UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

TIAGO SARDI MENDONÇA
Orientador: Prof. Dr. Reinaldo Silva Fortes

PROJETO E DESENVOLVIMENTO DE UMA PLATAFORMA DE GESTÃO DE QUESTÕES DINÂMICAS PARA O ENSINO DE PROGRAMAÇÃO DE COMPUTADORES

Ouro Preto, MG
2023

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

TIAGO SARDI MENDONÇA

PROJETO E DESENVOLVIMENTO DE UMA PLATAFORMA DE GESTÃO DE QUESTÕES DINÂMICAS PARA O ENSINO DE PROGRAMAÇÃO DE COMPUTADORES

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Reinaldo Silva Fortes

Ouro Preto, MG
2023

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

M539p Mendonca, Tiago Sardi.
Projeto e desenvolvimento de uma plataforma de gestão de questões dinâmicas para o ensino de programação de computadores. [manuscrito] / Tiago Sardi Mendonca. - 2023.
60 f.

Orientador: Prof. Dr. Reinaldo Silva FORTES.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Graduação em Ciência da Computação .

1. Programação. 2. Plataforma. 3. Computação. I. FORTES, Reinaldo Silva. II. Universidade Federal de Ouro Preto. III. Título.

CDU 004

Bibliotecário(a) Responsável: Luciana De Oliveira - SIAPE: 1.937.800

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

Tiago Sardi Mendonça

Projeto e desenvolvimento de uma plataforma de gestão de questões dinâmicas para o ensino de programação de computadores

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 25 de Agosto de 2023.

Membros da banca

Reinaldo Silva Fortes (Orientador) - Doutor - Universidade Federal de Ouro Preto
Pedro Henrique Lopes Silva (Examinador) - Doutor - Universidade Federal de Ouro Preto
Guilherme Augusto Lopes Silva (Examinador) - Mestre - Universidade Federal de Ouro Preto

Reinaldo Silva Fortes, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 25/08/2023.

Documento assinado eletronicamente por **Reinaldo Silva Fortes**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 28/08/2023, às 16:15, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0577038** e o código CRC **6970CC36**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591692 - www.ufop.br

Ao incentivo dos meus pais pela educação dos filhos.

Agradecimentos

Levo neste espaço a oportunidade de agradecer ao orientador Prof. Dr. Reinaldo Fortes pela sabedoria em me direcionar para este projeto e me ensinar a focar nos meus objetivos maiores. Agradeço também aos meus amigos, principalmente aos que não me deixaram desistir do curso quando já não tinha mais renda e estímulo para continuar. Levo a gratidão eterna ao Ensino Público por buscar incessantemente a verdade e fazer de seus discentes caçadores dela. Agradeço do fundo do meu coração à minha amada Nicolly, que tem sido meu grande apoio e fonte de inspiração durante todo o processo de elaboração desta monografia. Sua presença constante em minha vida é um dos maiores presentes que já recebi. Sua dedicação e amor incondicional me deram forças para superar as dificuldades que encontrei no caminho e continuar avançando em direção aos meus objetivos acadêmicos. Casa comigo?

"Educação não transforma o mundo. Educação muda as pessoas. Pessoas transformam o mundo."(FREIRE, 1987)

Resumo

A disciplina de introdução à programação de computadores é essencial em inúmeros cursos de engenharia e outras áreas de exatas. Por isso, ela está inclusa em um conjunto significativo de cursos. O professor encarregado por essa matéria necessita de um tempo significativo para elaborar e corrigir as questões práticas por conta da alta demanda. Além disso, as estatísticas comprovam que o desempenho dos alunos nessa disciplina é baixo.

Para solucionar parte desses problemas, um sistema online foi criado para que o aluno pudesse interagir com tarefas de questões e com o recurso de um corretor automático para que pudesse corrigir as questões rapidamente. No entanto, a plataforma carecia de um módulo que gerasse questões distintas para cada aluno, tornando as tarefas mais personalizadas. Diante dessa necessidade, este trabalho surgiu para a implementação de um módulo capaz de gerar uma infinidade de questões únicas, com o objetivo de aprimorar a plataforma e auxiliar os professores na criação de tarefas dinâmicas.

Palavras-chave: Introdução à programação. Gerador de questões. Corretor automático de código. Exercícios de programação. Questão baseada em modelo.

Abstract

The subject of introduction to computer programming is essential in many engineering courses and other exact areas. Therefore, it is included in a significant set of courses. The teacher in charge of this subject needs a significant amount of time to prepare and correct the practical questions due to the high demand. In addition, statistics prove that student performance in this subject is low.

To solve part of these problems, an online system was created so that the student could interact with question tasks and with the feature of an automatic corrector so that he could quickly correct the questions. However, the platform lacked a module that generated different questions for each student, making the tasks more personalized. Faced with this need, this work emerged for the implementation of a module capable of generating a multitude of unique questions, with the objective of improving the platform and helping teachers in the creation of dynamic tasks.

Keywords: Introduction to programming. Question Generator. Automatic code corrector. Programming exercises. Template based exercises.

Lista de Ilustrações

Figura 2.1 – Classificação dos problemas na Beecrowd.
Fonte: (BEECROWD, 2023). 6
Figura 2.2 – Histórico de submissões de um problema.
Fonte: (BEECROWD, 2023). 7
Figura 2.3 – Tela de publicação do problema.
Fonte: (BEECROWD, 2023). 8
Figura 2.4 – Adicionando um problema.
Fonte: (SPOJ, 2023) 9
Figura 2.5 – Caixa de texto do problema.
Fonte: (SPOJ, 2023) 9
Figura 2.6 – Definição do Juiz e das linguagens.
Fonte: (SPOJ, 2023) 10
Figura 2.7 – Interação do Juiz com uma pergunta.
Fonte: (CAMPOS, 2004) 11
Figura 2.8 – Representação das palavras que serão substituídas
Fonte: (GENCI, 2013) 12
Figura 2.9 – Entrada da questão por XML
Fonte: (GENCI, 2013) 13
Figura 3.1 – Questão criada de forma manual.
Fonte: Autoral (2023) 18
Figura 3.2 – TinyMCE: Editor de texto em formato HTML 19
Figura 3.3 – Enunciado com variáveis definidas 22
Figura 3.4 – Tabela de variáveis relacionadas ao enunciado 23
Figura 3.5 – Respostas do gerador de questão dinâmica. 24
Figura 3.6 – Metadados da questão-modelo. 26
Figura 3.7 – Metadado “Área de conhecimento”. 27
Figura 3.8 – Metadado “Tópicos Relacionados”. 27
Figura 3.9 – Metadado “Tags”. 28
Figura 3.10–Graus de dificuldade da questão. 29
Figura 3.11–Casos de Uso do Ator Educador
Fonte: Autoral (2023) 30
Figura 3.12–Modelo de entidade e relacionamento
Fonte: Autoral (2023) 31
Figura 3.13–Diagrama de Classes
Fonte: Autoral (2023) 32

Figura 3.14–Tela para Criar uma Questão
Fonte: Autoral (2023) 34
Figura 3.15–Tela para Visualizar e Buscar as Questões-Modelo
Fonte: Autoral (2023) 35
Figura 3.16–Tela para Atualizar uma Questão-Modelo
Fonte: Autoral (2023) 36

Lista de Tabelas

Tabela 2.1 – Gerador de valores em CSV 13
Tabela 3.1 – Matriz de opções de seleção 21

Lista de Abreviaturas e Siglas

DECOM	Departamento de Computação
UFOP	Universidade Federal de Ouro Preto
NLP	*Natural Language Processing*
ICPC	*International Collegiate Programming Contest*
XML	*eXtensible Markup Language*
HTML	*HyperText Markup Language*
GCC	*GNU Compiler Collection*
SPOJ	*Sphere Online Judge*
SGBD	Sistema Gerenciador de Banco de Dados
VCS	*Version Control System*
WYSIQYG	*What You See Is What You Get*
MER	Modelo de Entidade e Relacionamento
CRUD	*Create, Read, Update, Delete*

Sumário

1 Introdução 1
1.1 Justificativa 3
1.2 Objetivos 3
1.2.1 Objetivos Específicos 4
1.3 Organização do Trabalho 4
2 Revisão Bibliográfica 5
2.1 Trabalhos Relacionados 5
2.1.1 Juízes Online 5
2.1.1.1 Beecrowd 5
2.1.1.2 SPOJ 8
2.1.2 BOCA *Online Contest Administrator* 10
2.1.3 Geração de Questões Variadas 11
2.2 Fundamentação Teórica 14
2.2.1 Contextualização de Questões 14
2.2.2 Grau de Dificuldade de uma Questão 14
2.2.3 Tecnologias utilizadas no projeto 15
2.2.3.1 HTML5 e CSS3 15
2.2.3.2 PHP 16
2.2.3.3 Javascript 16
2.2.3.4 MySQL 16
3 Desenvolvimento 17
3.1 Metodologia 17
3.1.1 Editor de Texto para o Enunciado da Questão 18
3.1.2 Método de Identificação das Variáveis 20
3.1.2.1 Variável do tipo inteiro 20
3.1.2.2 Variável do tipo real 20
3.1.2.3 Variável do tipo seleção 21
3.1.2.4 Variável do tipo expressão 21
3.1.2.5 Tabela de definição das variáveis 23
3.1.3 Escolha pseudoaleatória dos valores das variáveis 24
3.1.4 Método de atribuição à variável 25
3.1.5 Metadados da questão 26
3.1.5.1 Área de Conhecimento 26
3.1.5.2 Tópicos Relacionados 27
3.1.5.3 Tags Relacionadas 28
3.1.5.4 Determinação do Grau de Dificuldade 28

3.2 Modelagem Conceitual 29
3.3 Banco de Dados 31
3.4 Operações Básicas em Questões 33
3.4.0.1 Criação de uma Questão-Modelo 33
3.4.0.2 Leitura e Busca por Questões-Modelo 35
3.4.0.3 Atualização de uma Questão-Modelo 36
3.4.0.4 Desabilitar uma Questão-Modelo 37
4 Discussões 38
5 Considerações Finais 40
5.1 Conclusão 40
5.2 Trabalhos Futuros 41

Referências 42

1 Introdução

Mesmo com diversas técnicas automatizadas de ensino e avaliação, o professor utiliza muito do seu tempo elaborando e corrigindo questões manualmente, ao mesmo tempo em que se preocupa em garantir a segurança e legitimidade da aplicação das questões para os alunos (JÚNIOR, 2019). Essa demanda sobrecarrega o tempo do educador que, costumeiramente, exerce suas atividades acadêmicas fora de seu horário de trabalho. Gomes e Mendes (2014) apontam que a consequência costuma ser uma menor quantidade de exercícios elaborados por professores quando se pretende efetuar a correção dessas questões manualmente, a fim de dar um feedback adequado. Devido ao excesso de turmas e alunos para o professor e a impossibilidade de um feedback ágil para seus alunos, o acompanhamento individual para o desenvolvimento do aluno fica comprometido (BLOOM, 1968). Para Vier, Gluz e Jaques (2015), essa dificuldade em fornecer um feedback rápido ao aluno pode afetar o desempenho da aprendizagem.

Aplicar os desafios mencionados acima ao contexto de disciplinas de programação de computadores torna a tarefa de envolver os alunos na educação em programação uma empreitada complexa, sobretudo no âmbito das disciplinas introdutórias. A alta taxa de reprovação e evasão de estudantes em cursos de programação é evidente, como apontado por Piteira e Haddad (2011). Conforme as pesquisas de Kollmansberger (2010), a adoção de uma abordagem pedagógica contextualizada à realidade do estudante pode ser motivador por facilitar a compreensão de texto. Isso se torna especialmente relevante, visto que um dos desafios enfrentados em disciplinas introdutórias de programação reside na dificuldade dos alunos em interpretar textos. Diversos outros fatores também contribuem para a baixa taxa de sucesso nas disciplinas iniciais de programação. Conforme destacado por Robins, Rountree e Rountree (2003), a falta de familiaridade do aluno com o universo da programação é um obstáculo adicional para a motivação na aprendizagem desse campo. Isso ocorre devido à dificuldade em compreender a finalidade dos programas e sua relação com a máquina, o que por sua vez influencia a falta de prática e aplicação desses conceitos.

Um mecanismo para atender essa demanda de forma eficaz são os corretores automáticos de questões, capazes de analisar as respostas dos alunos e fornecer correções instantâneas (BRITO, 2019). Nesse contexto, desponta o *opCoders Judge*, um corretor automático de questões de programação desenvolvido por Brito (2019). Seu funcionamento considera tanto a análise dinâmica, que avalia a saída gerada pela resposta do aluno através dos critérios predefinidos, quanto a análise estática, que estima a complexidade da solução proposta. Através dela, os alunos recebem correções precisas e instantâneas, permitindo-lhes aprimorar suas habilidades de programação de maneira mais efetiva. Além desse benefício, o feedback fornecido pelo corretor automático pode ser uma ferramenta valiosa para o educador, que pode adaptar sua abordagem de ensino de acordo e identificar possíveis obstáculos de aprendizado.

Em sua fase inicial, o *opCoders Judge* era disponibilizado como uma versão *offline*, na qual o docente executava o corretor em sua máquina local. Porém, um projeto já estava sendo desenvolvido para integrar esse corretor automático a um sistema de gerenciamento online, voltado para alunos e questões de programação. Como detalhado por Patrocínio (2023), ajustes e novas implementações foram introduzidos para transformar o *opCoders Judge* em um ambiente interativo. Nesse novo formato, os alunos tornaram-se capazes de submeter suas soluções para as questões e receber as avaliações fornecidas pelo corretor automático. Patrocínio (2023) também tratou de incorporar funcionalidades de administração ao professor para que possam gerenciar as questões, as turmas e também visualizar o desempenho dos estudantes por meio de seus resultados.

No trabalho de Patrocínio (2023), as questões do *opCoders Judge* são redigidas diretamente na plataforma e inseridas no banco de dados. As informações armazenadas abrangem o título da questão, o enunciado e os critérios de correção associados. Na seção de gestão de tarefas, os professores conseguem visualizar as tarefas já existentes no banco e selecioná-las para serem atribuídas a uma ou mais turmas. Uma "tarefa" é composta por um conjunto de questões distribuídas às turmas como uma lista de exercícios. É definido tanto a data de disponibilidade da tarefa para os alunos quanto o prazo final para a entrega de suas soluções. Há também a possibilidade de criar novas tarefas de maneira direta. Na página de criação de tarefas, o professor especifica o nome da tarefa, elabora o enunciado correspondente e seleciona as questões que serão parte integrante dessa tarefa.

Com a plataforma *opCoders Judge* agora disponível online e em produção, uma nova funcionalidade foi concebida para assegurar a autenticidade e originalidade das questões. Ao adicionar uma questão a uma tarefa e direcioná-la para várias turmas, surge a preocupação de que as respostas dos alunos possam ser copiadas, levando ao plágio. Além disso, questões idênticas podem prejudicar o que se espera do nível de dificuldade de uma questão. Para abordar essa questão, surgiu a ideia de incorporar o conceito de "questões dinâmicas". Isso envolve infundir elementos dinâmicos nos enunciados das questões. Esses elementos dinâmicos consistem em variáveis definidas no próprio texto do enunciado. Quando uma questão é gerada, essas variáveis são preenchidas com valores pseudoaleatórios gerados a partir de regras. Como resultado, cada instância da questão apresenta elementos distintos, garantindo que enunciados semelhantes tenham palavras ou valores únicos. Ou seja, a partir destas escolhas de valores, obtém-se um conjunto de diferentes versões de uma mesma questão. Essa abordagem permite a criação de tarefas nas quais cada aluno recebe uma coleção de questões, sendo que cada aluno possui elementos distintos em questões semelhantes. Assim, é possível ter uma ampla variedade de questões sem a necessidade de repetição, atendendo aos critérios de originalidade e enriquecendo a dinâmica do desafio da questão.

Além de planejar e implementar a funcionalidade mencionada anteriormente, foram consideradas outras características para serem incorporadas às questões, visando a classificação

por nível de dificuldade e o contexto que envolve o enunciado. Dessa forma, as questões podem ser direcionadas às áreas de interesse dos alunos ou aos tópicos estudados no momento. Desta forma, pretende-se fomentar uma abordagem pedagógica mais contextualizada, alinhada com a realidade dos estudantes. A ideia é que essa iniciativa promova a prática de aprendizado mais engajadora, permitindo que as questões se conectem de maneira direta com os interesses e vivências dos alunos. Esse enfoque, por sua vez, é conhecido por estimular um maior comprometimento e interesse dos alunos, conforme ressaltado por Corney, Teague e Thomas (2010).

A solução proposta por este trabalho pode trazer muitos benefícios para a prática educativa, pois permite que o educador dedique mais tempo para o acompanhamento individual dos alunos e outras atividades pedagógicas. Além disso, essa nova maneira de criar questões pode contribuir para a melhoria da qualidade do ensino, por proporcionar uma variedade maior de questões e possibilitar uma avaliação mais abrangente e diversificada. Dessa forma, espera-se que o educador otimize o tempo despendido na elaboração e correção de questões, além de contar com um apoio significativo na gestão das diversas turmas de disciplinas introdutórias de programação.

