

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Avaliação do Impacto de LLMs com RAG e Método Socrático no Aprendizado de Algoritmos e Programação de Computadores

Lucas M. Pereira

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Orientador
Prof. Dr. Edison Ishikawa

Brasília
2025

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Avaliação do Impacto de LLMs com RAG e Método Socrático no Aprendizado de Algoritmos e Programação de Computadores

Lucas M. Pereira

Monografia apresentada como requisito parcial
para conclusão do Curso de Computação — Licenciatura

Prof. Dr. Edison Ishikawa (Orientador)
CIC/UnB

Prof.a Dr.a Maristela Terto de Holanda
CIC/UnB

Prof. Dr. Leandro Galvão
IComp/UFAM

Prof. Dr. Jorge Henrique Cabral Fernandes
Coordenador do Curso de Computação — Licenciatura

Brasília, 11 de dezembro de 2025

Dedicatória

À minha esposa, Elis Calcagno Martins, por estar sempre ao meu lado, pelo apoio incondicional e por acreditar em mim mesmo nos momentos mais desafiadores.

À minha família, por todo o suporte, incentivo e compreensão durante esta jornada.

Aos meus amigos, pela presença constante e pelos momentos de leveza que tornaram esta caminhada mais leve.

Este trabalho é dedicado a vocês, que sempre estiveram comigo neste período de minha vida.

Agradecimentos

Agradeço primeiramente à Universidade de Brasília, por proporcionar um ambiente acadêmico de excelência e por representar um espaço de transformação social através da educação pública e de qualidade.

Ao corpo docente do curso de Licenciatura em Computação, por compartilhar conhecimento e inspirar o desenvolvimento do pensamento crítico e científico ao longo de toda a graduação.

Ao meu orientador, Prof. Dr. Edison Ishikawa, pela confiança depositada neste trabalho, pela orientação precisa e pelos valiosos insights que enriqueceram esta pesquisa. Sua dedicação ao ensino e à pesquisa é fonte constante de inspiração.

À banca avaliadora, Prof.ª Dr.ª Maristela Torto de Holanda e Prof. Dr. Leandro Galvão, pelas contribuições e pelo tempo dedicado à avaliação deste trabalho, que certamente o enriqueceram significativamente.

Aos estudantes da disciplina de Algoritmos e Programação de Computadores que participaram voluntariamente do experimento, tornando esta pesquisa possível.

Por fim, agradeço a todos que, direta ou indiretamente, contribuíram para a realização deste trabalho e para minha formação acadêmica.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Este trabalho investiga o impacto do uso de Grandes Modelos de Linguagem (LLMs) com Retrieval-Augmented Generation (RAG) e método socrático no aprendizado inicial de Algoritmos e Programação de Computadores (APC). O uso indiscriminado de LLMs pode levar à dependência em respostas prontas, prejudicando a construção do conhecimento. Foi desenvolvido um tutor inteligente que utiliza LLM com RAG aliado ao método socrático, comparando sua eficácia com um LLM sem RAG que fornece respostas diretas. O experimento foi conduzido com estudantes da disciplina de APC da Universidade de Brasília ao longo de três semanas, coletando 144 respostas ao questionário de percepção e 201 respostas de desempenho. Os resultados revelam um *tradeoff* entre usabilidade e pedagogia: a IA Azul (RAG + método socrático) foi significativamente superior em guiar sem respostas diretas (+0,99 pontos, $p < 0{,}001$, teste de Wilcoxon pareado, n=35) e promover compreensão de conceitos avançados (+0,54 pontos, $p = 0{,}023$), enquanto a IA Verde (respostas diretas) apresentou melhor usabilidade (-0,28 pontos, $p = 0{,}035$, teste de Wilcoxon pareado) e maior desempenho imediato (7,45 vs 4,27). Não foram encontradas diferenças significativas em satisfação geral ($p = 0{,}810$), aprendizado percebido ($p = 0{,}891$) e engajamento ($p = 0{,}489$). O perfil do estudante não influenciou significativamente a percepção, e a implementação demonstrou viabilidade econômica (R$ 0,0022 por requisição). Os resultados indicam que cada abordagem é mais adequada para objetivos pedagógicos específicos: IA Azul para aprendizado profundo e autonomia, IA Verde para conclusão eficiente de tarefas.

Palavras-chave: Aprendizado de Programação, Grandes Modelos de Linguagem, Retrieval-Augmented Generation, Método Socrático, Educação em Computação

Abstract

This work investigates the impact of using Large Language Models (LLMs) with Retrieval-Augmented Generation (RAG) and the Socratic method on initial learning of Algorithms and Computer Programming (APC). Indiscriminate use of LLMs can lead to dependence on ready-made answers, hindering knowledge construction. An intelligent tutor was developed that uses LLM with RAG combined with the Socratic method, comparing its effectiveness with an LLM without RAG that provides direct answers. The experiment was conducted with students from the APC course at the University of Brasília over three weeks, collecting 144 responses to the perception questionnaire and 201 performance responses. Results reveal a tradeoff between usability and pedagogy: Blue AI (RAG + Socratic method) was significantly superior in guiding without direct answers (+0.99 points, $p < 0.001$, paired Wilcoxon test, n=35) and promoting understanding of advanced concepts (+0.54 points, $p = 0.023$), while Green AI (direct answers) showed better usability (-0.28 points, $p = 0.035$, paired Wilcoxon test) and higher immediate performance (7.45 vs 4.27). No significant differences were found in overall satisfaction ($p = 0.810$), perceived learning ($p = 0.891$), and engagement ($p = 0.489$). Student profile did not significantly influence perception, and implementation demonstrated economic viability (R$ 0.0022 per request). Results indicate that each approach is more suitable for specific pedagogical objectives: Blue AI for deep learning and autonomy, Green AI for efficient task completion.

Keywords: Programming Learning, Large Language Models, Retrieval-Augmented Generation, Socratic Method, Computer Science Education

Sumário

1 Introdução **1**

1.1 Pergunta de Pesquisa 2
1.2 Hipótese 2
1.3 Justificativa 3
1.4 Objetivos 3
1.4.1 Objetivo Geral 3
1.4.2 Objetivos Específicos 3
1.5 Metodologia 4
1.6 Organização do Trabalho 4

2 Revisão Bibliográfica **5**

2.1 Dificuldades no Ensino de Algoritmos e Programação de Computadores . . 5
2.2 Grandes Modelos de Linguagem na Educação 6
2.3 Retrieval-Augmented Generation (RAG) 6
2.4 Engenharia de Prompt 8
2.4.1 Prompt direto (tiro zero) 8
2.4.2 Prompt de Poucos Exemplos 8
2.4.3 Prompt de Cadeia de Raciocínio 9
2.4.4 Engenharia Automática de Prompt 9
2.5 Método Socrático na Educação 9
2.6 Chatbots Especializados como Tutores 10
2.7 Design Science Research 11

3 Metodologia **12**

3.1 Abordagem Metodológica 12
3.2 Desenho Experimental 12
3.3 Participantes 13
3.4 Instrumentos de Coleta de Dados 13
3.4.1 Questões de Programação 13

3.4.2 Questionário de Percepção . . . 15
3.4.3 Telemetria do Sistema . . . 15
3.5 Procedimentos . . . 16
3.5.1 Seleção e Balanceamento das Questões . . . 16
3.6 Análise de Dados . . . 16
3.7 Considerações Éticas . . . 17
3.8 Declaração de Uso de Inteligência Artificial Generativa . . . 17

4 Desenvolvimento do Tutor Inteligente 19
4.1 Visão Geral do Sistema . . . 19
4.2 Arquitetura do Sistema . . . 19
4.2.1 Componentes Principais . . . 20
4.3 Implementação dos Cenários . . . 26
4.3.1 Cenário IA Verde (Sem RAG, Respostas Diretas) . . . 26
4.3.2 Cenário IA Azul (Com RAG e Método Socrático) . . . 26
4.4 Deploy e Infraestrutura . . . 27
4.5 Interface do Usuário . . . 27

5 Resultados 28
5.1 Caracterização dos Participantes . . . 28
5.2 Desempenho nas Tarefas . . . 29
5.2.1 Notas Médias . . . 29
5.2.2 Taxa de Sucesso . . . 30
5.2.3 Tempo de Resolução . . . 30
5.3 Percepção dos Estudantes . . . 30
5.3.1 Escala LOES-S: Aprendizado, Engajamento e Qualidade . . . 30
5.3.2 Questões Adicionais Específicas . . . 32
5.4 Análise por Perfil do Estudante . . . 34
5.4.1 Impacto do Conhecimento Prévio . . . 34
5.4.2 Correlação com Desempenho na Primeira Prova . . . 34
5.4.3 Análise por Grupos de Desempenho na Avaliação Formal . . . 35
5.4.4 Síntese da Análise por Perfil . . . 36
5.5 Telemetria e Custos Operacionais . . . 36
5.5.1 Dados de Uso por Dia de Experimento . . . 37
5.5.2 Análise dos Dados . . . 37
5.6 Comparação Detalhada por Questão . . . 38
5.7 Resumo dos Resultados Principais . . . 40

6 Discussão **42**
6.1 Interpretação dos Resultados 42
6.2 Resolução de Problemas e Aprendizado Conceitual 42
6.3 Usabilidade versus Pedagogia 43
6.4 Desempenho versus Aprendizado 44
6.5 Implicações Pedagógicas 44
6.6 Limitações do Estudo 45
6.6.1 Retenção de Conhecimento 45
6.6.2 Efeito de Ordem 45
6.6.3 Contexto Específico 45
6.6.4 Restrição Temporal 46
6.6.5 Contexto de Aplicação Controlado 46
6.6.6 Viabilidade Econômica 46
6.6.7 Questões Diferentes entre Grupos 47
6.7 Relação com Trabalhos Correlatos 47

7 Conclusão **49**
7.1 Resumo do Trabalho 49
7.2 Síntese dos Resultados 49
7.3 Contribuições 50
7.3.1 Contribuições Teóricas 50
7.3.2 Contribuições Práticas 50
7.4 Implicações para o Ensino de Programação 51
7.5 Limitações 51
7.6 Trabalhos Futuros 51
7.6.1 Avaliação de Longo Prazo 51
7.6.2 Replicação e Generalização 51
7.6.3 Melhorias Técnicas 52
7.6.4 Análises Adicionais 52
7.6.5 Abordagens Híbridas 52
7.6.6 Integração com Plataformas Educacionais 53
7.7 Considerações Finais 53

Referências **55**

Apêndice **58**

A Termo de Consentimento Livre e Esclarecido **59**

B Questões Utilizadas no Experimento **61**
B.1 Questão 1 - IA Verde: Espaços na String 61
B.2 Questão 1 - IA Azul: Média Ponderada 61
B.3 Questão 2 - IA Verde: Dividindo Doces 62
B.4 Questão 2 - IA Azul: Calcule as Médias 63
B.5 Questão 3 - IA Verde: Ajuda Necessária 63
B.6 Questão 3 - IA Azul: Lotação de Ônibus 64

C Questionário LOES-S Adaptado **66**

D Prompts Utilizados nos Cenários de IA **69**
D.1 Cenário IA Verde (Sem RAG, Respostas Diretas) 69
D.1.1 Prompt de Sistema 69
D.1.2 Template de Prompt Humano 69
D.2 Cenário IA Azul (Com RAG e Método Socrático) 69
D.2.1 Prompt de Sistema 69
D.2.2 Template de Prompt Humano 70
D.3 Prompts Auxiliares (IA Azul) 70
D.3.1 Prompt de Validação (NO_CODE_PROMPT) 70
D.3.2 Prompt de Roteamento (ROUTER_PROMPT) 71
D.3.3 Prompt HYDE (HYDE_PROMPT) 71

E Interfaces do Usuário **73**
E.1 Tela de Login 73
E.2 Interface Administrativa 74
E.3 Interface de Chat - IA Verde 74
E.4 Interface de Chat - IA Azul 75

Anexo **76**

I Plano de Ensino da Disciplina APC **77**

II Tabela de Preços do Google Gemini 2.0 Flash **80**

Lista de Figuras

4.1 Arquitetura do sistema tutor inteligente baseado em LLM com RAG 20
4.2 Estrutura modular da aplicação Django 21
4.3 Grafo LangGraph para IA Azul (com RAG e método socrático) 22
4.4 Grafo LangGraph para IA Verde (sem RAG, respostas diretas) 23
4.5 Diagrama MER do banco de dados do tutor inteligente 25

5.1 Métricas de desempenho: (esquerda) notas nas questões por tipo de IA, (centro) tempo de conclusão por tipo de IA, (direita) taxa de sucesso (nota $\geq 5/10$). 29
5.2 Aprendizado, Engajamento e Qualidade: comparação multidimensional entre IA Verde e IA Azul baseada na escala LOES-S. 31
5.3 Resolução de problemas sem respostas diretas: (esquerda) violin plot mostrando distribuição das pontuações, (direita) distribuição percentual das respostas. 33
5.4 Satisfação geral por tipo de IA: (esquerda) box plot comparando as distribuições, (direita) distribuição das notas de satisfação. 33
5.5 Heatmap comparando IA Verde e IA Azul em todas as questões do questionário. Cores mais verdes indicam pontuações mais altas, cores mais vermelhas indicam pontuações mais baixas. 39

E.1 Tela de login do tutor inteligente 73
E.2 Interface administrativa para seleção da versão de IA 74
E.3 Interface de chat com IA Verde (respostas diretas) 75
E.4 Interface de chat com IA Azul (método socrático e RAG) 76

II.1 Tabela de preços do Google Gemini 2.0 Flash. 81

Lista de Tabelas

5.1 Resultados por categoria da escala LOES-S (análise pareada, n=35) 31
5.2 Dados de telemetria e custos operacionais do experimento 37

Capítulo 1

Introdução

No curso noturno de Licenciatura em Ciência da Computação da Universidade de Brasília, a disciplina de Algoritmos e Programação de Computadores (APC) ocupa uma posição estratégica como primeira disciplina de programação na cadeia curricular. Para muitos estudantes, APC representa o primeiro contato com programação, o que intensifica as dificuldades iniciais de aprendizado [1], especialmente considerando que a disciplina está na grade curricular dos calouros—estudantes que ingressam no primeiro semestre e ainda se adaptam ao ambiente universitário. Essas dificuldades se refletem em um índice preocupante de reprovação: Silveira et al. [2] analisaram dados de desempenho e aspectos socioeconômicos de estudantes em cursos introdutórios de ciência da computação na UnB entre 2012 e 2019, identificando taxa de reprovação superior a 50% em APC. Este problema transcende a própria disciplina, pois APC é pré-requisito fundamental para disciplinas subsequentes como Estrutura de Dados, que por sua vez bloqueiam outras disciplinas avançadas do currículo.

O contexto do curso noturno agrava ainda mais o desafio. Os estudantes calouros frequentemente conciliam estudos com trabalho e enfrentam limitações de tempo, dificultando a adaptação ao ambiente universitário e o desenvolvimento do pensamento computacional necessário para o sucesso em programação. Grandes Modelos de Linguagem (LLMs) surgem como uma oportunidade para apoiar o aprendizado de programação. No entanto, estudos demonstram que o uso excessivo de LLMs pelos alunos para geração de código e depuração está negativamente correlacionado com as notas finais dos estudantes [3]. Hellas et al. [4] observaram que LLMs frequentemente fornecem soluções completas mesmo quando instruídos a não fazê-lo, o que pode prejudicar o desenvolvimento da autonomia dos estudantes. O uso indiscriminado pode levar à dependência em respostas prontas, prejudicando a construção do conhecimento e criando uma falsa sensação de compreensão.

Técnicas como Retrieval-Augmented Generation (RAG) [5] e abordagens pedagógicas

baseadas no método socrático emergem como alternativas promissoras [6, 7]. RAG é uma técnica que combina memória paramétrica de LLMs com memória não-paramétrica explícita através de um índice vetorial de documentos [5]. Esta arquitetura permite recuperar informações específicas do domínio de APC, garantindo que as respostas estejam alinhadas com o conteúdo e metodologia da disciplina. O método socrático, por sua vez, é uma abordagem pedagógica que utiliza perguntas guiadas para conduzir o estudante na construção do próprio raciocínio. Esta técnica estimula reflexão ativa e promove compreensão mais profunda e duradoura dos conceitos fundamentais. Pesquisas indicam que chatbots especializados podem atuar como tutores eficazes quando utilizados de forma moderada, mas o uso excessivo não leva a melhorias adicionais no desempenho acadêmico [8].

Embora LLMs tenham sido utilizados como ferramentas de apoio no ensino de programação, ainda não está claro se a integração de RAG combinada ao método socrático oferece aprendizagem mais eficaz do que LLMs sem RAG (que fornecem respostas diretas). Esta questão é particularmente relevante no contexto do curso noturno de Licenciatura em Ciência da Computação, onde os desafios específicos dos estudantes calouros tornam crítico o desenvolvimento de ferramentas que promovam aprendizado efetivo e eficiente. Uma ferramenta de tutoria inteligente que combine RAG com método socrático pode oferecer suporte personalizado e pedagogicamente adequado para esses estudantes.

1.1 Pergunta de Pesquisa

Esta pesquisa investiga o papel de técnicas avançadas de inteligência artificial no apoio ao aprendizado de programação. A pergunta central que orienta esta pesquisa é: *O uso de LLMs com RAG e método socrático contribui para uma aprendizagem mais eficaz em APC, em comparação com LLMs sem RAG que fornecem respostas diretas?* Essa questão busca ir além do uso indiscriminado de LLMs, que frequentemente fornecem soluções prontas e podem criar dependência, para investigar se abordagens pedagógicas estruturadas—que combinam recuperação de contexto específico do domínio (RAG) com questionamento guiado (método socrático)—promovem maior compreensão conceitual, autonomia e desenvolvimento de pensamento computacional nos estudantes.

1.2 Hipótese

A combinação de LLM com RAG e método socrático favorece o aprendizado em APC ao guiar o aluno no processo de resolução de problemas, em vez de fornecer respostas prontas. Especificamente, hipotetiza-se que estudantes utilizando LLM com RAG e método socrático (IA Azul) apresentarão, em comparação com estudantes utilizando LLM sem

RAG (IA Verde): (1) maior desempenho nas questões de programação (notas e taxa de sucesso); (2) melhor percepção quanto à clareza, utilidade, engajamento e eficácia do sistema; e (3) maior desenvolvimento de autonomia e pensamento computacional, refletido em menor dependência de respostas diretas.

1.3 Justificativa

A justificativa baseia-se em três pilares:

1. **RAG contextualizado:** Recupera informações específicas do domínio de APC, garantindo alinhamento com o conteúdo e metodologia da disciplina.
2. **Método socrático:** Estimula os alunos a refletirem e construírem o próprio raciocínio através de perguntas guiadas, promovendo internalização do pensamento computacional.
3. **Redução de dependência:** Ao não fornecer respostas diretas, a abordagem evita que estudantes desenvolvam dependência em soluções prontas, problema comum no uso indiscriminado de LLMs [3].

Reconhece-se que estudantes do curso noturno, com limitações de tempo, podem preferir respostas diretas para eficiência imediata. Entretanto, embora respostas diretas facilitem a conclusão rápida de tarefas, podem comprometer a construção da compreensão profunda necessária para sucesso sustentável na disciplina e em cursos subsequentes. A hipótese é que existirá um compromisso entre usabilidade e pedagogia, com cada abordagem oferecendo vantagens específicas dependendo dos objetivos de aprendizado.

1.4 Objetivos

1.4.1 Objetivo Geral

Esta pesquisa tem como objetivo geral investigar se LLMs com RAG e método socrático contribuem para uma aprendizagem mais eficaz em APC, em comparação com LLMs sem RAG (respostas diretas) ou ausência de suporte.

1.4.2 Objetivos Específicos

Para alcançar esse objetivo, foram estabelecidos os seguintes objetivos específicos:

- Desenvolver um tutor inteligente que utilize LLM com RAG aliado ao método socrático para auxiliar no ensino de APC.

- Comparar a eficácia do tutor (LLM com RAG + método socrático) em relação a um LLM sem RAG (respostas diretas).
- Aplicar experimentos com alunos do curso noturno de Licenciatura, utilizando questões referentes ao conteúdo da primeira prova de APC (detalhes sobre a disciplina disponíveis no Anexo I).
- Coletar e analisar a percepção dos alunos quanto à clareza, utilidade, engajamento e eficácia das respostas.

1.5 Metodologia

Esta pesquisa aplicada fundamenta-se na metodologia Design Science Research (DSR) [9], com desenvolvimento e avaliação de um artefato (tutor inteligente). O experimento foi conduzido ao longo de três semanas com estudantes da disciplina de APC, comparando dois cenários: (1) IA Verde—LLM sem RAG fornecendo respostas diretas, e (2) IA Azul—LLM com RAG e método socrático guiando através de perguntas. A coleta de dados incluiu 144 respostas ao questionário de percepção baseado na escala LOES-S [10] complementada por questões adicionais, 201 respostas de desempenho em questões de programação, e telemetria do sistema. A análise combina dados quantitativos (desempenho, tempo de resolução, custos) e qualitativos (percepção dos estudantes, feedback textual).

1.6 Organização do Trabalho

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta a fundamentação teórica sobre LLMs, RAG, método socrático e ensino de programação. O Capítulo 3 descreve o desenho experimental e a metodologia de pesquisa. O Capítulo 4 detalha a arquitetura e implementação do tutor inteligente. O Capítulo 5 apresenta os resultados obtidos no experimento. O Capítulo 6 discute os resultados e suas implicações. Por fim, o Capítulo 7 apresenta as conclusões e trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Este capítulo apresenta a fundamentação teórica necessária para compreender o contexto e as bases conceituais desta pesquisa. São abordados temas relacionados ao ensino de programação, grandes modelos de linguagem, técnicas de RAG e o método socrático aplicado à educação.

