 = 38,9%

Figura A.8: Resposta de um aluno na questão 4 da prova de Redes de Computadores

Questão 5

a-)
SHA-1 : tamanho do resumo 160 bits
MD5: tamanho do resumo 128 bits.
Não dependem do tamanho da mensagem.

b-)
1 byte.

c-)
SHA1: 5cb1c6e9fff86de89e3a5addd27c3e95adf47b88
MD5: 7289b2d941fd1ce079a7c213b0ac879b

d-)
SHA1: e8e1d4352d889f3fd621ad0fe7dd340a3a2ed2fb
MD5: 8968f2e256da927287ae939c675d9ace

e-)

Mensagem 1:
SHA1: 5 c b 1 c 6 e 9 f f f 8 6 d e 8 9 e 3 a 5 a d d d 2 7 c 3 e 9 5adf47b88
MD5: 7 2 8 9 b 2 d 9 4 1 f d 1 c e 0 7 9 a 7 c 2 1 3 b 0 a c 8 7 9 b

Número de caracteres idênticos: 5

Mensagem 2:
SHA1: e 8 e 1 d 4 3 5 2 d 8 8 9 f 3 f d 6 2 1 a d 0 f e 7 d d 3 4 0 a3a2ed2fb
MD5: 8 9 6 8 f 2 e 2 5 6 d a 9 2 7 2 8 7 a e 9 3 9 c 6 7 5 d 9 a c e

Número de caracteres idênticos: 3

Figura A.9: Resposta de um aluno na questão 5 da prova de Redes de Computadores

01. Resolução:

a. A transmissão será visível pelos seguintes itens: h5, h6, h8, s3, s6, s5, H3, H4, H1, S1 e S4.

b. Considere o desenho simplificado da rede apresentada para fins de esclarecimento. Note que podemos atingir no máximo 9 envios simultâneos. Este máximo é atingido quando ocorrem envios simultâneos para todos os extremos da rede. Um exemplo disso é o envio simultâneo de h1, s1, h4, h12, h10, s5, h5, h7 e s4.

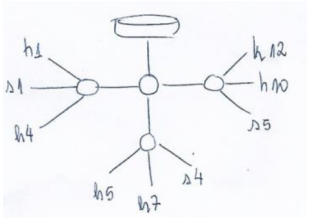


Figura A.10: Resposta de um aluno na questão 1 da prova de Redes de Computadores

Apêndice B

Prova de Física

B.1 Imagens das Questões da Prova

A seguir, apresentamos as imagens das questões da prova de Física.

Questão 1 (2,0 pontos)

Com base nas figuras esquemáticas abaixo, isole o bloco cinza claro (B) em cada um dos itens e indique **todas** as forças que atuam sobre ele. Considere o sistema sempre em equilíbrio. (faça o desenho no caderno de respostas, desenhos aqui não serão considerados)

