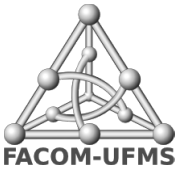
Fundação Universidade Federal de Mato Grosso do Sul

Faculdade de Computação

Processo de Desenvolvimento do Sistema de Ensino Gamificado

Leonardo de Souza e Silva Ramos



Campo Grande - MS

2026

Fundação Universidade Federal de Mato Grosso do Sul

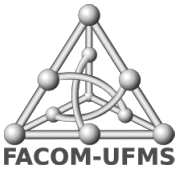
Faculdade de Computação

Leonardo de Souza e Silva Ramos

Processo de Desenvolvimento do Sistema de Ensino Gamificado

Trabalho de Conclusão de Curso submetido à uma banca da Faculdade de Computação para a obtenção do título de Bacharel em Sistemas de Informação.

Orientadora: Prof. Dra Ana Karina D. Salina de Oliveira



Campo Grande - MS

2026

Todo grande feito deve ter um começo, mas é a sua continuação até o fim, até que seja totalmente concluído, que traz a verdadeira glória.

(Sir Francis Drake, 1587)

Resumo

A gamificação aplicada ao ambiente educacional tem se mostrado uma estratégia eficaz para aumentar o engajamento e a retenção de conhecimento por parte dos estudantes. Este trabalho apresenta o desenvolvimento do Sistema de Ensino Gamificado (SEG), uma aplicação *web* concebida a partir de uma demanda prática identificada na Faculdade de Odontologia (FAODO) da Universidade Federal de Mato Grosso do Sul (UFMS). O projeto surge como uma alternativa gratuita e customizada a plataformas existentes de questionários educacionais, como o *Kahoot!*, cujos custos operacionais são atrelados ao dólar e as limitações de armazenamento de arquivos inviabilizam o uso contínuo no cotidiano universitário. A metodologia de desenvolvimento abrangeu o levantamento de requisitos, prototipação de interfaces e modelagem arquitetural baseada no Modelo C4. O sistema foi construído sob o padrão de arquitetura *Model-View-Controller* (MVC), utilizando HTML, CSS e *JavaScript* no *front-end* para a criação de telas dinâmicas adaptadas aos perfis de professores e alunos. O *back-end* foi estruturado em *Node.js*, integrado a um banco de dados relacional *MySQL* hospedado localmente no sistema Ubuntu. Como resultado, o SEG consolidou-se como uma ferramenta robusta, acessível e alinhada às necessidades pedagógicas institucionais, apta a dinamizar o processo de ensino-aprendizagem na UFMS.

Palavras-chave: Gamificação. *Node.js*. Algoritmo Jaro-Winkler. Banco de Dados Relacional. Ensino Superior. Modelo C4. UFMS.

Sumário

1. Introdução

O mundo vivencia constantes mudanças, frequentemente impulsionadas pela tecnologia aplicada às mais diversas áreas. Esse cenário traz inovações, novas possibilidades e ganhos de produtividade antes inimagináveis. A Tecnologia da Informação (TI) tem passado por um processo de grande expansão, estendendo-se a vários setores, inclusive àqueles que anteriormente relutavam em adotá-la. Atualmente, inovações tecnológicas podem ser encontradas em todos os lugares, desde o calendário de promoções de um supermercado até o fornecimento de acesso a novos recursos de comunicação, trabalho e ensino (Laudon, 2014).

Os métodos de ensino, desde a educação infantil até o ensino superior, têm sido cada vez mais apoiados e potencializados por sistemas informatizados, serviços web e aplicativos. Tais recursos estão presentes na vida de alunos e professores, integrando a rotina de estudos e trabalho por meio de sistemas de formatação de documentos acadêmicos, plataformas de disponibilização de materiais e avaliações, ou tecnologias que tornam o ensino mais interativo a partir de diferentes abordagens (Bacich e Moran, 2018). Nesse contexto, destaca-se o conceito de gamificação (do inglês, *gamification*), uma estratégia que utiliza a mecânica e o design de jogos para tornar o processo de aprendizagem mais dinâmico e atrativo, facilitando a compreensão e a fixação de um conteúdo didático (Busarello, 2016). O termo *gamificação* foi criado em 2002 pelo programador Nick Pelling, mas foi apenas na década de 2010 que a prática passou a ser amplamente estudada no meio acadêmico e corporativo. Em sua definição mais consolidada, proposta por Deterding et al. (2011), a gamificação consiste na utilização de elementos de *design* de jogos em contextos que não são jogos.