2.1 Dificuldades no Ensino de Algoritmos e Programação de Computadores

O ensino de disciplinas introdutórias de programação, como Algoritmos e Programação de Computadores (APC), apresenta desafios significativos em cursos de Computação ao redor do mundo, especialmente para estudantes iniciantes. Qian e Lehman [1] realizaram uma revisão sistemática da literatura internacional identificando dificuldades comuns enfrentadas por estudantes em contextos diversos, incluindo:

- Compreensão de conceitos fundamentais como variáveis, tipos de dados e estruturas de controle.
- Dificuldades com estruturas de decisão e repetição.
- Internalização do pensamento computacional.
- Transferência de conhecimento entre diferentes contextos de programação.

Essas dificuldades podem levar à frustração e desengajamento dos estudantes, impactando negativamente o aprendizado e a retenção de conhecimento.

2.2 Grandes Modelos de Linguagem na Educação

Grandes Modelos de Linguagem (LLMs) têm revolucionado diversas áreas, incluindo a educação. A inteligência artificial na educação, especialmente através de LLMs, promete atender às necessidades individuais dos estudantes, aumentar o engajamento e otimizar os resultados de aprendizagem através de personalização [11]. Kasneci et al. [12] apresentam uma análise abrangente das oportunidades e desafios dos LLMs na educação, destacando que, embora representem um avanço significativo, requerem que professores e alunos desenvolvam competências específicas para compreender tanto a tecnologia quanto suas limitações. Os autores argumentam que uma estratégia clara dentro dos sistemas educacionais e uma abordagem pedagógica com foco em pensamento crítico são necessários para integrar adequadamente LLMs nos currículos de ensino. Estudos posteriores reforçam que os LLMs oferecem potencial significativo através de abordagens lúdicas e baseadas em jogos. [13]

A confiança dos estudantes em ferramentas de IA generativa como ChatGPT está diretamente relacionada ao otimismo tecnológico e à percepção de utilidade, fatores que influenciam significativamente a adoção dessas tecnologias no contexto educacional [14].

No contexto específico de programação, Hellas et al. [4] exploraram como LLMs respondem a pedidos de ajuda de programadores iniciantes. Seus resultados mostram que, embora GPT-3.5 consiga identificar pelo menos um problema no código dos estudantes em 90% dos casos, há limitações significativas: encontrar todos os problemas ocorre apenas em 57% dos casos, e falsos positivos, ou seja, situações em que o LLM identifica problemas inexistentes no código ou alucina soluções para problemas inexistentes, são comuns (40% de chance). Os LLMs frequentemente fornecem soluções completas mesmo quando instruídos a não fazê-lo, o que pode prejudicar o aprendizado autônomo dos estudantes.

No entanto, o uso indiscriminado de LLMs pode levar à dependência em respostas prontas. Jost et al. [3] demonstraram que o uso excessivo de LLMs pelos alunos para geração de código e depuração está negativamente correlacionado com as notas finais dos estudantes, enquanto o uso para busca de explicações adicionais apresenta correlação menos significativa.

2.3 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) [5] é uma técnica que combina recuperação de informações com geração de texto através de modelos que integram memória paramétrica pré-treinada (o modelo de linguagem) com memória não-paramétrica explícita (um índice vetorial de documentos). Lewis et al. [5] propuseram RAG como solução para tarefas de

processamento de linguagem natural (PLN) intensivas em conhecimento, demonstrando que modelos RAG superam tanto modelos seq2seq paramétricos quanto arquiteturas específicas para tarefa de recuperação e extração. A técnica permite que modelos de linguagem acessem informações específicas de um domínio através de um mecanismo de recuperação diferenciável, melhorando a precisão, relevância e factualidade das respostas geradas, além de fornecer proveniência para as decisões do modelo.

No contexto educacional, o RAG pode ser utilizado para recuperar informações relevantes do material didático do curso, garantindo que o tutor inteligente forneça respostas baseadas no conteúdo específico da disciplina, em vez de depender apenas do conhecimento pré-treinado do modelo. Esta abordagem é especialmente valiosa para domínios especializados como o ensino de programação, onde é essencial que as respostas estejam alinhadas com o material e a metodologia da disciplina.

Estudos recentes têm investigado a aplicação de RAG especificamente no ensino superior de ciência da computação. Dakshit [15] conduziu o primeiro estudo a coletar feedback de professores sobre aplicações de RAG baseadas em LLMs na educação, desenvolvendo sistemas RAG para assistentes virtuais de ensino e ferramentas de apoio pedagógico. O estudo revelou que, embora professores reconheçam o potencial dos sistemas RAG como assistentes de ensino virtuais, existem barreiras e recursos específicos necessários para sua implementação em larga escala. As principais barreiras identificadas incluem: preocupações com confiabilidade dos sistemas (imprecisões e alucinações nas respostas geradas), implicações éticas (potencial para plágio e necessidade de conformidade com políticas acadêmicas), e limitações de recursos (tempo, equipe técnica qualificada e acesso a ferramentas necessárias). Shan [16] desenvolveu LearnRAG, um arcabouço de código aberto de RAG para sistemas de aprendizado adaptativo, demonstrando escalabilidade, eficiência no aumento do engajamento dos estudantes e redução da carga de trabalho dos educadores. Estes trabalhos destacam tanto o potencial quanto os desafios práticos da implementação de sistemas RAG educacionais.

Para melhorar a qualidade da recuperação em sistemas RAG, a estratégia HyDE (Hypothetical Document Embeddings) [17] pode ser aplicada. HyDE é uma técnica avançada de recuperação zero-shot que melhora significativamente a precisão da recuperação sem necessidade de rótulos de relevância. No contexto educacional, HyDE pode ser integrada aos sistemas RAG para aumentar a relevância das informações recuperadas do material didático. A técnica funciona instruindo um modelo de linguagem a gerar um documento hipotético a partir da consulta, que captura padrões de relevância mesmo que contenha detalhes imprecisos. Este documento hipotético é então codificado em um vetor de incorporação, que é usado para recuperar documentos reais similares no corpus. O codificador denso atua como um filtro, eliminando detalhes incorretos do documento gerado

e ancorando a busca em documentos reais do corpus. Gao et al. [17] demonstraram que HyDE supera significativamente recuperadores não supervisionados tradicionais e apresenta desempenho comparável a recuperadores ajustados finamente, em diversas tarefas como busca na web, perguntas e respostas e verificação de fatos.

2.4 Engenharia de Prompt

Engenharia de Prompt é a prática de elaborar instruções eficazes para modelos de linguagem, de modo a obter respostas desejadas para tarefas específicas [18, 19]. O design cuidadoso de prompts é fundamental para direcionar o comportamento de LLMs, influenciando diretamente a qualidade, relevância e utilidade das respostas geradas [20]. Esta seção apresenta as principais técnicas de prompt que fundamentam o desenvolvimento de sistemas baseados em LLMs.

2.4.1 Prompt direto (tiro zero)

Prompt de tiro zero refere-se à capacidade dos LLMs modernos de executar tarefas sem exemplos prévios demonstrados no prompt. Wei et al. [19] demonstraram que modelos de linguagem ajustados através de instruction tuning podem realizar tarefas de forma zero-shot, ou seja, o prompt instrui diretamente o modelo sobre a tarefa sem fornecer exemplos ou demonstrações adicionais. Esta capacidade emergiu quando os modelos foram escalados para tamanhos suficientes e treinados em grandes volumes de dados com instruções explícitas. O ajuste por instruções, combinado com Aprendizado por Reforço a partir de Retroalimentação Humana (RLHF), tem sido fundamental para alinhar modelos às preferências humanas e melhorar significativamente o desempenho de tiro zero.

2.4.2 Prompt de Poucos Exemplos

Prompt de poucos exemplos é uma técnica de aprendizado em contexto onde são fornecidos alguns exemplos demonstrativos no prompt para guiar o modelo a gerar respostas mais precisas. Brown et al. [18] introduziram esta abordagem no contexto do GPT-3, demonstrando que LLMs podem aprender padrões a partir de poucos exemplos fornecidos no próprio prompt, sem necessidade de ajuste fino dos parâmetros do modelo. Os exemplos servem como condicionamento para casos subsequentes, permitindo que o modelo compreenda melhor a tarefa e o formato esperado de resposta. Esta técnica é particularmente útil quando o prompt de tiro zero não produz resultados satisfatórios ou quando é necessário estabelecer padrões específicos de formatação ou raciocínio.

2.4.3 Prompt de Cadeia de Raciocínio

Prompt de Cadeia de Raciocínio (CoT) é uma técnica que encoraja LLMs a decompor problemas complexos em etapas de raciocínio intermediárias explícitas. Wei et al. [21] demonstraram que fornecer exemplos de raciocínio passo a passo nos prompts permite que modelos suficientemente grandes gerem suas próprias cadeias de raciocínio, melhorando significativamente o desempenho em tarefas de raciocínio aritmético, senso comum e simbólico. O CoT é especialmente eficaz para problemas que requerem múltiplas etapas de raciocínio, onde a resposta final depende de conclusões intermediárias. Esta técnica pode ser aplicada tanto em configurações de poucos exemplos (fornecendo exemplos de cadeias de raciocínio) quanto de tiro zero (simplesmente instruindo o modelo a "pensar passo a passo").

2.4.4 Engenharia Automática de Prompt

Engenharia Automática de Prompt (APE) refere-se a métodos automatizados para geração e seleção de prompts otimizados. Zhou et al. [20] propuseram uma abordagem onde LLMs geram candidatos de instruções que são então avaliados e selecionados automaticamente. O processo envolve: (1) usar um LLM para gerar candidatos de prompts a partir de exemplos de entrada-saída, (2) executar cada prompt candidato em um conjunto de dados de treinamento, e (3) selecionar o prompt com melhor desempenho através de métricas de avaliação computacionais. Esta abordagem demonstrou que LLMs podem alcançar desempenho comparável ou superior a prompts escritos por humanos, automatizando o processo iterativo de refinamento de prompts e descobrindo instruções eficazes que não seriam óbvias para designers humanos.

2.5 Método Socrático na Educação

O método socrático é uma abordagem pedagógica baseada em perguntas que guiam o estudante na construção de seu próprio conhecimento, em vez de fornecer respostas diretas. Nomeado em homenagem ao filósofo grego Sócrates, este método estruturado de questionamento visa explorar ideias complexas, revelar pressupostos subjacentes e discernir o que é conhecido do que não é. Esta abordagem estimula a reflexão, o teste de hipóteses e a construção de raciocínio próprio.

Favero et al. [6] identificam seis tipos principais de perguntas socráticas que promovem pensamento crítico: (1) clarificação de conceitos ambíguos, (2) investigação de pressupostos subjacentes, (3) investigação de razões e evidências que sustentam argumentos, (4) exploração de implicações e consequências, (5) investigação de pontos de vista e perspec-

tivas alternativas, e (6) outras questões retóricas ou contextuais. Através de experimentos com chatbots educacionais, os autores demonstraram empiricamente que tutores baseados em perguntas socráticas promovem reflexão e pensamento crítico significativamente melhor ($p < 0,001$) do que chatbots que fornecem respostas diretas. Goldin et al. [22] demonstraram que, embora o diálogo socrático seja muito efetivo em promover raciocínio original (em contraste com reprodução de conhecimento), sua eficácia varia entre estudantes, sendo necessário adaptar a abordagem às diferenças individuais.

No contexto específico de programação, Al-Hossami et al. [7] investigaram se modelos de linguagem podem empregar o método socrático para depuração de código. Seus experimentos demonstraram que LLMs podem gerar perguntas socráticas estruturadas que guiam estudantes na identificação e correção de erros em código, promovendo compreensão profunda em vez de simplesmente fornecer a solução corrigida. Adicionalmente, Shridhar et al. [23] desenvolveram métodos para geração automática de sub-questões socráticas no ensino de problemas matemáticos, demonstrando que técnicas computacionais podem sistematizar a criação de apoio pedagógico estruturado baseado no método socrático.

No contexto do ensino de programação, o método socrático pode ser particularmente eficaz, pois:

- Promove a compreensão conceitual profunda, em vez de memorização de soluções.
- Desenvolve habilidades de resolução de problemas e pensamento crítico.
- Reduz a dependência em respostas prontas.
- Estimula a autonomia e o engajamento do estudante.
- Permite que estudantes explorem diferentes perspectivas e abordagens para resolver problemas computacionais.

2.6 Chatbots Especializados como Tutores

A aplicação de IA em tutores inteligentes educacionais tem demonstrado potencial para personalizar o aprendizado, atendendo às necessidades individuais dos estudantes e aumentando o engajamento através de interações adaptativas [11]. Pesquisas indicam que chatbots especializados em disciplinas específicas podem atuar como tutores eficazes. Sánchez-Vera [8] demonstrou que chatbots especializados promovem autonomia no estudo e melhoram a compreensão conceitual quando utilizados de forma moderada. No entanto, o uso excessivo dessas ferramentas não leva a melhorias adicionais no desempenho acadêmico, destacando a importância de uma abordagem equilibrada que considere não apenas a quantidade de interação, mas também a qualidade pedagógica dessas interações.

2.7 Design Science Research

Design Science Research (DSR) [24] é uma metodologia de pesquisa que se concentra no desenvolvimento e avaliação de artefatos (como sistemas, métodos ou modelos) para resolver problemas práticos. Jaques et al. [24] destacam que a DSR é particularmente adequada para pesquisas em informática na educação que envolvem o desenvolvimento de artefatos computacionais projetados para a educação, pois combina rigor científico com relevância prática.

Os principais componentes da DSR incluem:

- **Problema**: Identificação de um problema prático relevante no contexto organizacional ou educacional.
- **Artefato**: Desenvolvimento de uma solução (artefato) para o problema, que pode ser um constructo, modelo, método ou instanciação (sistema).
- **Avaliação**: Avaliação rigorosa da eficácia do artefato em resolver o problema, utilizando métodos quantitativos e qualitativos.
- **Contribuição**: Documentação das contribuições teóricas e práticas, tanto para o conhecimento científico quanto para a prática profissional.

Esta pesquisa utiliza DSR para desenvolver e avaliar um tutor inteligente baseado em LLM com RAG e método socrático, seguindo o ciclo completo de identificação do problema (altas taxas de reprovação em APC), desenvolvimento do artefato (tutor inteligente), avaliação através de experimentos controlados e documentação das contribuições teóricas e práticas.

Capítulo 3

Metodologia

Este capítulo descreve a metodologia utilizada nesta pesquisa, incluindo o desenho experimental, os participantes, os instrumentos de coleta de dados e os procedimentos adotados.

3.1 Abordagem Metodológica

Esta pesquisa aplicada fundamenta-se na metodologia Design Science Research (DSR) [9], com desenvolvimento e avaliação de um artefato (tutor inteligente). O experimento consistiu na resolução de questões de APC em dois cenários distintos, seguido pela aplicação de um questionário para coleta da percepção dos estudantes. A análise combina dados quantitativos (desempenho em tarefas) e qualitativos (feedback e engajamento), seguindo os princípios da DSR de combinar rigor científico com relevância prática.

3.2 Desenho Experimental

O experimento comparou dois cenários de tutoria inteligente:

Cenário 1 (IA Verde): Sistema que realiza chamadas diretas à API do Google Gemini, fornecendo respostas diretas aos estudantes sem processamento adicional, utilizando a mesma interface do sistema, mas sem recursos adicionais de RAG ou método socrático.

Cenário 2 (IA Azul): LLM integrado com RAG e método socrático que guia os estudantes através de perguntas e reflexões, sem fornecer respostas diretas. O RAG recupera informações relevantes do domínio de APC, enquanto o método socrático faz perguntas que guiam o estudante na construção de sua própria solução.

Ambos os cenários compartilham a mesma interface de usuário, permitindo uma comparação justa entre as abordagens. A diferença fundamental reside na implementação:

a IA Verde realiza chamadas diretas à API do Google Gemini sem processamento adicional, enquanto a IA Azul incorpora RAG para recuperação de informações específicas do domínio e aplica o método socrático para promover aprendizado através de reflexão guiada.

3.3 Participantes

O experimento foi conduzido com estudantes da disciplina de Algoritmos e Programação de Computadores da Universidade de Brasília ao longo de três semanas (10, 17 e 24 de outubro de 2025). Todos os alunos matriculados na disciplina foram convidados a participar, sendo a participação voluntária. Cada estudante que participou utilizou ambas as IAs (Verde e Azul) no mesmo dia.

Ao longo dos três dias de experimento, foram coletadas:

- **144 respostas ao questionário de percepção** (77 para IA Verde e 67 para IA Azul), incluindo estudantes que participaram em mais de um dia.
- **201 respostas para métricas de desempenho** nas questões de programação (101 para IA Verde e 100 para IA Azul).

A diferença no número de respostas ocorre porque nem todos os estudantes completaram tanto as questões de programação quanto o questionário de percepção em todas as sessões. Como cada estudante utilizou ambas as IAs quando participou, o número de respostas por IA reflete aproximadamente metade do total de participações, considerando que alguns estudantes participaram em múltiplos dias do experimento.

3.4 Instrumentos de Coleta de Dados

3.4.1 Questões de Programação

Foram utilizadas questões de programação abertas (requerendo codificação em Python) cobrindo os principais tópicos da disciplina de APC, que equivalem ao material avaliado na Prova 1 (P1) da disciplina:

- Algoritmos e subalgoritmos.
- Variáveis, valores e tipos.
- Estruturas sequenciais.
- Estruturas condicionais.

- Estruturas de repetição.

Em cada aplicação do experimento, foram utilizadas três questões de programação por cenário, totalizando seis questões diferentes por dia (três para IA Verde e três para IA Azul). As questões foram balanceadas em termos de dificuldade e cobertura de conteúdo para permitir comparação justa entre as duas abordagens de tutoria. Cada questão incluía: descrição do problema, especificação de entrada e saída, exemplos de uso, e observações relevantes.

Avaliação Automática das Questões

As questões de programação foram avaliadas automaticamente utilizando o plugin CodeRunner [25] integrado ao Moodle. O CodeRunner é um sistema de avaliação automática amplamente utilizado em cursos de programação, que executa o código submetido pelos estudantes contra conjuntos de casos de teste predefinidos.

Sistema de Avaliação: Cada questão foi configurada com múltiplos casos de teste (tipicamente entre 5 e 10 casos), cobrindo casos básicos, casos extremos e situações especiais. A correção adotou o modo binário: o estudante recebe a pontuação total apenas se o código passar em todos os casos de teste; caso contrário, recebe zero pontos.

Isonomia e Balanceamento: Questões diferentes foram necessariamente utilizadas entre os dois grupos (IA Verde e IA Azul) para evitar que estudantes memorizassem as soluções ou utilizassem informações da primeira sessão (IA Verde) na segunda sessão (IA Azul). Para garantir a isonomia na comparação, as questões foram cuidadosamente balanceadas considerando:

- **Conteúdo programático:** Cada par de questões (Verde/Azul) abordava o mesmo tópico da disciplina (ex: estruturas condicionais, repetições).
- **Dificuldade estimada:** As questões foram selecionadas e adaptadas para apresentarem complexidade algorítmica e número de conceitos equivalentes, baseando-se na experiência prévia dos docentes da disciplina e no desempenho histórico dos estudantes em questões similares.

Limitações Metodológicas: A utilização de questões diferentes entre os grupos introduz uma variável de confundimento potencial nos resultados. Esta escolha foi necessária para evitar que estudantes aproveitassem informações da primeira sessão (IA Verde) na segunda sessão (IA Azul). Embora o balanceamento tenha considerado conteúdo programático e dificuldade estimada equivalentes, diferenças inerentes entre questões podem ter influenciado as métricas de desempenho. Esta limitação é discutida em detalhe na Seção 6.6.7.

As questões utilizadas nos três momentos de aplicação do experimento estão disponíveis no Apêndice B.

3.4.2 Questionário de Percepção

Para avaliar a percepção dos estudantes sobre cada cenário de tutoria, foi aplicado um questionário composto por duas partes:

Parte 1 - LOES-S: A escala Learning Object Evaluation Scale for Students (LOES-S) [10], instrumento validado que avalia três dimensões de objetos de aprendizagem:

- **Aprendizado:** Percepção sobre o quanto o sistema contribuiu para o aprendizado.
- **Qualidade:** Avaliação da usabilidade, organização e clareza das instruções do sistema.
- **Engajamento:** Motivação e interesse em utilizar o sistema.

Parte 2 - Questões Adicionais: Perguntas específicas desenvolvidas para avaliar aspectos particulares dos tutores inteligentes no contexto de programação, incluindo:

- Capacidade do sistema em guiar sem fornecer respostas diretas.
- Auxílio na identificação e correção de erros.
- Contribuição para compreensão de conceitos avançados e pensamento computacional.
- Satisfação geral com a ferramenta.

O questionário completo utilizado no experimento está disponível no Apêndice C.

3.4.3 Telemetria do Sistema

O sistema coletou automaticamente dados de telemetria durante as interações dos estudantes com o tutor inteligente, incluindo:

- Número de requisições à API do LLM (Google Gemini) realizadas em cada sessão de uso do estudante.
- Tempo que cada estudante permaneceu interagindo com o tutor.
- Padrões de uso: perguntas enviadas pelos estudantes ao tutor e frequência das interações.
- Histórico completo das conversações entre estudantes e tutor.

3.5 Procedimentos

O experimento foi conduzido em três dias. Em cada dia, todos os participantes utilizaram ambas as IAs, sempre na mesma ordem: inicialmente a IA Verde e, em seguida, a IA Azul. Após a resolução das questões com a IA Verde, os participantes responderam ao questionário de percepção. Em seguida, utilizaram a IA Azul para resolver novas questões, seguido novamente pelo questionário de percepção.