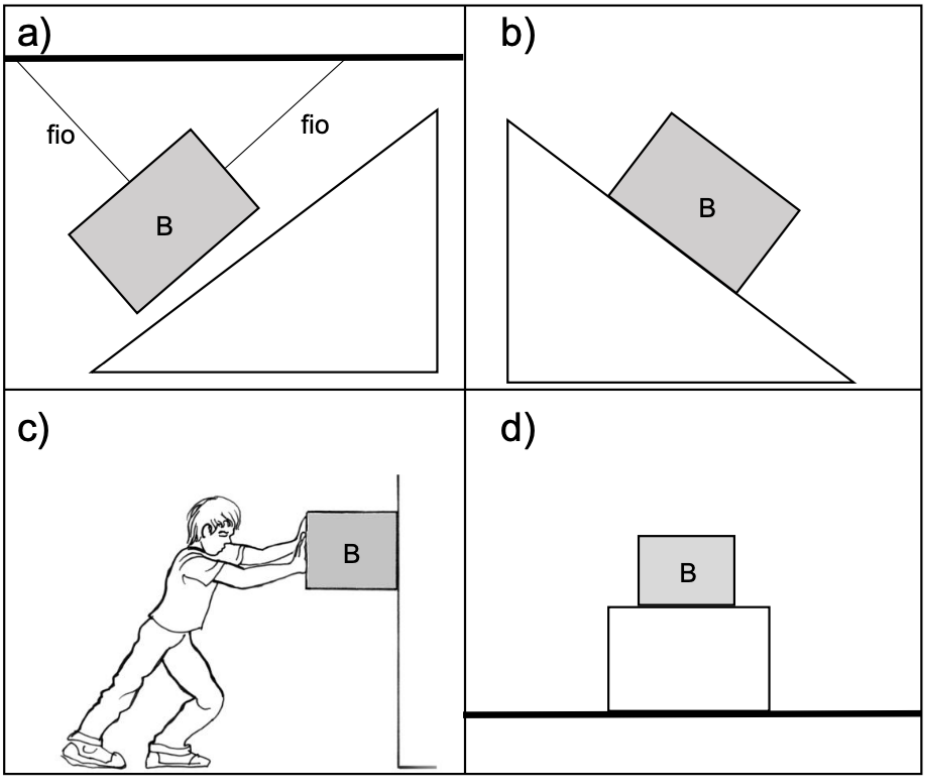
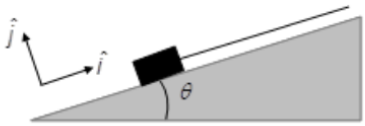


Figura B.1: Questão 1 da prova de Física

Questão 2 (2,0 pontos)

Um bloco de massa m = 4 kg está sobre uma superfície plana inclinada que forma um ângulo de 35º com a horizontal. O bloco sobe essa superfície com aceleração $\vec{a} = a\hat{\imath}$ puxado por uma corda que é paralela a esta superfície (ver figura). O módulo da força exercida pela corda é F = *35* N.

O coeficiente de atrito cinético entre o bloco e a superfície vale *0,3*. Considere a aceleração da gravidade como sendo g = *10* m/s² e despreze a resistência do ar.

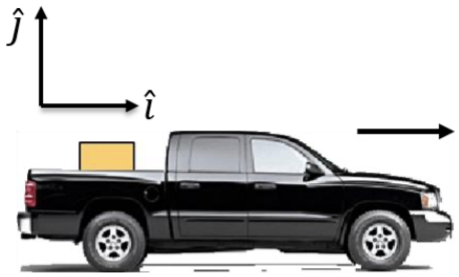


a) Isole a caixa e desenhe todas as forças que atuam sobre ela. Indique também onde atuam as forças de reação correspondentes.
b) Expresse todas as forças que atuam sobre o bloco, em termos dos vetores unitários.
c) Determine a aceleração com que o bloco sobre o plano inclinado, em termos dos vetores unitários.

Figura B.2: Questão 2 da prova de Física

Questão 3 (4,0 pontos)

Uma caminhonete com aceleração positiva na direção x está transportando uma caixa cuja massa é de m = 50 kg em uma estrada reta. A caixa está solta na carroceria da caminhonete, mas não desliza sobre ela. O coeficiente de atrito estático entre a caixa e a carroceria vale μ_e = 0,4. Despreze as forças que o ar exerce sobre a caixa. Considere a aceleração da gravidade como sendo g = 10 m/s². As direções x e y estão representadas na figura pelos seus vetores unitários î e ĵ, respectivamente



a) Desenhe a caixa isoladamente e coloque todas as forças que atuam sobre ela. Onde estão aplicadas as reações a essas forças?

b) Escreva a Segunda Lei de Newton na notação vetorial e na notação em componentes para a caixa (atenção: não confunda as componentes de uma força – que são números – com os vetores projetados nos eixos!)

c) Sabendo que o módulo da aceleração da caminhonete é a = *2,0* m/s², determine as componentes de todas as forças que atuam sobre a caixa.

d) Determine a maior intensidade possível da aceleração da caminhonete para que a caixa não deslize sobre a carroceria da mesma.

e) Uma vez que a caminhonete mantenha uma aceleração constante de a = *2,0* m/s², em quanto a sua velocidade escalar aumenta ao longo de 10 segundos?

