Integração de Inteligência Artificial e Ferramentas de Autoria em um Sistema Tutor Inteligente de Código Aberto

Alexandre Bittencourt Pigozzo[1], **Davi Jannotti Coelho Pinheiro**[1]

[1] Departamento de Ciência da Computação – Universidade Federal de São João del-Rei (UFSJ)
São João del-Rei – MG – Brasil

alexandre.pigozzo@ufsj.edu.br, davijannotti@gmail.com

Abstract. *Intelligent Tutoring Systems (ITSs) aim to personalize instruction but face barriers related to content creation and the provision of dynamic feedback to users. This work presents extensions to OATutor, an open-source ITS. A tool named OATutor Content Creator was developed using the Streamlit library to automate the creation of courses, problems, and lessons structured in JSON format, as well as to integrate Large Language Models (LLMs) for the generation of contextualized feedback. Applying the solutions in a simulated scenario for creating a Computer Networks course demonstrated technical feasibility, reducing authoring complexity and enriching pedagogical support for students.*

Resumo. *Sistemas Tutores Inteligentes (STIs) visam personalizar o ensino, mas enfrentam barreiras na criação de conteúdo e na oferta de feedbacks dinâmicos para os usuários. Este trabalho apresenta extensões ao OATutor, um STI de código aberto. Foi desenvolvida a ferramenta chamada OATutor Content Creator, usando a biblioteca Streamlit, para automatizar a criação de cursos, problemas e lições estruturados em JSON, além da integração de Modelos de Linguagem de Grande Porte (LLMs) para geração de feedback contextualizado. A aplicação das soluções em um cenário simulado de criação de um curso de Redes de Computadores demonstrou a viabilidade técnica, reduzindo a complexidade de autoria e enriquecendo o suporte pedagógico ao aluno.*

1. Introdução

Sistemas Tutores Inteligentes (STIs) têm como objetivo replicar, em ambiente computacional, o papel de um tutor humano, fornecendo instruções adaptativas baseadas no desempenho do aluno Crow et al. (2018). Plataformas de código aberto, como o OATutor (*Open Adaptive Tutor*), surgem para democratizar a pesquisa nessa área, permitindo a colaboração entre instituições Pardos et al. (2023) e o desenvolvimento de soluções *open source* para a aprendizagem adaptativa.

Apesar de seu potencial, o OATutor apresenta barreiras técnicas significativas para educadores sem experiência em programação. A criação de conteúdo exige a manipulação manual de múltiplos arquivos no formato JSON que são interdependentes, demandando um alto letramento digital dos docentes. Além disso, o *feedback* fornecido aos alunos limita-se à validação binária (certo/errado) das respostas enviadas através da plataforma, carecendo de explicações formativas que auxiliem na compreensão do erro.

Com base nesse contexto, o foco principal deste trabalho é apresentar novas funcionalidades que foram implementadas no software OATutor, além de realizar um estudo inicial da viabilidade prática das soluções e suas possíveis aplicações no ensino de

Computação. Neste estudo inicial, não foi avaliado o uso do OATutor em um cenário real de sala de aula. O objetivo foi entender como o sistema funciona e desenvolver extensões visando ampliar o potencial uso e aumentar os benefícios com a utilização da plataforma. Então, o objetivo deste trabalho é apresentar duas novas funcionalidades desenvolvidas para estender o OATutor: (i) uma ferramenta de autoria visual para abstrair a complexidade da edição manual dos arquivos de configuração; e (ii) a integração de Modelos de Linguagem de Grande Porte (LLMs) para gerar *feedbacks* explicativos automáticos. As contribuições visam modernizar a plataforma, tornando-a mais acessível a educadores e pedagogicamente rica para estudantes.

O restante do texto está organizado da seguinte forma: na Seção 2, são descritos os principais conceitos teóricos para o entendimento deste trabalho. O caminho científico adotado assim como o desenvolvimento das soluções propostas são apresentados na Seção 3. Os resultados são discutidos na Seção 4 e, por fim, as conclusões e os trabalhos futuros são apresentados na Seção 5.

2. Fundamentação Teórica

A maioria dos softwares educacionais encontrados na literatura são proprietários e possuem o código fechado, impossibilitando uma análise de como funcionam e, consequentemente, inviabilizando a proposição de extensões e adaptações para contextos educacionais específicos. Entre os primeiros sistemas desenvolvidos para apoiar o ensino e a aprendizagem mediada por tecnologia, destaca-se o Moodle (*Modular Object-Oriented Dynamic Learning Environment*) que é uma plataforma de código aberto criada no início dos anos 2000 com o objetivo de oferecer um ambiente virtual de aprendizagem colaborativo e acessível. O Moodle consolidou-se como uma das ferramentas mais utilizadas mundialmente em contextos educacionais, sendo amplamente estudado e aprimorado por pesquisadores ao longo das últimas duas décadas Gamage et al. (2022).