1.1 Justificativa

A criação de questões para avaliação, especialmente no contexto de exercícios de programação de computadores, pode ser bastante custosa em termos de tempo para o educador. É necessário pensar em um texto claro e preciso para a questão, além de estabelecer desafios que estejam no nível de conhecimento esperado para aquela avaliação. Muitas vezes, por falta de tempo ou recursos, as mesmas questões são utilizadas para avaliar todos os alunos de uma disciplina, o que pode levar a um processo de avaliação não tão efetivo quanto poderia ser.

Nesse contexto, o desenvolvimento do corretor automático de questões de programação, elaborado por Brito (2019), emerge como uma solução significativa para agilizar o retorno de feedback sobre as resoluções das questões por parte dos alunos. A subsequente contribuição de Patrocínio (2023) consistiu na criação de um ambiente interativo para os alunos interagirem com problemas de programação. No entanto, surgiu uma demanda por questões distintas para garantir que as soluções apresentadas pelos alunos sejam distintas, a fim de preservar a integridade da metodologia de avaliação.

1.2 Objetivos

Este trabalho procura contornar a complexidade da confecção manual de uma questão, agregando ao *opCoders Judge* um ambiente de formulação de questões dinâmicas de programação.

1.2.1 Objetivos Específicos

Com o intuito de atingir o objetivo principal deste trabalho, foram estabelecidos os seguintes objetivos específicos:

- Criar ambiente de criação de questões dinâmicas e simulação.
- Estender as características de questões para conter áreas de conhecimento, assuntos de interesse, tópicos relacionados à disciplina e graus de dificuldade;

1.3 Organização do Trabalho

O presente trabalho está estruturado como descrito a seguir. Capítulo 2, de Revisão Bibliográfica (Fundamentação Teórica e Trabalhos Relacionados), Capítulo 3, de Desenvolvimento (Metodologia e Modelagem), Capítulo 4, de Discussões e Capítulo 5, de Considerações Finais (Conclusão e Trabalhos Futuros).

2 Revisão Bibliográfica

A literatura relevante para o tema deste trabalho foi obtida por meio de pesquisas em bancos de dados de artigos acadêmicos disponíveis na internet. Além disso, foram obtidas as devidas autorizações para acessar todas as funcionalidades de plataformas semelhantes, como o Beecrowd, que é detalhado em uma das subseções a seguir.

Na Seção 2.1, são apresentadas alternativas de implementação para algumas funcionalidades compatíveis com a proposta deste trabalho. A Seção 2.2 apresenta algumas pesquisas de fundamentação a partir de trabalhos acadêmicos

2.1 Trabalhos Relacionados

Esta Seção apresenta uma análise detalhada de diversas ferramentas que possuem como propósito criar um ambiente de questões de programação e correções automáticas de código. A Subseção 2.1.1 traz uma análise de Juízes Online, que são plataformas amplamente utilizadas em universidades e competições de programação para avaliação do desempenho do participante. Já na Subseção 2.1.2, é mencionado o BOCA, uma ferramenta de apoio à maratonas de programação para avaliação em salas de aula presenciais. Na Subseção 2.1.3, apresenta-se um método de criação de um modelo gerador de questão. Todas essas ferramentas são importantes para o contexto desta pesquisa, pois apresentam diferentes abordagens para criação e correção de questões de programação.

2.1.1 Juízes Online

Os Juízes Online, ou *Online Judges*, oferecem um ambiente para elaboração de questões de programação seguindo os padrões do corretor automático integrado. Com isso, as estruturas de *Online Judges* são capazes de fornecer feedback imediato aos alunos sobre suas respostas às questões, além de utilizar métodos de *ranking* para avaliar o desempenho dos participantes. Neste contexto, destacamos duas plataformas populares: Beecrowd e SPOJ.

2.1.1.1 Beecrowd

A Beecrowd, conhecida anteriormente como URI Online Judge, é uma plataforma brasileira de programação competitiva que fornece um recurso de pontuações dos competidores nos desafios de programação com base no sistema de *ranking* da *International Collegiate Programming Contest* (ICPC), de acordo com Bez, Tonin e Rodegheri (2014). A Beecrowd possui um módulo em que professores possam elaborar listas de exercícios formadas por questões da própria Beecrowd ou customizadas. Os alunos podem ser convidados a resolver os problemas da lista de

exercícios com os mesmos recursos que a Beecrowd oferece em suas competições. Destacam-se a correção da questão no instante em que sua resposta é submetida e a compatibilidade com diversas linguagens de programação, tais como Java, C, Dart e Python.

Conforme mencionado anteriormente, a Beecrowd é uma plataforma que integra o ICPC, um concurso de programação com um modelo de avaliação único. Amplamente utilizado em universidades como método de ensino, o ICPC tem como objetivo incentivar os alunos a praticarem programação e aprimorarem suas habilidades (WIRAWAN et al., 2017).

Na Beecrowd, as questões são chamadas de "problemas" e possuem níveis de dificuldade. Além disso, são classificadas em nove categorias, como ilustrado na Figura 2.1.

Figura 2.1 – Classificação dos problemas na Beecrowd.
Fonte: (BEECROWD, 2023).

A Figura 2.2 apresenta o histórico de submissões na Beecrowd, que permite aos educadores analisarem as resoluções de um estudante para cada problema. Com essa funcionalidade, o educador pode ver o código-fonte enviado pelo aluno, o resultado obtido, o tempo de execução e a quantidade de memória utilizada pelo programa. Essas informações são úteis para identificar onde o estudante pode estar tendo dificuldades e oferecer orientações específicas para ajudá-lo a melhorar.

Além disso, o educador também pode acompanhar o progresso geral do estudante na plataforma, verificando quantos problemas foram resolvidos e em quanto tempo. Esses dados são importantes para avaliar o desempenho do aluno e fornecer feedback individualizado para ajudá-lo a desenvolver habilidades específicas de programação.

Em resumo, a funcionalidade de histórico de submissões da Beecrowd é uma ferramenta

valiosa para os educadores acompanharem o desempenho dos alunos e oferecerem feedback personalizado para melhorar o aprendizado.

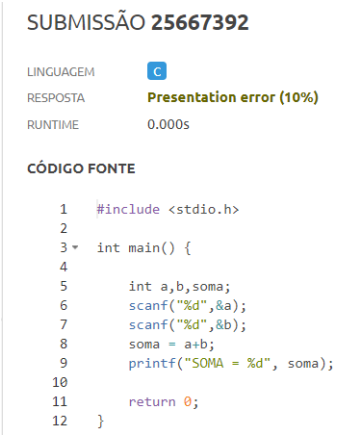


Figura 2.2 – Histórico de submissões de um problema.
Fonte: (BEECROWD, 2023).

Na Beecrowd, o módulo de aprendizagem aluno-professor permite que os educadores criem listas de exercícios a partir de problemas disponíveis no repositório. Ao criar uma lista, o educador pode definir uma data limite para a finalização da tarefa e selecionar os problemas que estarão inclusos, bem como as linguagens permitidas para a implementação pelo aluno.

Essa funcionalidade oferece flexibilidade para os educadores personalizarem as listas de acordo com o nível de dificuldade, o tempo disponível para a tarefa e as habilidades que desejam enfatizar. Além disso, a definição de uma data limite ajuda a manter o cronograma de aprendizado e incentiva os alunos a se dedicarem a cada tarefa.

Uma vez que a lista é criada, os alunos têm acesso aos problemas selecionados e podem enviar suas soluções para avaliação. O educador pode acompanhar o progresso dos alunos, fornecer feedback personalizado e avaliar o desempenho geral da turma. Essa funcionalidade é uma ótima maneira de incentivar a prática e aprimoramento das habilidades de programação dos alunos em um ambiente de aprendizagem interativo e personalizado.

A plataforma Beecrowd oferece aos usuários a opção de personalizar um problema. No entanto, o processo de publicação exige que o usuário siga uma série de etapas. A Figura 2.3 ilustra essas etapas, que incluem a especificação do enunciado do problema, as possíveis soluções aceitas e os possíveis erros de implementação, apresentação ou tempo excedido. Além disso, é necessário fornecer exemplos de entrada e saída esperadas para que o sistema possa compará-la com o algoritmo submetido pelo aluno. É fundamental também que os casos de teste sejam especificados para o problema. É importante notar que, diferentemente dos problemas padrões, que permitem a submissão de soluções em várias linguagens de programação, a submissão de problemas personalizados só é permitida na linguagem C (gcc 4.8.5).

Em resumo, a plataforma Beecrowd oferece diversas opções para que os usuários possam

aprimorar suas habilidades em programação e participar de competições, seja individualmente ou em times. Com suas funcionalidades flexíveis e variadas, a plataforma se destaca como uma excelente ferramenta para a aprendizagem e o desenvolvimento de programadores.

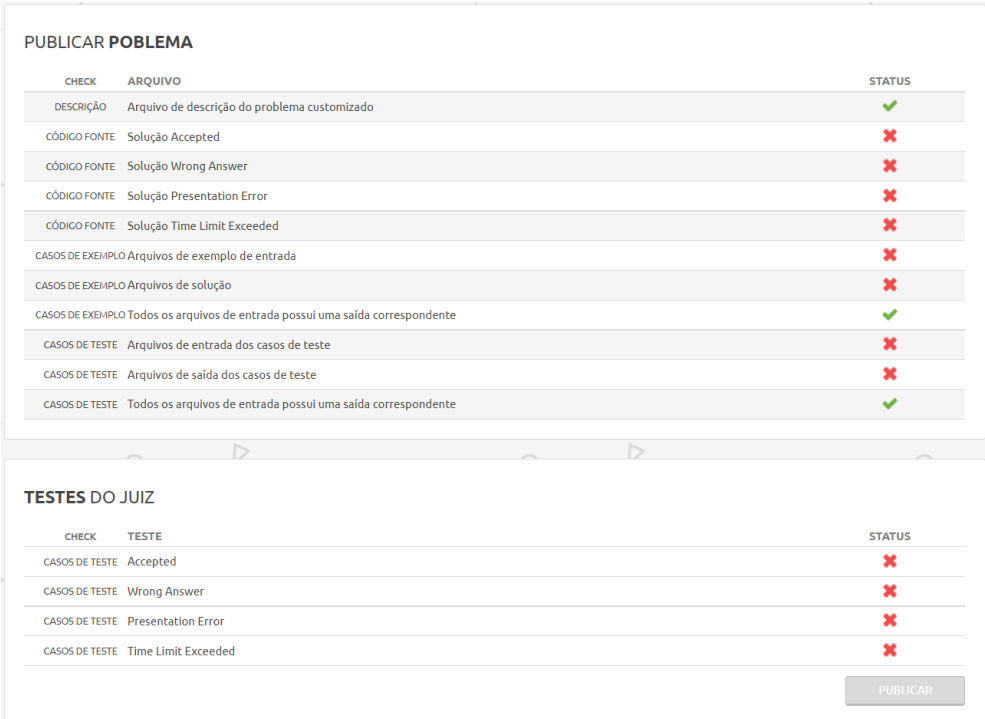


Figura 2.3 – Tela de publicação do problema.
Fonte: (BEECROWD, 2023).

Conforme mencionado anteriormente, a Beecrowd já dispõe de um vasto acervo de questões em seu banco de dados. No entanto, é possível personalizar uma questão, desde que se siga um padrão rigoroso para garantir compatibilidade com o corretor automático. Abrir essa discussão é fundamental para que o gerador automático de questões do *opCoders Judge* possa também explorar essa abordagem de compatibilidade com sistemas de avaliação.

Semelhantemente ao desejo adotado para o *opCoders Judge*, a criação de uma tarefa é realizada pelo professor, que seleciona previamente questões existentes no banco. No entanto, neste projeto, as questões já presentes no banco desempenham o papel de modelos, sendo usadas para gerar questões únicas para cada aluno. Além dessa diferenciação, as questões na Beecrowd são organizadas por meio de classificações baseadas em tópicos de programação. Por outro lado, neste trabalho, a categorização das questões é principalmente conduzida pelos "tópicos relacionados", seguindo a ementa da disciplina do Departamento de Computação da UFOP (DECOM).

2.1.1.2 SPOJ

Outra alternativa de *Online Judge* é o *Sphere Online Judge* (SPOJ). Além de atuar como *Online Judge*, o SPOJ também recebe título de *E-Learning Platform*, onde seus usuários podem

aprender conceitos de programação.

As questões de programação são definidas na plataforma como problemas. Para criar um problema, o usuário precisa de privilégio administrativo que pode ser concedido através de uma carta enviada aos moderadores. Um problema no SPOJ pode ter 3 tipos de julgamentos, o *Binary* classifica a solução apenas como correta ou incorreta, o *Maximise Score* pontua pela quantidade de similaridade com a solução esperada e *Minimise Score* retorna mais pontos para soluções mais distantes das esperadas. A Figura 2.4 mostra a parte inicial da criação de um problema que recebe como entrada os metadados padrões para identificar a questão e a opção *Resource* capaz de referenciar o autor do problema, caso tenha sido inspirado em algum problema disponível na internet.

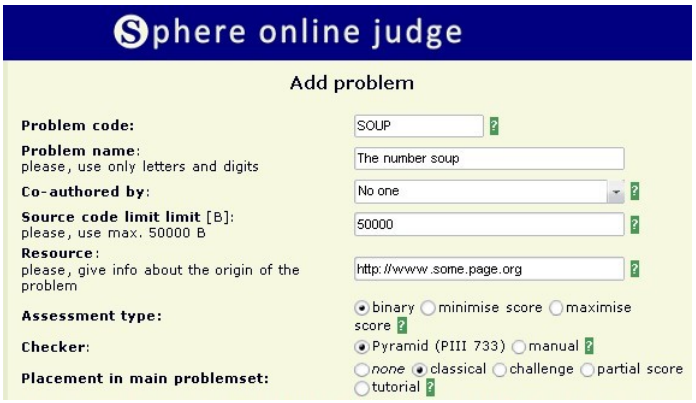


Figura 2.4 – Adicionando um problema.
Fonte: (SPOJ, 2023)

O editor de texto recebe o problema em formato HTML e, preferencialmente escrito em inglês. Na Figura 2.5 é representado o modelo de um problema sendo inserido no SPOJ e tornando-o disponível para utilização de terceiros em concursos a partir da marcação *Available for use in 3rd party contest*.

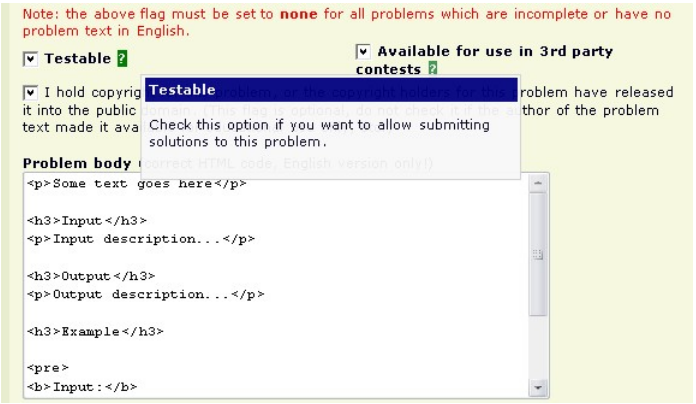


Figura 2.5 – Caixa de texto do problema.
Fonte: (SPOJ, 2023)

Uma característica que diferencia o SPOJ de outros Juízes Online é a liberdade concedida aos usuários para utilizar outros sistemas de correção automática de problemas. Isso significa que o SPOJ pode ser utilizado como plataforma para testar e avaliar outros Juízes Online. Além disso, a plataforma oferece uma grande variedade de linguagens de programação que podem ser utilizadas para solucionar os problemas. A Figura 2.6 ilustra esse cenário e demonstra como é possível adicionar novas linguagens à plataforma. Essa flexibilidade permite que os usuários possam escolher a linguagem de programação com a qual se sentem mais confortáveis e também possibilita a inclusão de novas linguagens na plataforma.

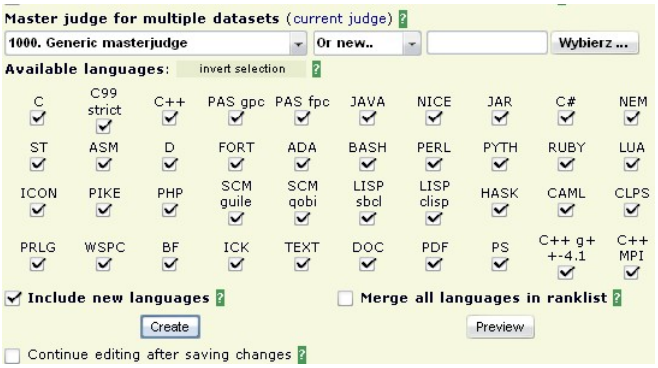


Figura 2.6 – Definição do Juiz e das linguagens.
Fonte: (SPOJ, 2023)

O método do SPOJ de inserção de problemas visa proporcionar um ambiente de diversas possibilidades de linguagens disponíveis para solução, juízes independentes da plataforma, entre outras alternativas de tratamento do problema. Dessa forma, o SPOJ torna-se dinâmico e cria outros interesses para o professor, como por exemplo, treinar juízes ou aplicar um estudo de caso para um problema que deva ser solucionado em mais de uma linguagem.