Figura B.3: Questão 3 da prova de Física

Questão 4 (2,0 pontos)

Analise as afirmativas abaixo e indique NA FOLHA DE RESPOSTAS se cada uma é verdadeira (V) ou falsa (F). **Caso seja verdadeira, explique o porquê**. **Caso seja falsa, escreva a versão correta** da frase correspondente.

Respostas sem justificativas não serão consideradas!

a) Os anos bissextos acontecem em um intervalo de 6 anos.

b) O eclipse lunar ocorre quando a Lua se alinha exatamente entre a Terra e o Sol.

c) A órbita elíptica da Terra em torno do Sol é a causa das estações do ano.

d) As marés na Terra são afetadas pela Lua devido à interação magnética entre elas.

e) Os movimentos retrógrados dos planetas são um sinal do heliocentrismo.

Figura B.4: Questão 4 da prova de Física

B.2 Exemplos de Respostas de Alunos

Nesta seção, apresentamos alguns exemplos de respostas fornecidas pelos alunos para as questões da prova, permitindo uma compreensão mais detalhada das respostas analisadas pelos LLMs.

As respostas dos alunos foram escritas de forma livre em folha de papel e digitalizadas posteriormente.

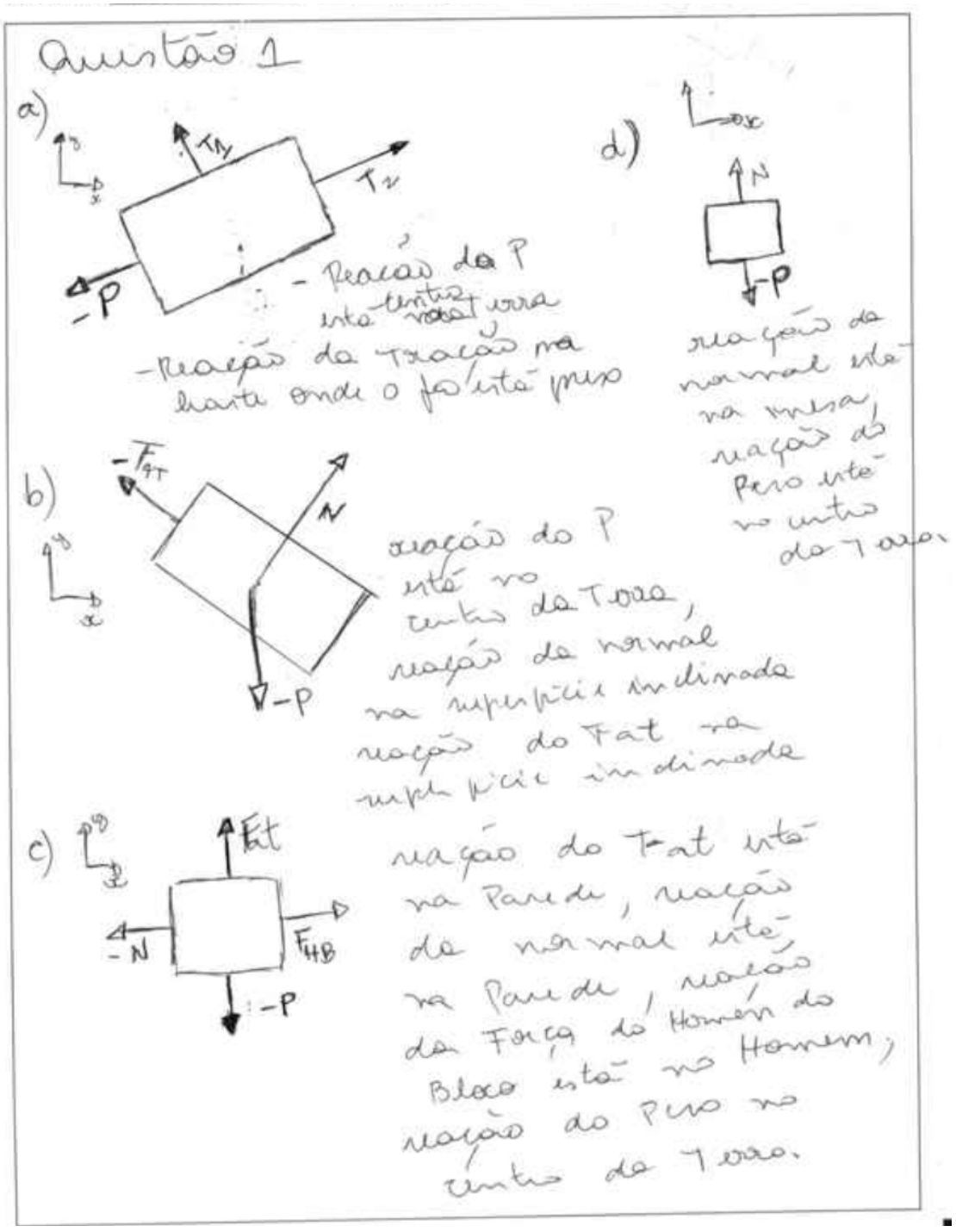


Figura B.5: Resposta de um aluno na questão 1 da prova de Física

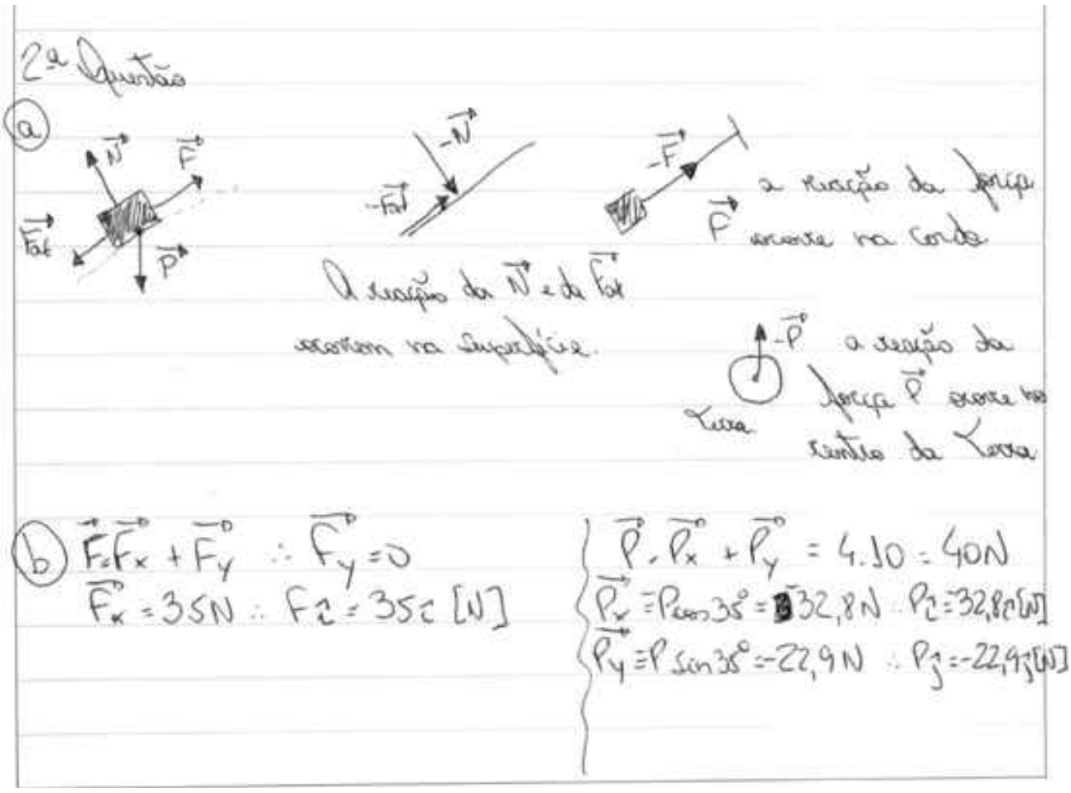
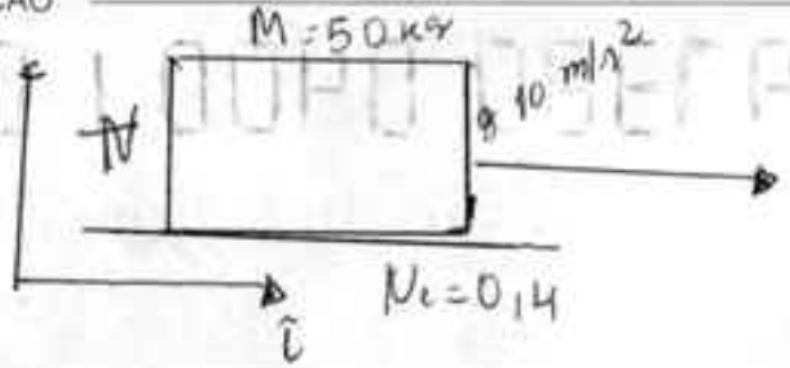