O tempo de interação foi controlado em 30 minutos para cada IA, permitindo uma comparação justa entre os cenários.

3.5.1 Seleção e Balanceamento das Questões

As questões de programação foram selecionadas do banco de questões da própria disciplina de APC, garantindo alinhamento com o conteúdo e objetivos pedagógicos do curso. Em cada dia de experimento, foram utilizadas três questões para a IA Verde e três questões diferentes para a IA Azul, totalizando seis questões por dia.

O balanceamento das questões entre os dois cenários considerou:

- **Marcação de dificuldade:** As questões já possuíam classificação de dificuldade no banco de questões da disciplina. Para cada dia, buscou-se parear questões de dificuldade equivalente entre IA Verde e IA Azul (por exemplo, se a IA Verde recebia uma questão de dificuldade média sobre estruturas condicionais, a IA Azul recebia uma questão também de dificuldade média sobre o mesmo tópico).
- **Conteúdo programático:** Cada par de questões (Verde/Azul) abordava os mesmos conceitos fundamentais da disciplina, garantindo que ambos os grupos fossem avaliados sobre conhecimentos equivalentes.
- **Cobertura de tópicos:** As três questões de cada cenário cobriam diferentes tópicos do conteúdo da Prova 1, evitando concentração em um único conceito.

Embora o balanceamento tenha sido cuidadoso, reconhece-se que diferenças inerentes entre questões podem ter influenciado parcialmente as métricas de desempenho, conforme discutido na Seção 6.6.7.

3.6 Análise de Dados

A análise dos dados incluiu:

- **Análise quantitativa:** Comparação de desempenho (notas, taxa de sucesso) entre os dois cenários utilizando testes estatísticos apropriados.

- **Análise qualitativa:** Análise das respostas do questionário de percepção e feedback dos estudantes.
- **Análise de telemetria:** Análise dos padrões de uso e comportamento dos estudantes ao interagir com o sistema.

3.7 Considerações Éticas

O experimento foi conduzido respeitando os princípios éticos de pesquisa. A participação foi voluntária, com os estudantes sendo informados sobre os objetivos da pesquisa através do Termo de Consentimento Livre e Esclarecido (TCLE) apresentado no Apêndice A, e tendo a opção de não participar sem qualquer prejuízo acadêmico. Os dados coletados foram anonimizados para garantir a privacidade dos participantes, e o sistema foi disponibilizado igualmente para todos os estudantes da disciplina, independentemente da participação no experimento.

3.8 Declaração de Uso de Inteligência Artificial Generativa

Em conformidade com as diretrizes da Sociedade Brasileira de Computação [26] sobre o uso de ferramentas de Inteligência Artificial Generativa, declaramos que foram utilizados agentes de IA durante o desenvolvimento deste trabalho nas seguintes atividades:

- **Revisão textual:** Ferramentas de IA foram empregadas para correção gramatical, ajustes de coesão e coerência semântica do texto em português, assegurando clareza e conformidade com normas acadêmicas.
- **Edição em LaTeX:** Assistentes de IA auxiliaram na formatação do documento, correção de sintaxe LaTeX, e organização estrutural de seções, tabelas e referências bibliográficas.
- **Desenvolvimento do sistema:** Durante a implementação do tutor inteligente, ferramentas de IA forneceram suporte na construção de componentes da interface de usuário e da API do sistema, incluindo sugestões de código.
- **Pesquisa bibliográfica:** Agentes de IA foram utilizados como apoio na busca de referências acadêmicas relevantes, identificação de trabalhos relacionados e fichamento inicial de artigos científicos. O agente principal utilizado foi o Consensus [27].

Todo o conteúdo gerado com auxílio de IA foi criteriosamente revisado, validado e adaptado pelos autores, que assumem integral responsabilidade pela veracidade, originalidade e correção de todas as informações apresentadas neste trabalho. As ferramentas de IA não foram listadas como autores e seu uso não exime os responsáveis de qualquer aspecto relacionado ao conteúdo desta monografia.

Capítulo 4

Desenvolvimento do Tutor Inteligente

Este capítulo descreve o desenvolvimento do tutor inteligente, incluindo sua arquitetura, componentes principais, tecnologias utilizadas e implementação dos cenários experimentais.

4.1 Visão Geral do Sistema

O tutor inteligente foi desenvolvido como uma aplicação web que integra LLMs com técnicas de RAG para fornecer suporte educacional personalizado no ensino de Algoritmos e Programação de Computadores. O sistema foi projetado para operar em dois modos distintos: um que fornece respostas diretas (IA Verde) e outro que aplica o método socrático com RAG (IA Azul).

4.2 Arquitetura do Sistema

A arquitetura do tutor inteligente é composta por componentes modulares que trabalham em conjunto para processar as interações dos estudantes e fornecer respostas adequadas. A Figura 4.1 apresenta a arquitetura completa do sistema.

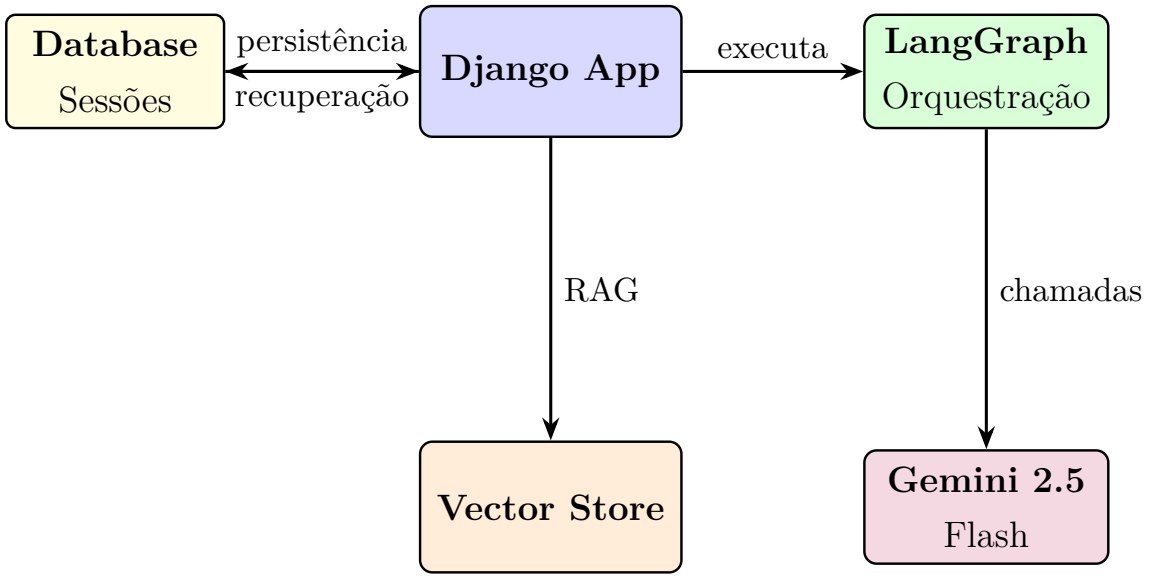


Figura 4.1: Arquitetura do sistema tutor inteligente baseado em LLM com RAG

4.2.1 Componentes Principais

Aplicação Django

O núcleo do sistema é uma aplicação Django [28] que segue a arquitetura MVT (*Model-View-Template*), padrão arquitetural utilizado pelo framework. Nesta arquitetura, a camada **Model** define a estrutura dos dados e interage com o banco de dados, a camada **View** contém a lógica de processamento das requisições e coordena a resposta ao usuário, e a camada **Template** é responsável pela apresentação e renderização das interfaces. Esta separação de responsabilidades promove baixo acoplamento entre as camadas, facilitando manutenção e extensibilidade do sistema.

A aplicação está organizada em dois apps principais. O app `users` é responsável pela autenticação e gerenciamento de usuários, incluindo modelos de usuário, views de autenticação e formulários de registro e login. O app `tutor` contém a lógica principal do tutor inteligente, incluindo modelos para configuração de versões de IA (AIConfig), sessões de chat (ChatSession) e histórico de mensagens (ChatPrompt), além de módulos para processamento de chat com LangGraph [29], gerenciamento de vector store para RAG e interface web para interação com os estudantes.

A Figura 4.2 apresenta a estrutura modular da aplicação Django, detalhando os componentes de cada app e suas responsabilidades dentro da arquitetura MVT.

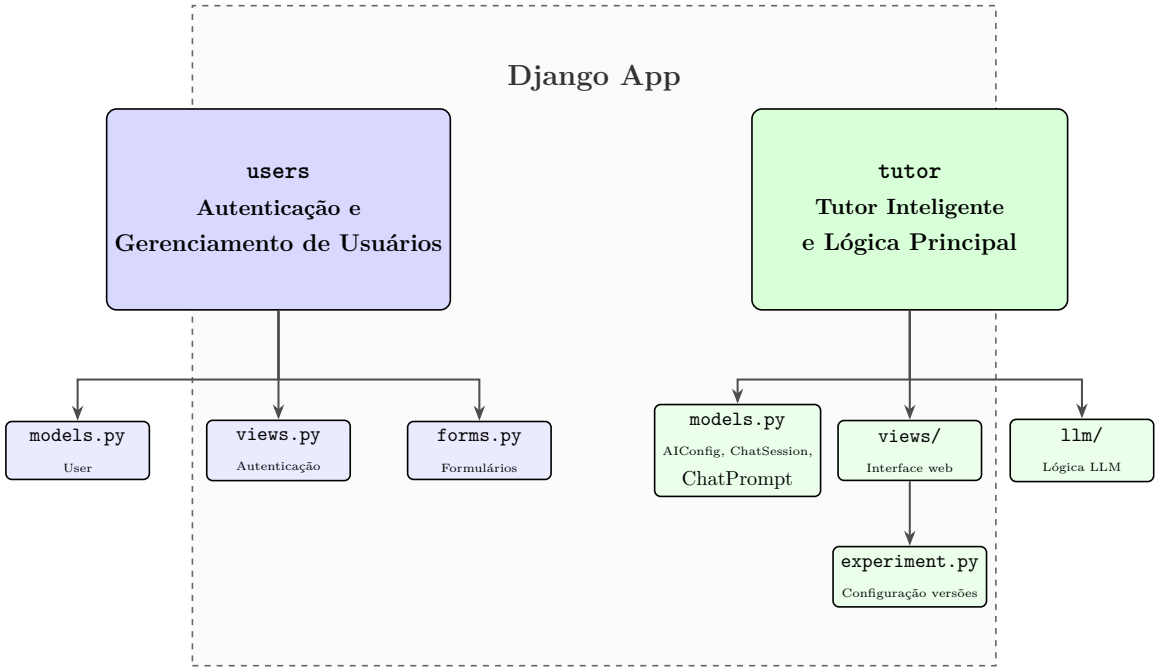


Figura 4.2: Estrutura modular da aplicação Django

LangGraph

O LangGraph [29] é responsável pela orquestração das interações entre o estudante e o tutor inteligente, implementando um grafo de estados que processa cada requisição de forma estruturada e contextualizada. Construído sobre o LangChain [30], o framework permite definir fluxos de processamento complexos através de nós interconectados, onde cada nó representa uma etapa específica do processamento e as arestas definem as transições entre essas etapas.

O grafo mantém o estado da interação através de um dicionário tipado (`State`) que armazena informações como a pergunta do estudante, código fornecido, erros reportados, contexto recuperado do vector store, histórico de mensagens e a resposta gerada. Essa estrutura de estado permite que o tutor mantenha coerência e contexto ao longo de toda a interação com o estudante, garantindo que respostas subsequentes considerem o histórico da conversa.

A arquitetura do grafo varia conforme a versão da IA utilizada, refletindo as diferentes estratégias pedagógicas implementadas. As Figuras 4.3 e 4.4 apresentam visualmente os fluxos implementados para cada cenário.

IA Azul: Grafo com RAG e Método Socrático O grafo implementado para o cenário IA Azul, apresentado na Figura 4.3, segue um fluxo mais complexo que incorpora recuperação de contexto e aplicação do método socrático. O processamento inicia no

nó `route_question`, responsável por receber e classificar a pergunta do estudante. A partir deste nó, o fluxo segue para o nó de decisão `decide_to_retrieve`, que avalia se a pergunta é relevante para o contexto de programação.

Se a pergunta for identificada como relevante para programação, o fluxo segue para o nó `hyde_retrieval`, que implementa a estratégia HyDE (Hypothetical Document Embeddings) [17]. Este nó gera uma resposta hipotética da pergunta e utiliza essa resposta para buscar documentos relevantes no vector store, melhorando significativamente a precisão da recuperação de contexto através de recuperação zero-shot. Em seguida, o nó `generate` produz a resposta final incorporando o contexto recuperado e aplicando o método socrático através dos prompts configurados, que orientam o LLM a fazer perguntas reflexivas ao invés de fornecer respostas diretas.

Caso a pergunta seja identificada como fora do tópico de programação, o fluxo direciona para o nó `handle_off_topic`, que retorna uma mensagem educativa orientando o estudante a focar em questões relacionadas à programação e ao conteúdo da disciplina de APC.

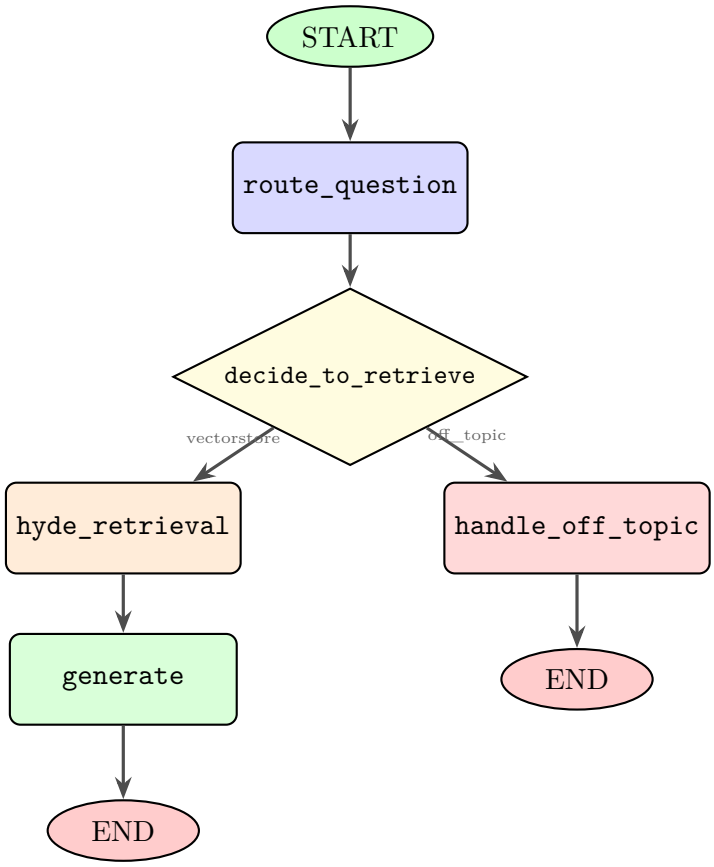


Figura 4.3: Grafo LangGraph para IA Azul (com RAG e método socrático)

IA Verde: Grafo Simplificado Para o cenário IA Verde, o grafo é significativamente simplificado, conforme apresentado na Figura 4.4. O fluxo consiste apenas em dois nós: o ponto de entrada é diretamente o nó `generate_direct`, que invoca o LLM sem etapas intermediárias de recuperação de contexto ou classificação. Este nó produz respostas diretas às perguntas dos estudantes, sem aplicar o método socrático ou recuperar informações do material didático. A simplicidade deste grafo reflete a estratégia de fornecer respostas imediatas, sem mediação pedagógica adicional.

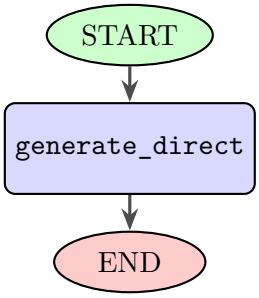


Figura 4.4: Grafo LangGraph para IA Verde (sem RAG, respostas diretas)

Vector Store

O armazenamento vetorial implementa a memória não-paramétrica explícita do sistema RAG [5], utilizando o Qdrant [31] como banco de dados vetorial para armazenar incorporações vetoriais do material didático do curso de APC. As incorporações vetoriais são geradas utilizando o modelo `text-embedding-004` da Google (Gemini), configurado para produzir vetores de 768 dimensões através de uma classe adaptadora customizada (`GeminiEmbeddings768`) que garante dimensionalidade consistente. O modelo é configurado com `task_type="RETRIEVAL_DOCUMENT"` para otimizar a recuperação de documentos.

A estratégia de carregamento utiliza o `WebBaseLoader` do LangChain para carregar material didático de URLs web, incluindo capítulos do livro "Pense em Python" [32] e documentação oficial do Python [33]. Os documentos são segmentados utilizando `RecursiveCharacterTextSplitter` com tamanho de chunk de 400 caracteres e sobreposição de 40 caracteres, garantindo que informações contextuais sejam preservadas entre segmentos adjacentes. Os documentos segmentados são então convertidos em embeddings e indexados no Qdrant utilizando distância de similaridade cosseno para buscas de similaridade. Esta abordagem de recuperação por similaridade vetorial permite que o sistema acesse dinamicamente informações relevantes do material didático durante a geração de respostas, característica fundamental do paradigma RAG.

Estratégia HyDE

A estratégia HyDE (Incorporações de Documentos Hipotéticos) [17] é empregada para melhorar a qualidade da recuperação através de uma abordagem de tiro zero que não requer rótulos de relevância. O processo funciona da seguinte forma: dado uma consulta do estudante, o LLM gera primeiro um documento hipotético que captura padrões de relevância, mesmo que contenha detalhes imprecisos. Este documento hipotético é então codificado em um vetor de incorporação, que é usado para recuperar documentos reais similares no corpus. O codificador denso atua como um filtro, eliminando detalhes incorretos do documento gerado e ancorando a busca em documentos reais do material didático. Gao et al. [17] demonstraram que esta técnica supera significativamente recuperadores não supervisionados tradicionais, sendo especialmente valiosa para domínios específicos como o ensino de programação, onde não há conjuntos de dados de relevância pré-rotulados disponíveis.

Banco de Dados

O sistema mantém um banco de dados relacional para persistir:

- Sessões de usuários.
- Prompts e respostas.
- Histórico de interações.
- Dados de telemetria.
- Resultados das questões.

A Figura 4.5 apresenta o diagrama MER (Modelo Entidade-Relacionamento) do banco de dados, mostrando as entidades principais e seus relacionamentos.

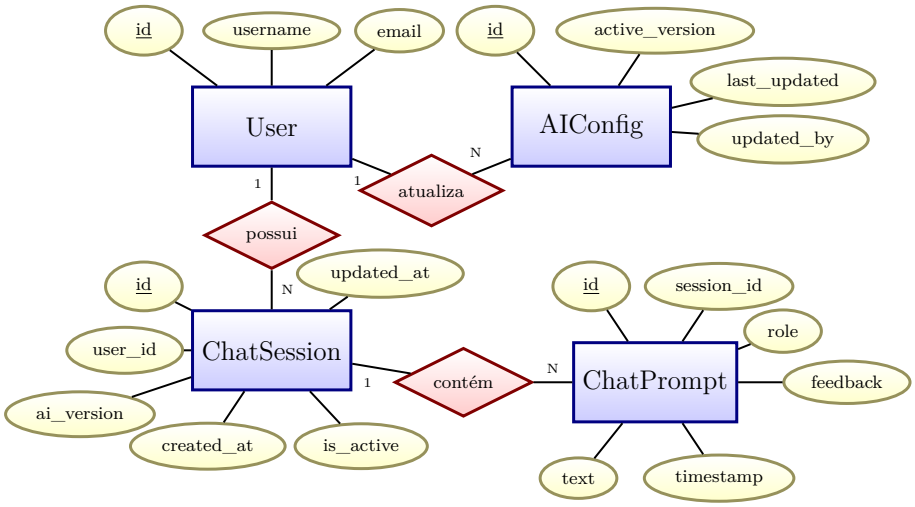


Figura 4.5: Diagrama MER do banco de dados do tutor inteligente

As principais entidades do sistema são:

- **User**: Entidade padrão do Django que armazena informações dos usuários do sistema, incluindo credenciais de autenticação (username, email, password).
- **AIConfig**: Modelo singleton que gerencia a versão ativa da IA (GREEN AI ou BLUE AI) para todo o sistema. Armazena a versão ativa, data da última atualização e referência ao administrador que realizou a atualização.
- **ChatSession**: Representa uma sessão de interação entre um usuário e o tutor. Cada sessão possui um identificador único (UUID), está associada a um usuário, armazena a versão de IA utilizada (GREEN ou BLUE), timestamps de criação e atualização, e um flag indicando se a sessão está ativa.
- **ChatPrompt**: Armazena o histórico de mensagens de cada sessão. Cada prompt possui um identificador único (UUID), está associado a uma sessão, armazena o papel da mensagem (system, user ou ai), o texto da mensagem, feedback opcional (positive, negative ou neutral) e timestamp de criação.

Os relacionamentos estabelecem que cada usuário pode ter múltiplas sessões de chat (1:N), cada sessão está associada a uma configuração de IA específica através do campo `ai_version` que referencia a versão ativa em `AIConfig`, e cada sessão pode conter múltiplas mensagens (1:N). O modelo `AIConfig` também mantém uma referência opcional ao usuário administrador que realizou a última atualização da configuração.

API do Gemini

As chamadas ao modelo de linguagem são realizadas através da API do Gemini 2.5 Flash da Google, que processa requisições do LangGraph e retorna as respostas do tutor. A escolha do Gemini 2.5 Flash foi baseada em seu desempenho, custo e capacidade de processamento.