Um sistema gamificado oferece a alunos e professores dinâmicas como questionários interativos, nos quais os estudantes são estimulados a pensar de forma ágil e a interagir tanto com os colegas quanto com os conteúdos ministrados em aula. A resolução dessas atividades gera pontuações e *feedbacks* imediatos, o que estimula o aprendizado, corrige eventuais falhas de raciocínio e promove a interação entre os discentes. Além disso, possibilita aos docentes uma percepção rápida sobre a absorção do conteúdo pelas turmas. Diante desse contexto, este trabalho propõe a implementação do Sistema de Ensino Gamificado (SEG), desenvolvido para fornecer tais recursos a toda a comunidade acadêmica da Universidade Federal de Mato Grosso do Sul (UFMS).

Projetado em colaboração com os docentes da instituição, o SEG busca oferecer uma experiência de uso intuitiva e acessível, disponibilizando ferramentas que auxiliem tanto o aprendizado quanto o processo de ensino.

Além dos aspectos funcionais relacionados à gamificação, o desenvolvimento do SEG considera requisitos fundamentais de segurança da informação e segurança de software. Como o sistema armazena dados de autenticação, informações acadêmicas e resultados de atividades realizadas pelos usuários, torna-se essencial adotar mecanismos que garantam a confidencialidade, a integridade e a disponibilidade dessas informações. Entre as medidas implementadas destacam-se o armazenamento seguro de senhas por meio de funções criptográficas de hash, a utilização de controle de acesso baseado em perfis de usuários para restringir funcionalidades conforme o nível de privilégio, a validação de entradas para prevenção de falhas e ataques, o tratamento adequado de erros e exceções para evitar exposição de informações sensíveis e a adoção de práticas de desenvolvimento seguro voltadas à proteção dos dados dos usuários. Dessa forma, o sistema busca atender não apenas aos requisitos pedagógicos, mas também às boas práticas de segurança recomendadas para aplicações web modernas.

1.1 Objetivos

Este trabalho tem como objetivo documentar as etapas de desenvolvimento de um *software* de gamificação educacional e disponibilizá-lo gratuitamente para a comunidade acadêmica da Universidade Federal de Mato Grosso do Sul (UFMS). Busca-se, com isso, fornecer uma ferramenta tecnológica que auxilie no aprimoramento do processo de ensino-aprendizagem de alunos e professores, consolidando-se como um recurso didático de apoio para alunos e professores. Ambientes de hospedagem para sistemas *web* (como *AWS* e *Digital Ocean*) possuem custos na faixa de 20 a 35 reais por mês, o que oneraria a universidade, mas existem opções gratuitas, como o *Oracle Cloud* e o *Render,* que são o foco de utilização para garantir a gratuidade pretendida.

2. Trabalhos relacionados

2.1 Gamificação no Ensino de Programação de Computadores em turmas do Ensino Médio: uma experiência com o software Kahoot!

Madureira *et al.* (2021) identificaram grande dificuldade por parte dos estudantes para compreender conceitos de lógica e algoritmos, visto que os mesmos também já tinham problemas em lidar com a grade curricular do ensino médio. Diante disso, encontrou-se um espaço para aplicação de ensino gamificado, para motivar os alunos e criar uma forma de aprendizado mais acessível. Usando o sistema Kahoot!, foram criados questionários com conteúdo pertinente ao currículo escolar, testando e orientando estudantes a respeito de conhecimentos básicos de um programa (entrada, saída, processamento), problemas algorítmicos ou matemáticos, lógica booleana e fluxo de condicionais (*if, else*). Ao final do estudo, foi identificado um grande aumento na motivação e no engajamento de estudantes no ensino, com ideia de interação e competição sendo um fator de descontração. Os autores concluíram que o Kahoot! foi uma forma lúdica e eficaz de avaliar a aprendizagem. Os professores conseguiram mapear conceitos em que a turma estava errando mais, sem fazer uso de testes tradicionais que geram mais pressão psicológica.