Figura B.6: Resposta de um aluno na questão 2 da prova de Física

CONTINUAÇÃO



b) $\vec{F}_n = m \cdot \vec{a} \cdot T$ $\quad g = 10\,m/s^2$

$\vec{F}_n = 0,4$ $\quad \vec{N} = x \cdot y$

$m = 50\,kg$

$\vec{a} = 2,0\,m/s^2$

c) $\vec{N} = \vec{a} = 2,0\,m/s^2 \cdot \vec{f}_n\,0,4$

$\vec{N} = 2,0\,m/s^2 \cdot 0,4$

$\vec{N} = 0,8\,m/s^2$

$\vec{a} = 0,8\,m/s^2$

d) $\vec{a} = 2,0\,m/s^2 : 0,8\,m/s^2$

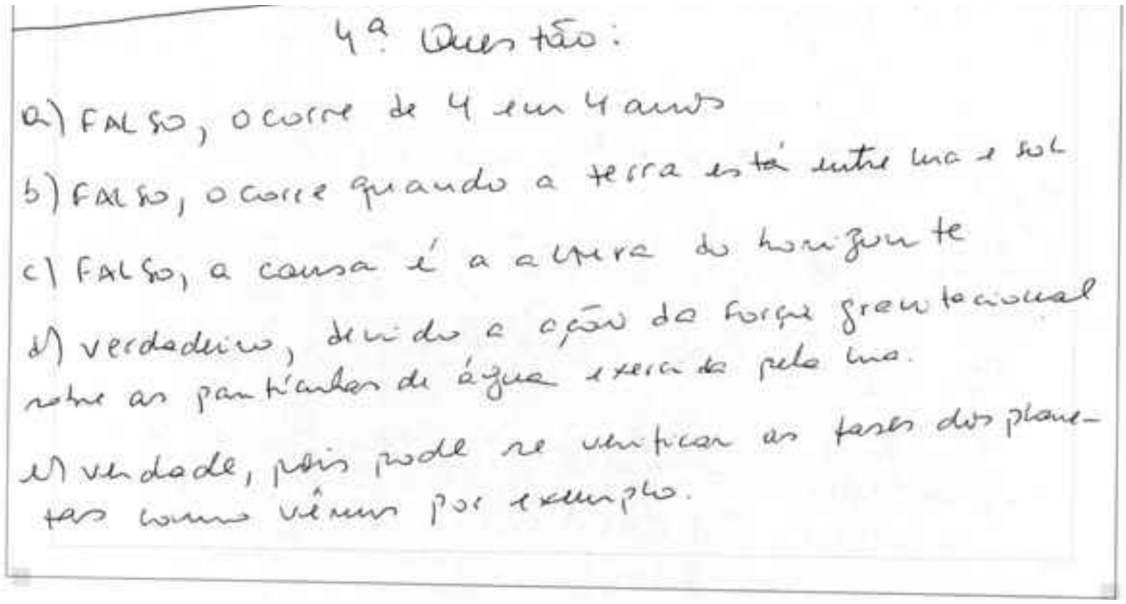
$\vec{a} = 4,0 + 16$

$\vec{a} = 20\,m/s$

e) $\vec{a} = 2,0\,m/s^2 + 10\,s$

$a = 20\,m/s^2 \cdot 10$

$a = 10\,m/s^2$


4ª Questão:

a) FALSO, ocorre de 4 em 4 anos

b) FALSO, ocorre quando a terra está entre lua e sol

c) FALSO, a causa é a altura do horizonte

d) verdadeiro, devido a ação da força gravitacional sobre as partículas de água exercida pela lua.

e) verdade, pois pode se identificar as fases dos planetas como vênus por exemplo.