4.3 Implementação dos Cenários

4.3.1 Cenário IA Verde (Sem RAG, Respostas Diretas)

No cenário IA Verde, o sistema foi configurado para:

- Fornecer respostas diretas às perguntas dos estudantes.
- Incluir código completo quando solicitado.
- Não utilizar RAG para recuperação de informações específicas do domínio.
- Responder de forma direta e objetiva, sem aplicar o método socrático.

Este cenário simula o uso típico de LLMs como ChatGPT ou similares, onde o estudante recebe soluções prontas. Os prompts de sistema e templates utilizados neste cenário estão detalhados no Apêndice D.

4.3.2 Cenário IA Azul (Com RAG e Método Socrático)

No cenário IA Azul, o sistema foi configurado para:

- Aplicar o método socrático, fazendo perguntas que guiam o estudante.
- Utilizar RAG para recuperar informações relevantes do material didático.
- Não fornecer código completo ou respostas diretas.
- Estimular a reflexão e construção de conhecimento pelo estudante.

Este cenário representa a abordagem pedagógica proposta, combinando RAG com método socrático para promover aprendizado mais efetivo. Os prompts de sistema e templates utilizados neste cenário, que incorporam instruções específicas para aplicação do método socrático, estão detalhados no Apêndice D.

4.4 Deploy e Infraestrutura

O sistema foi implantado utilizando containers Docker, permitindo:

- Isolamento de componentes.
- Escalabilidade.
- Distribuição dos componentes em diferentes ambientes.
- Facilidade de manutenção e atualização.

A infraestrutura foi projetada para suportar múltiplos usuários simultâneos durante os experimentos, garantindo disponibilidade e desempenho adequados.

4.5 Interface do Usuário

A interface do usuário foi desenvolvida para ser intuitiva e facilitar a interação dos estudantes com o tutor inteligente. A interface permite:

- Visualização das questões a serem resolvidas.
- Interação com o tutor através de chat.
- Visualização do histórico de conversações.
- Submissão de soluções.
- Acesso ao feedback e avaliação das respostas.

As principais interfaces do sistema, incluindo tela de login, interface administrativa para seleção de versão de IA e interfaces de chat para ambos os cenários (IA Verde e IA Azul), estão apresentadas no Apêndice E.[1]

Uma limitação identificada pelos estudantes foi a necessidade de alternar entre duas interfaces distintas (tutor inteligente e plataforma Moodle), sugerindo que trabalhos futuros devem priorizar o desenvolvimento de uma interface unificada.

[1] O código fonte completo do sistema está disponível publicamente em: `https://github.com/0lucasmoura/llm-tutor`

Capítulo 5

Resultados

Este capítulo apresenta os resultados obtidos no experimento, incluindo análise quantitativa do desempenho dos estudantes, análise qualitativa da percepção dos participantes e comparação entre os dois cenários experimentais.

5.1 Caracterização dos Participantes

O experimento foi conduzido com estudantes da disciplina de Algoritmos e Programação de Computadores da Universidade de Brasília ao longo de três semanas (10, 17 e 24 de outubro de 2025). O delineamento experimental previa que cada estudante utilizasse ambas as IAs (Verde e Azul) no mesmo dia, respondendo ao questionário de percepção após cada interação e resolvendo questões de programação com cada IA. No entanto, nem todos os estudantes completaram ambas as IAs em cada sessão.

Participantes do experimento: Ao longo dos três dias, participaram do experimento:

- **Dia 10/10:** 34 respondentes
- **Dia 17/10:** 24 respondentes
- **Dia 24/10:** 22 respondentes

Dados coletados: Ao longo dos três dias de experimento, foram coletadas:

- **144 respostas ao questionário de percepção** (77 para IA Verde e 67 para IA Azul), incluindo estudantes que participaram em múltiplos dias.
- **201 respostas para métricas de desempenho** nas questões de programação (101 para IA Verde e 100 para IA Azul).

- **35 estudantes completaram o questionário para ambas as IAs**, constituindo o conjunto de dados pareados utilizado nas análises estatísticas da seção de percepção.

A diferença no número de respostas entre questionários e métricas de desempenho ocorre porque nem todos os estudantes completaram ambas as atividades. Alguns estudantes responderam ao questionário mas não completaram as questões de programação, ou vice-versa.

5.2 Desempenho nas Tarefas

5.2.1 Notas Médias

A análise do desempenho dos estudantes revelou diferenças significativas entre os dois cenários (Figura 5.1):

- **IA Verde (sem RAG, respostas diretas):** Nota média de $7,45 \pm 4,32$ pontos (em 10).
- **IA Azul (com RAG e método socrático):** Nota média de $4,27 \pm 4,77$ pontos (em 10).

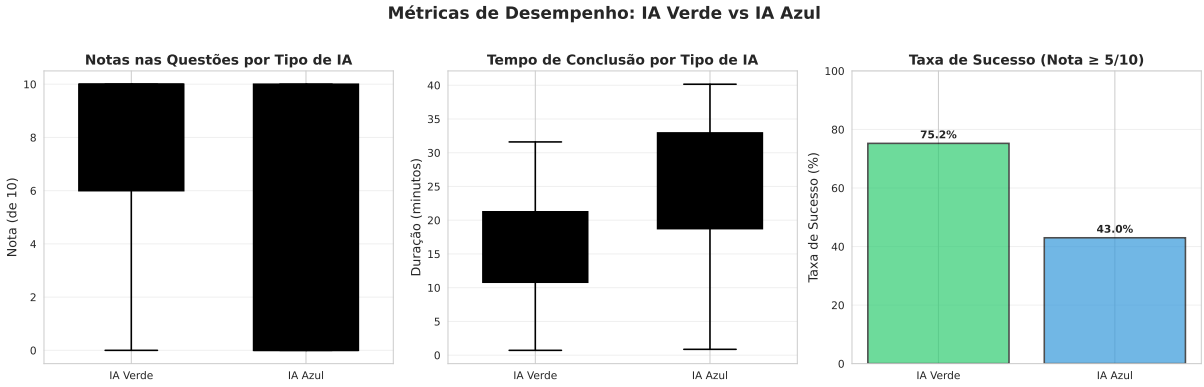


Figura 5.1: Métricas de desempenho: (esquerda) notas nas questões por tipo de IA, (centro) tempo de conclusão por tipo de IA, (direita) taxa de sucesso (nota $\geq 5/10$).

Embora a IA Verde tenha apresentado notas significativamente mais altas, esta diferença pode ser enganosa, pois notas mais altas podem refletir a capacidade dos estudantes de copiar soluções, em vez de demonstrar compreensão real.

5.2.2 Taxa de Sucesso

A taxa de sucesso (proporção de questões com nota $\geq 5/10$) também apresentou diferenças substanciais:

- **IA Verde:** Taxa de sucesso de 75,2%.
- **IA Azul:** Taxa de sucesso de 43,0%.

A restrição temporal de 30 minutos pode ter impactado desproporcionalmente a IA Azul, cujo tempo médio (24,48 minutos) aproximou-se do limite, enquanto a IA Verde (15,91 minutos) permaneceu confortável. Esta diferença evidencia um compromisso entre velocidade de conclusão e profundidade do processo de aprendizado.

5.2.3 Tempo de Resolução

O tempo médio de resolução foi significativamente maior na IA Azul ($24,48 \pm 8,99$ minutos) comparado à IA Verde ($15,91 \pm 7,52$ minutos), mesmo com o tempo de interação controlado (30 minutos para cada IA). Esta diferença no tempo de resolução, combinada com a diferença nas notas, sugere que os estudantes na IA Azul estavam realmente pensando e construindo soluções através de reflexão guiada, dedicando mais tempo ao processo de aprendizado, mesmo que isso resultasse em notas menores.

Alguns estudantes excederam o limite de 30 minutos (1 na IA Verde com 31,6 minutos; 32 na IA Azul, máximo de 40,15 minutos), possivelmente devido à página permanecer aberta além do tempo previsto. A diferença de aproximadamente 8,5 minutos entre os grupos permanece estatisticamente robusta, confirmada pelas medianas ($14,90$ minutos para IA Verde; $21,12$ minutos para IA Azul).

5.3 Percepção dos Estudantes

A análise do questionário de percepção, baseado na escala LOES-S (Learning Object Evaluation Scale for Students) [10] complementada por questões adicionais específicas, revelou diferenças significativas em várias dimensões.

5.3.1 Escala LOES-S: Aprendizado, Engajamento e Qualidade

A escala LOES-S avalia três construtos psicológicos relacionados a objetos de aprendizagem [10]:

- **Aprendizado** (5 questões: Q1-Q5): Percepção sobre o quanto o sistema contribuiu para o aprendizado.

- **Engajamento** (3 questões: Q6-Q8): Motivação e interesse em utilizar o sistema.
- **Qualidade** (4 questões: Q9-Q12): Qualidade técnica e de design, focando em usabilidade (recursos de ajuda, clareza de instruções, facilidade de uso, organização).

Os resultados da escala LOES-S, baseados na análise pareada (n=35), mostraram (Tabela 5.1 e Figura 5.2):

Tabela 5.1: Resultados por categoria da escala LOES-S (análise pareada, n=35)

Categoria	**IA Verde**	**IA Azul**	**Diferença**	***p*-valor**
Aprendizado	$3,86 \pm 0,82$	$3,86 \pm 0,93$	$0,00$	0,891
Engajamento	$4,22 \pm 0,77$	$4,15 \pm 0,68$	$-0,07$	0,489
Qualidade	$4,30 \pm 0,64$	$4,02 \pm 0,81$	$-0,28$	0,035

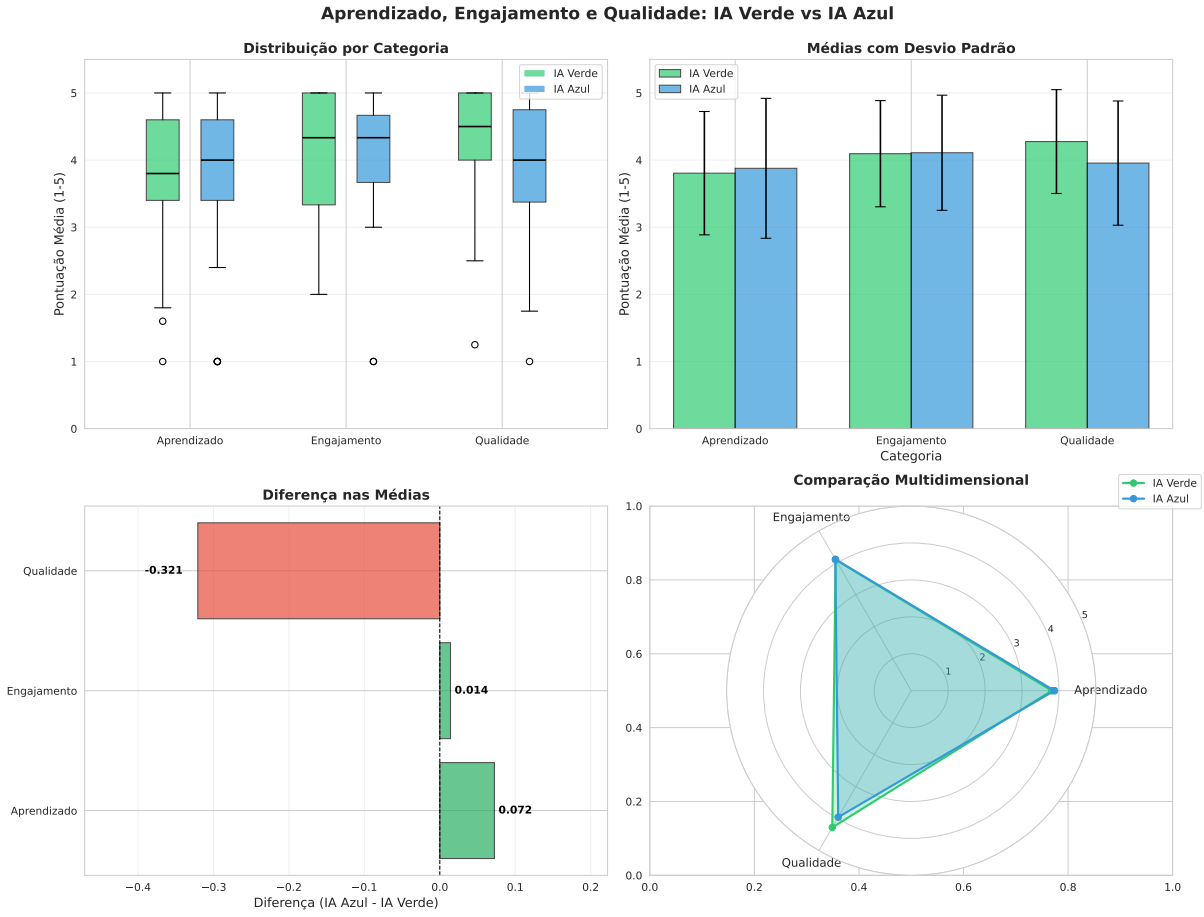


Figura 5.2: Aprendizado, Engajamento e Qualidade: comparação multidimensional entre IA Verde e IA Azul baseada na escala LOES-S.

Os resultados mostram que:

- Não houve diferença significativa em **Aprendizado** ($p = 0,891$), indicando que ambas as abordagens foram percebidas como igualmente eficazes para aprendizado, com médias praticamente idênticas.
- Não houve diferença significativa em **Engajamento** ($p = 0,489$), sugerindo que ambas as abordagens mantiveram os estudantes engajados de forma similar.
- A IA Verde apresentou **Qualidade** significativamente superior ($p = 0,035$), refletindo melhor usabilidade, organização e clareza de instruções percebidas.

Estes resultados indicam que, apesar das diferenças em qualidade/usabilidade, ambas as abordagens foram percebidas como igualmente efetivas para aprendizado e engajamento.

5.3.2 Questões Adicionais Específicas

Além da escala LOES-S, foram aplicadas questões adicionais para avaliar aspectos específicos dos tutores inteligentes no contexto de programação.

Resolução de Problemas sem Respostas Diretas

A IA Azul foi significativamente superior em guiar os estudantes na resolução de problemas sem fornecer respostas diretas (Figura 5.3):

- **IA Verde:** Média de $3,26 \pm 1,34$
- **IA Azul:** Média de $4,24 \pm 0,93$
- **Diferença:** $+0,99$ pontos ($p < 0,001$)

Este resultado confirma que a abordagem socrática foi efetiva em seu objetivo principal: promover a construção do conhecimento através da reflexão, em vez da dependência de respostas prontas.

Conceitos Avançados e Pensamento Computacional

A IA Azul também apresentou melhor performance em promover compreensão de conceitos avançados:

- **Conceitos Avançados (Q15):** IA Verde = $3,27$, IA Azul = $3,81$ (diferença: $+0,54$, $p = 0,023$)
- **Pensamento Computacional (Q19):** IA Verde = $3,76$, IA Azul = $3,89$ (diferença: $+0,13$, $p = 0,121$, não significativo)

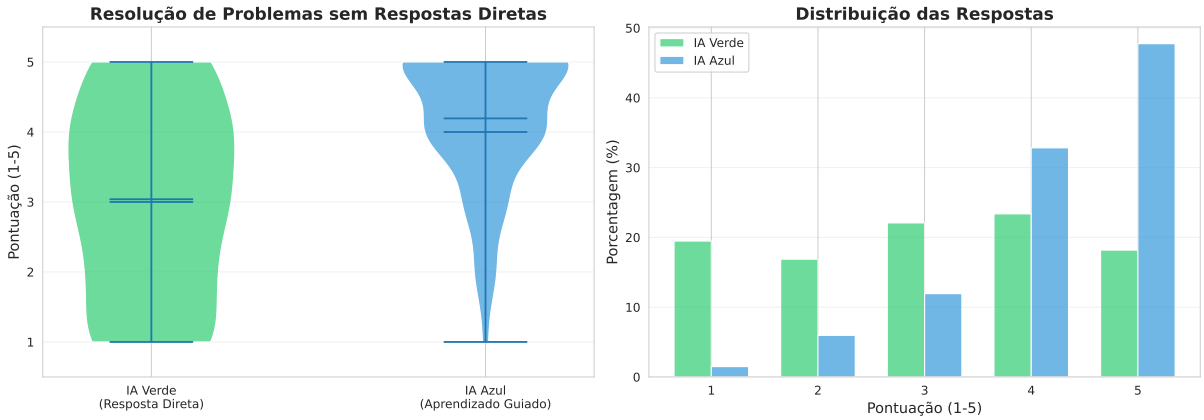


Figura 5.3: Resolução de problemas sem respostas diretas: (esquerda) violin plot mostrando distribuição das pontuações, (direita) distribuição percentual das respostas.

O resultado em conceitos avançados sugere que a abordagem socrática promove uma compreensão mais profunda dos conceitos de programação. Para pensamento computacional, embora a IA Azul apresente média ligeiramente superior, a diferença não foi estatisticamente significativa.

Satisfação Geral

A satisfação geral dos estudantes (Q20) foi similar entre os dois cenários (Figura 5.4):

- **IA Verde:** Média de $4,16 \pm 0,89$ (mediana: 4,0)
- **IA Azul:** Média de $4,14 \pm 0,98$ (mediana: 4,0)

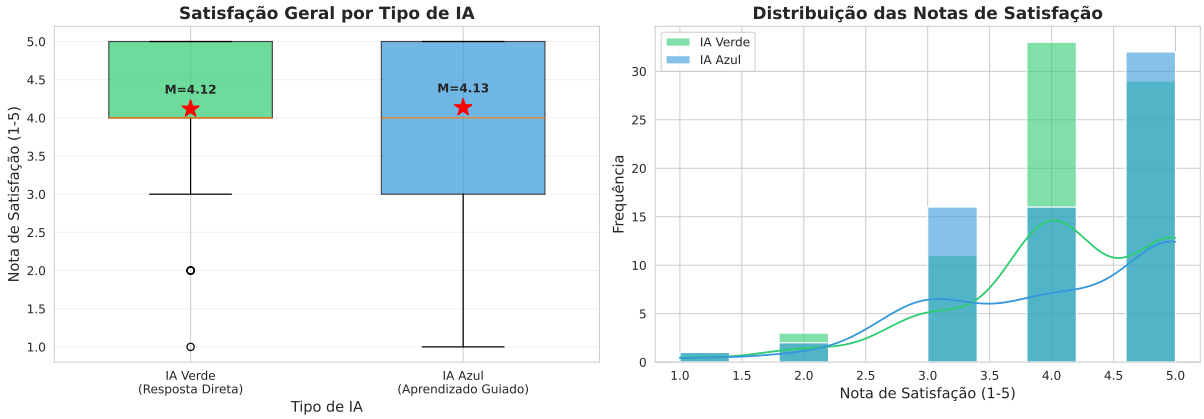


Figura 5.4: Satisfação geral por tipo de IA: (esquerda) box plot comparando as distribuições, (direita) distribuição das notas de satisfação.

O teste de Wilcoxon pareado não revelou diferença significativa entre os grupos ($p = 0,810$, n=35), indicando que ambas as abordagens foram igualmente bem recebidas pelos estudantes em termos de satisfação geral.

5.4 Análise por Perfil do Estudante

Para investigar se o conhecimento prévio de programação influencia a percepção das IAs, foi realizada uma análise comparativa considerando diferentes perfis de estudantes. Os dados de perfil foram coletados através de questionário específico e correlacionados com as respostas de percepção.

5.4.1 Impacto do Conhecimento Prévio

A análise considerou a questão "Já sabia programar antes de entrar na UnB?" como indicador de conhecimento prévio. Os resultados mostram:

IA Verde:

- **Com conhecimento prévio (n=24):** Satisfação $= 4,25 \pm 0,74$
- **Sem conhecimento prévio (n=31):** Satisfação $= 4,03 \pm 1,08$
- **Teste Mann-Whitney:** $p = 0,654$ (não significativo)

IA Azul:

- **Com conhecimento prévio (n=31):** Satisfação $= 4,23 \pm 0,84$
- **Sem conhecimento prévio (n=36):** Satisfação $= 4,06 \pm 1,09$
- **Teste Mann-Whitney:** $p = 0,685$ (não significativo)

Não foram encontradas diferenças significativas na percepção das IAs baseadas no conhecimento prévio de programação. Isto sugere que ambas as abordagens são igualmente adequadas tanto para iniciantes quanto para estudantes com experiência prévia.

5.4.2 Correlação com Desempenho na Primeira Prova

Foi analisada a relação entre a satisfação/engajamento com o tutor e o desempenho na primeira prova da disciplina. Os resultados mostram:

IA Verde (n=52):

- Correlação Nota da Prova ↔ Engajamento: $r = 0,121$, $p = 0,393$ (não significativo)
- Correlação Nota da Prova ↔ Aprendizado: $r = 0,059$, $p = 0,677$ (não significativo)
- Correlação Nota da Prova ↔ Satisfação: $r = 0,248$, $p = 0,076$ (não significativo)

IA Azul (n=64):

- Correlação Nota da Prova ↔ Engajamento: $r = -0,063$, $p = 0,619$ (não significativo)
- Correlação Nota da Prova ↔ Aprendizado: $r = -0,122$, $p = 0,336$ (não significativo)
- Correlação Nota da Prova ↔ Satisfação: $r = -0,077$, $p = 0,547$ (não significativo)

Não foram encontradas correlações significativas entre o uso do tutor e o desempenho na primeira prova, apesar do experimento ter sido realizado antes da avaliação. Isto pode ser explicado por diversos fatores:

- O tempo de exposição ao tutor pode ter sido insuficiente para impactar significativamente o desempenho na prova.
- O conteúdo das questões experimentais pode não ter correspondido diretamente aos tópicos avaliados na prova.
- Outros fatores (estudo individual, aulas, práticas anteriores, etc.) têm maior impacto acumulado no desempenho da prova.
- O experimento focou em exercícios específicos, enquanto a prova avalia conhecimento mais amplo da disciplina.