Portanto, os *Online Judges* são ferramentas que permitem criar ambientes para elaboração de questões de programação e correção automática de código. O SPOJ se diferencia por permitir a inserção de outros Online Judges para correção dos problemas, além de oferecer uma variedade de linguagens para solucioná-los. Ao criar alternativas para correção de um problema, o SPOJ abre um novo objeto de estudo para este trabalho, que por sua vez, decidiu-se não utilizar alternativas de correções em seus modelos de geração de questão.

2.1.2 BOCA *Online Contest Administrator*

Desenvolvido por Campos (2004), o BOCA é uma plataforma que funciona como um sistema de entrega de exercícios e correção a partir de mediadores, pessoas denominadas juízes. Os exercícios são criados de forma manual, juntamente com um arquivo de testes, e então são inseridos em um banco de dados relacional. Os juízes são pessoas convidadas a corrigir as questões utilizando o arquivo de testes e seu conhecimento na área. Além disso, fica a cargo do juiz sanar as dúvidas dos competidores.

Durante a maratona, dúvidas sobre os exercícios podem ocorrer. Para respondê-las sem comprometer a atenção dos demais participantes, o juiz possui uma interface na qual pode interagir com os competidores, respondendo suas dúvidas e enviando para todos ou apenas para os autores da pergunta. Essa interface é dada como exemplo na Figura 2.7.

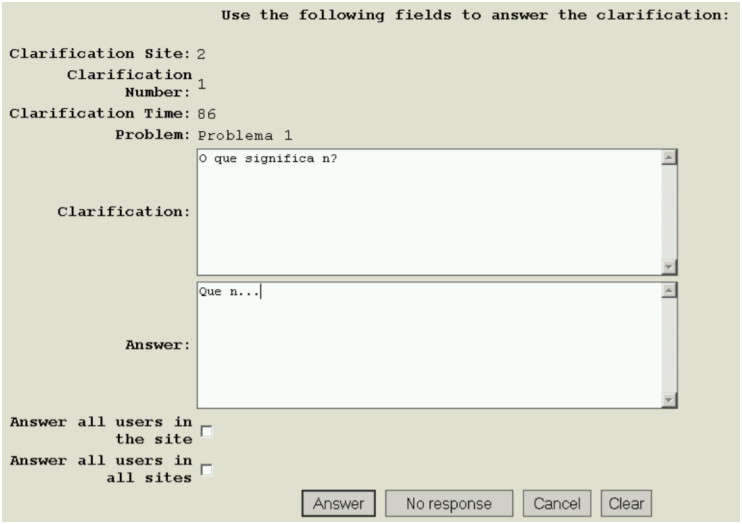


Figura 2.7 – Interação do Juiz com uma pergunta.
Fonte: (CAMPOS, 2004)

Por ser uma ferramenta de uso em maratonas, o BOCA tem foco na execução em servidor local para evitar fraudes com uso da internet, no entanto, também é possível criar uma competição multi-site.

Em resumo, pode-se concluir que o BOCA é uma plataforma de entrega e correção de exercícios, utilizada em maratonas de programação, que tem como objetivo evitar fraudes na resolução das provas. A ferramenta funciona a partir da participação de juízes convidados, que corrigem as questões utilizando um arquivo de testes e seu conhecimento na área. Durante a maratona, os juízes podem responder às dúvidas dos competidores através de uma interface específica, sem comprometer a atenção dos demais participantes. Este recurso é um diferencial que abre objeto de discussão para este trabalho. Agora, partindo para a análise de como o BOCA evita fraude nas provas executando toda a avaliação em servidor local, neste trabalho, a fraude é evitada utilizando um gerador automático de questões para que alunos tenham questões diferentes uns dos outros.

2.1.3 Geração de Questões Variadas

Em seu artigo, Genci (2013) aborda um desafio comumente encontrado em plataformas virtuais: a repetição de questões para os alunos. O autor identifica uma lacuna na existência de um sistema que possa oferecer questões quase idênticas aos alunos, com exceção de algumas variações no texto da questão. Essa abordagem visa manter as questões consistentes em seu grau de dificuldade, enquanto introduz alterações nas entradas, resultando, assim, em diferentes

respostas. Essa técnica pode ser aplicada em ambientes virtuais de apoio à aprendizagem, como o Moodle LMS usado pelo autor.

Para criar questões com diferença em seus valores de entrada, Genci (2013) emprega um método que se baseia em dois arquivos distintos. O primeiro desses arquivos contém a questão, com simbolos inseridos no texto para indicar os pontos onde ocorrerão substituições de termos. O segundo arquivo, formatado em *Comma-separated values* (CSV), contém todas as opções de entrada para serem inseridas no primeiro arquivo. No exemplo dado na Figura 2.8, uma questão da disciplina de Sistemas Operacionais teve algumas palavras selecionadas para serem substituídas por outras contidas no segundo arquivo. O algoritmo identifica essas palavras a partir de um marcador definido dentro do texto no formato *eXtensible Markup Language* (XML).

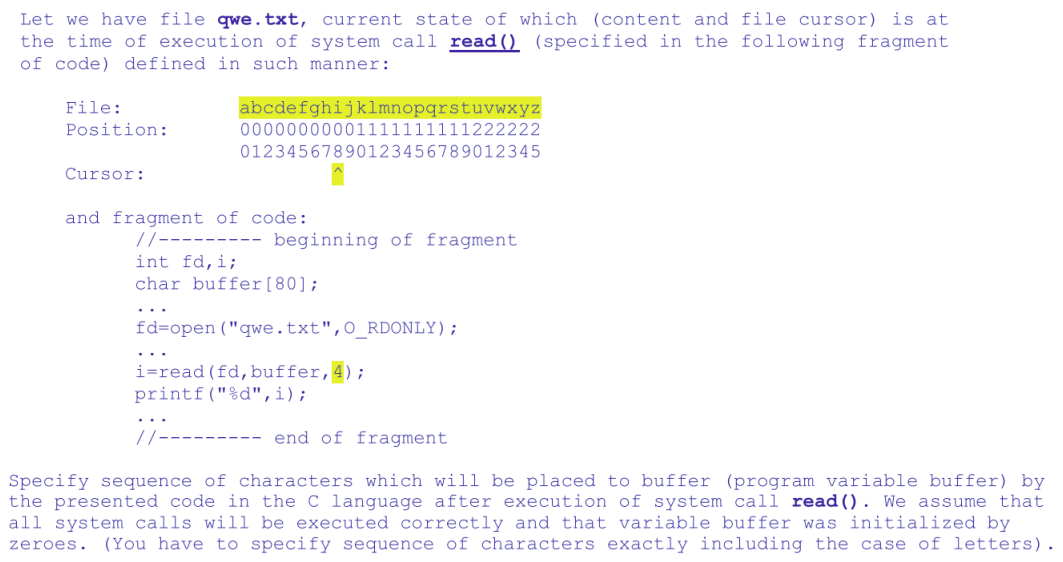


Figura 2.8 – Representação das palavras que serão substituídas
Fonte: (GENCI, 2013)

Para criar uma questão baseada em modelo, (GENCI, 2013) utilizou como padrão a escrita de questões em formato XML. Conforme ilustrado na Figura 2.9, é possível marcar com uma tag pré-definida as palavras do texto que receberão novos valores. Essas tags são, posteriormente, substituídas por palavras correspondentes presentes no segundo arquivo.

Após escolher as palavras a serem substituídas no arquivo da questão, o segundo arquivo é elaborado usando o ambiente do MS Excel. Na Tabela 2.1, a coluna PC exibe a criação de sete possíveis valores, os quais serão empregados no texto da questão. Note que estes arquivos precisam ser configurados para atender especificamente uma questão. Ainda na Tabela 2.1, os campos da coluna String são preenchidos manualmente com os valores possíveis, dos quais um será escolhido para compor o texto de uma versão da questão. As colunas P1 e P2 determinam o índice e a extensão dos caracteres extraídos da String, usados para gerar os valores em “Resultado” e determinar a posição do cursor, conforme especificado na coluna “Cursor do Arquivo”. Com

```
<!-- question: 0  -->
  <question type="category">
    <category>
        <text>$module$/Source for Pokus/Q01-001</text>
    </category>
  </question>

<!-- question: 7884  -->
  <question type="shortanswer">
    <name><text>Q01-0«PC»</text>
</name>
    <questiontext format="html">
<text> Let we have file qwe.txt, current state of which (content and file cursor)
is at the time of execution of system call read() (specified in the following
fragment of code) defined in such manner:

&lt;pre&gt;
Chars:          «String»
Position:       00000000001111111111222222
                01234567890123456789012345
Cursor: «File_cursor»

and fragment of code:
      //--------- beginning of fragment
      int fd,i;
      char buffer[80];
      ...
      fd=open(&quot;qwe.txt&quot;,O_RDONLY);
      ...
      i=read(fd,buffer,«P2»);
      printf(&quot;%d&quot;,i);
      ...
```

Figura 2.9 – Entrada da questão por XML
Fonte: (GENCI, 2013)

esses passos concluídos, o algoritmo é executado, resultando na criação da mesma questão, porém com algumas variações entre as versões entregues aos estudantes.

PC	String	P1	P2	Result	File Cursor
01	oneceauugfycnbwavaokfkvvkb	1	4	nece	_^
02	qwertyuiopasdfghjklzxcvbnm	4	4	tyui	____^
03	abcdefghijklmnopqrstuvwxyz	14	4	opqr	_____^
04	kjihgfedcbapqrstuvwxyzonml	16	5	uvwxy	_____^
05	yxwvutsrqponmlkjihgfedcba	5	5	tsrqp	_____^
06	bcdefghijklmnopqrstuvwxyza	8	5	jklmn	_____^
07	zabcdefghijklmnopqrstuvwxy	4	3	def	____^

Tabela 2.1 – Gerador de valores em CSV

Logo em seguida, é realizada uma mala direta entre o arquivo XML e a tabela criada no MS Excel. Esse processo automatizado permite criar diferentes variações da mesma questão, garantindo a diversidade das questões e diminuindo a quantidade de provas idênticas para os alunos.

Ainda que o método permita a criação de questões com certo grau de variação e personalização, ele está restrito a uma questão específica. Consequentemente, o professor é obrigado a configurar um novo XML e uma nova tabela de palavras para cada nova série de questões. Apesar do investimento considerável do tempo do professor na preparação manual das perguntas, essa estratégia limita o leque de possibilidades a um número finito de versões de questões. Para con-

tornar um obstáculo semelhante ao identificado no trabalho de Genci (2013), o estudo trabalhado no módulo proposto se possibilita a inserção de um valor dentro de um espectro infinito. No trabalho de Genci (2013), a aleatoriedade de valores não pode ser explorada com a utilização desta metodologia de construção de geração de questões.

2.2 Fundamentação Teórica

A abrangência temporal do trabalho se amplia a partir desta seção. Isso se justifica pela necessidade de analisar a trajetória pedagógica e de implementação tecnológica que instigou e iniciou o desenvolvimento de funcionalidades para o *opCoders Judge*.

A Subseção 2.2.1 trata do embasamento de que uma questão cujo assunto esteja próximo à realidade do aluno facilita a sua compreensão. Na Subseção 2.2.2, é discutida a complexidade em definir um grau de dificuldade para uma questão. Em seguida, as tecnologias utilizadas no projeto são apresentadas na Subseção 2.2.3.

2.2.1 Contextualização de Questões

A partir dos estudos realizados por Frossard (2018), percebe-se a importância de buscar estratégias pedagógicas que possibilitem a aproximação do conteúdo com a realidade do aluno. Através desse processo, o estudante é instigado a relacionar uma avaliação com sua própria experiência de vida, facilitando a compreensão de um enunciado de questão e desenvolvendo outras habilidades, como a de resolver um problema usando a programação.

Nesse sentido, a utilização de questões de avaliação que abordem temáticas relacionadas ao cotidiano do aluno pode ser uma estratégia eficaz para aproximar o conteúdo com sua realidade. Ao relacionar a questão com experiências de interesse do estudante, é possível favorecer uma compreensão mais profunda sobre o problema proposto e estimular a construção de soluções mais simples. Dessa forma, a combinação de uma questão com o cotidiano do aluno pode ser uma ferramenta poderosa para favorecer um aprendizado mais significativo e contextualizado.

Neste trabalho, as questões-modelo são criadas categorizando-as por área de conhecimento. Desta forma, é possível fazer com que as turmas de determinados cursos recebam questões cujo assunto do enunciado esteja próximo da realidade cotidiana de estudo do aluno.

2.2.2 Grau de Dificuldade de uma Questão

Ao determinar o nível de dificuldade de uma série de questões, é essencial manter uma consistência e padrão ao decidir o nível de dificuldade entre elas.Considerando o escopo deste trabalho, que visa fornecer questões semelhantes, porém com variáveis, a uma turma, o objetivo é assegurar que os alunos recebam perguntas diferentes, porém com a mesma complexidade.

De acordo com Santos et al. (2019), a dificuldade de uma questão pode variar entre cursos e turmas por conta do desempenho durante as aulas. Mesmo que o desempenho pudesse ser equivalente, é necessário analisar se o curso utiliza a computação como atividade meio ou como atividade fim. Entrelaçado a isso, segundo Barbosa, Costa e Brito (2017), a avaliação manual traz consigo o custo dispendioso e a experiência pessoal de cada avaliador.

O processo de avaliação de questões é uma tarefa complexa que envolve muitos desafios. Para abordar esses desafios, os pesquisadores têm se dedicado a desenvolver métricas de avaliação de dificuldade de questões. Santos et al. (2019) destaca a importância de métricas como o *Flesch Reading Ease* e as Métricas do Coh-Metrix-Port. O *Flesch Reading Ease* é uma métrica amplamente divulgada no Brasil, que calcula a complexidade dos textos dos enunciados usando o Índice Flesch. Já as Métricas do Coh-Metrix-Port utilizam técnicas avançadas de *Natural Language Processing* (NLP) e *Machine Learning* para definir atributos do texto do enunciado, oferecendo uma abordagem mais sofisticada para a avaliação de questões. Essas métricas são valiosas para garantir que as questões apresentem um nível de dificuldade adequado e sejam acessíveis a todos os alunos, independentemente do seu nível de conhecimento ou habilidade.

A definição do nível de dificuldade das questões disponíveis na plataforma deste trabalho será de responsabilidade do elaborador da questão. Para tanto, serão fornecidos padrões pré-definidos, com o intuito de orientar o usuário na escolha do nível de dificuldade apropriado. Dessa forma, espera-se que o usuário possa selecionar o nível de dificuldade que melhor se adeque ao perfil do público-alvo da questão.

2.2.3 Tecnologias utilizadas no projeto

As tecnologias de apoio para implementação deste trabalho estão descritas nesta subseção e traz consigo a definição de cada uma delas e como foram utilizadas.

Na Subseção 2.2.3.1, são apresentados os conceitos fundamentais do HTML5 e CSS3. Na Subseção 2.2.3.2, a linguagem PHP é abordada, incluindo seu papel no desenvolvimento deste trabalho. A seguir, na Subseção 2.2.3.3, o conceito da linguagem e sua relevância neste projeto são delineados. A Subseção 2.2.3.4 introduz o banco de dados MySQL.

2.2.3.1 HTML5 e CSS3

O HTML5 é uma atualização do HTML4. Todas as versões mantem os atributos básicos do HTML. No entanto, com o objetivo de facilitar a manpulação dos elementos, foram criadas seções para encurtar o trabalho de se criar rodapés, cabeçalhos e outras seções de estrutura da página web (SILVA, 2019).

Enquanto o HTML5 atua como uma linguagem de marcação para estruturar páginas da web, o CSS3 assume o papel de estilizá-las. Com a introdução dos módulos do CSS3 em sua

terceira versão, é possível criar classes de forma dinâmica, simplificando o desenvolvimento e a responsividade das páginas web (VEROU, 2015).

2.2.3.2 PHP

O PHP é uma linguagem interpretada capaz de embutir no código HTML scripts para fornecer uma lógica à página (CONVERSE; PARK, 2003). Um recurso muito útil do PHP é buscar informações de solicitações HTML feitas nas páginas. Sua forma simples e direta de armazenar as informações de requisições HTML contribui para uma boa comunicação com o banco de dados (BENTO, 2021).