O software OATutor é o primeiro sistema tutor inteligente totalmente de código aberto, projetado para apoiar pesquisas em aprendizagem adaptativa e sistemas de tutoria. Desenvolvido pela Universidade da Califórnia em Berkeley, o OATutor integra uma biblioteca de problemas educacionais licenciados sob *Creative Commons*, um mecanismo de *Bayesian Knowledge Tracing* (BKT) e suporte a experimentos A/B Pardos et al. (2023). Segundo Pardos et al. (2023), a principal motivação do projeto é democratizar a pesquisa em aprendizagem adaptativa, eliminando a dependência de plataformas proprietárias e oferecendo uma base reproduzível para experimentação científica. O caráter aberto da ferramenta a torna ideal para trabalhos colaborativos e para contribuições de pesquisadores e educadores de diferentes instituições.

O OATutor utiliza o modelo *Bayesian Knowledge Tracing* (BKT) para rastrear o domínio de habilidades dos alunos. O BKT modela o conhecimento do estudante como uma variável latente, atualizada a cada interação Pelánek (2017) do usuário com a plataforma. O modelo baseia-se em quatro parâmetros probabilísticos fundamentais para cada habilidade:

1. $P(L_0)$ (*Initial Mastery*): A probabilidade de o aluno já conhecer a habilidade antes de iniciar a lição.
2. $P(T)$ (*Transit*): A probabilidade de o aluno aprender a habilidade após uma oportunidade de prática.

3. $P(S)$ (*Slip*): A probabilidade de o aluno cometer um erro acidental, mesmo dominando a habilidade.
4. $P(G)$ (*Guess*): A probabilidade de o aluno acertar a resposta por acaso "chute"), sem dominar a habilidade.

Esses parâmetros permitem que o sistema decida quando uma habilidade foi dominada, adaptando a sequência de exercícios apresentada. No OATutor, esses valores são configurados estaticamente em arquivos JSON, o que adiciona uma camada de complexidade para autores de conteúdo que desejam calibrar a dificuldade de seus cursos.

2.1. Estrutura de Dados e Desafios na Autoria

A arquitetura de dados do OATutor é hierárquica e desacoplada. Um curso é composto por referências a lições que por sua vez referenciam problemas. O problema em si é armazenado isoladamente e contém metadados como, por exemplo, o corpo da questão (texto e imagens), o tipo de resposta esperada e as dicas associadas. No OATutor, os dados são organizados com base na interconexão de diferentes arquivos JSON, que definem quatro entidades principais no sistema:

- **Habilidades (*Skills*):** Representam as competências ou conhecimentos específicos que o aluno deve dominar. Cada habilidade é definida pelos quatro parâmetros probabilísticos do modelo *Bayesian Knowledge Tracing* (BKT) explicado anteriormente.
- **Lições (*Lessons*):** Agrupam um conjunto de problemas sob um mesmo tema. Cada lição possui objetivos de aprendizagem associados às habilidades que pretende desenvolver, sendo definida no arquivo `coursePlans.json`.
- **Problemas (*Problems*):** São as atividades interativas que compõem as lições. Cada problema é armazenado em sua própria pasta dentro do diretório `content-pool/` e descrito por arquivos JSON que definem o enunciado, tipo de resposta, opções, gabarito, dicas e etapas com seus respectivos caminhos de tutoria.
- **Cursos (*Courses*):** Representam a estrutura de mais alto nível. Cada curso reúne um conjunto de lições relacionadas, além de conter metadados como nome, licença e fonte OER (*Open Educational Resources*). Os cursos são definidos no arquivo `coursePlans.json` e servem como ponto de entrada para a navegação do aluno na plataforma.

Para ilustrar a complexidade de criação de novos conteúdos na plataforma, sendo este o fator que serviu de motivação para o desenvolvimento da ferramenta de autoria, o Código apresentado na Listagem 1 apresenta um trecho simplificado da estrutura JSON exigida para definir um único problema. Observa-se a necessidade de identificadores únicos e a estrutura aninhada de passos.

Listing 1. Exemplo simplificado da estrutura JSON de um problema no OATutor.

```
{
  "id": "rede_core_01",
  "title": "Nucleo da Rede",
  "body": "Qual tecnica utiliza a reserva de recursos dedicada durante
     a comunicacao?",
  "type": "MultipleChoice",
  "step": {
```

```
    "id": "rede_core_01a",
    "answerType": "string",
    "answers": ["Comutacao de Circuitos"],
    "hints": [
       {"id": "h1", "title": "Pense no sistema telefonico antigo."},
       {"id": "h2", "title": "Ha um caminho fisico dedicado."}
    ]
  }
}
```

A edição manual desses arquivos, multiplicada por dezenas de questões em um curso, torna o processo propenso a erros de sintaxe (como vírgulas ou chaves mal posicionadas) e inconsistências de referência entre arquivos, dificultando a adoção por professores sem conhecimento em programação.

2.2. LLMs na Educação

Modelos de Linguagem de Grande Porte, como o GPT-4, demonstraram capacidade de gerar texto contextualizado, oferecendo novas possibilidades para STIs. Estudos recentes indicam que LLMs podem apoiar a geração de explicações e dicas, aproximando o sistema da tutoria humana, embora apresentem desafios como alucinações e custos computacionais Filippi and Motyl (2024); Rosa et al. (2025).