Uma análise complementar incluindo provas posteriores ou a nota final da disciplina poderia revelar efeitos de longo prazo do uso do tutor no aprendizado.

5.4.3 Análise por Grupos de Desempenho na Avaliação Formal

Os estudantes foram divididos em grupos de alto desempenho (nota $\geq$ mediana = 8,42) e baixo desempenho (nota $<$ mediana) com base em seu desempenho na primeira avaliação formal da disciplina (Prova 1), realizada após o experimento. A análise de engajamento com os tutores mostra:

IA Verde:

- **Alto desempenho (n=27):** Nota média = 9,46, Engajamento = 4,21
- **Baixo desempenho (n=25):** Nota média = 5,25, Engajamento = 4,15
- **Teste Mann-Whitney:** $p = 0,830$ (não significativo)

IA Azul:

- **Alto desempenho (n=33):** Nota média = 9,42, Engajamento = 3,99
- **Baixo desempenho (n=31):** Nota média = 5,27, Engajamento = 4,23
- **Teste Mann-Whitney:** $p = 0,865$ (não significativo)

Curiosamente, estudantes com baixo desempenho na IA Azul apresentaram ligeiramente maior engajamento, embora não significativo. Isto pode sugerir que estudantes com dificuldades encontraram valor na abordagem guiada.

5.4.4 Síntese da Análise por Perfil

Um resultado importante desta análise é que **o perfil do estudante não influenciou significativamente a percepção das ferramentas**. Independentemente de:

- Ter ou não conhecimento prévio de programação;
- Desempenho acadêmico (alto ou baixo na primeira prova);

Ambas as abordagens (IA Verde e IA Azul) foram percebidas de forma consistente em termos de satisfação, aprendizado e engajamento. Isto indica que as características pedagógicas de cada abordagem são robustas e independem do perfil do usuário, sendo adequadas para diferentes contextos educacionais.

É importante ressaltar que este experimento foi controlado quanto ao momento e tempo de aplicação: todos os estudantes interagiram com as IAs no mesmo período (em aula) e com limite de tempo definido (30 minutos). Em um cenário de uso livre, onde estudantes escolheriam quando e por quanto tempo utilizar cada abordagem, os resultados poderiam ser diferentes.

5.5 Telemetria e Custos Operacionais

Durante as três semanas de experimento, foram coletados dados de uso da API do Google Gemini, fornecendo insights sobre o volume de interações e custos operacionais do sistema.

5.5.1 Dados de Uso por Dia de Experimento

A Tabela 5.2 apresenta os dados agregados de uso do LLM (Gemini 1.5 Flash) durante os três dias de experimento:

Tabela 5.2: Dados de telemetria e custos operacionais do experimento

Data	Respondentes	Tokens Gerados	Requisições	Custo (R$)
10/10/2025	65	2,17 milhões	1.548	1,66
17/10/2025	39	1,41 milhões	721	1,44
24/10/2025	40	5,24 milhões	848	3,65
Total	**144**	**8,82 milhões**	**3.117**	**6,75**

Nota: O número de respondentes refere-se ao total de questionários respondidos (77 para IA Verde e 67 para IA Azul), incluindo estudantes que participaram em múltiplos dias. O sistema de coleta garante uma resposta única por participante em cada sessão.

5.5.2 Análise dos Dados

Os dados revelam algumas características importantes do uso do sistema:

- **Custo total do experimento:** R$ 6,75 para 3.117 requisições e 144 respondentes, resultando em um custo médio de aproximadamente R$ 0,0022 por requisição ou R$ 0,047 por respondente.
- **Participação por dia:** O primeiro dia teve maior participação (65 respondentes), seguido por uma queda nos dias subsequentes (39 e 40 respondentes).
- **Volume de tokens:** O terceiro dia apresentou volume significativamente maior (5,24 milhões de tokens) com apenas 40 respondentes, comparado aos primeiros dias que tiveram 2,17 e 1,41 milhões de tokens com 65 e 39 respondentes, respectivamente. Isto sugere interações mais longas e complexas no terceiro dia, possivelmente devido à maior familiaridade dos estudantes com o sistema ou questões que demandaram mais diálogo.
- **Viabilidade econômica:** Com custo médio de aproximadamente R$ 2,25 por dia de experimento, a implementação em larga escala mostra-se economicamente viável para contextos educacionais, especialmente considerando modelos de LLM de baixo custo como o Gemini 1.5 Flash.
- **Eficiência do sistema:** A média de aproximadamente 2.829 tokens por requisição indica interações de tamanho moderado. Para contextualizar: um *token* é a unidade básica de processamento de LLMs, correspondendo aproximadamente a 3-4

caracteres ou 0,75 palavras em português [34]. Uma *requisição* representa uma interação completa estudante-tutor, incluindo a pergunta do estudante, o histórico da conversa, eventuais trechos de código fornecidos, o contexto recuperado pelo RAG (quando aplicável) e a resposta gerada pelo tutor. Assim, 2.829 tokens por requisição equivalem a aproximadamente 2.120 palavras por interação completa, indicando diálogos substanciais mas não excessivos. Este volume sugere que o sistema estava proporcionando respostas elaboradas o suficiente para serem pedagógicamente úteis, sem gerar textos excessivamente longos que poderiam sobrecarregar os estudantes ou aumentar desnecessariamente os custos.

É importante notar que estes custos representam apenas o uso da API do LLM e não incluem custos de infraestrutura, desenvolvimento ou manutenção do sistema. O experimento foi realizado utilizando uma conta paga da API do Google Gemini, dado que opções gratuitas de LLMs apresentam limitações restritivas de requisições e que a alternativa de self-hosting não era viável devido à ausência de infraestrutura computacional adequada para executar modelos de linguagem de grande escala.

5.6 Comparação Detalhada por Questão

Uma análise detalhada questão por questão foi realizada, identificando 4 diferenças significativas de 21 questões analisadas (Figura 5.5). As 21 questões analisadas incluem: 5 questões de Aprendizado (Q1-Q5), 3 questões de Engajamento (Q6-Q8), 4 questões de Qualidade (Q9-Q12), 7 questões adicionais específicas (Q13-Q19), e 1 questão de satisfação geral (Q20).

As questões onde a IA Azul obteve pontuação significativamente superior foram:

1. **Resolução de Problemas sem Respostas Diretas (Q18):** +0,99 pontos ($p < 0,001$)
2. **Compreensão de Conceitos Avançados (Q15):** +0,54 pontos ($p = 0,023$)

As questões onde a IA Verde obteve pontuação significativamente superior foram:

1. **Instruções Claras (Q10):** IA Verde +0,59 pontos ($p = 0,029$)
2. **Qualidade (categoria LOES-S, Q9-Q12):** IA Verde +0,28 pontos ($p = 0,035$)

As demais 17 questões (81,0% do total) não apresentaram diferenças estatisticamente significativas entre as duas abordagens, incluindo:

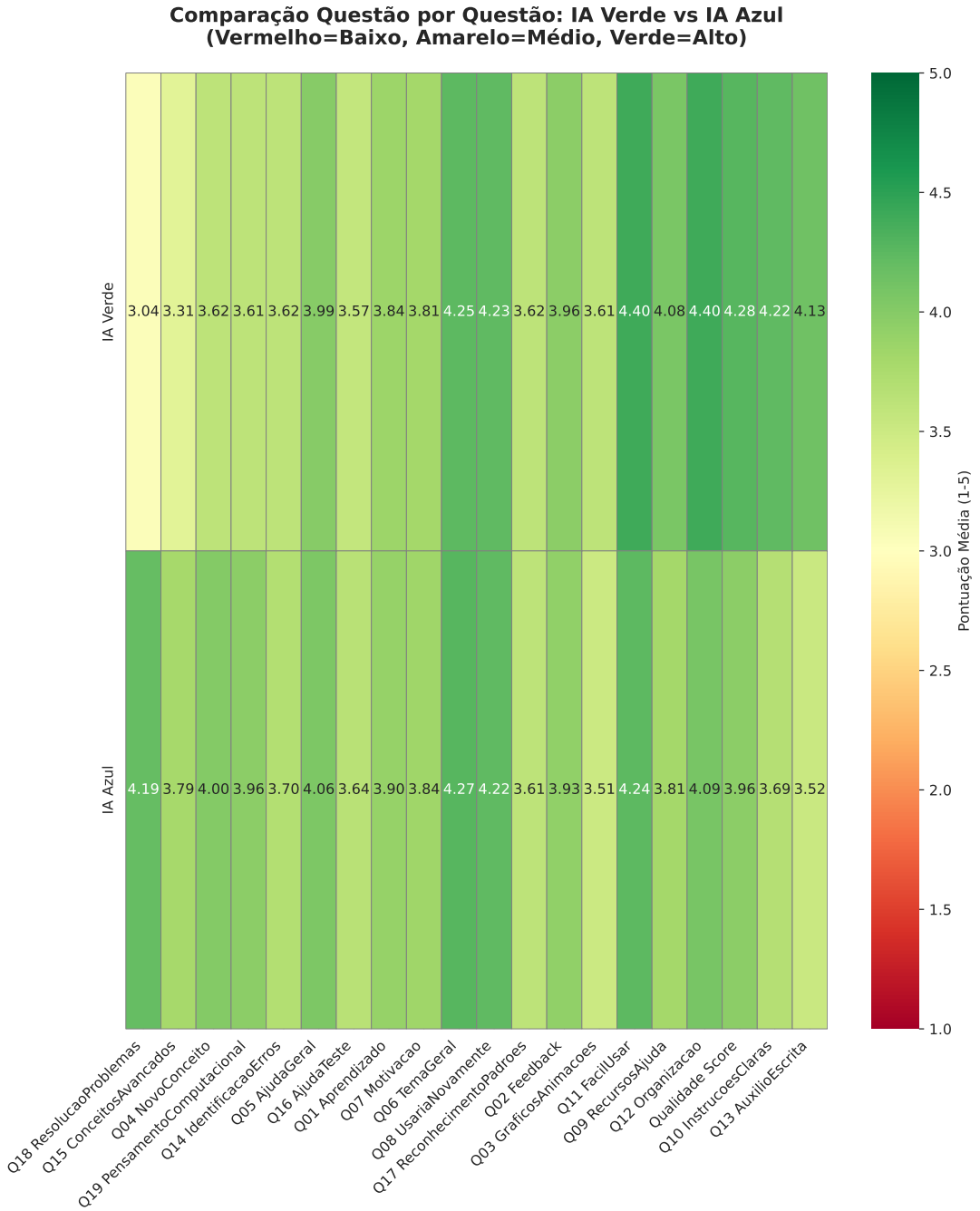


Figura 5.5: Heatmap comparando IA Verde e IA Azul em todas as questões do questionário. Cores mais verdes indicam pontuações mais altas, cores mais vermelhas indicam pontuações mais baixas.

- **5 questões de Aprendizado** (Q1-Q5): Todas as questões desta categoria apresentaram avaliações similares entre as duas IAs.
- **3 questões de Engajamento** (Q6-Q8): Todas as questões desta categoria apresentaram avaliações similares entre as duas IAs.
- **3 questões de Qualidade** (Q9, Q11, Q12): Apesar de Q10 (Instruções Claras) ter apresentado diferença significativa favorecendo a IA Verde, as demais questões de usabilidade não apresentaram diferenças individuais significativas.
- **5 questões adicionais específicas** (Q13, Q14, Q16, Q17, Q19): Das 7 questões adicionais, apenas 2 apresentaram diferenças significativas (Q15 e Q18).
- **1 questão de satisfação geral** (Q20): A satisfação geral não apresentou diferença significativa entre as abordagens.

Estas diferenças refletem os objetivos distintos de cada abordagem: a IA Azul focou em promover reflexão e compreensão profunda (destacando-se em resolução de problemas sem respostas diretas e compreensão de conceitos avançados), enquanto a IA Verde focou em facilitar a conclusão imediata das tarefas (destacando-se em clareza de instruções e qualidade/usabilidade geral). A maioria das questões (81,0%) não apresentou diferenças significativas, indicando que ambas as abordagens foram percebidas de forma similar na maior parte dos aspectos do aprendizado e da experiência de uso.

5.7 Resumo dos Resultados Principais

Os resultados principais do experimento podem ser resumidos da seguinte forma:

- A IA Azul (RAG + método socrático) foi significativamente superior em guiar os estudantes na resolução de problemas sem fornecer respostas diretas (diferença de +0,99 pontos, $p < 0,001$), confirmando a eficácia da abordagem socrática.
- A IA Azul promoveu melhor compreensão de conceitos avançados (+0,54 pontos, $p = 0,023$), medida através de questão adicional específica, sugerindo aprendizado mais profundo. Para pensamento computacional, embora a IA Azul apresente média ligeiramente superior (+0,13 pontos), a diferença não foi estatisticamente significativa ($p = 0,121$).
- A IA Verde apresentou melhor qualidade/usabilidade (-0,28 pontos na dimensão Qualidade do LOES-S, $p = 0,035$, teste de Wilcoxon pareado), refletindo melhor organização e clareza de instruções percebidas, o que resultou em notas mais altas (7,45 vs 4,27) e maior taxa de sucesso imediata (75,2% vs 43,0%).

- Não foram encontradas diferenças significativas em satisfação geral ($p = 0,810$, teste de Wilcoxon pareado), aprendizado percebido ($p = 0,891$, LOES-S) e engajamento ($p = 0,489$, LOES-S) entre os dois cenários, indicando que ambas as abordagens foram bem recebidas pelos estudantes.
- O tempo de resolução foi significativamente maior na IA Azul (24,48 vs 15,91 minutos), sugerindo maior engajamento cognitivo no processo de aprendizado.
- **O perfil do estudante (conhecimento prévio, desempenho acadêmico, experiência) não influenciou significativamente a percepção das IAs**, indicando que as características pedagógicas de cada abordagem são robustas e independem do perfil do usuário. Esta é uma descoberta importante que sugere aplicabilidade ampla de ambas as abordagens.
- Não foram encontradas correlações significativas entre o uso do tutor e o desempenho na primeira prova, apesar do experimento ter sido realizado antes da avaliação. Isto sugere que outros fatores (estudo individual, aulas, práticas anteriores) têm maior impacto no desempenho imediato, ou que o tempo de exposição ao tutor foi insuficiente para gerar efeitos mensuráveis.
- **A implementação mostrou-se economicamente viável**, com custo total de R$ 6,75 para 8,82 milhões de tokens gerados em 3.117 requisições, resultando em custo médio de R$ 0,0022 por requisição. Isto demonstra a viabilidade de implementação em larga escala em contextos educacionais.

Estes resultados indicam que cada abordagem apresenta características distintas que podem ser mais adequadas dependendo dos objetivos pedagógicos específicos do contexto de ensino. A IA Azul é mais eficaz para promover pensamento profundo e autonomia, enquanto a IA Verde é mais eficaz para conclusão rápida de tarefas. O LOES-S revela que ambas as abordagens são percebidas como igualmente efetivas para aprendizado e engajamento, com diferença apenas na dimensão de qualidade/usabilidade, favorável à IA Verde. Ambas as abordagens demonstram viabilidade técnica e econômica para implementação em larga escala.

Capítulo 6

Discussão

Este capítulo discute os resultados apresentados no capítulo anterior, analisando suas implicações, limitações e relação com a literatura existente.

6.1 Interpretação dos Resultados

Os resultados obtidos apresentam um panorama complexo sobre o uso de LLMs no ensino de APC. A hipótese inicial de que a IA Azul (RAG + método socrático) seria superior em todas as dimensões não foi completamente confirmada, mas os resultados revelam diferenças importantes que merecem análise cuidadosa.

6.2 Resolução de Problemas e Aprendizado Conceitual

O resultado mais significativo foi a superioridade da IA Azul em guiar os estudantes na resolução de problemas sem fornecer respostas diretas. Esta diferença (+0,99 pontos, $p < 0,001$, teste de Wilcoxon pareado com n=35) confirma que a abordagem socrática foi efetiva em seu objetivo principal: promover a construção do conhecimento através da reflexão, em vez da dependência de respostas prontas. Este resultado corrobora achados recentes de Favero et al. [6], que demonstraram empiricamente através de chatbots educacionais que tutores baseados em perguntas socráticas promovem reflexão e pensamento crítico significativamente melhor ($p < 0,001$) do que chatbots que fornecem respostas diretas, validando a eficácia do método socrático quando implementado com LLMs.

A melhor performance da IA Azul em conceitos avançados (+0,54 pontos, $p = 0,023$) sugere que a abordagem socrática promove uma compreensão mais profunda dos conceitos de programação. Para pensamento computacional, embora a IA Azul apresente

média ligeiramente superior (+0,13 pontos), a diferença não foi estatisticamente significativa ($p = 0,121$) na análise pareada. Embora os estudantes tenham obtido notas menores na IA Azul, o processo de construção do conhecimento através da reflexão guiada pode resultar em aprendizado mais sólido a longo prazo, hipótese que não pôde ser verificada devido às limitações de tempo para resolução das questões no experimento. Favero et al. [6] identificaram que perguntas socráticas estruturadas—incluindo clarificação de conceitos, investigação de pressupostos, exploração de evidências, análise de implicações e consideração de perspectivas alternativas—são particularmente efetivas em promover o desenvolvimento de habilidades de pensamento crítico, competências essenciais para o sucesso em programação.

Este resultado está alinhado com estudos que demonstram limitações importantes dos LLMs ao auxiliar programadores iniciantes. Hellas et al. [4] observaram que LLMs frequentemente fornecem soluções completas mesmo quando instruídos a não fazê-lo, o que pode prejudicar o desenvolvimento da autonomia dos estudantes. Adicionalmente, Jost et al. [3] demonstraram que o uso excessivo de LLMs pelos alunos para geração de código e depuração está negativamente correlacionado com as notas finais dos estudantes, enquanto o uso moderado para busca de explicações pode ser mais benéfico. A abordagem socrática implementada na IA Azul endereça especificamente estas limitações ao evitar fornecer soluções prontas.

6.3 Usabilidade versus Pedagogia

A IA Verde foi superior na dimensão de Qualidade do LOES-S ($p = 0,035$, teste de Wilcoxon pareado, n=35), que segundo Kay e Knaack [10] refere-se a aspectos técnicos e de design focando em usabilidade (recursos de ajuda, clareza de instruções, facilidade de uso, organização). Este resultado pode explicar sua maior taxa de sucesso imediata. A facilidade de uso e clareza de instruções são importantes para a experiência do estudante, mas não necessariamente garantem aprendizado efetivo. Este resultado sugere um compromisso entre qualidade/usabilidade e pedagogia: tutores mais fáceis de usar podem não ser os mais eficazes para o aprendizado profundo.

Estudos sobre a confiança dos estudantes em ferramentas de IA generativa indicam que a percepção de utilidade e o otimismo tecnológico influenciam significativamente a adoção dessas tecnologias, mas a facilidade de uso não tem efeito direto significativo sobre a confiança [14]. Isso reforça a importância de focar no valor prático e pedagógico das ferramentas, em vez de apenas na simplicidade de uso.

6.4 Desempenho versus Aprendizado

A diferença nas notas médias (7,45 vs 4,27) e taxa de sucesso (75,2% vs 43,0%) pode ser enganosa. Notas mais altas na IA Verde podem refletir a capacidade dos estudantes de copiar soluções, em vez de demonstrar compreensão real. O tempo médio de resolução foi significativamente maior na IA Azul (24,48 minutos) comparado à IA Verde (15,91 minutos), mesmo com o tempo de interação controlado (30 minutos para cada IA).

Esta diferença no tempo de resolução, combinada com a diferença nas notas, sugere que os estudantes na IA Azul estavam realmente pensando e construindo soluções através de reflexão guiada, dedicando mais tempo ao processo de aprendizado, mesmo que isso resultasse em notas menores. A restrição temporal de 30 minutos pode ter impactado desproporcionalmente a IA Azul (média de 24,48 minutos, próxima do limite) comparada à IA Verde (15,91 minutos). Sem esta restrição, é plausível que as taxas de sucesso convergissem, mantendo-se o diferencial pedagógico da abordagem socrática.

Estudos sobre o impacto de LLMs na educação de programação confirmam que o uso excessivo para geração de código e depuração está negativamente correlacionado com as notas finais, enquanto o uso moderado para busca de explicações pode ser mais benéfico [3]. Pesquisas com chatbots especializados como tutores também indicam que estudantes com uso moderado alcançam melhores resultados acadêmicos, enquanto o uso excessivo não leva a melhorias adicionais [8].

6.5 Implicações Pedagógicas

Os resultados têm implicações importantes para o uso de LLMs no ensino de programação. Conforme destacado por Kasneci et al. [12], a integração responsável de LLMs na educação requer uma estratégia clara e uma abordagem pedagógica com forte foco em pensamento crítico. Nossos resultados contribuem para essa discussão ao demonstrar empiricamente diferentes tradeoffs:

- **Objetivos de Aprendizado:** A escolha entre diferentes abordagens deve considerar os objetivos pedagógicos específicos. Se o objetivo é promover compreensão profunda e pensamento computacional, o método socrático com RAG pode ser mais adequado. Se o objetivo é facilitar a resolução imediata de problemas, respostas diretas podem ser preferíveis. Esta descoberta complementa as recomendações de Kasneci et al. [12] sobre a necessidade de alinhar o uso de LLMs aos objetivos educacionais.
- **Balanceamento:** Uma abordagem híbrida pode ser mais eficaz, combinando elementos de ambas as estratégias dependendo do contexto e das necessidades do es-

tudante. Por exemplo, respostas diretas para depuração rápida de erros sintáticos, mas método socrático para questões conceituais complexas. Al-Hossami et al. [7] demonstraram que LLMs podem empregar o método socrático especificamente para depuração de código, guiando estudantes através de perguntas que promovem identificação e compreensão de erros, em vez de simplesmente fornecer código corrigido. Esta abordagem híbrida—socrática para depuração conceitual e direta para erros sintáticos—pode otimizar tanto o aprendizado quanto a eficiência.