O PHP teve um papel fundamental na arquitetura do *opCoders Judge*. Esta pesquisa deu continuidade ao uso da linguagem, uma vez que ela constitui a base estrutural do sistema. O PHP foi utilizado principalmente na manipulação das operações associadas ao banco de dados, bem como na busca das informações das solicitações HTML.

2.2.3.3 Javascript

Javascript é uma linguagem de alto nível, dinâmica, interpretada e não tipada (FLANAGAN, 2012). O JavaScript desempenha um papel crucial na interatividade das páginas web, Flanagan (2012) reforça que o Javascript permite a criação de comportamentos dinâmicos para os usuários. Como linguagem de programação do lado do cliente, ele viabiliza animações, atualizações em tempo real e respostas a eventos.

A linguagem JavaScript desempenhou um papel fundamental no desenvolvimento deste projeto. Sua aplicação abrangeu a identificação de variáveis no texto, facilitou a gestão da inserção dessas variáveis em uma tabela dedicada à tela de questões e foi essencial na criação de valores pseudoaleatórios e na geração de diferentes versões de questões a partir da questão-modelo.

2.2.3.4 MySQL

O MySQL, ou *Structured Query Language*, é um sistema gerenciador de banco de dados (SGBD) que utiliza o modelo relacional. Foi escrito para obter grande compatibilidade com sistemas operacionais, além de garantir confiabilidade na sua forma de armazenamento e processamento de dados (MILANI, 2007).

No Mysql, é simples configurar o mecanismo de manipulação de dados *(Storage Engine)*. Ela define o modo de funcionamento e armazenamento da tabela no banco (MILANI, 2007).

3 Desenvolvimento

Este trabalho visa a criação de um módulo no *opCoders Judge* que permita ao professor elaborar questões de programação para alunos evitando questões idênticas distribuídas para diversos alunos. Para isso, foi desenvolvido um método para o professor elaborar um modelo de questão e a partir deste modelo, as versões de questões serão criadas pelo sistema. Além disso, as questões-modelo possuem metadados que auxiliam na sua classificação, como a área de conhecimento, o nível de dificuldade da questão, os tópicos relacionados ao ensino de programação e as tags que permitem ao professor estabelecer um padrão de características aplicáveis às questões.

A metodologia adotada para a criação desse ambiente é detalhada na Seção 3.1, que está dividida em subseções que complementam a proposta apresentada. A modelagem conceitual do módulo é apresentada na Seção 3.2. O Banco de Dados possui uma descrição mais detalhada na Seção 3.3. Por fim, a Seção 3.4 explora as Operações Básicas em Questões.

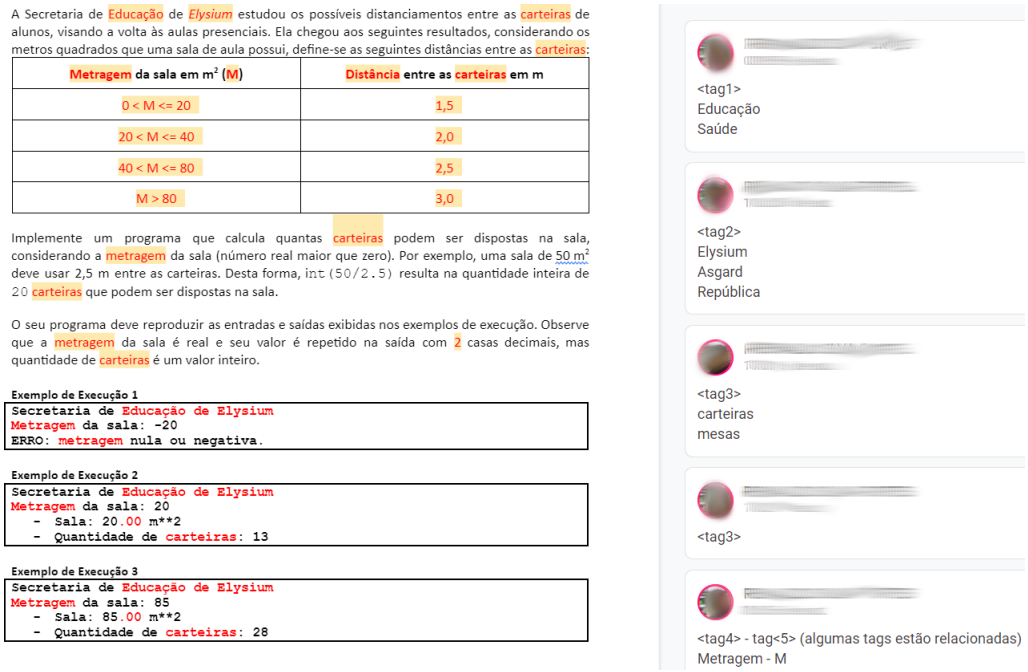
3.1 Metodologia

Na disciplina de Programação de Computadores na Universidade Federal de Ouro Preto (UFOP), alguns professores utilizam a ferramenta Google Docs, da Google (2023) para esboçar o modelo de uma questão. Com a intenção de tornar a questão suscetível a mudanças em algumas palavras ou valores, os professores marcam os elementos do texto que serão substituídos por outras palavras. Para fazer a substituição de fato, um algoritmo escrito em Python recebe a questão e as palavras candidatas à troca e, em seguida, é feita uma análise combinatória para gerar uma lista com todas as possibilidades de questões baseadas na substituição dos elementos do texto. Dessa forma, é possível criar versões de questões que apresentam distinções.

Na Figura 3.1, é apresentado um exemplo de questão-modelo formulada no Google Docs, onde as palavras candidatas a serem inseridas em uma versão de questão estão definidas nos comentários do documento. Esse método permite que os professores planejem a criação de questões personalizadas para seus alunos.

Embora o método de criação de versões de questões seja eficaz, havia o desejo de conceber um módulo que pudesse gerar versões individuais das questões para os alunos, garantindo que cada um recebesse uma questão com aspectos diferenciados em comparação com as questões dos colegas. Esta necessidade identificada propiciou a elaboração e implementação deste trabalho de forma a criar um módulo que permitisse a formulação de questões-modelo e, a partir delas, a geração de múltiplas versões de questões.

Para obter essa variedade de versões de questões, a questão-modelo elaborada pelo professor utiliza variáveis em seu texto que podem receber diversos valores. Dessa forma, é



A Secretaria de Educação de *Elysium* estudou os possíveis distanciamentos entre as carteiras de alunos, visando a volta às aulas presenciais. Ela chegou aos seguintes resultados, considerando os metros quadrados que uma sala de aula possui, define-se as seguintes distâncias entre as carteiras:

Metragem da sala em m^2 (M)	Distância entre as carteiras em m
0 < M <= 20	1,5
20 < M <= 40	2,0
40 < M <= 80	2,5
M > 80	3,0

Implemente um programa que calcula quantas carteiras podem ser dispostas na sala, considerando a metragem da sala (número real maior que zero). Por exemplo, uma sala de 50 m^2 deve usar 2,5 m entre as carteiras. Desta forma, `int(50/2.5)` resulta na quantidade inteira de 20 carteiras que podem ser dispostas na sala.

O seu programa deve reproduzir as entradas e saídas exibidas nos exemplos de execução. Observe que a metragem da sala é real e seu valor é repetido na saída com 2 casas decimais, mas quantidade de carteiras é um valor inteiro.

Exemplo de Execução 1

```
Secretaria de Educação de Elysium
Metragem da sala: -20
ERRO: metragem nula ou negativa.
```

Exemplo de Execução 2

```
Secretaria de Educação de Elysium
Metragem da sala: 20
  - Sala: 20.00 m**2
  - Quantidade de carteiras: 13
```

Exemplo de Execução 3

```
Secretaria de Educação de Elysium
Metragem da sala: 85
  - Sala: 85.00 m**2
  - Quantidade de carteiras: 28
```


Figura 3.1 – Questão criada de forma manual.
Fonte: Autoral (2023)

possível gerar versões de questões com valores distintos. Quanto maior o número de variáveis no texto, maior será a diversidade de elementos em uma versão da questão.

Uma questão-modelo requer metodologias específicas para sua elaboração. Para entender melhor essas metodologias, este tópico foi dividido em subtópicos. A Subseção 3.1.1 apresenta a escolha do editor de texto ideal para manter a estrutura e a estilização do enunciado da questão inserida. Em seguida, na Subseção 3.1.2, a estratégia para identificar e configurar as variáveis contidas no texto do enunciado foi detalhada. Já na Subseção 3.1.3, é possível compreender como o algoritmo escolhe pseudoaleatoriamente os valores a serem atribuídos às variáveis do texto. Na Subseção 3.1.4 é explicado como as variáveis são identificadas no texto da questão. Na Subseção 3.1.5, as informações descritivas sobre a questão são usadas para caracterizá-la por meio de Metadados.

3.1.1 Editor de Texto para o Enunciado da Questão

Para que o professor possa elaborar uma questão-modelo, é necessário que ele utilize um editor de texto capaz de receber uma formatação para estruturar e estilizar todo o enunciado, de acordo com a preferência do professor. Dessa forma, o professor consegue organizar o texto da questão em seções, utilizando diferentes tamanhos e estilos de fonte, além de adicionar imagens e tabelas para enriquecer o conteúdo do enunciado.

Também é importante que o editor de texto retorne o texto do enunciado ao algoritmo em um formato padrão, como o *HyperText Markup Language* (HTML). Isso garante a integridade

da estrutura do enunciado, assim como a estilização, que pode ser mantida mesmo após o armazenamento no banco de dados e a recuperação quando for necessária. A utilização de um editor de texto com essa funcionalidade simplifica o processo de criação de questões e torna mais fácil a elaboração de enunciados claros e bem estruturados.

Através da interface de entrada, o educador pode descrever uma questão-modelo e definir variáveis dentro do próprio enunciado. Essas variáveis podem ser utilizadas para tornar a questão dinâmica, permitindo que diversas questões sejam geradas com alterações nos valores definidos em cada uma delas. Essa funcionalidade é especialmente útil para promover a prática e o aprimoramento dos conceitos ensinados, já que cada aluno poderá ter acesso a uma versão única da questão.

Com base nessas premissas, foi investigada a utilização do editor de texto TinyMCE. Segundo TinyMCE (2023), o TinyMCE é um editor *Rich Text* de código aberto, escrito em Javascript, amplamente utilizado para criar e editar conteúdo de texto com personalização de estrutura e estilização. Ele possui uma interface do tipo *What You See Is What You Get* (WYSIWYG) com plugins para formatar textos, criar tabelas, links, adicionar imagens e outros recursos que enriquecem o texto. Sua interface gráfica pode ser vista na Figura 3.2.

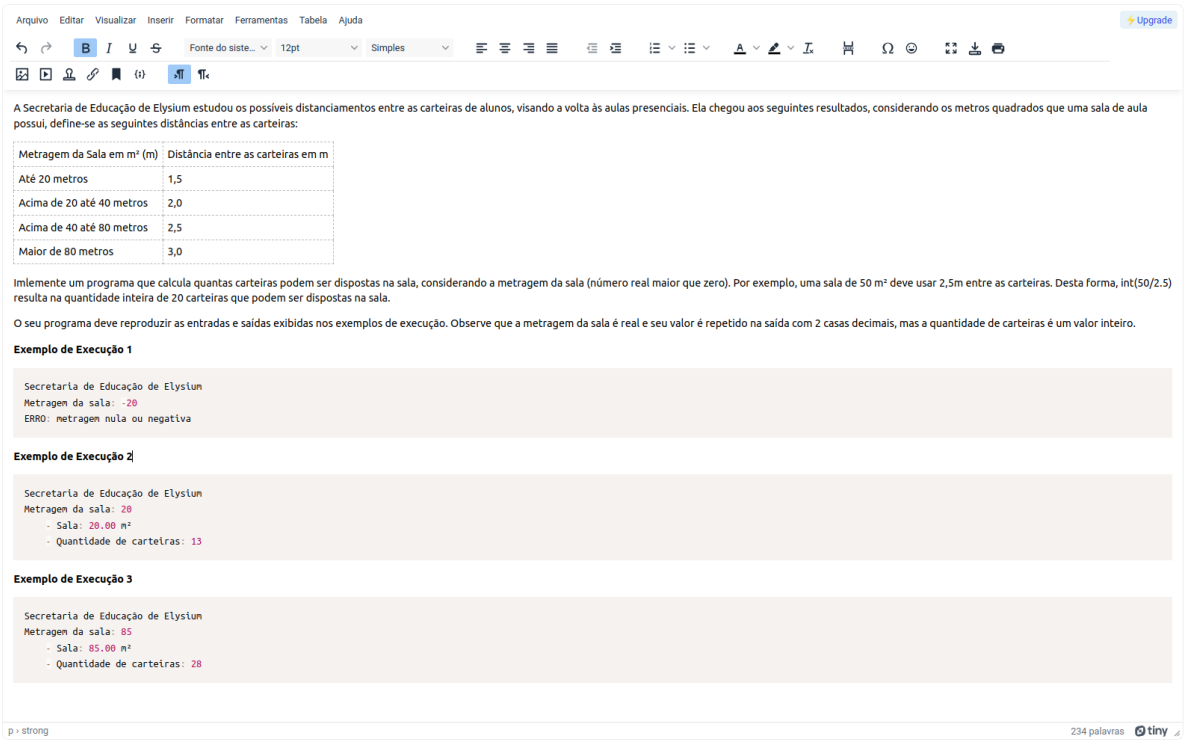


Figura 3.2 – TinyMCE: Editor de texto em formato HTML

O TinyMCE é capaz de extrair o conteúdo escrito no formato HTML, permitindo que seja armazenado no banco de dados com a integridade da estrutura do texto e sua estilização preservadas.

Logo, foi decidido utilizar o TinyMCE por se tratar de um editor eficiente e atualizado,

no que tange à elaboração da questão no módulo proposto neste trabalho.

3.1.2 Método de Identificação das Variáveis

Para elaborar uma questão-modelo, é necessário que partes do textos sejam reservados para receber valores específicos. Essas partes são chamadas por este trabalho de variáveis e são identificadas pelo algoritmo através de uma palavra precedida do símbolo "@". Posteriormente, o professor que estiver elaborando a questão define o tipo da variável, podendo ser numérico (inteiro ou real), do tipo "seleção" ou ainda uma "expressão".

Esta Subseção será dividida para explicar detalhadamente os tipos de variáveis Inteiro, Real, Seleção e Expressão.

3.1.2.1 Variável do tipo inteiro

Para adicionar variáveis no texto da questão-modelo com a intenção de gerar valores inteiros, é necessário pensar em um intervalo de números no qual faça sentido para a atribuição do valor à variável. Um valor mínimo e um valor máximo devem ser estipulados pelo professor. Dessa forma, é possível garantir que os valores gerados estejam dentro de um intervalo aceitável e correspondam às necessidades do texto da questão.

Para ilustrar, considere o caso em que o texto da questão elaborada pelo professor inclui a sentença "Pela rua passaram @quantidade carros amarelos". Nesse contexto, a variável corresponde à palavra "quantidade", precedida pelo símbolo "@". Essa variável será preenchida posteriormente com um valor numérico inteiro. Como já mencionado, um intervalo é manualmente definido para essa variável. Por exemplo, o valor mínimo pode ser 2 e o valor máximo pode ser 15. Esse intervalo garante que "@quantidade" pode ser preenchido com valores como 2, 10 ou 15, já que esses valores estão dentro do domínio estabelecido para essa variável.

3.1.2.2 Variável do tipo real

A ideia por trás do funcionamento das variáveis do tipo real é semelhante a de variáveis do tipo inteiro, no entanto com um acréscimo. Para que as variáveis do tipo real sejam geradas com precisão, é importante que o elaborador da questão defina um intervalo de valores e defina também a quantidade de casas decimais que serão usadas para representar os valores gerados. Essa informação é crucial para que o enunciado da questão tenha a precisão necessária para o contexto em que será aplicado. Para tornar a variável com possibilidade infinita de entrada, basta deixar em branco o campo para o valor mínimo, o valor máximo do intervalo, ou ambos.

Por isso, é fundamental que o professor que elabora a questão-modelo tenha um bom conhecimento sobre o contexto do problema e as exigências da questão para definir corretamente os valores mínimo e máximo e a quantidade de casas decimais necessárias para cada variável do tipo real.

3.1.2.3 Variável do tipo seleção

Para atribuir valores textuais a variáveis, é preciso ter atenção especial, pois é necessário respeitar as regras morfológicas da língua portuguesa. A flexão de gênero do substantivo é um exemplo que mostra a complexidade dessa tarefa. Em alguns casos, é necessário substituir mais de uma palavra para evitar erros de gênero. Além disso, a formação do plural é outra flexão que pode exigir alteração em um conjunto de palavras. Por exemplo, considere a frase "Quatro carros passaram na rua". É possível notar que algumas palavras formam uma estrutura de pluralidade. Se a troca da palavra "rua" por "estacionamento" acontecesse, a flexão de gênero não estaria adequada às regras da língua portuguesa. Portanto, é importante estar atento a todas essas questões para que o texto gerado pelo algoritmo seja morfologicamente correto.