Filippi e Motyl destacam que os LLMs já estão sendo utilizados em cursos de engenharia e em outras áreas do conhecimento, com foco em personalização do ensino e geração automática de conteúdo Filippi and Motyl (2024). No contexto brasileiro, pesquisas como a de Rosa *et al.* discutem os impactos do uso de LLMs no ensino de programação, apontando ganhos em acessibilidade e autonomia dos alunos, mas também riscos associados à dependência tecnológica e à confiabilidade das respostas Rosa et al. (2025). Além disso, estudos como o de Ribeiras, Dorotea e Esteves revelam percepções contrastantes entre professores e estudantes sobre o uso dessas tecnologias, com maior receptividade entre os alunos e ceticismo entre docentes Ribeiras et al. (2025).

3. Metodologia e Desenvolvimento

As soluções propostas neste trabalho foram desenvolvidas seguindo uma abordagem iterativa, com validação técnica através de um cenário simples simulando o papel do professor ou tutor de uma disciplina de graduação em Computação. A disciplina escolhida foi a de Redes de Computadores. A ferramenta de autoria desenvolvida e a integração com a inteligência artificial generativa foram testadas considerando esse cenário. Os principais objetivos do cenário simulado foram testar as funcionalidades da solução desenvolvida, analisar se ela facilitou a criação de conteúdo para o sistema OATutor e também verificar se a integração com o modelo LLM estava funcional.

3.1. Ferramenta de Autoria

Para mitigar a complexidade da edição manual de vários arquivos de configuração, desenvolveu-se a ferramenta de autoria *OATutor Content Creator* utilizando a biblioteca *Streamlit* em Python. A ferramenta oferece uma interface gráfica (*Graphical User Interface* ou GUI) que guia o educador por um fluxo linear: definição de habilidades, criação da lição e cadastro de problemas (Figura 1).

A aplicação gerencia automaticamente a estrutura de diretórios e a validação da sintaxe JSON, garantindo que o arquivo final esteja em conformidade com o esquema esperado pelo OATutor. Recursos como o *upload* de imagens, que antes exigiam que o autor salvasse o arquivo manualmente na pasta correta e escrevesse a referência no código, agora são feitos via interface gráfica.

Mais detalhes sobre a ferramenta de autoria e a sua aplicação na criação do curso de Redes serão apresentados e discutidos na Seção de Resultados (Seção 4).

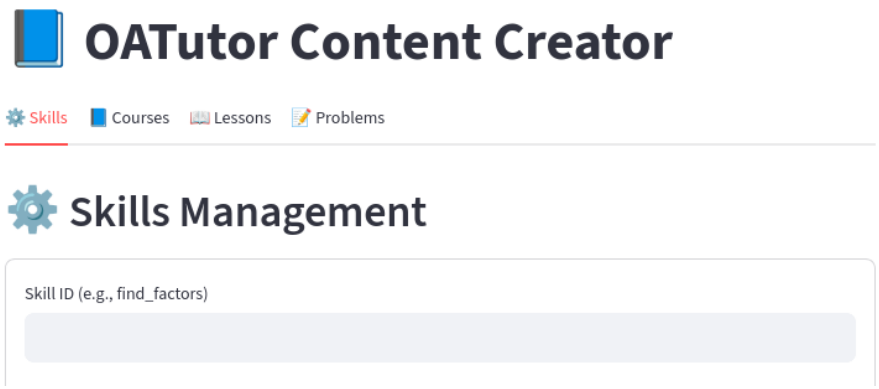

Figura 1. Parte da interface gráfica da ferramenta *OATutor Content Creator* com destaque para as abas disponíveis.

3.2. Integração com LLMs

Para realizar a integração da ferramenta OATutor com um modelo LLM, foi implementada uma arquitetura de microsserviço utilizando um *middleware* em Node.js para intermediar a comunicação entre o OATutor (*frontend*) e a API do modelo LLM. Essa camada intermediária protege as chaves de API e centraliza a engenharia de *prompt*. A arquitetura da solução desenvolvida é apresentada na Figura 2.

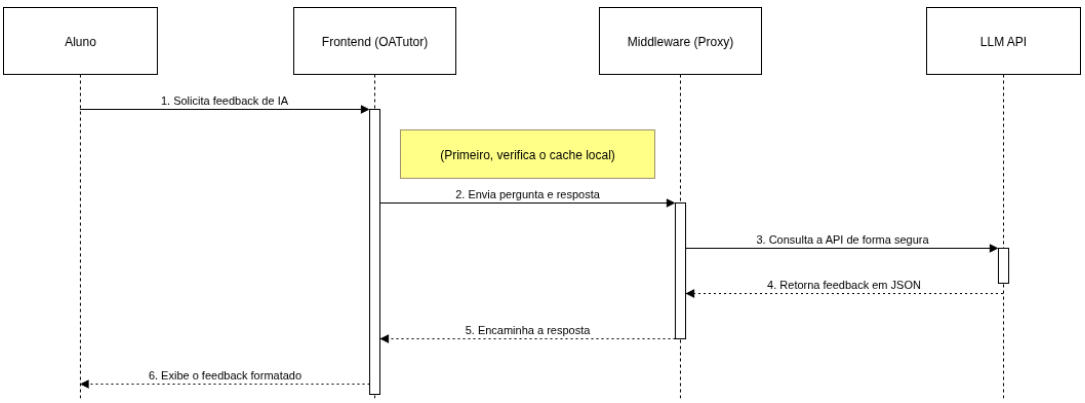

Figura 2. Arquitetura de integração entre o OATutor, o Middleware e o modelo LLM.