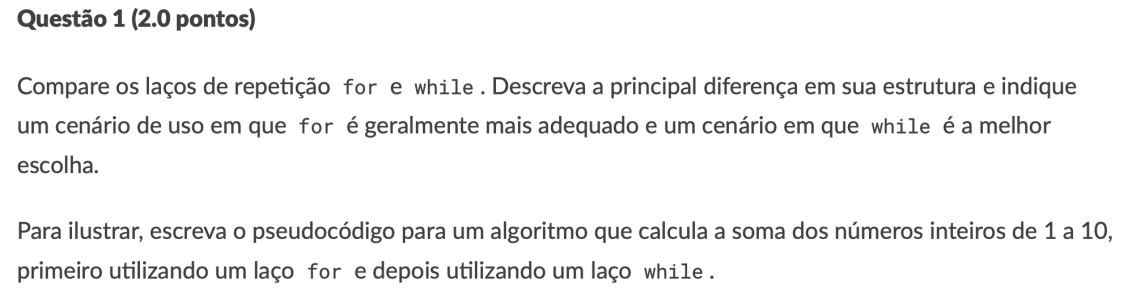
Figura B.8: Resposta de um aluno na questão 4 da prova de Física

Apêndice C

Prova Sintética de Introdução à Programação

Esta prova foi gerada inteiramente por IA (*Gemini-2.5-pro*) com o objetivo de isolar o componente de **raciocínio textual**, sem ruído de caligrafia ou digitalização. Tanto o enunciado quanto as respostas "dos alunos" são textos vetoriais perfeitos em PDF, possibilitando avaliar o desempenho dos LLMs quando o desafio visual é mínimo.

C.1 Imagens das Questões da Prova



Questão 1 (2.0 pontos)

Compare os laços de repetição `for` e `while`. Descreva a principal diferença em sua estrutura e indique um cenário de uso em que `for` é geralmente mais adequado e um cenário em que `while` é a melhor escolha.

Para ilustrar, escreva o pseudocódigo para um algoritmo que calcula a soma dos números inteiros de 1 a 10, primeiro utilizando um laço `for` e depois utilizando um laço `while`.

Figura C.1: Questão 1 — Diferença entre `for` e `while`

Questão 2 (2.0 pontos)

Discorra sobre os conceitos de **Recursão** e **Iteração** como duas abordagens para resolver problemas que envolvem repetição. Apresente as vantagens e desvantagens de cada método.

Como exemplo, descreva (sem precisar escrever o código completo) como uma função para calcular o fatorial de um número (ex: Fatorial de 5 = 5 * 4 * 3 * 2 * 1) funcionaria utilizando uma abordagem recursiva.

Figura C.2: Questão 2 — Recursão *vs.* iteração

Questão 3 (2.0 pontos)

Explique a diferença fundamental entre as estruturas de dados **Pilha (Stack)** e **Fila (Queue)**. Sua resposta deve incluir:

a) O princípio de funcionamento de cada uma (ex: LIFO, FIFO).
b) As operações principais de cada estrutura (ex: `push` / `pop` , `enqueue` / `dequeue`).
c) Um exemplo prático de aplicação computacional para cada uma delas.

Figura C.3: Questão 3 — Fila *vs.* pilha

Questão 4 (2.0 pontos)

A Programação Orientada a Objetos (POO) é um paradigma fundamental na computação moderna. Descreva os três pilares principais da POO: **Encapsulamento**, **Herança** e **Polimorfismo**.

Para cada pilar, forneça uma breve explicação teórica e uma analogia com o mundo real para ilustrar o conceito.