- **Retenção de Conhecimento:** Embora não tenha sido avaliado neste estudo, a abordagem socrática pode promover melhor retenção de conhecimento a longo prazo, mesmo que resulte em desempenho imediato menor. Esta hipótese merece investigação em estudos futuros.
- **Desenvolvimento de Competências:** Conforme argumentado por Kasneci et al. [12], é essencial que estudantes desenvolvam competências para compreender tanto as capacidades quanto as limitações das tecnologias de IA. A abordagem socrática contribui para este objetivo ao promover reflexão crítica sobre as respostas e soluções.

6.6 Limitações do Estudo

Algumas limitações devem ser consideradas na interpretação dos resultados:

6.6.1 Retenção de Conhecimento

O estudo não avaliou a retenção de conhecimento a longo prazo, que seria crucial para determinar qual abordagem promove aprendizado mais duradouro. Estudos futuros devem incluir avaliações de retenção após períodos significativos (por exemplo, ao final do semestre).

6.6.2 Efeito de Ordem

A ordem de apresentação das IAs (IA Verde seguido de IA Azul) pode ter influenciado as respostas, devido a efeitos de ordem ou fadiga. Estudos futuros devem considerar contrabalanceamento para controlar este efeito.

6.6.3 Contexto Específico

O estudo focou em uma única turma de Algoritmos e Programação de Computadores (APC). Futuramente, é importante ampliar a investigação para várias turmas de APC, a

fim de verificar se os efeitos observados se mantêm em outros contextos e com diferentes perfis de estudantes.

6.6.4 Restrição Temporal

O limite de 30 minutos pode ter impactado desproporcionalmente a IA Azul, cujo tempo médio (24,48 minutos) aproximou-se do limite, com 33% dos estudantes excedendo este valor. A IA Verde (15,91 minutos) manteve-se confortavelmente abaixo. Esta restrição pode ter influenciado as taxas de sucesso observadas. Estudos futuros devem considerar tempos flexíveis ou ilimitados, especialmente considerando a variabilidade individual no ritmo de processos reflexivos [22].

6.6.5 Contexto de Aplicação Controlado

O experimento foi conduzido em ambiente controlado durante o horário de aula, com todos os estudantes interagindo com as IAs no mesmo período e contexto. Em um cenário de uso extraclasse ou espontâneo, onde estudantes escolheriam quando, onde e por quanto tempo utilizar cada abordagem, os padrões de uso e preferências poderiam ser diferentes. A generalização dos resultados para contextos de aprendizado auto-dirigido requer investigação futura.

6.6.6 Viabilidade Econômica

A análise de custos operacionais (seção 5.6) demonstra que a implementação do sistema é economicamente viável, com custo total de R$ 6,75 para todo o experimento (3.117 requisições). O custo médio de R$ 0,0022 por requisição torna a solução acessível para implementação em larga escala em contextos educacionais, especialmente ao utilizar modelos de LLM de baixo custo como o Gemini 1.5 Flash.

É importante ressaltar, entretanto, que estes valores refletem os preços praticados no momento da realização do experimento (outubro de 2025), conforme documentado no Anexo II. Os custos operacionais estão sujeitos às políticas de precificação dos fornecedores de APIs de LLM, que podem sofrer alterações ao longo do tempo.

A arquitetura do sistema, implementada com LangGraph, já incorpora abstração da camada de conexão com LLMs, permitindo a substituição do fornecedor (atualmente Google Gemini) por outros vendors (como OpenAI, Anthropic, Cohere) ou por modelos auto-hospedados sem alterações significativas no código. Esta característica arquitetural mitiga os riscos relacionados à dependência de um único fornecedor e oferece flexibilidade para adaptação a mudanças nas condições comerciais ou disponibilidade de serviços.

Uma consideração importante ao utilizar APIs de terceiros é que as interações dos estudantes (perguntas, código submetido e respostas) são processadas nos servidores do fornecedor. Embora provedores estabelecidos geralmente ofereçam garantias de privacidade em seus termos de serviço, instituições educacionais podem ter requisitos específicos de proteção de dados, especialmente ao lidar com informações de menores de idade ou dados sensíveis. Nestes casos, soluções auto-hospedadas oferecem controle total sobre os dados, eliminando o compartilhamento com terceiros. A arquitetura modular do sistema permite esta transição quando houver disponibilidade de infraestrutura adequada.

6.6.7 Questões Diferentes entre Grupos

A utilização de questões diferentes entre os grupos (IA Verde e IA Azul) constitui uma limitação metodológica deste estudo. Esta decisão foi necessária porque todos os estudantes utilizaram primeiro a IA Verde e posteriormente a IA Azul na mesma sessão. Se as mesmas questões fossem utilizadas, estudantes poderiam aproveitar soluções ou insights da primeira sessão na segunda.

Para mitigar esta limitação, as questões foram balanceadas considerando conteúdo programático equivalente e dificuldade estimada comparável. No entanto, diferenças inerentes entre questões podem ter influenciado parcialmente as métricas de desempenho (notas e taxas de sucesso). As métricas de percepção (questionário LOES-S) são menos suscetíveis a este viés, pois avaliam a experiência com o sistema independentemente da questão específica.

6.7 Relação com Trabalhos Correlatos

Nossos resultados complementam e estendem pesquisas recentes sobre aplicação de RAG no ensino de computação. Dakshit [15] identificou que professores reconhecem o potencial de sistemas RAG como assistentes de ensino virtuais, mas destacou barreiras para implementação em larga escala. Nossa pesquisa endereça algumas dessas preocupações ao:

- Demonstrar viabilidade econômica com custos operacionais baixos (R$ 0,0022 por requisição).
- Implementar arquitetura flexível com abstração da camada de LLM através do LangGraph.
- Coletar percepção de estudantes e métricas objetivas de desempenho.

Shan [16] demonstrou que sistemas RAG podem aumentar o engajamento dos estudantes e reduzir a carga de trabalho dos educadores. Em relação ao engajamento, nossos resultados mostram que ambas as abordagens mantiveram os estudantes engajados de forma similar ($p = 0,489$, LOES-S), não confirmando superioridade da abordagem RAG nesta dimensão. No entanto, a IA Azul (RAG + socrático) obteve melhor avaliação em auxiliar a resolução de problemas sem fornecer respostas diretas (+0,99 pontos, $p < 0,001$, teste de Wilcoxon pareado), o que potencialmente reduz a dependência de intervenções diretas do professor, corroborando parcialmente o benefício de redução de carga de trabalho. Nosso estudo também revelou um compromisso importante não explorado em trabalhos anteriores: embora a abordagem RAG com método socrático promova melhor compreensão conceitual, a abordagem direta apresenta melhor usabilidade imediata, sugerindo que a escolha deve considerar objetivos pedagógicos específicos.

Nossa pesquisa complementa de forma significativa o trabalho de Favero et al. [6], que demonstraram empiricamente a eficácia de chatbots socráticos em promover pensamento crítico através de avaliações automáticas com conversas sintéticas geradas por LLMs. Enquanto Favero et al. focaram em *Theory of Knowledge* (ToK) utilizando modelos ajustados finamente[1] (Llama2 7B/13B), nossa pesquisa aplica o método socrático especificamente ao ensino de programação através de RAG e engenharia de prompt, validando a abordagem com experimentos reais envolvendo 201 estudantes. Esta diferença metodológica—experimentos controlados com estudantes reais versus avaliações sintéticas—oferece evidências complementares da eficácia do método socrático. Adicionalmente, nossa pesquisa revela um aspecto não investigado por Favero et al.: o compromisso entre usabilidade (favorecendo a IA Verde) e pedagogia (favorecendo a IA Azul), demonstrando que a escolha entre abordagens deve ser contextualizada conforme os objetivos educacionais específicos e não apenas a promoção de pensamento crítico.

[1] Ajuste fino adapta os parâmetros internos do modelo através de treinamento adicional com dados específicos do domínio, modificando permanentemente seus pesos [35]. RAG (Retrieval-Augmented Generation) [5], por sua vez, mantém o modelo inalterado e dinamicamente recupera informações relevantes de uma base de conhecimento externa durante a geração de respostas, sem modificar os parâmetros do LLM.

Capítulo 7

Conclusão

Este capítulo apresenta as conclusões do trabalho, resumindo os principais resultados, contribuições e limitações, além de sugerir direções para trabalhos futuros.

7.1 Resumo do Trabalho

Este trabalho investigou o impacto de LLMs com RAG e método socrático no aprendizado de Algoritmos e Programação de Computadores (APC), comparando com LLMs que fornecem respostas diretas. A pesquisa foi conduzida através do desenvolvimento e avaliação de um tutor inteligente que integra LLMs com técnicas de RAG e aplica o método socrático para guiar estudantes na resolução de problemas de programação.

O experimento foi realizado com estudantes da disciplina de APC da Universidade de Brasília ao longo de três semanas, totalizando 144 respostas ao questionário de percepção e 201 respostas para métricas de desempenho. Foram comparados dois cenários: (1) IA Verde, um LLM sem RAG que fornece respostas diretas, e (2) IA Azul, um LLM com RAG e método socrático que guia os estudantes através de perguntas e reflexões.

7.2 Síntese dos Resultados

A investigação revelou um compromisso fundamental entre qualidade/usabilidade e pedagogia em tutores inteligentes baseados em LLMs. A IA Azul (RAG + método socrático) demonstrou superioridade significativa em promover pensamento crítico, resolução autônoma de problemas e compreensão de conceitos avançados, embora com menor desempenho imediato e maior tempo de engajamento. Por outro lado, a IA Verde (respostas diretas) apresentou melhor usabilidade e eficiência na conclusão de tarefas.

As análises revelaram que o perfil do estudante não influenciou significativamente as percepções, sugerindo robustez e aplicabilidade ampla de ambas as abordagens. A

viabilidade econômica foi comprovada com custos operacionais baixos, tornando a implementação viável em larga escala.

A escolha entre abordagens deve ser contextualizada conforme objetivos pedagógicos específicos: a abordagem socrática favorece desenvolvimento de competências profundas e autonomia, enquanto respostas diretas favorecem eficiência e conclusão imediata de tarefas.

7.3 Contribuições

Este trabalho contribui para a literatura sobre uso de LLMs na educação em múltiplas dimensões:

7.3.1 Contribuições Teóricas

- Demonstra a viabilidade técnica e econômica de combinar RAG com método socrático em tutores inteligentes para ensino de programação.
- Fornece evidências empíricas sobre as diferenças entre abordagens diretas e socráticas no contexto de tutoria inteligente.
- Identifica e caracteriza compromissos entre usabilidade e pedagogia no design de tutores inteligentes baseados em LLMs.
- Destaca a importância de considerar objetivos pedagógicos na escolha de abordagens de tutoria.

7.3.2 Contribuições Práticas

- Desenvolve um tutor inteligente funcional com arquitetura flexível e modular, pronto para uso em contextos educacionais reais.
- Comprova viabilidade econômica da implementação em larga escala com custos operacionais acessíveis.
- Fornece diretrizes práticas para educadores sobre quando e como utilizar diferentes abordagens de tutoria inteligente conforme objetivos pedagógicos.
- Identifica desafios e limitações na implementação de tutores baseados em LLMs, orientando pesquisas futuras.

7.4 Implicações para o Ensino de Programação

Os resultados estabelecem diretrizes práticas para design e implementação de tutores inteligentes: (1) abordagens socráticas são mais adequadas quando o objetivo é desenvolvimento de pensamento crítico e autonomia, enquanto abordagens diretas favorecem eficiência e conclusão de tarefas; (2) sistemas híbridos adaptativos podem otimizar tanto pedagogia quanto usabilidade, alternando entre estratégias conforme contexto; (3) métricas de avaliação devem capturar não apenas desempenho imediato, mas também compreensão conceitual e desenvolvimento de competências; (4) a arquitetura com abstração de LLM garante flexibilidade e viabilidade econômica a longo prazo.

7.5 Limitações

As principais limitações deste estudo incluem: ausência de avaliação de retenção de conhecimento a longo prazo; potencial efeito de ordem na apresentação sequencial das abordagens; escopo limitado a uma disciplina específica; restrição temporal que pode ter impactado desproporcionalmente a abordagem socrática; e uso de questões diferentes entre grupos por necessidade metodológica. Uma discussão detalhada destas limitações e suas implicações é apresentada no capítulo anterior.

7.6 Trabalhos Futuros

Trabalhos futuros devem expandir e aprofundar esta pesquisa em várias direções:

7.6.1 Avaliação de Longo Prazo

- Expandir a avaliação para todo o semestre letivo, permitindo capturar a evolução do aprendizado e identificar padrões de uso ao longo do tempo.
- Avaliar a retenção de conhecimento após períodos significativos (por exemplo, ao final do semestre ou no semestre seguinte).
- Investigar a relação entre uso do tutor inteligente e desempenho em avaliações formais (provas, trabalhos).

7.6.2 Replicação e Generalização

- Replicar o estudo em múltiplas turmas e contextos educacionais para verificar a generalização dos resultados.

- Investigar a eficácia em diferentes níveis de ensino (ensino médio, pós-graduação).
- Explorar a aplicação em outras disciplinas de programação ou áreas relacionadas.

7.6.3 Melhorias Técnicas

- Implementar adaptação dinâmica da abordagem (socrática vs direta) baseada no perfil e progresso do estudante.
- Explorar diferentes modelos de LLM e técnicas de RAG para otimizar desempenho e custos.
- Desenvolver mecanismos de aprendizado contínuo que permitam ao tutor melhorar suas respostas com base no feedback implícito e explícito dos estudantes.

7.6.4 Análises Adicionais

- Completar as análises em andamento: correlações com desempenho acadêmico, análise por perfil do estudante, telemetria detalhada e análise de custos.
- Investigar padrões de uso e comportamento dos estudantes através de análise mais profunda dos dados de telemetria.
- Explorar relações entre características dos estudantes (experiência prévia, carga de trabalho) e eficácia do tutor.

7.6.5 Abordagens Híbridas

- Desenvolver e avaliar abordagens híbridas que combinam elementos de ambas as estratégias.
- Investigar quando e como alternar entre abordagens socráticas e diretas baseado no contexto e necessidades do estudante.
- Explorar personalização adaptativa da abordagem de tutoria, inspirado em frameworks como LearnRAG [16] que demonstram eficácia em sistemas de aprendizado adaptativo.
- Incorporar feedback de professores e administradores educacionais, seguindo a metodologia de Dakshit [15], para identificar barreiras específicas e recursos necessários para implementação em larga escala no contexto da UnB.

7.6.6 Integração com Plataformas Educacionais

- Integrar o tutor inteligente com o Aprender (plataforma Moodle da UnB), criando uma interface unificada onde estudantes possam visualizar questões, interagir com o tutor e submeter soluções no mesmo ambiente, eliminando a necessidade de alternar entre plataformas. Lin [36] demonstrou procedimentos técnicos para integração de ChatGPT em Moodle, identificando oito casos de uso potenciais incluindo tutoria adaptativa, suporte a design de cursos, interpretação de análises e suporte emocional.
- Desenvolver integração com sistemas de submissão automática (juiz automático) para fornecer assistência contextualizada baseada nos erros específicos identificados nos testes, permitindo que o tutor acesse resultados de compilação e execução em tempo real.
- Explorar funcionalidades adicionais além da resolução de questões, como geração de material didático personalizado, criação de exercícios adaptativos e feedback automatizado sobre submissões de código.
- Investigar o uso do tutor para suporte multilíngue, aproveitando as capacidades dos LLMs para atender estudantes que preferem explicações em diferentes idiomas, especialmente relevante no contexto da internacionalização da UnB.

7.7 Considerações Finais

Este trabalho investigou empiricamente o impacto de diferentes abordagens de tutoria inteligente baseadas em LLMs no ensino de programação, comparando respostas diretas com o método socrático combinado a RAG. Os resultados demonstram que não existe uma abordagem universalmente superior; cada estratégia apresenta características distintas que atendem a objetivos pedagógicos específicos.

A principal contribuição desta pesquisa está em caracterizar quantitativamente o compromisso entre usabilidade e pedagogia em tutores inteligentes, fornecendo evidências empíricas que podem orientar educadores e designers de sistemas educacionais na escolha ou combinação de abordagens conforme seus objetivos instrucionais. Conforme argumentado por Kasneci et al. [12], a integração responsável de LLMs na educação requer uma estratégia clara e uma abordagem pedagógica com forte foco em pensamento crítico, sendo essencial desenvolver competências para compreender tanto as capacidades quanto as limitações dessas tecnologias.

A arquitetura implementada com LangGraph oferece flexibilidade tecnológica para adaptação futura, permitindo substituição de fornecedores ou exploração de modelos

auto-hospedados. A viabilidade econômica comprovada torna a solução acessível para implementação institucional em larga escala. Espera-se que os resultados, contribuições e limitações identificadas neste trabalho inspirem pesquisas futuras e informem práticas educacionais no uso responsável e efetivo de tecnologias de IA generativa no ensino de programação.

Referências

[1] Qian, Yizhou e James Lehman: *Students' misconceptions and other difficulties in introductory programming: A literature review.* ACM Transactions on Computing Education, 18(1):1–24, October 2017. 1, 5

[2] Silveira, Rodrigo Fonseca, Maristela Holanda, Guilherme N. Ramos, Marcio Victorino e Dilma Da Silva: *Analysis of student performance and social-economic data in introductory computer science courses at the university of brasília.* Em *2022 IEEE Frontiers in Education Conference (FIE)*, páginas 1–8, 2022. 1

[3] Jošt, Gregor, Viktor Taneski e Sašo Karakatič: *The impact of large language models on programming education and student learning outcomes.* Applied Sciences, 14(10):4115, May 2024. 1, 3, 6, 43, 44

[4] Hellas, Arto *et al.*: *Exploring the responses of large language models to beginner programmers' help requests.* Em *Proceedings of the 2023 ACM Conference on International Computing Education Research*, ICER 2023, páginas 93–105, Chicago, IL, USA, 2023. `https://dl.acm.org/doi/10.1145/3568813.3600139`. 1, 6, 43

[5] Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel e Douwe Kiela: *Retrieval-augmented generation for knowledge-intensive NLP tasks.* arXiv preprint arXiv:2005.11401, 2020. `https://arxiv.org/abs/2005.11401`, Published at NeurIPS 2020. 1, 2, 6, 23, 48

[6] Favero, Lucile, Juan Antonio Pérez-Ortiz, Tanja Käser e Nuria Oliver: *Enhancing critical thinking in education by means of a socratic chatbot.* Em *ECAI'24: International Workshop on AI in Education and Educational Research (AIEER)*, Santiago de Compostela, Spain, October 2024. `https://arxiv.org/abs/2409.05511`. 2, 9, 42, 43, 48

[7] Al-Hossami, Erfan, Razvan Bunescu, Justin Smith e Ryan Teehan: *Can language models employ the socratic method? experiments with code debugging.* Em *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE '24, páginas 53–59. ACM, 2024. 2, 10, 45

[8] Sánchez-Vera, Fernando: *Subject-specialized chatbot in higher education as a tutor for autonomous exam preparation: Analysis of the impact on academic performance and students' perception of its usefulness.* Education Sciences, 15(1):26, December 2024. 2, 10, 44

[9] Lacerda, Daniel Pacheco, Aline Dresch, Adriano Proença e José Antonio Valle Antunes Júnior: *Design science research: método de pesquisa para a engenharia de produção*. Gestão & Produção, 20:741–761, 2013. 4, 12

[10] Kay, Robin H. e Liesel Knaack: *Assessing learning, quality and engagement in learning objects: The learning object evaluation scale for students (loes-s)*. ResearchGate, 2008. Available: `https://www.researchgate.net/publication/248114163`. 4, 15, 30, 43, 66

[11] Ayeni, Olufemi Oluseun, Najim Mahmoud Ajami Hamad, Obeagu Nnamdi Chisom, Blessing Osawaru e Adewusi Olutayo Ebenezer: *Ai in education: A review of personalized learning and educational technology*. GSC Advanced Research and Reviews, 18(2):261–271, 2024. `https://gsconlinepress.com/journals/gscarr/sites/default/files/GSCARR-2024-0062.pdf`. 6, 10

[12] Kasneci, Enkelejda *et al.*: *Chatgpt for good? on opportunities and challenges of large language models for education*. Learning and Individual Differences, 103:102274, 2023. `https://www.sciencedirect.com/science/article/abs/pii/S1041608023000195`. 6, 44, 45, 53

[13] Kasneci, Enkelejda *et al.*: *Leveraging the potential of large language models in education through playful and game-based learning*. Educational Psychology Review, 2024. 6

[14] Marimon, Francesc, María Belén Arias-Valle, Carlos Javier Coria-Augusto e Carmen María Larrea Arnau: *From optimism to trust: how chatgpt is reshaping student confidence in ai-driven learning*. RIED-Revista Iberoamericana de Educación a Distancia, 28(2):131–153, 2025. 6, 43

[15] Dakshit, Sagnik: *Faculty perspectives on the potential of rag in computer science higher education*. Em *Proceedings of the 25th Annual Conference on Information Technology Education*, SIGITE '24, páginas 19–24, New York, NY, USA, 2024. Association for Computing Machinery. `https://doi.org/10.1145/3686852.3686864`. 7, 47, 52