Para que a mudança das palavras no texto seja realizada ao mesmo tempo em que se mantenha a estrutura linguística correta, foi preciso pensar em um algoritmo capaz de fazer mudanças de mais de uma palavra, respeitando as regras morfológicas. Para isso, o tipo "seleção" funciona como uma matriz. Por exemplo, o trecho "Os carros" está no gênero masculino e no plural. Uma alternativa para torná-lo uma forma dinâmica reconhecida pelo algoritmo seria "@palavra[0] @palavra[1]". Agora, pode-se adicionar uma regra à variável "palavra", especificando que ela pode receber "Os, automóveis | A, moto | O, carro". Nesse caso, o algoritmo escolherá aleatoriamente entre "Os automóveis", "A moto" ou "O carro". Supondo que o algoritmo escolha pela primeira opção, seu índice será 0, portanto a busca para atribuir o valor à variável será por "palavra[0][0] @palavra[0][1]" e seu resultado será "Os automóveis". Como representado na Tabela 3.1, a Matriz de Seleção é a estrutura criada pelo algoritmo para permitir a seleção aleatória de valores sem perder a referência de todos os valores que precisam ser modificados simultaneamente para que não ocorra uma violação às regras morfológicas da língua portuguesa.

	@palavra[x][0]	@palavra[x][1]
x=0	Os	automóveis
x=1	A	moto
x=2	O	carro

Tabela 3.1 – Matriz de opções de seleção

3.1.2.4 Variável do tipo expressão

Os textos de enunciado podem conter cálculos ou representações numéricas que dependem de valores pré-definidos. Um exemplo disso é o enunciado da Figura 3.3, no qual foram destacadas em amarelo todas as variáveis para facilitar a identificação pelo leitor. O algoritmo reconhece as variáveis a partir do símbolo "@" que as precede. No exemplo, podemos identificar as variáveis "calculo1", "indice1" e "distancia1". A variável "calculo1" é obtida pela divisão entre "indice1" e "distancia1", conforme ilustrada mais a frente na Figura 3.4. Por isso, é classificada como do

tipo “expressão”. Para informar isso ao algoritmo, foi definida a variável como tal e indicada sua entrada como “@indice1 / @distancia1”.

Ainda na Figura 3.4, a variável “calculo1” é um exemplo do uso de funções e expressões matemáticas para realizar cálculos em variáveis. A função *math.ceil()* é utilizada para arredondar o resultado da divisão de “indice1” por “distancia1” para o próximo número inteiro maior.

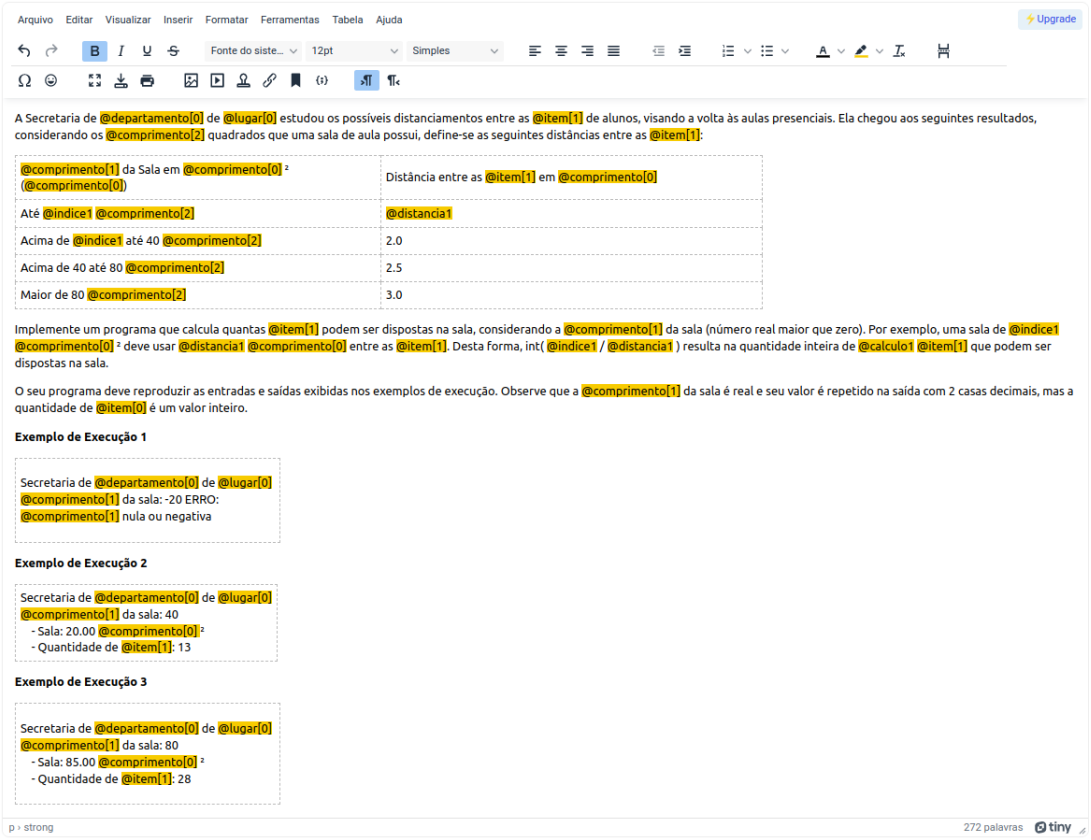


Figura 3.3 – Enunciado com variáveis definidas

As expressões condicionais são comumente usadas em programação e são uma maneira de permitir que o código tome decisões com base em certas condições. Na utilização de variáveis em conjunto, as expressões condicionais podem ser usadas para definir o valor de uma variável com base em uma condição específica.

No exemplo mencionado no texto, a variável “comprimento[0]”, vista na Figura 3.4, pode assumir dois valores diferentes, “m” ou “pés”. Nesse caso, poderia ter utilizado uma expressão condicional na variável “indice” que recebe o valor do comprimento e informar que se a unidade de medida fosse “pés” o valor de “indice” seria convertido de forma a manter a proporcionalidade da medida.

O processo de tornar partes do texto dinâmicas é fundamental para criar questões-modelo que possam ser respondidas de forma automática por um algoritmo. É importante que o elaborador da questão identifique corretamente as variáveis e estabeleça as regras para cada uma delas, a fim de garantir que o algoritmo forneça uma resposta precisa e coerente.

3.1.2.5 Tabela de definição das variáveis

Para elaborar uma questão-modelo, é preciso definir as regras para preencher as variáveis presentes no enunciado. Essas regras são definidas por meio da tabela de variáveis, como ilustrado na Figura 3.4. Nessa tabela, é necessário inserir o nome da variável e seu tipo, que pode ser seleção, inteiro, real ou expressão. Além disso, é preciso definir o conjunto de valores que a variável pode assumir.

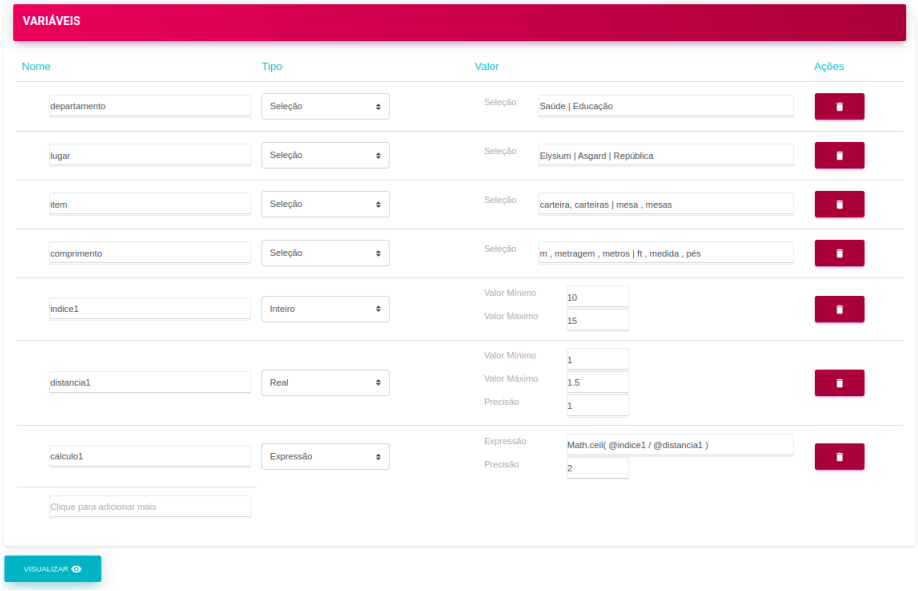


Figura 3.4 – Tabela de variáveis relacionadas ao enunciado

Após realizar o preenchimento das variáveis na tabela, o próximo passo é visualizar como os valores serão atribuídos ao texto da questão, substituindo o nome de suas variáveis pelos seus respectivos valores. Essa etapa é crucial para verificar se os valores atribuídos às variáveis estão corretos e coerentes com a proposta da questão. Na Figura 3.5, é possível ver duas possibilidades de questões geradas dinamicamente por meio de algoritmos pseudoaleatórios, com a cor de fundo verde destacando todas as variáveis que receberam valores.

No entanto, é importante ressaltar que, caso alguma variável presente no enunciado não tenha sido substituída devido à ausência de preenchimento na tabela de variáveis, a cor de fundo correspondente será vermelha. Isso significa que é necessário revisar a tabela de variáveis e preencher as informações faltantes para que a questão fique completa e coerente.

Além disso, a utilização de algoritmos pseudoaleatórios permite a criação de questões diferentes a cada execução do programa, tornando o processo de geração de questões mais dinâmico e interessante. Isso possibilita a elaboração de um grande número de questões de forma rápida e eficiente, o que torna especialmente útil para professores e instituições de ensino que precisam produzir uma grande quantidade de avaliações.

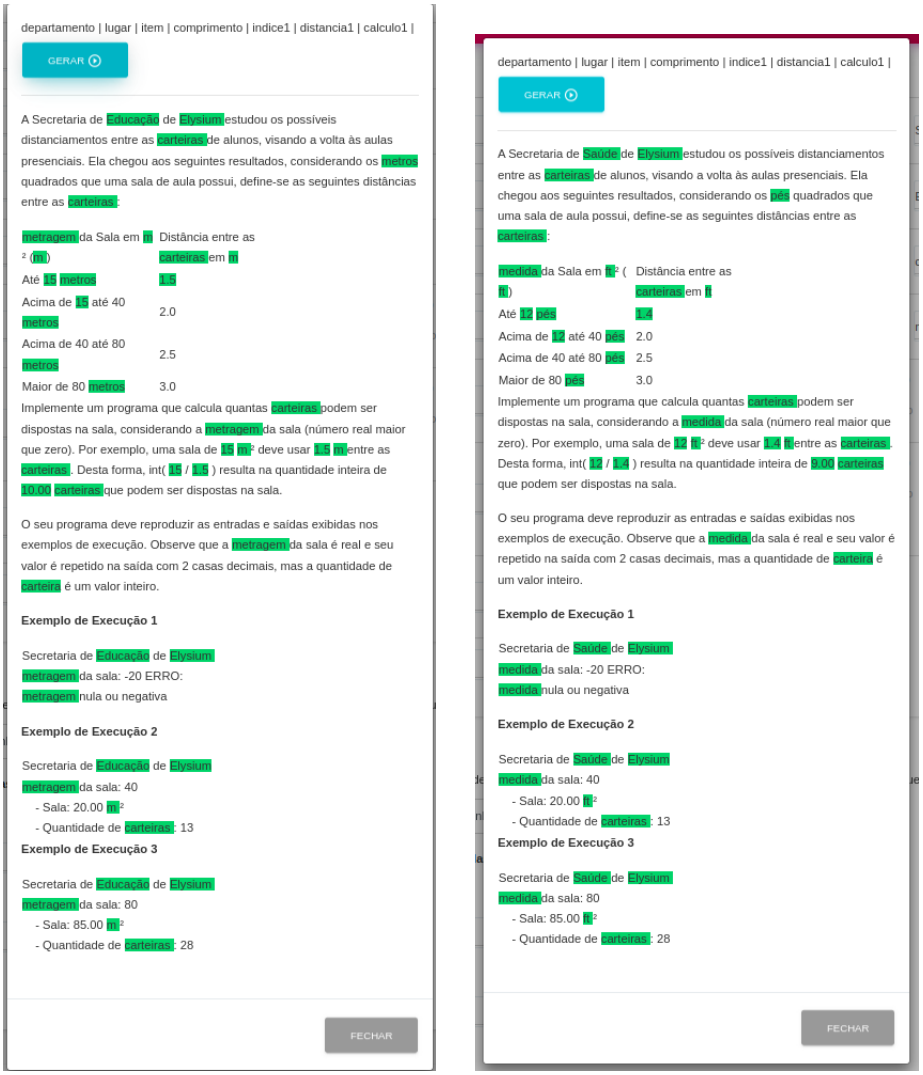


Figura 3.5 – Respostas do gerador de questão dinâmica.

3.1.3 Escolha pseudoaleatória dos valores das variáveis

Para criar questões dinâmicas, valores aleatórios são gerados para as variáveis. No entanto, é importante que esses valores estejam dentro de um domínio específico, para garantir que a solução seja viável e faça sentido com o texto. Para isso, é comum utilizar o termo "pseudoaleatório", que significa que o valor é gerado por um algoritmo que segue um padrão, mas que para fins práticos, pode ser considerado aleatório.

A faixa de possíveis valores para variáveis inteiras ou reais pode ser infinita, estando sujeita às diretrizes definidas pelo professor no domínio da variável. Por outro lado, no contexto das variáveis do tipo "seleção", a probabilidade de escolha é calculada como $\frac{1}{n}$, em que n representa o número de opções possíveis determinado pelo professor para essa variável.

3.1.4 Método de atribuição à variável

Com o texto da questão redigido e as variáveis definidas, o algoritmo está pronto para substituir o nome das variáveis no texto por seus respectivos valores gerados aleatoriamente.

Para que o algoritmo entenda a variável no texto que contém o símbolo "@" é a mesma definida sem o mesmo símbolo, utilizou-se uma busca através da expressão regular "/@\w+\b/g". Esta expressão regular busca por todas as ocorrências de palavras iniciadas por "@" seguidas de uma ou mais letras, números ou underscores, considerando apenas o final da palavra como um limite. O "g" no final da expressão indica que a busca deve ser global, ou seja, aplicada em todo o texto, e não apenas na primeira ocorrência encontrada. Vale ressaltar que variáveis do tipo seleção têm seus índices ignorados pelo algoritmo, a fim de que seja capturado apenas o nome da variável.

Após identificar as variáveis presentes no texto do enunciado, o próximo passo do algoritmo é buscar o nome de cada uma delas na tabela de variáveis. Essa tabela contém as informações necessárias para a geração dos valores correspondentes às variáveis, como o tipo e, no caso de variáveis numéricas, o domínio possível de valores.

Para gerar um valor aleatório dentro do domínio predefinido para variáveis numéricas, o algoritmo utiliza funções específicas da linguagem de programação para gerar um número dentro desse intervalo. Em seguida, o nome da variável no texto é substituído pelo valor gerado aleatoriamente.

No caso de variáveis do tipo seleção, o algoritmo escolhe aleatoriamente uma das opções predefinidas para a variável e substitui o nome da variável no texto pelo valor correspondente. Isso é feito por meio da escolha de uma posição no conjunto de palavras possíveis para aquela variável e utilizando o índice para fazer a substituição no texto.

Já para as variáveis do tipo expressão, o processo é um pouco mais complexo. O algoritmo verifica todas as variáveis já existentes antes do surgimento da variável do tipo expressão no enunciado para capturar seus valores e fazer com que a expressão substitua os nomes das variáveis pelos seus respectivos valores. Em seguida, a expressão é avaliada e o resultado é utilizado para substituir o nome da variável no texto.

Para validar essas variáveis inseridas no texto da questão, o professor as define manualmente, digitando os nomes correspondentes na tabela de variáveis. Dessa maneira, o professor assegura que apenas as palavras precedidas pelo símbolo "@" que ele deseja como variáveis serão tratadas como tais. Esse método previne potenciais conflitos no caso de o professor precisar usar o caractere "@" sem que ele seja interpretado como um indicador de variável. Simultaneamente, essa abordagem confere maior segurança ao processo de construção de questões-modelo, proporcionando ao professor um controle mais detalhado.

É importante ressaltar que todo esse processo de atribuição à variável é feito de forma dinâmica, ou seja, a cada vez que o algoritmo é executado, novos valores aleatórios são gerados e

novas expressões podem ser avaliadas, permitindo a geração de diferentes versões do mesmo enunciado.