O fluxo de interação entre a plataforma e o modelo LLM ocorre da seguinte forma:

1. O aluno submete uma resposta incorreta.
2. O sistema captura o enunciado da questão e a resposta do aluno.

3. O *middleware* envia esses dados ao LLM com um *prompt* instrucional, solicitando uma análise pedagógica mais detalhada. O prompt utilizado é apresentado na Listagem 2.
4. O LLM retorna um JSON estruturado contendo: a explicação do erro, uma dica para auxiliar na resolução da questão e uma ou mais sugestões de conteúdos para estudar com relação ao assunto abordado na questão.

Os elementos retornados pelo modelo LLM são renderizados em um componente visual de *feedback* (Figura 3). Para otimização de custos e latência, implementou-se um sistema de cache que armazena respostas para erros comuns, evitando requisições repetidas ao modelo para a mesma dúvida. Destaca-se que o *feedback* da IA fica habilitado somente quando o aluno erra uma questão. A IA tem acesso apenas à questão e à resposta do aluno para retornar uma explicação sobre o erro. Não foi possível utilizar o histórico do aluno na consulta ao modelo LLM porque não há suporte no sistema OATutor para o armazenamento e uso desse histórico.

A API Groq (`api.groq.com/openai/v1`) foi usada para acesso ao modelo Llama `llama-3.1-8b-instant`. Para os parâmetros da LLM, foram escolhidos valores comumente usados e alguns como, por exemplo, $max\_tokens = 300$ foram ajustados para que o modelo retornasse uma resposta mais concisa e objetiva de forma a ser exibida em uma janela de tamanho moderado evitando ocupar muito espaço na tela.

Listing 2. Prompt utilizado para fornecer um feedback formativo gerado pelo modelo LLM.

```
const messages = [
    {
        role: "system",
        content:
            "You are a general tutor. Provide pedagogical feedback
               strictly in JSON format without revealing the correct
               answer.",
    },
    {
        role: "user",
        content: `
        Question: ${question_stem}.
        Student answer: ${student_answer}.
        Knowledge components: ${knowledge_components.join(", ")}.

        Return a strict JSON object in this format:
        {
          "evaluation": "correct" | "incorrect",
          "feedback": {
            "message": "An explanation message for the student.",
            "hint": "A hint to help the student reach the correct
               answer.",
            "study_tips": [
              {
                "topic": "The study topic related to the mistake.",
                "tip": "A practical study tip for the topic."
              }
            ]
          }
        }
```

```
        Rules:
        - If the answer is correct, set "feedback" to null.
        - Do NOT reveal the full solution.
        - When writing math expressions, ALWAYS wrap inline LaTeX in
            single dollar signs.
          Example: $x = 4$, evaluate $x^2$.
        - Never use double dollar signs ($$...$$).
        - Keep the JSON valid and strictly follow the schema above.
        - Do not add extra keys or text outside the JSON.
        `,
    },
];
```

Além da solução implementada para integração com um modelo LLM, o suporte a diferentes tipos de questões foi expandido neste trabalho. Originalmente restrito a questões de múltipla escolha com resposta única, o sistema foi modificado para suportar questões do tipo Múltipla Seleção, isto é, com múltiplas respostas corretas. Para implementar essa extensão, foi necessário alterar tanto a lógica de validação no *backend* quanto a renderização de componentes (*checkboxes*) no *frontend*. O código fonte completo das soluções implementadas neste trabalho pode ser obtido no Github: `github.com/davijannotti/OATutor-LLM-Gen`.