Figura C.4: Questão 4 — Quatro pilares da POO

Questão 5 (2.0 pontos)

Em muitas linguagens de programação, a memória de um programa é gerenciada em duas áreas principais: a **Pilha de Execução (Stack)** e o **Monte (Heap)**. Descreva a finalidade de cada uma dessas áreas de memória.

Cite pelo menos três diferenças importantes entre elas, considerando aspectos como o tipo de dado que armazenam, velocidade de acesso e como a alocação e desalocação de memória são gerenciadas.

Figura C.5: Questão 5 — *Stack vs. Heap*

C.2 Exemplos de Respostas de "Alunos"

As respostas foram sintetizadas para representar performances variadas — excelente, mediana e fraca — permitindo testar a sensibilidade do corretor a níveis de competência distintos.

Resposta à Questão 1:

`for` é para quando você tem uma sequência finita e quer passar por ela. É determinístico. `while` é para quando você está esperando um estado mudar. É sobre uma condição, não sobre uma contagem. `for` é "faça isso N vezes", `while` é "faça isso até que X aconteça".

Pseudocódigo:

Usando `for`:

```
funcao soma_com_for()
    soma = 0
    para i de 1 ate 10
        soma += i
    retornar soma
```

Usando `while`:

```
funcao soma_com_while()
    soma = 0
    i = 1
    enquanto i <= 10
        soma += i
```

Figura C.6: Resposta "A" (alto desempenho) na Questão 1

Resposta à Questão 3:

a) Pilha é LIFO (ultimo a entrar primeiro a sair) e Fila é FIFO (primeiro a entrar primeiro a sair). Eu acho que é isso, ou é o contrário? Acho que é isso.

b) push e pop... e enqueue. Não lembro a outra.

c) Pilha é o botao de voltar. Fila é fila de banco.

Figura C.7: Resposta "B" (desempenho mediano) na Questão 3

Resposta à Questão 4:

A POO é um paradigma de programação muito importante, baseado em objetos. Os pilares são o encapsulamento, a herança e o polimorfismo.

O **encapsulamento** tem a ver com encapsular os dados para ter mais segurança. A **herança** é quando uma classe herda coisas da outra, para reaproveitar o código. O polimorfismo eu não lembro.

Figura C.8: Resposta "C" (baixo desempenho) na Questão 4

Resposta à Questão 5:

A memória de um programa é gerenciada em duas áreas principais: a **Pilha de Execução (Stack)** e o **Monte (Heap)**.

Finalidade:

- **Pilha de Execução (Stack):** É uma região de memória usada para armazenar variáveis locais e chamadas de função. A memória é alocada e desalocada em uma ordem Last-In, First-Out (LIFO).
- **Monte (Heap):** É uma região de memória usada para alocação dinâmica de memória. A memória é alocada e desalocada de forma não ordenada.

Diferenças Importantes:

1. **Gerenciamento:** A Stack é gerenciada pelo compilador, enquanto a Heap é gerenciada pelo programador ou por um coletor de lixo.
2. **Velocidade de acesso:** O acesso à memória da Stack é mais rápido que o acesso à memória da Heap.

---

3. **Tamanho:** A Stack tem um tamanho fixo e limitado, enquanto a Heap tem um tamanho maior e é limitada pela memória virtual do sistema.

Figura C.9: Resposta "A" na Questão 5

Por serem vetoriais e semanticamente bem-formadas, essas respostas exercitam principalmente a **capacidade de avaliação conceitual** dos LLMs, servindo como linha de base para comparar com provas manuscritas ou ricas em diagramas.