3.1.5 Metadados da questão

Para criar um banco de questões-modelo eficiente, é importante considerar a organização e classificação das mesmas através de metadados. Como explica Ikematu (2001), os metadados são importantes para documentar e organizar dados com a finalidade de facilitar a identificação desse dados. Em outras palavras, são informações que descrevem os dados que estão sendo armazenados, assim como uma etiqueta que identifica o conteúdo de um arquivo físico, por exemplo.

Para aplicabilidade nas questões-modelo, os metadados são informações capazes de categorizar e identificar as questões de acordo com sua área de conhecimento, nível de dificuldade, tópicos relacionados ao assunto e tags que informam mais livremente palavras que estejam relacionadas à questão. A Figura 3.6 ilustra os metadados citados anteriormente. Com os metadados, é possível criar uma estrutura que facilite a busca e seleção das questões mais adequadas para cada situação. Neste contexto, os metadados desempenham um papel fundamental para a organização e gestão do banco de questões dinâmicas.

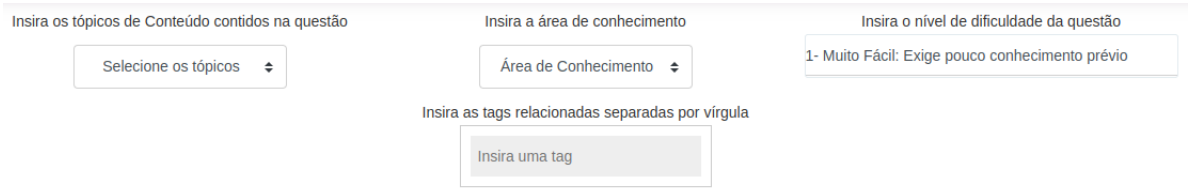


Figura 3.6 – Metadados da questão-modelo.

3.1.5.1 Área de Conhecimento

O metadado "Área de Conhecimento" é uma categoria utilizada para identificar e classificar as questões dinâmicas em áreas específicas do conhecimento. Essa categoria é preenchida pelo professor no momento da elaboração da questão, selecionando as áreas de conhecimento relacionadas ao tema abordado na questão.

Como ilustra a Figura 3.7, as opções de áreas de conhecimento são pré-definidas e podem incluir, por exemplo, engenharia de produção, engenharia de automação industrial, engenharia ambiental, estatística e química industrial. O professor também pode selecionar mais de uma área de conhecimento, caso a questão aborde temas que se enquadrem em mais de uma categoria.

O uso do metadado "Área de Conhecimento" é importante para facilitar a organização e recuperação das questões dinâmicas, tornando mais fácil a busca por questões relacionadas a uma

Figura 3.7 – Metadado "Área de conhecimento".

área específica do conhecimento. Além disso, esse metadado ajuda a garantir que as questões sejam utilizadas de forma mais eficiente e direcionada ao curso.

3.1.5.2 Tópicos Relacionados

O metadado "Tópicos Relacionados" tem como objetivo reunir informações que possam caracterizar a questão de programação como uma questão que esteja relacionada a um conteúdo específico do ensino de programação, como por exemplo, funções ou estrutura condicional. Isso permite que os professores possam filtrar questões de acordo com os tópicos que estão ensinando em sala de aula, facilitando a criação de tarefas e avaliações.

É comum que uma questão de programação envolva vários tópicos específicos do assunto. Para caracterizar melhor a questão, é possível selecionar diversos tópicos relacionados, como demonstrado na Figura 3.8.

Figura 3.8 – Metadado "Tópicos Relacionados".

Ao utilizar esse metadado, os professores também podem identificar questões que possuem similaridades em relação aos tópicos lecionados em sala de aula, o que pode ser útil na hora de planejar aulas e atividades que envolvam a revisão desses tópicos.

Em resumo, o metadado "Tópicos Relacionados" é uma ferramenta importante para organização e categorização de questões de programação, permitindo uma melhor identificação dos conteúdos presentes nas questões e facilitando a criação de atividades e avaliações.

3.1.5.3 Tags Relacionadas

Os metadados apresentados anteriormente formam um escopo fechado para assegurar agrupamentos mais definidos das questões-modelo. No entanto, foi considerado relevante incluir um metadado com uma abordagem mais aberta.

Inicialmente, o metadado “Tag” não possui uma regra ou orientação pré-definida para sua inserção. Essa entrada será utilizada como um meio de pesquisa para desenvolver um padrão de classificação de questões em colaboração com os professores, quando o módulo de questões estiver em funcionamento no ambiente de produção.

Para explicar o funcionamento prático, espera-se que o professor inclua uma ou mais tags, separadas por vírgula, que estejam de alguma forma relacionadas ao conteúdo da questão. Um exemplo apropriado de conjunto de tags se encontra na Figura 3.9, na qual foram inseridas as tags "mru, física, velocidade, veículo"para uma questão de exatas relacionada a esses assuntos.

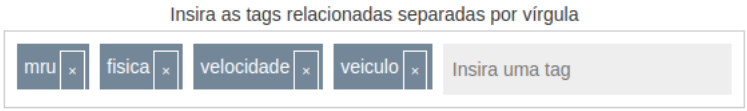


Figura 3.9 – Metadado “Tags”.

3.1.5.4 Determinação do Grau de Dificuldade

O nível de dificuldade da questão é um metadado importante na elaboração de questões de programação para a plataforma. Ele é determinado pelo próprio professor que está criando a questão, e é utilizado para indicar o grau de dificuldade da questão. É comum que, ao criar uma questão, o professor tenha em mente um determinado nível de dificuldade que seja adequado, e é através do metadado de nível de dificuldade que ele pode comunicar essa informação aplicada à própria questão.

Para ajudar a determinar o grau de dificuldade da questão, foram utilizadas descrições para cada nível de 1 a 10, a fim de padronizar um pouco as decisões de nível aplicado nas questões pelos professores. Por exemplo, uma questão com nível de dificuldade 1 foi descrita como “1- Muito Fácil: Exige pouco conhecimento prévio”, enquanto uma questão com nível de dificuldade 10 pôde ser descrita como “10- Épica: Alto nível de complexidade e exigência de conhecimento”.

Essas descrições podem ser visualizadas na Figura 3.10. Elas são úteis para que os professores possam criar questões que estejam de acordo com as habilidades e conhecimentos dos seus alunos, além de auxiliar na seleção das questões mais adequadas para uma determinada avaliação. O uso correto desse metadado pode fazer uma grande diferença na qualidade das avaliações e no sucesso dos alunos.

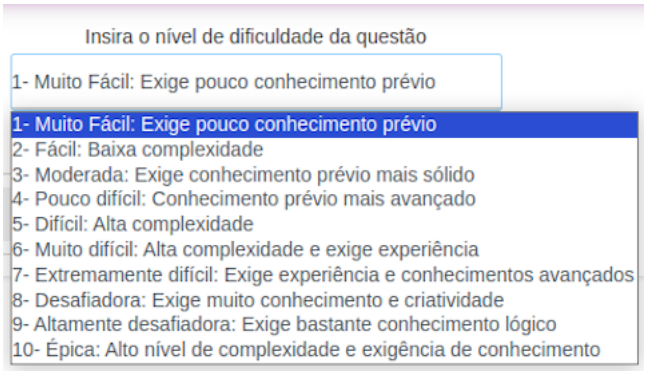


Figura 3.10 – Graus de dificuldade da questão.

3.2 Modelagem Conceitual

Com o objetivo de oferecer uma compreensão mais precisa da dinâmica de interação entre o professor e o módulo, a Figura 3.11 ilustra os casos de uso específicos para o professor dentro do módulo. A partir da criação de uma questão-modelo, o professor também define a área temática correspondente, além dos tópicos relacionados ao conteúdo de programação abordado e as tags, que seriam as etiquetas associadas. A seguir, é necessário configurar o domínio de cada variável inserida no texto da questão. Também é permitido fazer a edição de uma questão, no entanto, a possibilidade de excluir uma questão-modelo foi repensada e optou-se por criar o recurso de desabilitar uma questão.

Na etapa de modelagem conceitual de dados, a primeira abordagem escolhida consistiu em desenvolver a estrutura fundamental que envolve uma questão utilizando o Modelo de Entidade e Relacionamento (MER), conforme ilustrado na Figura 3.12. Essa figura representa o primeiro modelo concebido para a estrutura do módulo. É importante observar que as variáveis eram originalmente designadas como “Elementos Dinâmicos” e tratadas como entidades independentes. Posteriormente, essas variáveis foram integradas como atributos da entidade “Questão Dinâmica”. Os elementos dinâmicos eram identificados no texto por meio da captura da posição inicial e final do caractere correspondente, que então eram armazenadas no banco de dados. Para viabilizar essa proposta, optou-se na época por identificar esses elementos dinâmicos considerando que deveriam sempre começar com os caracteres “<|” e terminar com “|>”. Inicialmente, essa abordagem foi validada e parecia bem estruturada. Contudo, à medida que o desenvolvimento avançava, surgiram desafios significativos para essa estratégia, o que levou à criação de uma nova abordagem para o módulo.

Após revisões sucessivas do módulo até alcançar a solução mais adequada, o que antes era denominado “Elemento Dinâmico” e atualmente é conhecido como “Variáveis”, evoluiu para um atributo que recebe dados no formato JSON contendo as configurações de domínio das variáveis. Esse atributo é associado à entidade “Questão Dinâmica”, que por sua vez foi renomeada como “Questão-Modelo”. Além disso, houve uma simplificação significativa para

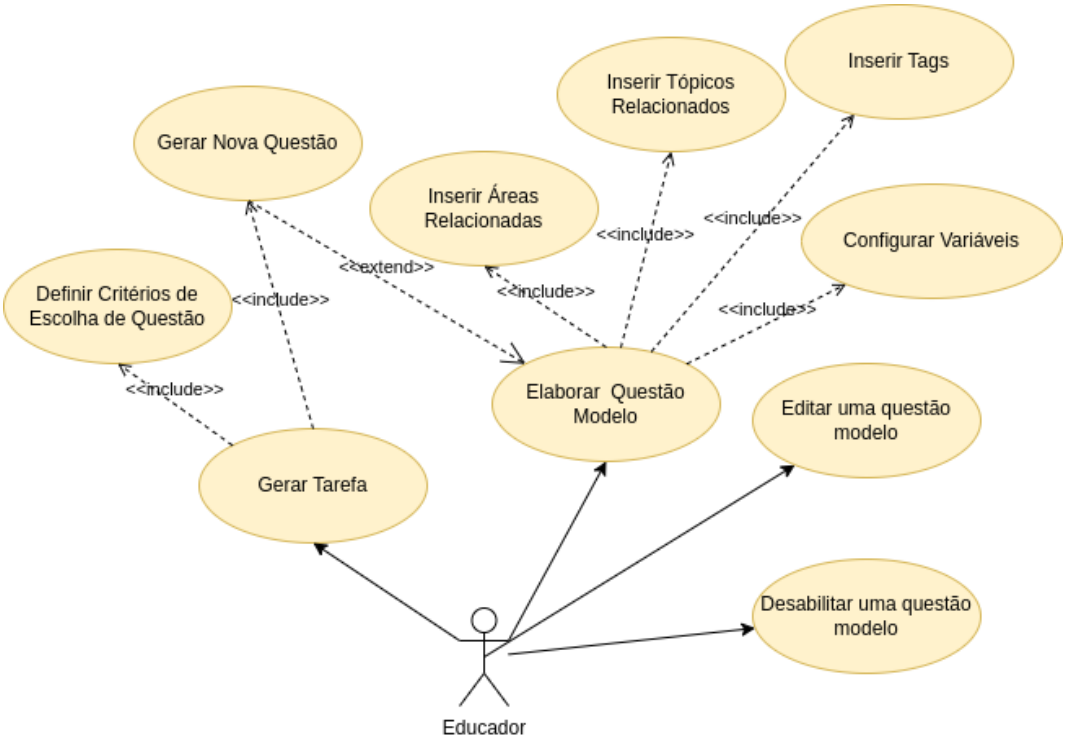


Figura 3.11 – Casos de Uso do Ator Educador
Fonte: Autoral (2023)

eliminar a necessidade de usar posições para armazenar a localização das variáveis no texto.

A Figura 3.13 exibe o Diagrama de Dados, destacando a estrutura final do módulo. Essa representação visual oferece uma solução para um problema identificado após uma parte significativa do desenvolvimento ter sido concluída. Vale notar a inclusão de uma tabela destinada a armazenar o histórico das questões. A cada vez que uma questão-modelo passava por edição, tornava-se necessário preservar o histórico das modificações para não comprometer as relações das questões com os alunos, as quais tiveram interações em algum momento. Essa abordagem foi adotada devido à perda de informações que ocorria durante a atualização de uma questão.

Ao examinar a Figura 3.13 com maior detalhe, é possível observar as tabelas individualizadas para cada metadado. Todas essas tabelas possuem relacionamentos que permitem receber um conjunto de metadados para uma questão-modelo específica. Comparando o Modelo de Entidade e Relacionamento da versão inicial do módulo com o Diagrama de Classes que apresenta a estrutura mais recente, torna-se evidente que os atributos associados às variáveis foram simplificados para conter apenas o nome da questão e um indicador de ativação. Essa nova adição, denominada “ativa”, foi introduzida para resolver um desafio relacionado à preservação das informações das questões. Optou-se por um método em que as questões permanecessem desativadas e nunca removidas, evitando a perda de informações. Essa abordagem foi implementada de maneira que, ao gerar uma tarefa em que o sistema selecionasse as questões, estas só seriam consideradas se estivessem marcadas como ativas no banco de dados. Consequentemente, a opção de remover uma questão foi eliminada do escopo do projeto.

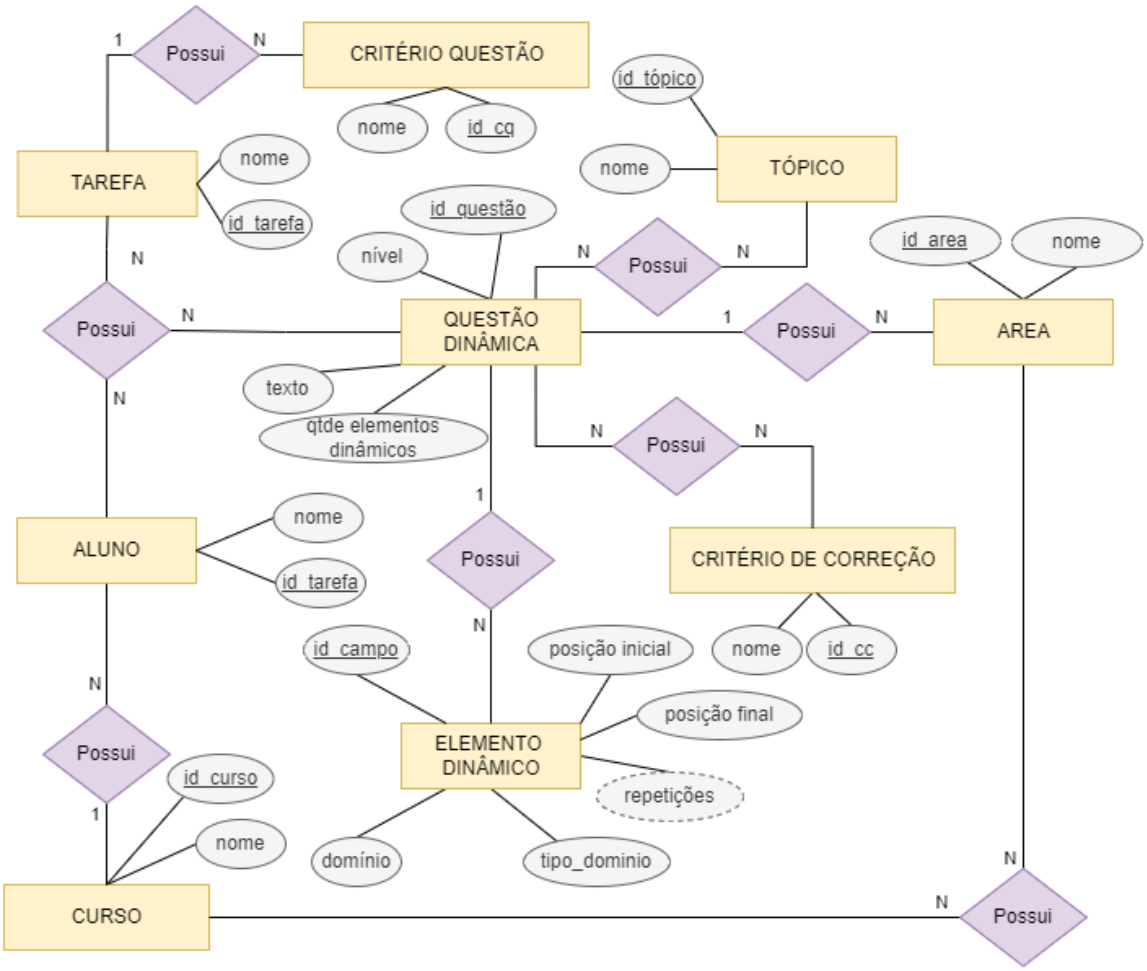


Figura 3.12 – Modelo de entidade e relacionamento
Fonte: Autoral (2023)