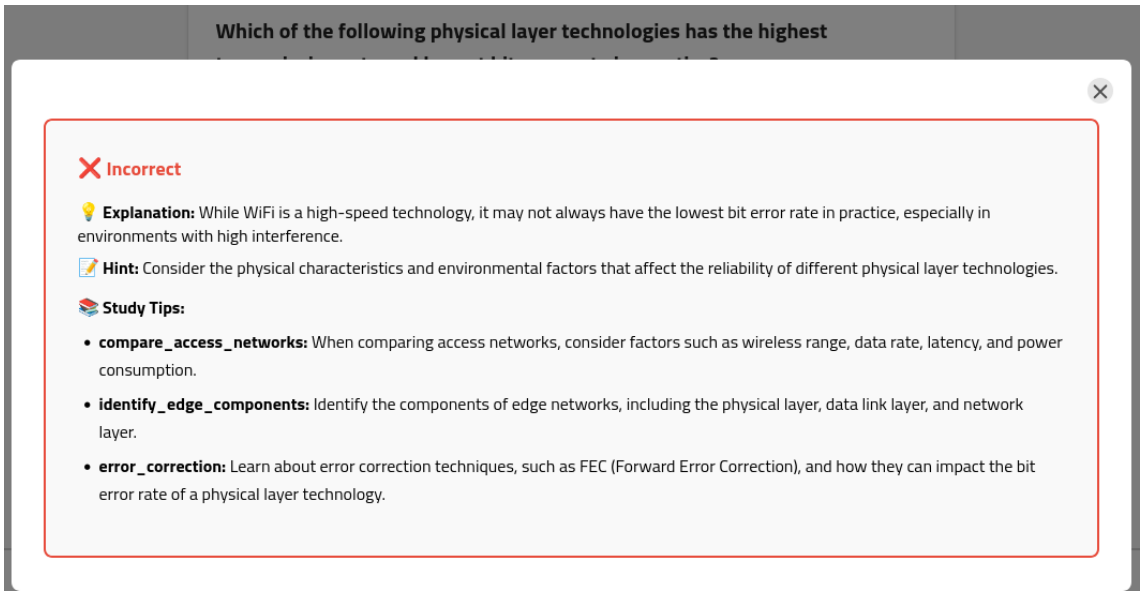

Figura 3. Exemplo de *feedback* formativo gerado pela IA exibido ao aluno após um erro.

4. Resultados e Discussão

Para testar e validar as soluções desenvolvidas, conduziu-se um estudo da viabilidade prática considerando a disciplina Redes de Computadores e utilizando os exercícios baseados no livro de Kurose e Ross que encontram-se disponíveis online em `gaia.cs.umass.edu/kurose_ross/online_lectures.htm`. Utilizando a ferramenta *OATutor Content Creator*, foi criado um módulo completo cobrindo alguns tópicos fundamentais da disciplina de Redes como, por exemplo, “Núcleo da Rede”, “Atrasos e Perdas”

e “Camadas de Protocolo”. A partir dos recursos e exercícios online disponíveis no *website* do livro, foram criadas todas as questões correspondentes aos seguintes subtemas do Capítulo Introdutório:

- **What is the Internet?**
- **The Network Edge**
- **The Network Core**
- **Performance: Delay, Loss and Throughput in Computer Networks**
- **Protocol Layers and Their Service Models**

Cada conjunto de questões foi estruturado como uma lição independente dentro do curso “Redes de Computadores”. As questões foram adaptadas e reconstruídas no formato interativo do OATutor, contemplando diferentes tipos de exercícios, como Múltipla Escolha (*Multiple Choice*), Múltipla Seleção (*Multiple Select*) e Caixa de Texto (*TextBox*) e, quando necessário, incorporando imagens de apoio e dicas estruturadas (*hints*) para orientar o estudante durante a resolução.

A Figura 4 apresenta a página inicial do curso desenvolvido no OATutor, enquanto a Figura 5 ilustra um exemplo de problema criado.

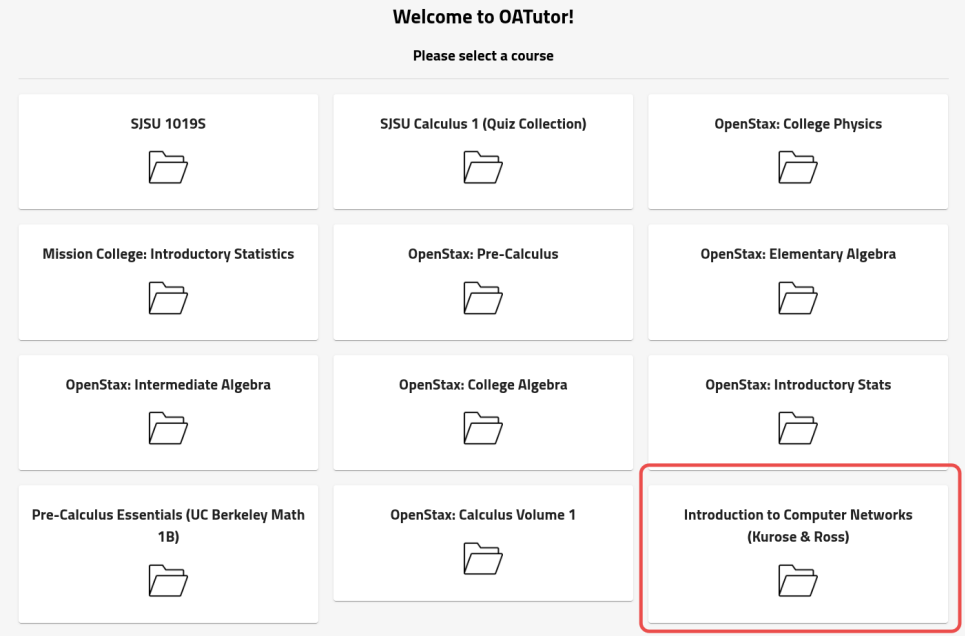

Figura 4. Página inicial do OATutor exibindo o curso de Redes de Computadores desenvolvido como parte do cenário testado.