Apêndice D

Prompts e JSON Schema Agentes LLM

D.1 Prompt Agent Avaliador

D.1.1 Schema JSON do output do Avaliador

```
GRADING_SCHEMA = {
    "name": "grading_schema",
    "schema": {
        "type": "object",
        "properties": {
            "questoes": {
                "type": "array",
                "description": "Uma lista de feedbacks para
                   cada questão.",
                "items": {
                    "type": "object",
                    "properties": {
                        "question_numero": {
                            "type": "integer",
                            "description": "O número da questã
                               o a qual o feedback se refere."
                        },
                        "feedback": {
                            "type": "string",
                            "description": "O feedback
                               detalhado direcionado ao
                               estudante."
                        },
```

```
                        "nota": {
                            "type": "number",
                            "description": "A nota final do
                               estudante na questão, com duas
                               casas decimais."
                        }
                    },
                    "required": ["question_numero", "feedback
                       ", "nota"],
                    "additionalProperties": False
                }
            }
        },
        "required": ["questoes"],
        "additionalProperties": False
    },
    "strict": True
}
```

D.1.2 System Prompt do Avaliador

```
GRADER_SYSTEM_PROMPT = """Você é um avaliador de provas
   especializado. Você fornece correções detalhadas,
   atribuindo notas precisas com base em critérios claros
e dando feedback construtivo para ajudar os estudantes a
   entenderem seus erros e melhorarem continuamente.

Você receberá as questões da prova, as rubricas (critérios) de
    avaliação e as respostas do estudante.

Seu objetivo é analisar a questão original, as rubricas de
   avaliação e a resposta do aluno e, com base nisso, fornecer
    um feedback para o aluno, seguido de
uma nota para a resposta.

Diretrizes a serem seguidas:
- As rubricas de avaliação determinam quantos pontos cada
   questão, alternativa e critério valem.
- A nota deve ter duas casas decimais, utilizando \".\" como
   separador. Exemplo: 2.00, 0.75, 1.50.
- Sua avaliação e nota deve ser baseada na rubrica de avaliaçã
```

```
   o e no gabarito.
- Seu feedback deve ser falando diretamente para o aluno ler e
    detalhado para o aluno entender, principalmente, o que ele
    errou.
- Seu feedback deve conter a nota final da questão, a nota por
    critério e a nota por alternativa (se houver).
- Você deve dar sua nota para cada critério ou alternativa da
   questão. A nota final deve ser a soma das notas em cada
   alternativa ou critério da questão.
- Preste particular atenção a imagens e diagramas nas
   respostas do estudante. Preste atenção nos detalhes.
- Mesmo que o estudante não responda a uma questão na prova,
   avalie essa questão, dando nota 0 para ele na questão.
- Não precisa falar "Olá" ou cumprimentar o estudante ao dar
   seu feedback, dê o feedback diretamente.
- Se o estudante desenhou ou escreveu algo que não faz sentido
   , não tente advinhar o que ele queria dizer, seja explícito
    dizendo que não ficou claro.
"""
```

D.2 Prompt Agent Revisor

D.2.1 Schema JSON do output do Revisor

```
REVIEW_SCHEMA = {
    "name": "review_schema",
    "schema": {
        "type": "object",
        "properties": {
            "quality_score": {
                "type": "integer",
                "description": "Nota de 1 a 5 para a qualidade
                    da avaliação do avaliador."
            },
            "overall_feedback": {
                "type": "string",
                "description": "Feedback geral sobre a avaliaç
                   ão do avaliador, o que melhorar, o que está
                    bom."
            }
        },
```

```
        "required": ["quality_score", "overall_feedback"],
        "additionalProperties": False
    },
    "strict": True
}
```

D.2.2 System Prompt do Revisor

```
REVIEWER_SYSTEM_PROMPT = """Você é um revisor crítico de
   avaliações. Seu papel:
- Ver a prova, o gabarito, as respostas do aluno e a avaliação
    feita pelo avaliador.
- Verificar se o feedback para o aluno é claro, alinhado com a
    rubrica/gabarito.
- Verificar se as notas dadas fazem sentido.
- Sugerir melhorias específicas se algo estiver errado,
   faltando ou puder ser mais claro.
- Retornar uma nota de qualidade (1-5) para a avaliação, onde
   5 é excelente.
- Se o estudante cometeu algum erro na questão e o avaliador n
   ão identificou, deixe claro que a questão precisa ser
   corrigida novamente.
- Se houver qualquer uma das questões que precisem de correção
    novamente, dê uma nota abaixo de 4 para o avaliador.
- Não é para você criticar o gabarito, trate o gabarito como a
    fonte da verdade.
Responda no schema especificado."""
```