3.3 Banco de Dados

De acordo com Patrocínio (2023), a implementação do banco de dados já havia sido desenvolvida em um trabalho não concluído por outro aluno e que o autor deu continuidade. Nessa construção, o banco de dados não tinha conexão entre as tabelas com uso de chave estrangeira. Portanto, foi implementado um novo tipo de conexão de forma que utilizasse as chaves estrangeiras. Neste trabalho, o relacionamento das tabelas e seu diagrama com detalhes das colunas das tabelas foi resgatado através de uma engenharia inversa feita no script mysql do banco existente. Neste processo, as tabelas são esboçadas automaticamente através do MySQL Workbench e percebeu-se que os relacionamentos foram feitos entre tabelas com mecanismo de dados MySAM. Esse mecanismo é apropriado para tabelas com baixo volume de transações de dados e é eficaz para proteção de tabelas (MILANI, 2007). No entanto, para as tabelas utilizadas neste projeto, que abrangem todas as relações de gerenciamento de questões, optou-se pelo uso do mecanismo INNODB. O INNODB é mais recomendado para tabelas que requerem transações mais frequentes. Além disso, o INNODB é mais adequado para manter a integridade dos relacionamentos, uma vez que o mecanismo MyISAM não suporta chaves estrangeiras, tornando necessário criar os

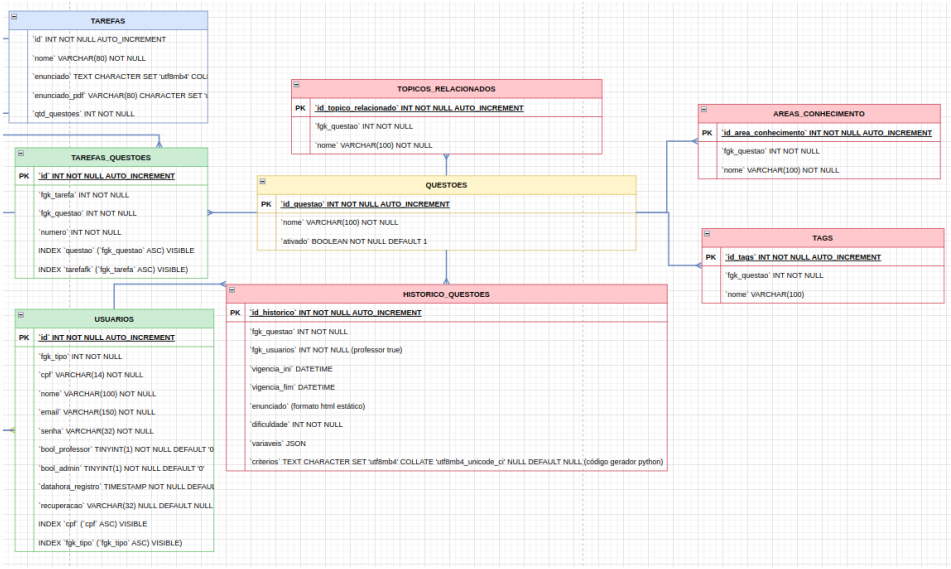


Figura 3.13 – Diagrama de Classes
Fonte: Autoral (2023)

relacionamentos de forma implícita.

Inicialmente, o banco de dados incluía um arquivo compactado no formato ZIP, conhecido como o "critério de correção". Durante a criação de uma questão, esse arquivo era carregado por meio de um processo de *upload*. Esse arquivo desempenhava um papel fundamental ao permitir que o corretor automático de questões do *opCoders Judge* executasse o processo de avaliação para a questão correspondente. No entanto, essa abordagem era adequada apenas para questões estáticas, e com a introdução do dinamismo na criação de questões, surgiu a necessidade de repensar a forma como o "critério de correção" era tratado. Para isso, o "critério de correção" passou a receber um arquivo Python que, no futuro, permitirá a configuração dinâmica dos critérios de correção. A fim de armazenar essa informação no banco de dados, foi escolhida a abordagem de armazenar apenas o script do código Python como um texto extenso. Portanto, quando é necessário apresentar o arquivo correspondente, o valor armazenado no banco de dados é utilizado no algoritmo para construir o arquivo Python, possibilitando ao usuário o download desse arquivo de maneira conveniente.

Para tornar o banco de dados mais consistente e menos esparso, uma avaliação crítica do rumo da implementação foi conduzida. Em uma etapa específica, a decisão foi tomar uma abordagem que envolvesse a criação de uma tabela para as variáveis, onde cada variável corresponderia a uma coluna individual e cada tipo de valor associado à variável teria sua própria coluna. Entretanto, essa abordagem resultou em uma tabela de variáveis excessivamente fragmentada, pois embora um valor específico pudesse ser inserido para um determinado tipo, os demais tipos permaneceriam com valores nulos. Esse cenário conduziu a uma situação em que a tabela continha mais valores nulos do que valores efetivamente preenchidos e relevantes. Logo, para remediar essa situação, a tabela de variáveis foi removida e as variáveis foram armazenadas como

uma coluna da tabela denominada “historico_questoes”, formatada em JSON. Explicando mais detalhadamente, as variáveis eram recebidas pelo banco de dados como uma entrada em JSON e passou a conter as informações do nome, tipo e valor de cada variável da questão-modelo. Por meio dessa abordagem, a consistência do banco de dados foi substancialmente aprimorada no que diz respeito ao tratamento das variáveis.

3.4 Operações Básicas em Questões

Para acessar o gerenciamento de questões-modelo, é imprescindível possuir uma conta de professor para obter privilégios adequados. As operações básicas para manipulação das informações armazenadas são a criação, leitura, atualização e remoção, conhecidas como CRUD (*Create, Read, Update, Delete*). Essas operações possibilitam que os usuários interajam efetivamente com o banco de dados (POLO; PIATTINI; RUIZ, 2001).

A fim de explicar cada uma dessas operações de forma abrangente, esta seção foi subdividida em subseções, sendo acompanhada por recortes de telas que auxiliam na ilustração. Na Subseção 3.4.0.1, é fornecida uma explicação detalhada sobre o processo de criação de uma questão-modelo. Em seguida, na Subseção 3.4.0.2, apresentamos como a lista de questões-modelo é visualizada no banco de dados e como funciona o mecanismo de busca por uma questão específica. Passando para a próxima operação, a Subseção 3.4.0.3 oferece um entendimento mais profundo sobre como ocorre a atualização de uma questão-modelo e explora mais detalhes acerca do histórico das questões, previamente delineado no Diagrama de Classes. Por fim, na Subseção 3.4.0.4, é apresentada uma alternativa para a exclusão de uma questão, juntamente com a definição de seu processo.

3.4.0.1 Criação de uma Questão-Modelo

Ao entrar na tela para adicionar uma questão, o editor de texto TinyMCE é inicializado e todas os campos para formar uma questão-modelo é disponibilizado. A Figura 3.14 proporciona uma representação visual dessa interface. O primeiro campo a ser preenchido refere-se ao nome da questão, uma etapa obrigatória. Imediatamente abaixo, encontra-se um botão que permite habilitar ou desabilitar a questão, conferindo a flexibilidade de mantê-la apta ou não a ser considerada em seleções para compor as tarefas destinadas aos alunos.

A seção para o enunciado ou texto da questão é implementada por meio do editor de texto TinyMCE, que proporciona um espaço de formatação para a construção da questão. É importante ressaltar que a utilização de variáveis não é obrigatória, porém torna-se imprescindível configurá-las na tabela de variáveis caso exista no decorrer do enunciado.

O botão de visualização permite ao professor gerar simulações de versões distintas da questão, fornecendo uma prévia do que os alunos encontrarão. Na sequência, os metadados

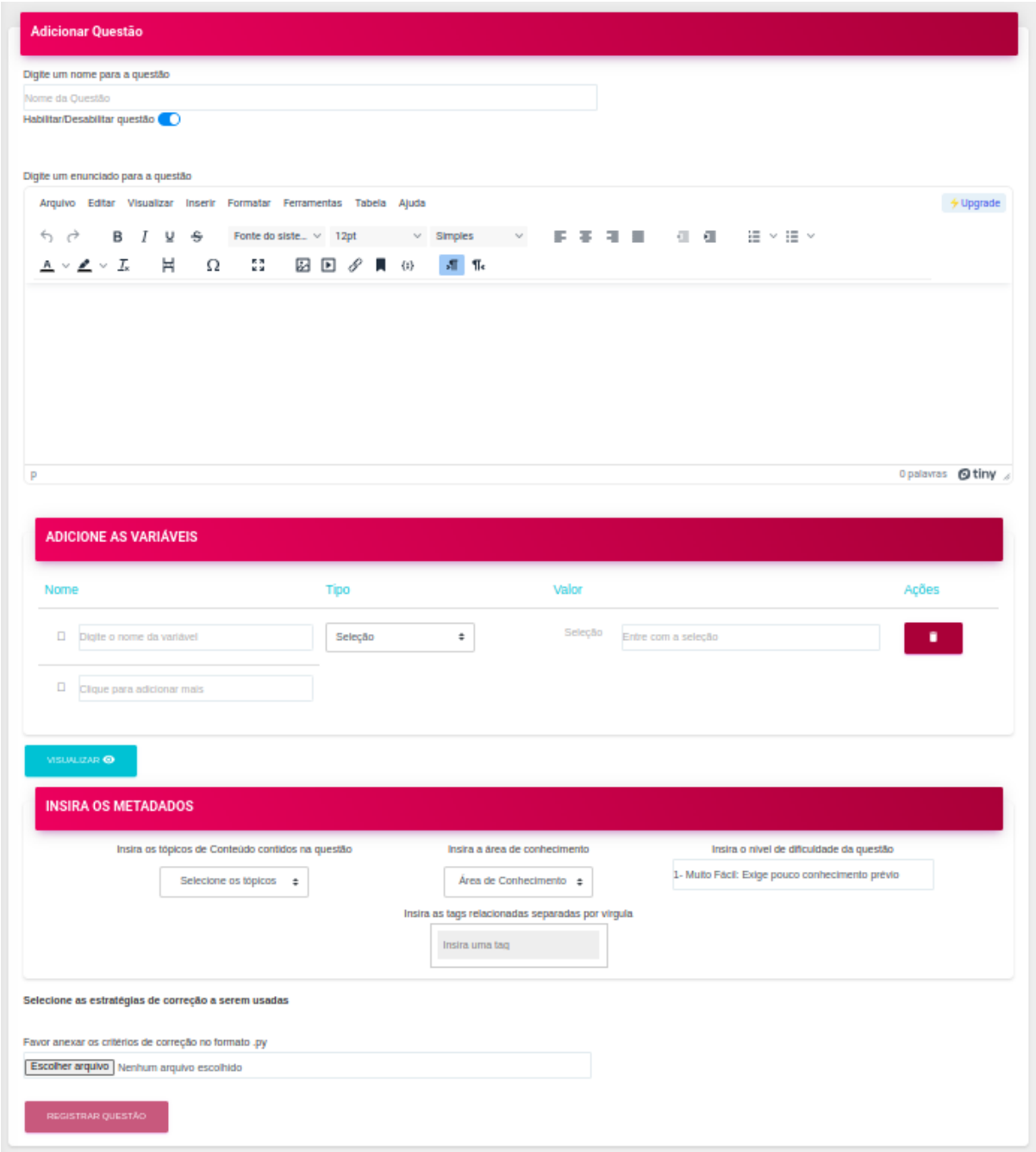


Figura 3.14 – Tela para Criar uma Questão
Fonte: Autoral (2023)

da questão podem ser inseridos por meio de campos de seleção e inserção, permitindo uma categorização precisa e facilitando a busca posterior.

Por fim, é indispensável que um arquivo com critério de correção em Python seja enviado. Após concluir a configuração da questão, o professor pode finalizar a operação ao clicar no botão "Registrar Questão", o que o redirecionará para a tela de lista de questões, onde poderá verificar o resultado da operação e avaliar a questão recém-criada em contexto com outras questões já existentes.

3.4.0.2 Leitura e Busca por Questões-Modelo

A interface retratada na Figura 3.15 apresenta uma tabela contendo as questões-modelo cadastradas. Na primeira coluna, é exibido o código correspondente a cada questão. A coluna subsequente apresenta o nome da questão. Cada linha é acompanhada de um botão "Visualizar"na coluna "Enunciado", permitindo que o professor acesse o texto completo da questão-modelo ao clicar nesse botão.

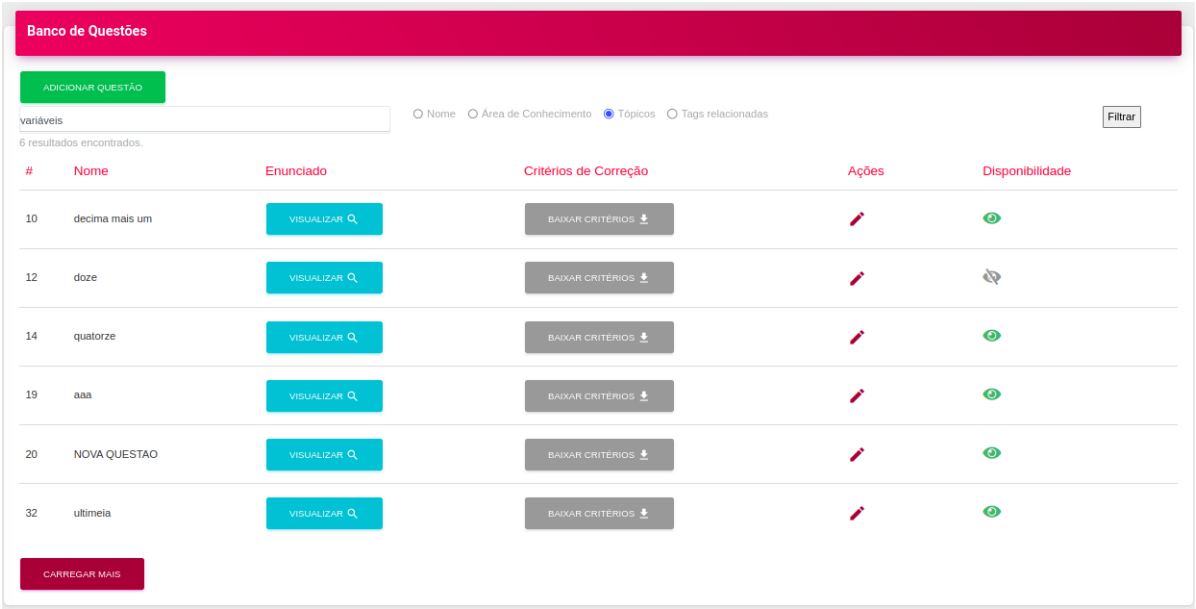


Figura 3.15 – Tela para Visualizar e Buscar as Questões-Modelo
Fonte: Autoral (2023)

Na coluna “Critérios de Correção”, encontra-se um recurso que possibilita o download do algoritmo em formato Python, o qual é responsável pela correção da questão-modelo relacionada. A coluna “Ações” oferece a opção de editar a questão-modelo, garantindo a flexibilidade de ajustes sempre que necessário. A coluna “Disponibilidade” é composta por um ícone indicativo que esclarece se a questão está habilitada para ser incluída em Tarefas ou não.

Ainda na Figura 3.15, é disponibilizado um campo de busca que permite a pesquisa por uma questão-modelo específica ou por um conjunto delas. Essa funcionalidade possibilita a pesquisa por nome da questão ou pelos metadados relacionados. O resultado da pesquisa é apresentado como uma lista de uma ou mais questões que contêm a chave de pesquisa parcial ou completa inserida pelo professor.

Para evitar uma sobrecarga visual com um grande volume de questões exibidas simultaneamente, foi incorporado o botão “Carregar mais”, permitindo ao usuário controlar a quantidade de questões exibidas por vez. Adicionalmente, logo abaixo do campo de inserção da chave de pesquisa, é fornecida uma informação quantitativa que indica o número de questões encontradas na busca, proporcionando uma noção imediata da amplitude dos resultados obtidos. Essa abordagem visa aprimorar a usabilidade e a experiência do usuário.

3.4.0.3 Atualização de uma Questão-Modelo

Após o cadastro da questão no banco de dados, é possível realizar edições nas mesmas informações previamente preenchidas durante a fase de criação da questão. A Figura 3.16 ilustra que os campos permanecem os mesmos da tela de criação, porém, o conteúdo da questão já é carregado automaticamente em seus respectivos campos.