4.1. Criação de Problemas e Anexação de Imagens

Durante o processo de autoria, um dos pontos de maior destaque foi a criação de problemas ilustrados com figuras, como topologias de rede e diagramas de pacotes. No OATutor original, esse procedimento era totalmente manual: o autor precisava inserir as imagens na pasta do problema (`content-pool/`) e, em seguida, referenciá-las no corpo do enunciado utilizando a sintaxe específica `##imagem.jpg##`. Além disso, as demais configurações, como definição das respostas corretas, tipos de questão e estrutura

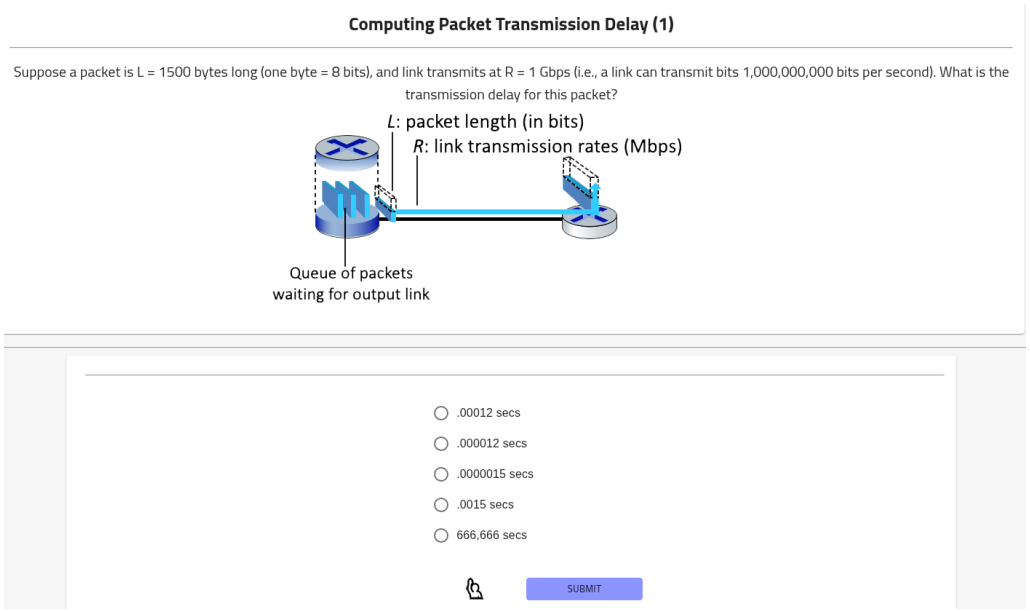

Figura 5. Exemplo de problema sobre o cálculo no atraso de transmissão de pacotes para o curso de Redes.

de dicas, exigiam a edição direta dos arquivos JSON, o que tornava o processo técnico, demorado e propenso a erros.

A ferramenta desenvolvida, denominada **OATutor Content Creator**, automatiza e integra todas essas etapas em uma interface gráfica simples e intuitiva, permitindo criar e editar problemas sem qualquer necessidade de manipulação manual dos arquivos de configuração. Por meio de campos interativos na interface gráfica da ferramenta, o autor pode:

- Associar o problema a cursos e lições existentes;
- Definir o título, enunciado e tipo de problema;
- Anexar imagens ilustrativas por meio de um botão de *upload*, sem necessidade de editar o JSON manualmente;
- Selecionar respostas corretas de forma visual;
- Adicionar e configurar dicas.

A Figura 6 mostra a etapa inicial da criação, onde o autor associa o problema a um curso e lição específicos, e envia uma imagem de apoio. A imagem é automaticamente salva na pasta correta e referenciada no corpo do problema. Na Figura 7, observa-se o preenchimento dos campos principais. Essa abordagem unificada simplifica todas as etapas de autoria, reduz a possibilidade de inconsistências nos arquivos, e permite que docentes e pesquisadores foquem no aspecto pedagógico do conteúdo, e não em detalhes técnicos da estrutura interna do OATutor.

4.2. Eficiência na Autoria

A utilização do *OATutor Content Creator* eliminou a necessidade de interação direta com o código fonte dos dados. A Tabela 1 apresenta uma estimativa de esforço comparando os métodos manual e com o uso da ferramenta.