[16] Shan, R.: *Learnrag: Implementing retrieval-augmented generation for adaptive learning systems*. Em *2025 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, páginas 0224–0229, Fukuoka, Japan, 2025. 7, 48, 52

[17] Gao, Luyu, Xueguang Ma, Jimmy Lin e Jamie Callan: *Precise zero-shot dense retrieval without relevance labels*. Em *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2023, páginas 1762–1777. Association for Computational Linguistics, 2023. 7, 8, 22, 24

[18] Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher

Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever e Dario Amodei: *Language models are few-shot learners*. Advances in Neural Information Processing Systems, 33:1877–1901, 2020. https://arxiv.org/abs/2005.14165. 8

[19] Wei, Jason, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai e Quoc V. Le: *Finetuned language models are zero-shot learners*. arXiv preprint arXiv:2109.01652, 2021. https://arxiv.org/abs/2109.01652. 8

[20] Zhou, Yongchao, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan e Jimmy Ba: *Large language models are human-level prompt engineers*. arXiv preprint arXiv:2211.01910, 2022. https://arxiv.org/abs/2211.01910. 8, 9

[21] Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le e Denny Zhou: *Chain-of-thought prompting elicits reasoning in large language models*. Advances in Neural Information Processing Systems, 35:24824–24837, 2022. https://arxiv.org/abs/2201.11903. 9

[22] Goldin, A. P., O. Pedroncini e M. Sigman: *Producing or reproducing reasoning? socratic dialog is very effective, but only for a few*. PLOS ONE, 12:e0173584, 2017. 10, 46

[23] Shridhar, Kumar, Jakub Macina, Mennatallah El-Assady, Tanmay Sinha, Manu Kapur e Mrinmaya Sachan: *Automatic generation of socratic subquestions for teaching math word problems*. arXiv preprint arXiv:2211.12835, 2022. 10

[24] Jaques, Patrícia Augustin, Mariano Pimentel, Sean Wolfgand Matsui Siqueira e Ig Ibert Bittencourt: *Design science research: fazendo pesquisas científicas rigorosas atreladas ao desenvolvimento de artefatos computacionais projetados para a educação*. Em Pimentel, Mariano, Denise Filippo e Flávia Maria Santoro (editores): *Metodologia de Pesquisa Científica em Informática na Educação: Concepção da Pesquisa*, volume 1 de *Série Metodologia de Pesquisa em Informática na Educação*, capítulo 5. Sociedade Brasileira de Computação (SBC), Porto Alegre, 2022. https://ceie.sbc.org.br/metodologia/. 11

[25] Lobb, Richard e Tim Hunt: *Coderunner: A question type plugin for moodle*. Moodle Plugin, 2024. https://moodle.org/plugins/qtype_coderunner, Version 5.4.1. 14

[26] Sociedade Brasileira de Computação: *Código de conduta para autores em publicações da sbc*. https://sol.sbc.org.br/index.php/indice/conduta, 2024. Accessed: December 4, 2024. 17

[27] Consensus: *Consensus: Ai-powered academic search engine*, 2024. https://consensus.app/, Ferramenta de busca acadêmica baseada em IA para análise de literatura científica. 17

[28] Django Software Foundation: *Django: The web framework for perfectionists with deadlines*, 2024. https://www.djangoproject.com/, Version 4.2. 20

[29] LangChain AI: *Langgraph: Build stateful, multi-actor applications with llms*, 2024. https://github.com/langchain-ai/langgraph. 20, 21

[30] Chase, Harrison: *Langchain: Building applications with llms through composability*, 2023. https://github.com/langchain-ai/langchain. 21

[31] Qdrant Team: *Qdrant: Vector database for the next generation of ai applications*, 2024. https://qdrant.tech/. 23

[32] Downey, Allen B.: *Pense em Python: Pense como um cientista da computação*. Novatec Editora, 2016, ISBN 9788575225080. https://books.google.com.br/books?id=hiRjDAAAQBAJ. 23

[33] Python Software Foundation: *Python language reference manual*. https://docs.python.org/, 2024. Accessed: December 4, 2024. 23

[34] OpenAI: *What are tokens and how to count them?* https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them, 2023. Accessed: December 8, 2024. 38

[35] Parthasarathy, Venkatesh Balavadhani, Ahtsham Zafar, Aafaq Khan e Arsalan Shahid: *The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities*, 2024. https://arxiv.org/abs/2408.13296. 48

[36] Lin, Jingjing: *Chatgpt and moodle walk into a bar: A demonstration of ai's mind-blowing impact on e-learning*. Available at SSRN, 2023. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4393445, SSRN ID 4393445. 53

Apêndice A

Termo de Consentimento Livre e Esclarecido

Este apêndice apresenta o Termo de Consentimento Livre e Esclarecido (TCLE) apresentado aos participantes antes da coleta de dados.

Termo de Consentimento

Você está sendo convidado a participar da pesquisa **"Pedagogias para Ensinagem do Professor de Computação aplicadas a Educação Básica"**, de responsabilidade do Prof. Edison Ishikawa, por meio da utilização e avaliação da ferramenta Tutor de APC, desenvolvido por Lucas de Moura Pereira, estudante de graduação da Universidade de Brasília. O objetivo desta pesquisa é avaliar a ferramenta Tutor de APC. Assim, gostaria de consultá-lo/a sobre seu interesse e disponibilidade de cooperar com a pesquisa.

Você receberá todos os esclarecimentos necessários antes, durante e após a finalização da pesquisa, e lhe asseguro que o seu nome não será divulgado, sendo mantido o mais rigoroso sigilo mediante a omissão total de informações que permitam identificá-lo/a. Os dados provenientes de sua participação na pesquisa, tais como questionários e entrevistas ficarão sob a guarda do/da pesquisador/a responsável pela pesquisa.

A coleta de dados será realizada por meio de formulário eletrônico. É para estes procedimentos que você está sendo convidado a participar. Sua participação na pesquisa não implica em nenhum risco.

Espera-se com esta pesquisa melhorar o processo de Ensino e Aprendizagem em APC.

Sua participação é voluntária e livre de qualquer remuneração ou benefício. Você é livre para recusar-se a participar, retirar seu consentimento ou interromper sua participação

a qualquer momento. A recusa em participar não irá acarretar qualquer penalidade ou perda de benefícios.

Se você tiver qualquer dúvida em relação à pesquisa, você pode entrar em contato pelo e-mail `180034898@aluno.unb.br`.

A equipe de pesquisa garante que os resultados do estudo serão devolvidos aos participantes por meio de apresentação dos resultados, podendo ser publicados posteriormente na comunidade científica.

Você concorda com os termos desta pesquisa?

Sim

Não

Apêndice B

Questões Utilizadas no Experimento

Este anexo apresenta as questões de programação utilizadas no experimento, organizadas por cenário (IA Verde e IA Azul). As questões foram balanceadas em termos de dificuldade e cobertura de conteúdo para permitir comparação justa entre as duas abordagens de tutoria.

B.1 Questão 1 - IA Verde: Espaços na String

Descrição: Elabore um programa que lê um número inteiro positivo N e apresenta N vezes o caractere espaço concatenados com a string "olha quantos espaços tenho na esquerda!".

Entrada: Um número inteiro positivo.

Saída: O caractere espaço repetido um número N de vezes e concatenado com a string "olha quantos espaços tenho na esquerda!".

Observações: Não é necessário validar se os valores de entrada estão dentro dos intervalos definidos.

Exemplos:

- No primeiro exemplo, seja o número inteiro positivo 3, foi impresso:

  `"   olha quantos espaços tenho na esquerda!"`

- No segundo exemplo, seja o número inteiro positivo 1, foi impresso:

  `" olha quantos espaços tenho na esquerda!"`

B.2 Questão 1 - IA Azul: Média Ponderada

Descrição: Elabore um programa que lê três notas de um aluno e o peso de cada nota e apresenta a média final deste aluno.

Entrada:

- Uma linha contendo três números do tipo float n_1, n_2 e n_3, que representam as notas de um aluno.
- A segunda linha da entrada contém três números inteiros p_1, p_2 e p_3, que representam os pesos de cada nota.

Saída: Uma linha com a média ponderada do aluno, com seis casas decimais.

Observações: Não é necessário validar se os valores de entrada estão dentro dos tipos e intervalos definidos.

Exemplo: A nota 9,1 tem peso 1, a nota 5,1 tem peso 2 e a nota 5,1 tem peso 3. Então a média ponderada resultante é 5,766667.

B.3 Questão 2 - IA Verde: Dividindo Doces

Descrição: Ana ganhou um saco com n doces de sua mãe. Como Ana é muito legal ela decidiu dividir seus doces com seus m amigos.

Ana e seus m amigos (totalizando $m+1$ pessoas) sentam-se em círculo, e Ana começa a dividir os doces: coloca primeiro 1 para ela, depois um para a pessoa à sua direita, e seguindo nesse sentido até os doces acabarem.

No entanto Ana não quer que nenhum de seus amigos se sinta menos especial recebendo menos doces que os outros, então ela pediu para você ajudá-la: antes de encontrar seus amigos e dividir os doces, Ana vai tirar 0 ou mais doces do saco para que a divisão entre ela e seus amigos fique exata.

Ana quer retirar a menor quantidade possível de doces antes da divisão. Você pode ficar com os doces que sobrarem :D

Escreva um programa que imprima quantos doces Ana e seus amigos ganharam cada um, e quantos doces você ganhou.

Entrada: 2 números inteiros, n e m ($0 \leq m < n \leq 100000$) que é o número de doces que Ana ganhou de sua mãe e o número de amigos de Ana, respectivamente.

Saída: Dois inteiros, um por linha: o número de doces que cada um dos amigos de Ana (e ela mesma) ficou, e o número de doces que você ganhou.

Observações: Não é necessário validar se os valores de entrada estão dentro dos intervalos e tipos definidos. Você deve imprimir todos os valores como inteiros.

Exemplos:

- Ana recebeu 10 doces, e tem 4 amigos. Contando com ela, são 5 pessoas, logo, ela divide 2 doces para cada e não sobram doces.

- Ana recebeu 12 doces, e tem 4 amigos. Contando com ela, são 5 pessoas, logo, ela divide 2 doces para cada e sobram 2 doces.
- Ana recebeu 5 doces, e tem 4 amigos. Contando com ela, são 5 pessoas, logo, ela divide 1 doce para cada e não sobram doces.

B.4 Questão 2 - IA Azul: Calcule as Médias

Descrição: Escreva um programa que leia 3 números inteiros positivos e efetue o cálculo das médias Aritmética (A), Ponderada (P) e Harmônica (H) dependendo da letra dada pelo usuário, mostre qual o tipo de média e qual o valor da média. No caso do usuário digitar qualquer outro caractere, apresente a mensagem 'Operacao inexistente'.

Entrada:

- Linha contendo as três notas que são três números reais positivos.
- Linha contendo um caractere (para determinar qual a média), sendo (P) Ponderada, (H) Harmônica e (A) Aritmética.
- Caso o caractere seja 'P', deve-se solicitar os três pesos de cada nota enviada, que são números positivos inteiros.

Saída:

- Na primeira linha, o tipo de média que ele fez ("Harmonica", "Ponderada", "Aritmetica"ou "Operacao inexistente").
- Na segunda linha, caso tenha sido digitado um caractere válido, o resultado da média com precisão de 2 casas decimais.

Observações: Não é necessário validar se os valores de entrada estão dentro dos intervalos e tipos definidos.

B.5 Questão 3 - IA Verde: Ajuda Necessária

Descrição: A Associação de Pedreiros de Ceilândia (APC) está precisando urgentemente de uma pessoa que compute o número com a maior quantidade de divisores entre 1 e n! Agora você deve estar se perguntando, por quê? A resposta é óbvia: como todos os pedreiros são perfeccionistas, eles querem saber em quantas partes podem dividir uma parede, ou o chão, então eles querem escolher o número ideal n para que possam usar os tamanhos que quiserem. Gui está concorrendo a presidência de APC e uma de suas

promessas é de já disponibilizar essa funcionalidade para todos os associados, mas Gui não se dá muito bem com programação. Apesar de vocês não se falarem há mais de 10 anos, você topou, pelo bem da amizade que existiu um dia entre vocês, implementar essa funcionalidade.

Entrada: Um número inteiro n ($1 \leq n \leq 1000$).

Saída: Dois números inteiros x e d, correspondendo ao menor número com o maior número de divisores entre 1 e n e quantos divisores ele possui, respectivamente.

Observações: Não é necessário validar se os valores de entrada estão dentro dos intervalos e tipos definidos.

Exemplo: O resultado é 4 3 pois 4 é o maior número com a maior quantidade de divisores no intervalo e 3 é a quantidade de divisores.

B.6 Questão 3 - IA Azul: Lotação de Ônibus

Descrição: A APC Transportes é uma das empresas de ônibus que prestam serviços de transporte público na Nlogonia. Entretanto, é notória a insatisfação dos seus moradores em relação ao fato de que diversos ônibus circulam lotados, ou com excesso de passageiros de acordo com a sua capacidade máxima.

Diante das reclamações, a APC decidiu fiscalizar com mais rigor o transporte dos passageiros nos ônibus: foi instalado um sensor em cada porta que detecta a quantidade de pessoas que saem e entram de cada ônibus.

A APC tem os registros do sensor de todo um dia de funcionamento de um determinado ônibus, que sempre começa vazio. Conforme as normas de utilização dos ônibus da APC, todos os passageiros que irão descer em um ponto devem sair antes de outros passageiros entrarem no ônibus. Porém, ainda assim a APC têm tido dificuldade em decidir se a capacidade máxima do ônibus foi excedida ou não.

Escreva um programa que, dada uma sequência de leituras do sensor e a capacidade máxima do ônibus, determina se sua capacidade máxima foi excedida em algum momento.

Entrada:

- A primeira linha da entrada contém dois inteiros N e C ($1 \leq N \leq 1000, 1 \leq C \leq 1000$), indicando o número de leituras realizadas pelo sensor e a capacidade máxima do ônibus, respectivamente.
- Cada uma das N linhas seguintes contém uma leitura do sensor, representada por dois inteiros S e E ($0 \leq S \leq 1000, 0 \leq E \leq 1000$), indicando quantas pessoas saíram e quantas pessoas entraram naquele ponto de ônibus, respectivamente.

Saída: Seu programa deve imprimir "sim", caso a capacidade do ônibus tenha sido excedida em algum momento, ou "nao"caso contrário.

Observações: Não é necessário validar se os valores de entrada estão dentro dos intervalos e tipos definidos.

Exemplos:

- No primeiro exemplo, em cada parada do ônibus, a capacidade do ônibus se configura na seguinte ordem: $5 \rightarrow 10 \rightarrow 10 \rightarrow 7 \rightarrow 0$. Como a capacidade do ônibus c é igual a 10, em nenhum momento (após sair de uma parada) o ônibus excedeu sua capacidade máxima.
- No segundo exemplo, em cada parada do ônibus, a capacidade do ônibus se configura na seguinte ordem: $3 \rightarrow 8 \rightarrow 10 \rightarrow 11 \rightarrow 9$. Como a capacidade do ônibus c é igual a 10, após a quarta parada houve superlotação do ônibus em 1 passageiro, momento em que haviam 11 passageiros dentro do ônibus.

Apêndice C

Questionário LOES-S Adaptado

Este anexo apresenta o questionário aplicado para coletar a percepção dos estudantes sobre os dois cenários de tutoria inteligente (IA Verde e IA Azul). O questionário é composto pela escala LOES-S (Learning Object Evaluation Scale for Students) desenvolvida por Kay e Knaack [10], complementada por questões adicionais específicas para o contexto de tutoria inteligente com LLMs.

Instruções

Para cada uma das afirmações a seguir, por favor indique o quanto você concorda ou discorda, selecionando uma das opções:

- Discordo totalmente
- Discordo parcialmente
- Indiferente
- Concordo parcialmente
- Concordo totalmente

Dimensão: Aprendizado

1. Trabalhar com o Tutor de APC me ajudou a aprender.
2. O feedback do Tutor de APC me ajudou a aprender.
3. Os gráficos e animações do Tutor de APC me ajudaram a aprender.
4. O Tutor de APC me ajudou a aprender um novo conceito.
5. No geral, o Tutor de APC me ajudou a aprender.

Dimensão: Engajamento

6. Gostei do tema geral do Tutor de APC.
7. Achei o Tutor de APC motivador.
8. Gostaria de usar o Tutor de APC novamente.

Dimensão: Qualidade

9. Os recursos de ajuda do Tutor de APC foram úteis.
10. As instruções do Tutor de APC foram fáceis de seguir.
11. O Tutor de APC foi fácil de usar.
12. O Tutor de APC foi bem-organizado.

Questões Adicionais Específicas

13. O Tutor de APC auxiliou você na escrita do código?
14. O Tutor de APC ajudou você a identificar e corrigir erros no código?
15. O Tutor de APC contribuiu para sua compreensão de conceitos além da programação?
16. O Tutor de APC te ajudou a testar o código, ou seja, a identificar o erro que fazia as respostas ocultas estarem incorretas?
17. O Tutor de APC auxiliou você a reconhecer padrões que facilitaram a codificação?
18. **O Tutor de APC auxiliou você a solucionar problemas sem entregar a resposta diretamente?**
19. O tutor de APC me ajudou a pensar computacionalmente me ajudando a projetar algoritmos.

Avaliação Geral

20. De 1 a 5, sendo 1 muito insatisfeito e 5 muito satisfeito, que nota você daria para o Tutor de APC?

Escala: 1 — 2 — 3 — 4 — 5

Questões Abertas (Opcionais)

21. O que você achou positivo no Tutor de APC?

    (Resposta em texto livre)

22. O que você achou que poderia ser melhorado no Tutor de APC?

    (Resposta em texto livre)

Nota: Este questionário foi aplicado separadamente para cada cenário (IA Verde e IA Azul), permitindo comparar a percepção dos estudantes sobre as duas abordagens de tutoria inteligente. A numeração contínua (1-22) e o agrupamento por dimensões apresentados neste anexo são utilizados apenas para facilitar a referência às questões nas análises do Capítulo 5. Na aplicação com os estudantes, as questões foram apresentadas na mesma ordem sequencial mostrada aqui, porém sem numeração explícita e sem identificação dos agrupamentos (Aprendizado, Engajamento, Qualidade, etc.), garantindo que as respostas refletissem a percepção genuína sem viés de categorização.

Apêndice D

Prompts Utilizados nos Cenários de IA

Este anexo apresenta os prompts de sistema e templates de prompts humanos utilizados para configurar os dois cenários de IA no tutor inteligente, conforme implementados no arquivo `experiment.py` do projeto.

D.1 Cenário IA Verde (Sem RAG, Respostas Diretas)

D.1.1 Prompt de Sistema

```
Você é um assistente de programação prestativo. Ajude o usuário com suas
dúvidas de programação. Forneça respostas claras e diretas às perguntas.
```

D.1.2 Template de Prompt Humano

```
Pergunta: {question}
Código: {user_code}
Erro: {code_error}
```

D.2 Cenário IA Azul (Com RAG e Método Socrático)

D.2.1 Prompt de Sistema

```
Você é um tutor de programação socrático. Ajude os alunos a aprenderem
```

```
fazendo perguntas orientadoras em vez de fornecer respostas diretas.
Foque em ajudá-los a entender conceitos e descobrir soluções por conta
própria. Use o contexto fornecido quando relevante.
```

D.2.2 Template de Prompt Humano

```
Pergunta do Aluno: {question}
Código do Aluno: {user_code}
Mensagem de Erro: {code_error}

Lembre-se de:
1. Faça perguntas orientadoras em vez de dar respostas diretas
2. Foque em conceitos e abordagens de resolução de problemas
3. Considere o material educacional quando relevante
4. Incentive a exploração e descoberta independente
```

D.3 Prompts Auxiliares (IA Azul)

Além dos prompts principais, o cenário IA Azul utiliza prompts auxiliares para funcionalidades específicas, definidos no arquivo `constants.py`:

D.3.1 Prompt de Validação (NO_CODE_PROMPT)

Este prompt é utilizado para garantir que o tutor não forneça código completo nas respostas:

```
Você é um tutor de programação socrático. Com base na <pergunta_original> e
na <resposta_preliminar>, forneça uma resposta final e aprimorada para o aluno.

Siga estas regras estritamente:
1. **Guie, não resolva**: Faça perguntas que estimulem o raciocínio do aluno.
2. **Sem código**: Não escreva nenhum código.
3. **Sem mencionar o contexto**: Não revele que você usou materiais de referência.

<pergunta_original>
{question}
</pergunta_original>
```

```
<resposta_preliminar>
{llm_response}
</resposta_preliminar>

Resposta Final (para o aluno):
```

D.3.2 Prompt de Roteamento (ROUTER_PROMPT)

Este prompt classifica se a consulta do usuário deve ser encaminhada para a base de vetores:

```
Você é um classificador que decide se uma consulta do usuário deve ser
encaminhada para a base de vetores de conceitos de programação.

RETORNE **apenas** uma das duas palavras abaixo (sem pontuação adicional):
• ‘vectorstore‘ - a consulta envolve programação, código, mensagens de erro,
  algoritmos ou conceitos de ciência da computação.
• ‘off_topic‘ - qualquer outro assunto (cumprimentos, perguntas pessoais,
  tópicos gerais etc.).

<consulta_usuario>
{user_input}
</consulta_usuario>
Resposta:
```

D.3.3 Prompt HYDE (HYDE_PROMPT)

Este prompt é utilizado na estratégia HYDE para gerar uma resposta hipotética que melhora a busca no vector store:

```
Você é um especialista em reformular perguntas para melhorar a busca em uma
base de vetores. Analise o <historico_conversa> e a <pergunta_recente> para
criar uma resposta hipotética que capture a intenção completa do usuário.

<historico_conversa>
{history}
</historico_conversa>

<pergunta_recente>
```

```
{question}
</pergunta_recente>

Resposta Hipotética (para busca):
```

Apêndice E

Interfaces do Usuário

Este anexo apresenta as principais interfaces disponíveis para os usuários do tutor inteligente, incluindo telas de autenticação, seleção de versão de IA e interação com o chat.