Como mencionado anteriormente, este trabalho passou por aprimoramentos ao longo do desenvolvimento para abordar diversos problemas. Um desses desafios estava relacionado à possibilidade de a questão-modelo ser editada, mas perder a referência de sua relação com os usuários. Além disso, era crucial manter registrado no banco de dados o nome de usuário do professor responsável pela modificação da questão. Para solucionar esse problema, foi criada a tabela denominada “historico_questoes” no banco de dados. Portanto, sempre que ocorre uma edição em uma questão, uma nova entrada é adicionada a essa tabela, incluindo informações como o nome de usuário do professor e a data de modificação na coluna "vigencia_fim"da tabela mencionada. Essa abordagem garante um histórico preciso das alterações realizadas em questões.

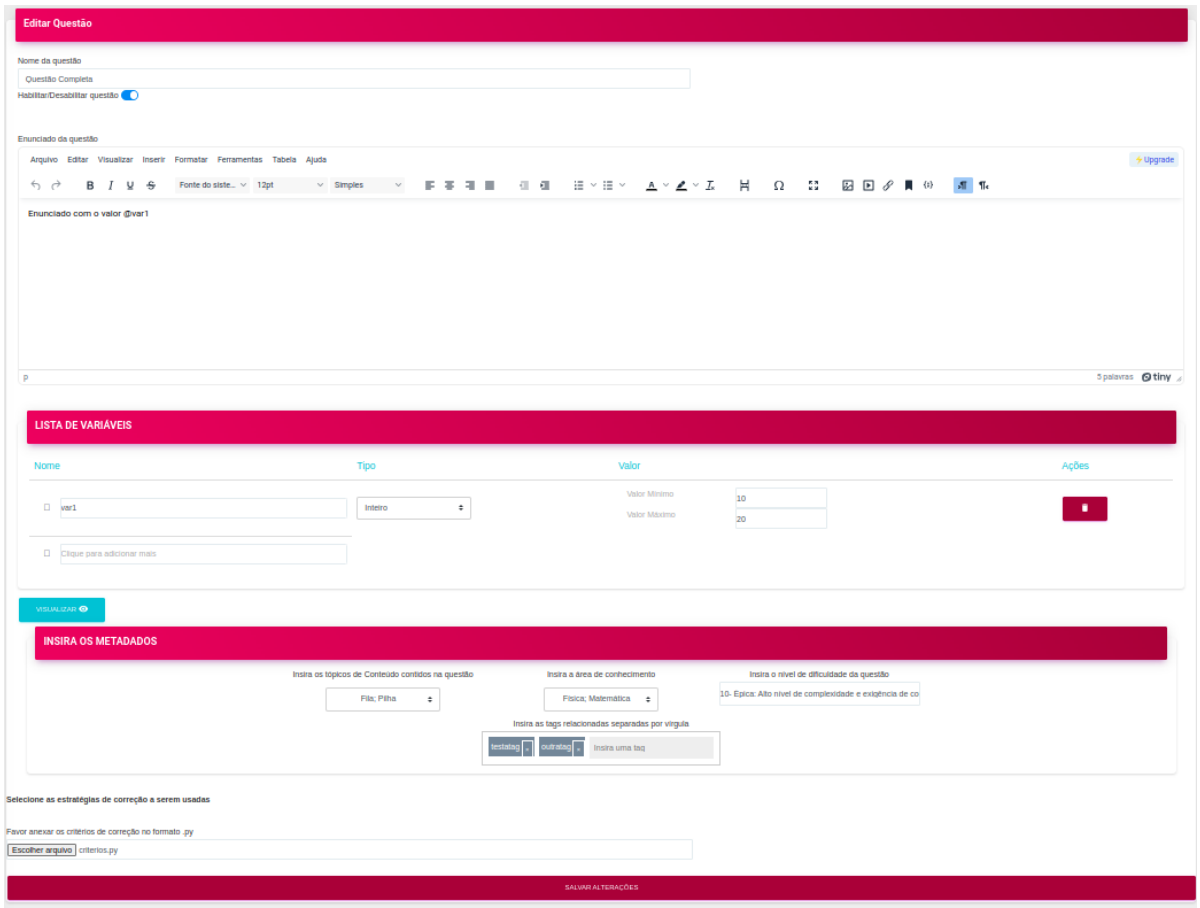


Figura 3.16 – Tela para Atualizar uma Questão-Modelo
Fonte: Autoral (2023)

3.4.0.4 Desabilitar uma Questão-Modelo

Inicialmente, a questão era considerada como uma informação que poderia ser removida do banco de dados sem maiores preocupações com possíveis perdas. No entanto, percebeu-se que ao remover uma questão-modelo, os usuários que já haviam interagido com ela poderiam perder acesso às informações associadas. Com o intuito de evitar essa situação, uma nova abordagem foi adotada para essa operação.

A opção de remoção completa da questão foi substituída pela funcionalidade de desabilitá-la. Em outras palavras, ao invés de remover uma questão, ela pode se tornar indisponível, de modo que não seja mais considerada para seleção em uma Tarefa. E como exemplificado anteriormente na Figura 3.15, é possível visualizar o status de uma questão, e para alterar esse status, é necessário acessar a tela de edição da questão. Ao efetuar essa alteração, a questão passa por uma revisão manual completa antes que a decisão de trocar o seu status seja confirmada. Dessa forma, é proporcionada uma oportunidade de consideração mais cuidadosa antes de alterar o estado da questão, contribuindo para a manutenção da integridade das informações.

4 Discussões

As pesquisas referenciadas neste trabalho foram substanciais para sustentar a viabilidade de desenvolvimento do módulo. Através do conjunto de informações encontrados na literatura, assegura-se uma maior consistência do módulo. Dito isto, compreende-se que foi estabelecida uma metodologia de aprendizado de programação em que possibilita ao educador acompanhar o crescimento de seu aluno, ou turma, na disciplina de introdução à programação de computadores.

Para compreender plenamente a transformação que o módulo de geração de questões trouxe ao *opCoders Judge*, é essencial fazer uma comparação. Inicialmente, a tela de criação de questões não dispunha de um editor de texto com recursos de formatação e exportação em HTML. O editor disponível era básico e trabalhava diretamente com HTML, cujo conteúdo era armazenado na tabela de questões do banco de dados. Além disso, a funcionalidade de edição de questões não estava implementada, visto que a questão não era buscada no banco de dados. Era permitido excluir questões anteriormente. No que diz respeito ao banco de dados, a inserção de novas questões era registrada na tabela "questões", que continha todas as colunas relacionadas à questão, sem contemplar uma tabela de histórico de questões para registrar informações como data de criação, autor da questão e assim criar uma forma segura de armazenamento de dados. As variáveis das questões não existiam, tampouco os recursos para criar uma questão-modelo. A inserção de metadados, incluindo a especificação do nível de dificuldade da questão, também não estava presente. Em decorrência disso, um sistema de busca por questões-modelo não havia sido desenvolvido. A incorporação do módulo de geração de questões trouxe mudanças fundamentais a esses aspectos, enriquecendo a funcionalidade do *opCoders Judge*.

A discussão acerca da associação de um nível de dificuldade a uma questão revelou um desafio significativo devido à complexidade em determinar o grau de complexidade apropriado. Diante disto, foi concluido que no primeiro momento de utilização do módulo em ambiente de produção, será interessante analisar como os professores definirão o nível de dificuldade para suas questões-modelo elaboradas. A partir dessa análise, será possível avaliar a eficácia do método atualmente empregado e, se necessário, propor aprimoramentos ou atualizações para esse recurso, visando a uma abordagem mais satisfatória.

Durante o processo de desenvolvimento, houve extensas discussões em relação à escolha do símbolo a ser utilizado para identificar as variáveis no texto. Inicialmente, optou-se por adotar o símbolo "<|" para marcar o início de uma variável e "|>" para indicar o seu término. Contudo, essa abordagem não seguia as convenções usuais de marcação em textos, o que poderia resultar em confusões e erros sintáticos por parte dos usuários ao redigirem as questões-modelo contendo variáveis. Além disso, a criação da expressão regular para capturar essas variáveis seria mais complexa, uma vez que envolveria o reconhecimento de fechamento de tags. Com o

decorrer da implementação, a discussão voltou à tona e o símbolo foi definido como “@”, como convencionalmente é utilizado em editores com o nome de “menção”.

5 Considerações Finais

Através de um estudo da literatura e de discussões em relação ao processo de implementação, foi desenvolvido um módulo capaz de gerar múltiplas variações de questões de programação a partir de um modelo de questão. Esse módulo foi devidamente integrado à plataforma *opCoders Judge* da UFOP, representando uma fase de testes e validação.

Com uma visão voltada para o futuro, espera-se que em breve a conclusão da implementação da tela de gerenciamento de tarefas e o ajuste do script dos critérios de avaliação resultem na funcionalidade plena do módulo, habilitando seu uso em um ambiente de produção. Destina-se a ser um recurso acessível a todos os professores envolvidos nas disciplinas introdutórias de programação, enriquecendo as experiências de aprendizagem e permitindo uma maior personalização e diversificação das avaliações.

A partir de estudos advindos da literatura e bastante discussão sobre o processo de implementação, foi desenvolvido um módulo capaz de criar versões de questões de programação a partir de uma questão-modelo. Este módulo se integrou à plataforma *opCoders Judge* da UFOP como ambiente de homologação. Espera-se que em breve a implementação da tela de gestão de Tarefas e o ajuste no script dos critérios de correção possam dar ao módulo deste trabalho seu funcionamento em ambiente de produção para uso de todos os professores das disciplinas introdutórias de programação.

5.1 Conclusão

A relevância de apresentar aos alunos questões inéditas e mais contextualizadas vai além de uma simples tarefa para avaliar conhecimento, por visar buscar maior compreensão dos conceitos abordados e com mais autonomia para o aluno que realiza a tarefa. Esse princípio, que almeja um aprendizado mais eficaz, encontra seu respaldo na literatura. Se destaca neste trabalho de monografia a importância da criação de questões únicas para otimizar o processo de aprendizagem.

A evolução do banco de dados por meio de reformulações constantes durante seu planejamento é uma história que ilustra a busca por consistência e eficiência. Foi possível alcançar um banco de dados menos complexo, menos esparso e mais consistente.

A importância da modelagem conceitual revelou-se fundamental para a concepção do módulo. Através dela, foi possível visualizar a estrutura de dados de maneira coerente.

Portanto, fica evidente a riqueza da colaboração entre conhecimento teórico e prático. O êxito na criação de mais um módulo para o *opCoders Judge* enriquece a experiência educacional ao proporcionar variedade e personalização nas questões.

5.2 Trabalhos Futuros

Uma meta importante para ser considerada como trabalho futuro é a criação de critérios de correção dinâmicos. Ao personalizar os critérios de correção de acordo com cada questão, o módulo se adaptará precisamente aos diferentes tipos de questões propostas. No momento deste trabalho, o critério de correção está estático.

Entre os objetivos prioritários, destaca-se a criação da parte do módulo que permitirá a geração de Tarefas. Com essa funcionalidade, os professores poderão criar listas personalizadas de questões para seus alunos.

Um caminho a ser explorado é aprimorar a contextualização das questões, aproximando-as da realidade do aluno.

A integração de um analisador morfológico poderá ser útil também para o momento em que o professor redige uma questão.

A implementação de testes, incluindo testes unitários e de aceitação, se apresenta como uma tarefa importante. Essa abordagem garantirá que todas as partes do módulo estejam funcionando harmoniosamente.

A criação de um sistema de busca mais eficaz para a lista de questões proporcionaria aos professores uma maneira mais eficiente de encontrar e gerenciar as questões-modelo, facilitando o processo de criação de Tarefas.

Referências

BARBOSA, A.; COSTA, E.; BRITO, P. Uma abordagem adaptativa para gerar agrupamento de códigos em disciplinas de programação introdutória. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2017. v. 28, n. 1, p. 1427.

BEECROWD. *Beecrowd Platform*. 2023. Disponível em: <https://www.beecrowd.com.br/academic/>. Acesso em: 15 mar. 2023.

BENTO, E. J. *Desenvolvimento web com PHP e MySQL*. [S.l.]: Editora Casa do Código, 2021.

BEZ, J. L.; TONIN, N. A.; RODEGHERI, P. R. Uri online judge academic: A tool for algorithms and programming classes. In: IEEE. *2014 9th International Conference on Computer Science & Education*. [S.l.], 2014. p. 149–152.

BLOOM, B. S. Learning for mastery. instruction and curriculum. regional education laboratory for the carolinas and virginia, topical papers and reprints, number 1. *Evaluation comment*, ERIC, v. 1, n. 2, p. n2, 1968.

BRITO, P. S. S. O uso de ferramentas computacionais para o ensino de programação para alunos de engenharia. In: . [S.l.]: Departamento de Computação, Universidade Federal de Ouro Preto, 2019.

CAMPOS, C. de. Boca: um sistema de apoio a competiç oes de programaç ao. 2004.

CONVERSE, T.; PARK, J. *PHP: a bíblia*. [S.l.]: Gulf Professional Publishing, 2003.

CORNEY, M.; TEAGUE, D.; THOMAS, R. Engaging students in programming. In: AUSTRALIAN COMPUTER SOCIETY. *Proceedings of the Twelfth Australasian Computing Education Conference*. [S.l.], 2010. p. 63–72.

FLANAGAN, D. *JavaScript: o guia definitivo*. [S.l.]: Bookman Editora, 2012.

FREIRE, P. *Pedagogia do Oprimido*. [S.l.]: Paz e Terra, 1987. I.

FROSSARD, A. C. *Conhecendo a Pedagogia da Alternância: contextualização, questões teóricas e práticas*. [S.l.]: Pragma Livros, 2018.

GENCI, J. Methods to ensure higher variability of knowledge tests in the moodle lms environment. In: SOBH, T.; ELLEITHY, K. (Ed.). *Emerging Trends in Computing, Informatics, Systems Sciences, and Engineering*. New York, NY: Springer New York, 2013. p. 447–453. ISBN 978-1-4614-3558-7.

GOMES, A.; MENDES, A. A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations. In: IEEE. *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. [S.l.], 2014. p. 1–8.

GOOGLE. *Google Docs*. 2023. Disponível em: <https://www.google.com/intl/pt-BR/docs/about/>. Acesso em: 11 jan. 2023.

IKEMATU, R. S. Gestão de metadados: sua evolução na tecnologia da informação. *DataGramaZero-Revista de Ciência da Informação*, v. 2, n. 6, 2001.

JÚNIOR, J. R. d. L. Desafios, dificuldades e incertezas no trabalho do professor universitário: Estudo de caso em uma universidade pública no interior do estado de são paulo. Universidade Estadual Paulista (UNESP), 2019.

KOLLMANSBERGER, S. Helping students build a mental model of computation. In: *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*. [S.l.: s.n.], 2010. p. 128–131.

MILANI, A. *MySQL-guia do programador*. [S.l.]: Novatec Editora, 2007.

PATROCÍNIO, J. A. d. Opcoders judge: uma versão online para o corretor automático de exercícios de programação do projeto opcoders. 2023.

PITEIRA, M.; HADDAD, S. R. Innovate in your program computer class: an approach based on a serious game. In: *Proceedings of the 2011 Workshop on Open Source and Design of Communication*. [S.l.: s.n.], 2011. p. 49–54.

POLO, M.; PIATTINI, M.; RUIZ, F. Reflective persistence (reflective crud: reflective create, read, update and delete). In: UNIVERSITÄTSVERLAG KONSTANZ GMBH IRSEE, GERMANY. *Sixth European Conference on Pattern Languages of Programs (EuroPLOP)*. [S.l.], 2001. p. 69–85.

ROBINS, A.; ROUNTREE, J.; ROUNTREE, N. Learning and teaching programming: A review and discussion. *Computer science education*, Taylor & Francis, v. 13, n. 2, p. 137–172, 2003.

SANTOS, P. dos; CARVALHO, L. S. G. de; OLIVEIRA, E.; FERNANDES, D. Classificação de dificuldade de questões de programação com base na inteligibilidade do enunciado. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2019. v. 30, n. 1, p. 1886.

SILVA, M. S. *HTML5: a linguagem de marcação que revolucionou a web*. [S.l.]: Novatec Editora, 2019.

SPOJ. *Tutorial for problem-setters*. 2023. Disponível em: <https://www.spoj.com/tutorials/PS>. Acesso em: 15 jul. 2023.

TINYMCE. *TinyMCE*. 2023. Disponível em: <https://www.tiny.cloud/docs/tinymce/6/introduction-to-tinymce/>. Acesso em: 11 ago. 2023.

VEROU, L. *CSS Secrets: Better Solutions to Everyday Web Design Problems*. [S.l.]: "O'Reilly Media, Inc.", 2015.

VIER, J.; GLUZ, J.; JAQUES, P. A. Empregando redes bayesianas para modelar automaticamente o conhecimento dos alunos em lógica de programação. *Revista Brasileira de Informática na Educação*, v. 23, n. 02, p. 45, 2015.

WIRAWAN, I. M.; TAUFANI, A. R.; WAHYONO, I. D.; FADLIKA, I. Online judging system for programming contest using um framework. In: *2017 4th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*. [S.l.: s.n.], 2017. p. 230–234.