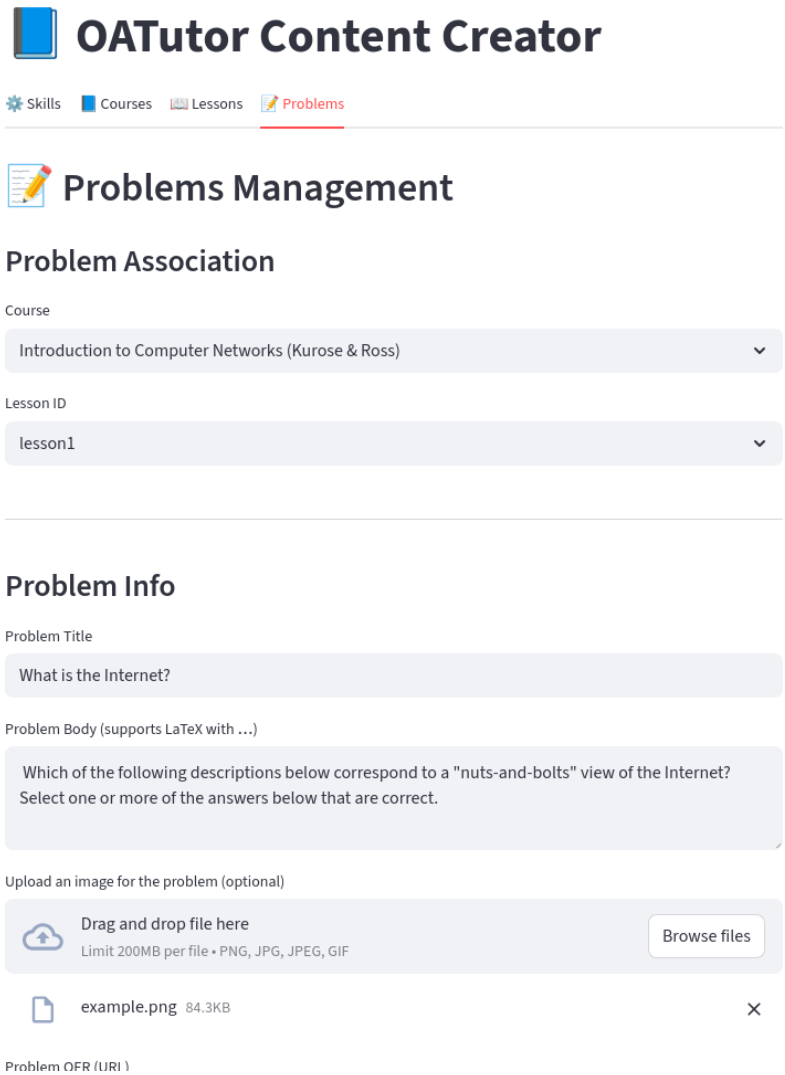

Figura 6. Interface da ferramenta de autoria para cadastro de problemas, abstraindo a edição do JSON.

A ferramenta garantiu a integridade dos dados, permitindo a geração rápida de lições complexas contendo imagens e equações *LaTeX*. A automação do processo permitiu que o foco do trabalho se mantivesse no design instrucional das questões, e não na depuração de erros de sintaxe de arquivos de configuração.

4.3. Qualidade do Feedback e Limitações

A integração com IA proporcionou *feedbacks* detalhados que não seriam possíveis com o sistema original de gabarito estático. No entanto, observou-se o fenômeno de alucinação em casos específicos. Em um teste envolvendo cabeçalhos de pacotes (onde variáveis eram identificadas como `h1`, `h2`), o modelo confundiu os termos com *tags* HTML, gerando explicações incorretas sobre desenvolvimento web. Isso reforça que, embora a ferramenta amplie significativamente a capacidade pedagógica, a supervisão humana na curadoria dos *prompts* e do conteúdo permanece essencial para garantir a precisão técnica.

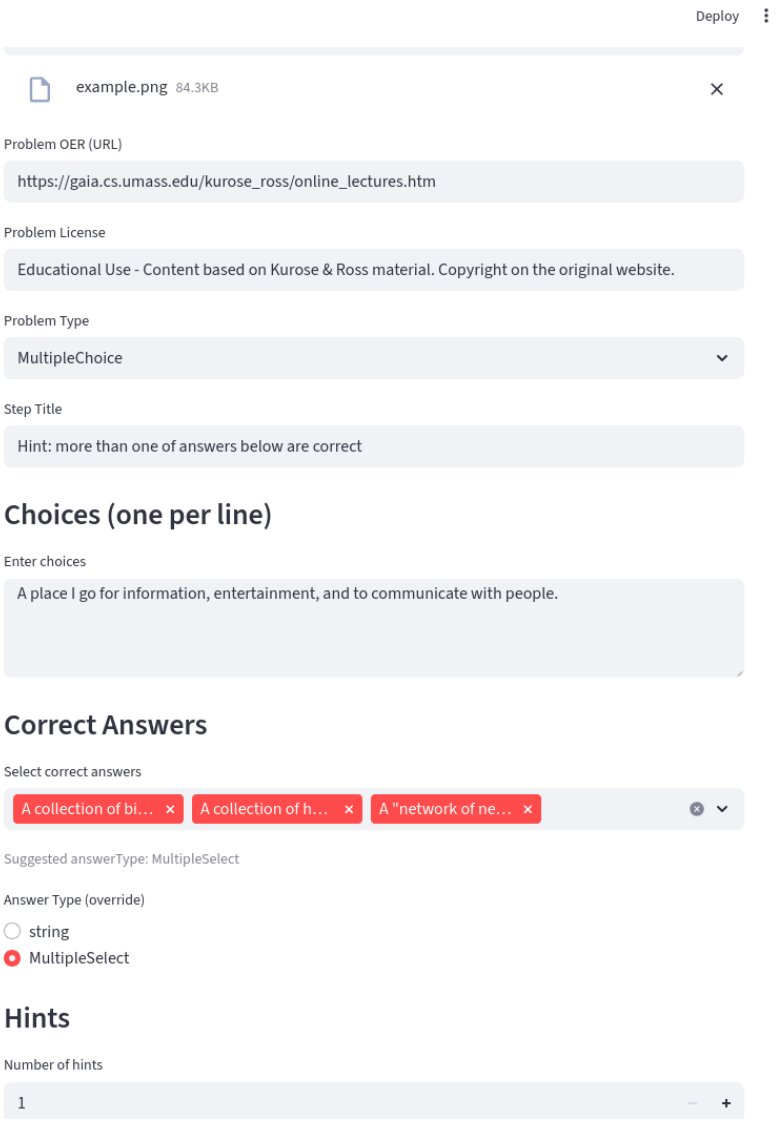

Figura 7. Etapa de definição do enunciado, tipo de problema e respostas corretas.