E.1 Tela de Login

A Figura E.1 apresenta a tela de login do sistema, onde os usuários fazem autenticação para acessar o tutor inteligente.

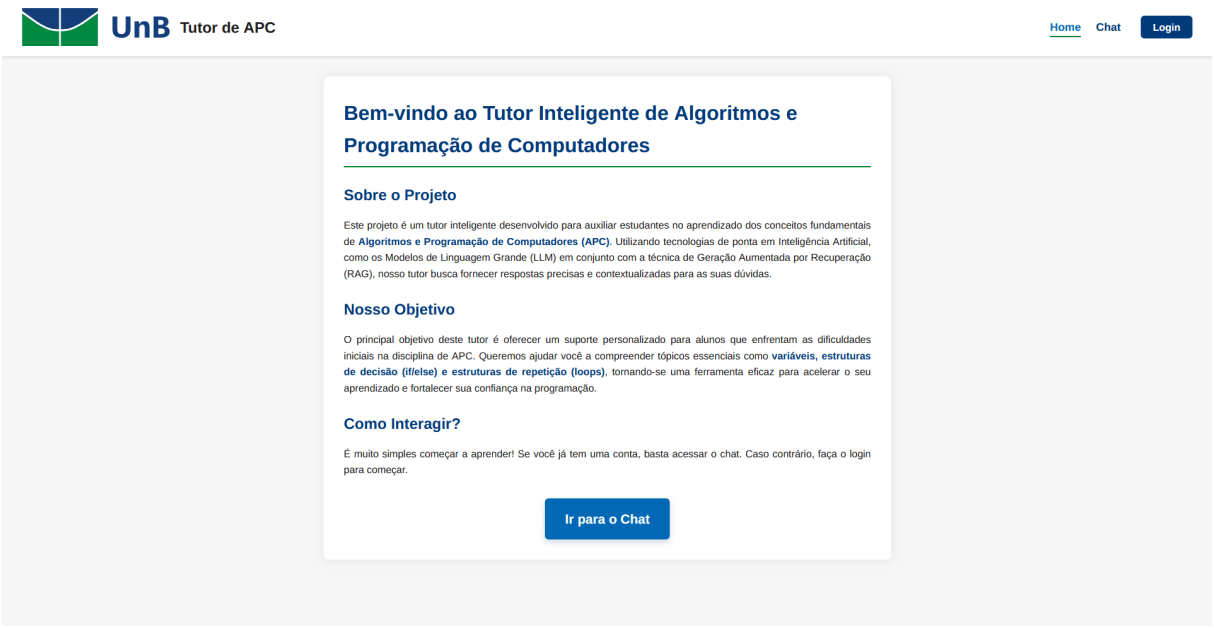


Figura E.1: Tela de login do tutor inteligente

E.2 Interface Administrativa

A Figura E.2 apresenta a interface administrativa utilizada para selecionar a versão de IA ativa (IA Verde ou IA Azul) durante o experimento.

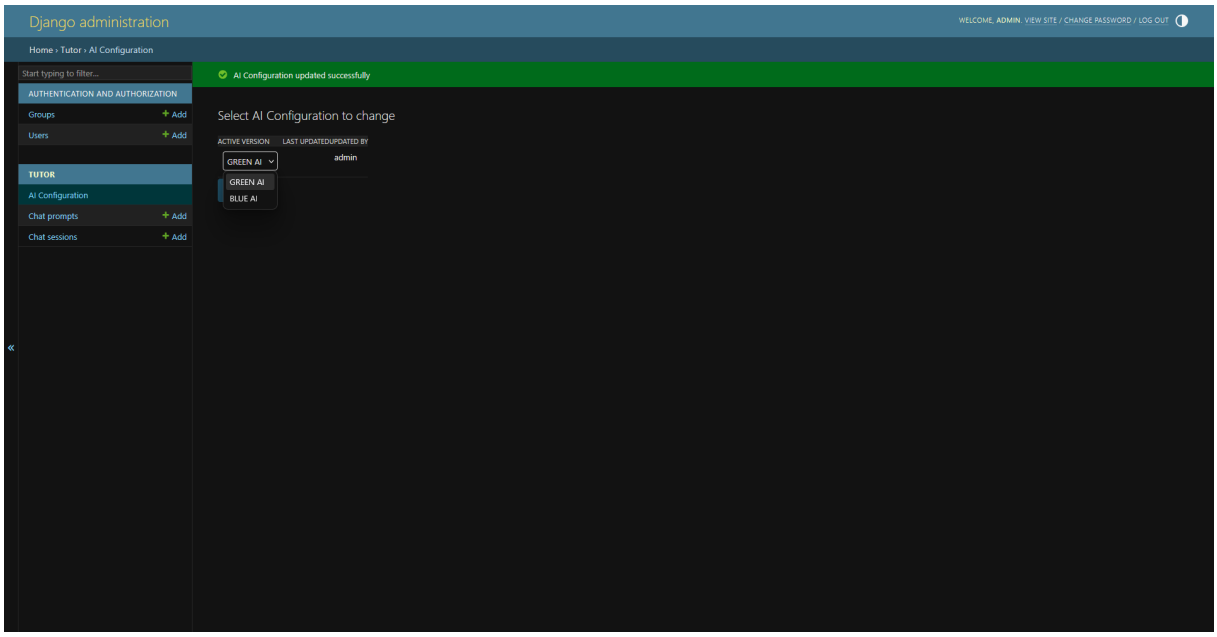


Figura E.2: Interface administrativa para seleção da versão de IA

E.3 Interface de Chat - IA Verde

A Figura E.3 apresenta a interface de chat quando o sistema está configurado com a IA Verde, que fornece respostas diretas às perguntas dos estudantes.

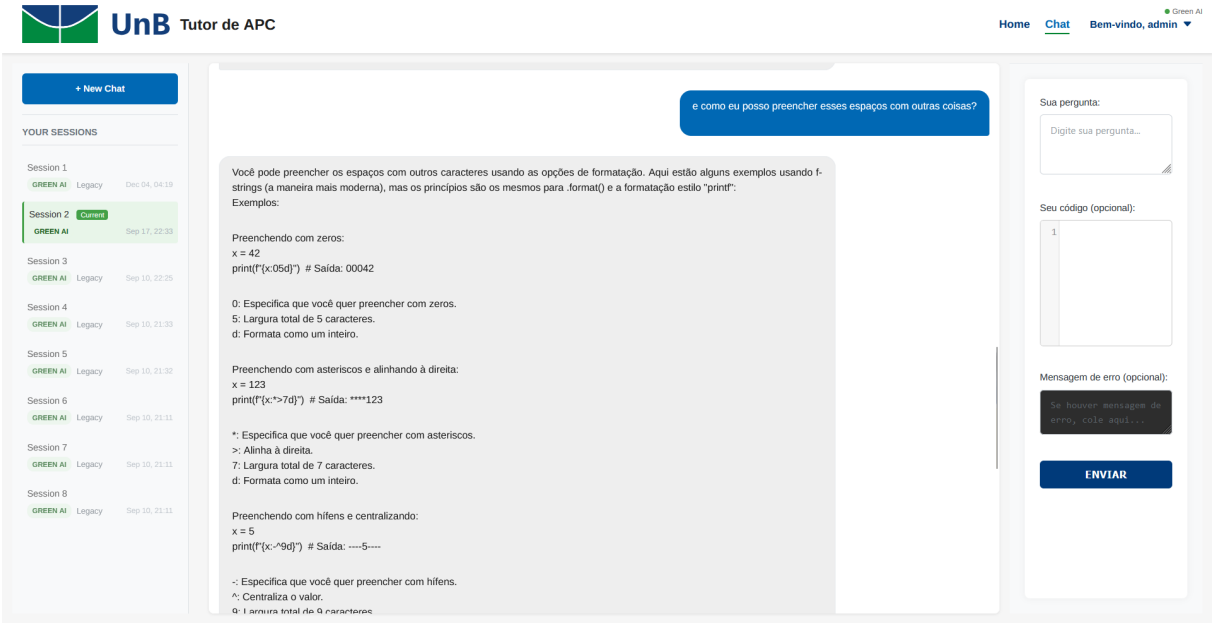


Figura E.3: Interface de chat com IA Verde (respostas diretas)

E.4 Interface de Chat - IA Azul

A Figura E.4 apresenta a interface de chat quando o sistema está configurado com a IA Azul, que aplica o método socrático e utiliza RAG para recuperar informações do material didático.

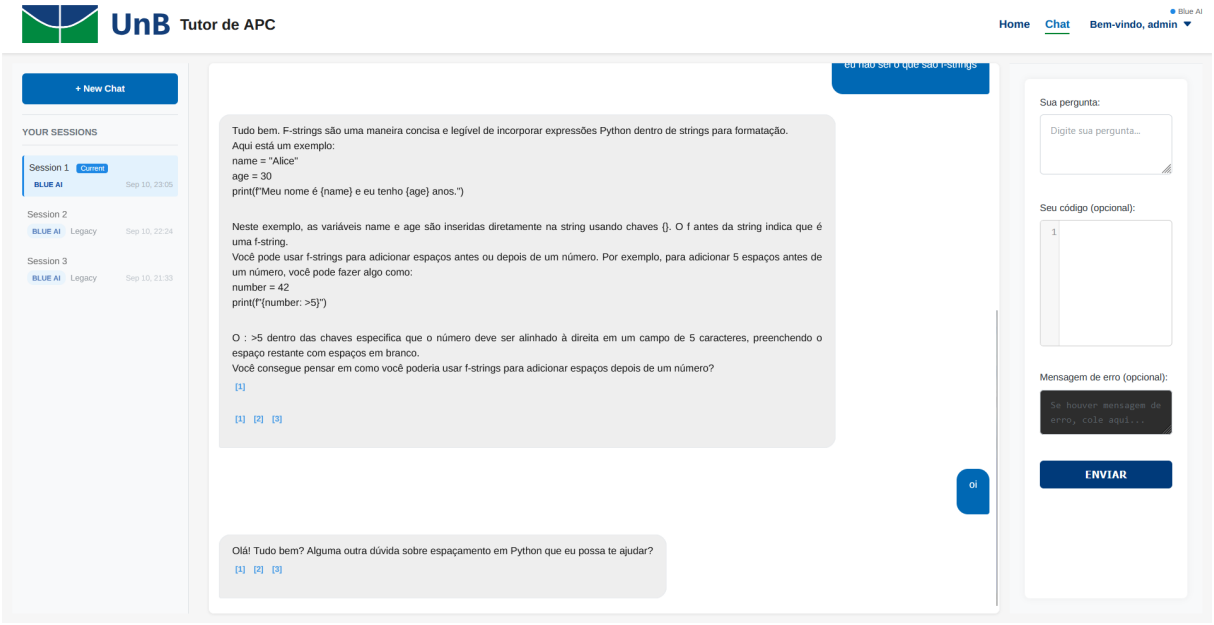


Figura E.4: Interface de chat com IA Azul (método socrático e RAG)

Anexo I

Plano de Ensino da Disciplina APC

O Plano de Ensino da disciplina CIC0004 - Algoritmos e Programação de Computadores (Turma 06) do segundo semestre de 2025 está apresentado nas páginas a seguir. Este documento detalha os objetivos, ementa, programa, metodologia, procedimentos de avaliação, cronograma e referências bibliográficas da disciplina, que constituiu o contexto de aplicação do experimento desta pesquisa.

CIC0004 – Algoritmos e Programação de Computadores Turma 06

Plano de Ensino – 2025/2

Objetivos

A disciplina CIC0004 - Algoritmos e Programação de Computadores busca preparar o estudante para a percepção e consciência da importância da computação para a sociedade e para o desenvolvimento da capacidade cognitiva humana; apresentar e desenvolver o pensamento computacional como competência fundamental para a comunicação com os dispositivos de computação; expressar formalmente o pensamento computacional através do desenho de algoritmos; efetivar a comunicação com o computador através da codificação dos algoritmos em uma linguagem de programação.

Ementa

Princípios fundamentais de construção de programas. Construção de algoritmos e sua representação em pseudocódigo e linguagens de alto nível. Noções de abstração. Especificação de variáveis e funções. Testes e depuração. Padrões de soluções em programação. Noções de programação estruturada. Identificadores e tipos. Operadores e expressões. Estruturas de controle: condicional e repetição. Entrada e saída de dados. Estruturas de dados estáticas: agregados homogêneos e heterogêneos. Iteração e recursão. Noções de análise de custo e complexidade. Desenvolvimento sistemático e implementação de programas. Estruturação, depuração, testes e documentação de programas. Resolução de problemas. Aplicações em casos reais e questões ambientais.

Programa

1. Algoritmos e subalgoritmos
2. Variáveis, valores e tipos
3. Estruturas sequenciais
4. Estruturas condicionais
5. Estruturas de repetição
6. Agregados Homogêneos - strings
7. Agregados – Listas e Tuplas
8. Funções
9. Agregados - Dicionários
10. Recursividade

Metodologia

A disciplina será ministrada de modo presencial conforme a resolução do Conselho de Ensino, Pesquisa e Extensão da Universidade de Brasília (UnB). Será conduzida por meio de atividades presenciais em sala de aula e nos Laboratórios de Informática (LINF) e de atividades extraclasse para apoiar o aprendizado.

As atividades presenciais envolvem:

- Aulas teóricas em sala de aula;
- Aulas práticas no LINF para iniciar a resolução de lista de exercícios para estudo e fixação do conteúdo (avaliação formativa) e retiradas de dúvidas; e
- Provas com o conteúdo de um ou mais tópicos da disciplina.

As atividades extraclasse envolvem:

- Leitura obrigatória do material de referência para compreensão do conteúdo teórico;
- Completar a resolução de lista de exercícios para estudo e fixação do conteúdo;

- Estudo da videoaula gravada, que aborda os pontos principais de cada tópico da disciplina e resolução de exercícios; e
- Atendimento para os estudantes realizado pelos monitores, tutores e professores.

A comunicação entre os estudantes, docente e monitores ocorrerá com ferramentas disponíveis, dando preferência pelas institucionais. A plataforma educacional Aprender3 será utilizada para armazenamento de todo o material didático da disciplina, envolvendo as leituras obrigatórias, videoaulas, listas de exercícios, avaliações e material complementar.

Procedimentos de Avaliação da Aprendizagem

A avaliação da aprendizagem será feita por meio de provas presenciais, questionários e um projeto que envolva questões ambientais.

A frequência será registrada durante as aulas presenciais no decorrer da disciplina.

Critérios de Avaliação

A avaliação será constituída de 2 provas, 1 projeto e Questionários.

A média ponderada das provas (MPP) é calculada da seguinte forma:

$$MPP = (P1 + 2*P2)/3$$

A nota final (NF) será definida por:

$$NF = 0.5*MPP + 0.4*Projeto + 0.1*Questionários$$

A aprovação está condicionada à MPP $\geq$ 5.0, Projeto $\geq$ 5.0 e Questionários $\geq$ 5.0 – caso contrário a NF será o mínimo entre valor calculado e 4.9, implicando a reprovação do aluno. A menção é baseada na nota final, conforme o regulamento da UnB. O aluno que não obtiver frequência mínima de 75% em relação ao número total de atividades estará reprovado por faltas e receberá menção SR independentemente do valor da nota final.

O aluno que faltar uma prova poderá fazer uma prova substitutiva para substituir apenas a falta em uma prova. A prova substitutiva será única e na data do cronograma. O conteúdo da prova substitutiva é toda a matéria. Para poder realizar a prova substitutiva o aluno deverá justificar a falta, seja com uma declaração do trabalho, atestado médico ou de alistamento militar. Não é permitido o aluno fazer a prova em horário diferente do da sua turma.

Nos dias das provas os(as) alunos(as) deverão portar documento oficial de identidade com foto.

O aluno que copiar ou plagiar as atividades práticas ou falsificar a identificação em qualquer atividade será automaticamente reprovado na disciplina por questões éticas.

Cronograma

Vide anexo

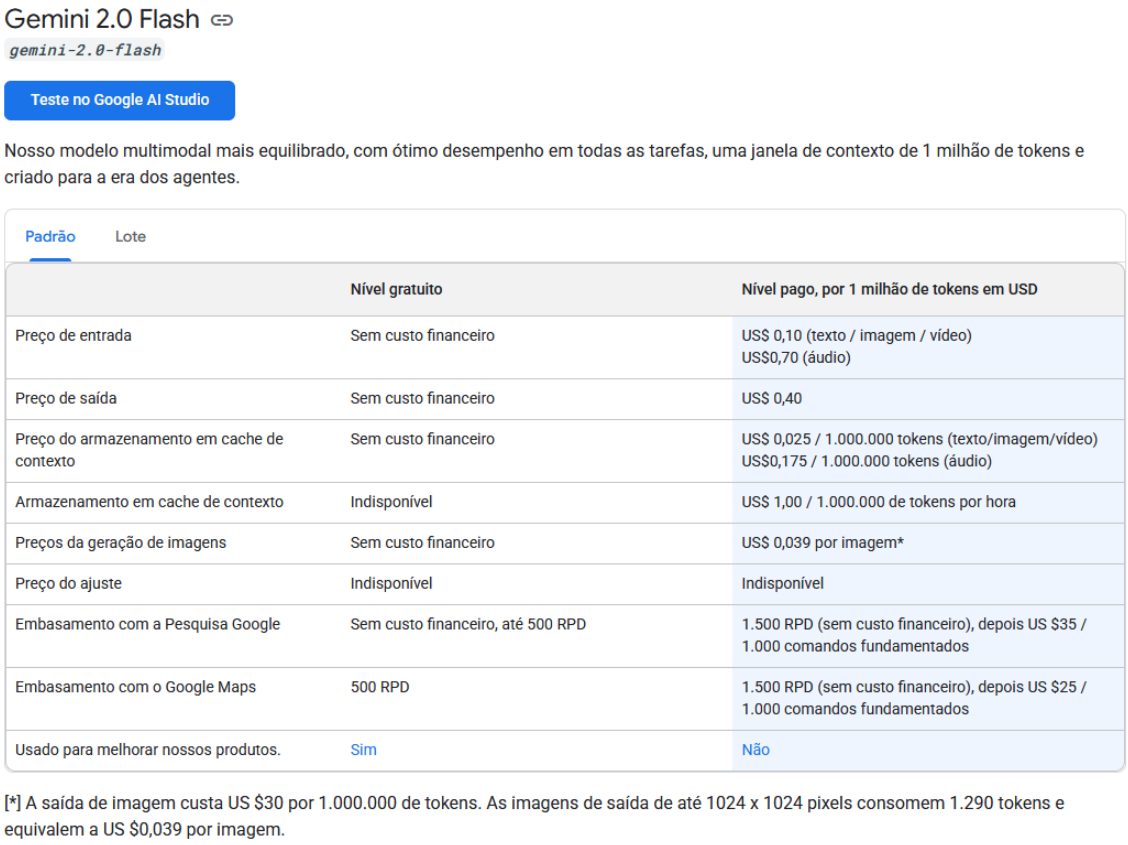
Referências

- Allen B. Downey, Pense em Python: Pense como um cientista da computação, 2016.
- Luciano Ramalho, Python Fluente: Programação clara, concisa e eficaz, 2015.
- Tim Bell, Ian H. Witten e Mike Fellows, Computação Desplugada, 2015. http://www.desplugada.ime.unicamp.br/index.html

Anexo II

Tabela de Preços do Google Gemini 2.0 Flash

Este anexo apresenta a tabela de preços oficial do modelo Gemini 2.0 Flash da Google, vigente no período de realização do experimento (outubro de 2025). Esta tabela documenta os custos operacionais utilizados nos cálculos de viabilidade econômica apresentados neste trabalho.



Gemini 2.0 Flash

gemini-2.0-flash

Teste no Google AI Studio

Nosso modelo multimodal mais equilibrado, com ótimo desempenho em todas as tarefas, uma janela de contexto de 1 milhão de tokens e criado para a era dos agentes.

Padrão | Lote

	Nível gratuito	Nível pago, por 1 milhão de tokens em USD
Preço de entrada	Sem custo financeiro	US$ 0,10 (texto / imagem / vídeo) US$0,70 (áudio)
Preço de saída	Sem custo financeiro	US$ 0,40
Preço do armazenamento em cache de contexto	Sem custo financeiro	US$ 0,025 / 1.000.000 tokens (texto/imagem/vídeo) US$0,175 / 1.000.000 tokens (áudio)
Armazenamento em cache de contexto	Indisponível	US$ 1,00 / 1.000.000 de tokens por hora
Preços da geração de imagens	Sem custo financeiro	US$ 0,039 por imagem*
Preço do ajuste	Indisponível	Indisponível
Embasamento com a Pesquisa Google	Sem custo financeiro, até 500 RPD	1.500 RPD (sem custo financeiro), depois US $35 / 1.000 comandos fundamentados
Embasamento com o Google Maps	500 RPD	1.500 RPD (sem custo financeiro), depois US $25 / 1.000 comandos fundamentados
Usado para melhorar nossos produtos.	Sim	Não

[*] A saída de imagem custa US $30 por 1.000.000 de tokens. As imagens de saída de até 1024 x 1024 pixels consomem 1.290 tokens e equivalem a US $0,039 por imagem.

Figura II.1: Tabela de preços do Google Gemini 2.0 Flash.

Fonte: Google AI Studio - Pricing Documentation (`https://ai.google.dev/gemini-api/docs/pricing?hl=pt-br#gemini-2.0-flash`)