5. Conclusões e Trabalhos Futuros

Este trabalho apresentou o desenvolvimento de extensões para o *OATutor*, visando mitigar barreiras técnicas na criação de conteúdo e enriquecer o *feedback* fornecido aos estudantes. As soluções propostas, a ferramenta *Content Creator* e a integração com LLMs, foram implementadas e tecnicamente verificadas através de uma aplicação prática na criação de módulos para um curso de Redes de Computadores.

Os resultados obtidos demonstram a viabilidade técnica da automação do processo de autoria. A ferramenta desenvolvida abstraiu a complexidade dos muitos arquivos de configuração, permitindo a estruturação de cursos e a inserção de recursos multimídia através de uma interface gráfica. O cenário simulado indicou que a solução é funcional e capaz de gerar conteúdos compatíveis com o ecossistema do OATutor, sugerindo uma potencial redução no tempo de desenvolvimento de material didático em comparação ao método manual.

Tabela 1. Comparativo estimado de esforço para criação do módulo de Redes.

Ação	Método Manual	Content Creator
Manipulação de arquivos	$\approx$ 15 arquivos dispersos	1 Interface Unificada
Tempo médio por questão	10-15 min	2-3 min
Gestão de Imagens	Upload manual + Referência	Upload direto na UI
Risco de Erro de Sintaxe	Alto (ex: vírgula em JSON)	Nulo (geração automática)

A integração com Modelos de Linguagem de Grande Porte introduziu a capacidade de *feedbacks* explicativos dinâmicos. Embora promissora, essa funcionalidade evidenciou desafios inerentes ao uso de IA Generativa, como a possibilidade de alucinações, reforçando a necessidade de supervisão humana na curadoria dos conteúdos educacionais.

Em suma, o trabalho entrega um conjunto de ferramentas funcionais que preparam o terreno para futuras investigações pedagógicas, contribuindo para a evolução do OATutor como uma plataforma mais moderna e acessível.

Como limitação, destaca-se a ausência de validação quantitativa e qualitativa com usuários finais (professores e alunos) em ambiente real de sala de aula. Trabalhos futuros devem focar em experimentos controlados para mensurar o impacto na aprendizagem e a usabilidade da ferramenta de autoria, além de explorar o uso de LLMs locais para redução de custos de operação.

Agradecimentos

Os autores gostariam de agradecer todo o apoio da Universidade Federal de São João del-Rei (UFSJ) e do Programa de Pós-Graduação em Ciência da Computação da UFSJ.

Uso de Inteligência Artificial

Os autores declaram que foi feito o uso de tecnologias de inteligência artificial generativa para guiar e auxiliar a escrita do texto nas Seções de Fundamentação Teórica e em partes da Metodologia. Declaramos que todo o texto foi revisado antes da inclusão na versão final do trabalho.

Referências

Crow, T., Luxton-Reilly, A., and Wuensche, B. (2018). Intelligent tutoring systems for programming education: a systematic review. In *Proceedings of the 20th Australasian Computing Education Conference*, ACE '18, page 53–62, New York, NY, USA. Association for Computing Machinery.

Filippi, S. and Motyl, B. (2024). Large language models (llms) in engineering education: A systematic review and suggestions for practical adoption. *Information*, 15(6).

Gamage, S. H. P. W., Ayres, J. R., and Behrend, M. B. (2022). A systematic review on trends in using moodle for teaching and learning. *International Journal of STEM Education*, 9(1):9.

Pardos, Z. A., Tang, M., Anastasopoulos, I., Sheel, S. K., and Zhang, E. (2023). Oatutor: An open-source adaptive tutoring system and curated content library for learning sciences research. In *Proceedings of the 2023 CHI Conference on Human Factors in*

Computing Systems, CHI '23, New York, NY, USA. Association for Computing Machinery.

Pelánek, R. (2017). Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modeling techniques. *User Modeling and User-Adapted Interaction*, 27(3):313–350.

Ribeiras, C., Dorotea, N., and Esteves, Y. (2025). Uso e perceção de llm pelos estudantes e professores do ensino superior - revisão sistemática da literatura. *Revista Lusófona de Educação*, 65(65):85–108.

Rosa, Y., Garcia, P., Constantino, K., and Figueiredo, E. (2025). Reflexões sobre o uso de llms no ensino de programação. In *Anais do V Simpósio Brasileiro de Educação em Computação*, pages 741–749, Porto Alegre, RS, Brasil. SBC.