2.2 Gamificação no Processo de Aprendizagem: Desenvolvimento e Aplicação de uma Plataforma Lúdica Educacional

No trabalho de Gomes (2025), não se buscou utilizar um sistema existente, mas sim desenvolver uma solução própria e documentar todo o processo, desde a concepção da ideia até a conclusão, buscando cumprir o objetivo de preencher lacunas deixadas por softwares de gamificação já existentes. O estudo se diferencia pela contribuição da detalhada fase de modelagem, apresentando uma definição rigorosa dos requisitos necessários para atender aos usuários. Foram especificadas restrições de desempenho, segurança do banco de dados e usabilidade no dispositivo móvel. O autor estabeleceu o que era essencial, importante e o que era desejável para o funcionamento do sistema.

2.3 Quizzes Gamificados como Recursos Tecnológicos no Ensino-Aprendizagem de Programação

O estudo desenvolvido por Sousa (2023) se baseou em algo recorrente em cursos de computação em universidades e institutos federais: os altos

índices de reprovação e evasão nas disciplinas iniciais de programação. A autora identificou que muitos estudantes não conseguem perceber, no início de seus cursos, qual a importância desse conhecimento para as suas carreiras. Destaca-se também destaca que aprender programação exige a construção de um pensamento abstrato, e que as aulas dessa área tendem a ser muito teóricas, o que dificulta a retenção da atenção da geração atual. A pesquisa se diferencia por buscar capacitar os professores antes dos alunos, o que foi realizado de forma remota através de uma oficina que abordou o que é gamificação, a comparação entre diferentes plataformas de *quizzes* e a construção de perguntas estimulantes. Após a aplicação da gamificação nas disciplinas iniciais de programação, foi constatado que os estudantes foram cativados e fortemente estimulados pelas competições amigáveis geradas pelos *quizzes*, o que incentivou a dedicação a estas "listas de exercícios". O uso dos *games* de ensino também serviu como uma poderosa avaliação formativa, permitindo aos docentes visualizar quais conceitos básicos a turma inteira estava errando e possibilitando um melhor direcionamento do conteúdo nas aulas.

2.4 Fundamentos da Gamificação: Origem, Conceitos e Benefícios Educacionais

A proposta de inserir a gamificação no processo educacional surgiu como uma resposta direta aos modelos tradicionais de ensino, que frequentemente enfrentam altos índices de desmotivação e passividade por parte dos alunos. Segundo Kapp (2012), a gamificação propõe uma mudança de paradigma ao engajar os estudantes através da motivação intrínseca e extrínseca. A teoria se apoia no fato de que os jogos estimulam a resolução de problemas de forma contínua e fornecem feedback imediato às ações do usuário. Os ganhos proporcionados pela adoção dessa estratégia são múltiplos. Kapp (2012) destaca que a gamificação cria um ambiente seguro para o erro, onde a falha deixa de ser um fator punitivo e passa a ser compreendida como uma etapa natural do aprendizado e da progressão. Além disso, a subdivisão de conteúdos complexos em missões ou metas menores gera pequenas recompensas que mantêm o foco e a resiliência do aluno. Consequentemente, observa-se não apenas um aumento no tempo de retenção e na participação ativa, mas também uma melhora significativa na assimilação do conhecimento abordado.

3. Requisitos do Sistema

3.1 Visão Geral do Sistema

O Sistema de Ensino Gamificado (SEG) foi desenvolvido para a Faculdade de Odontologia (FAODO) para aplicação no ensino em diversas áreas do conhecimento. O sistema permite que os docentes da instituição criem questões no formato de jogos, de forma incentivar a interação dos discentes e estimular o raciocínio e a construção do conhecimento. O SEG possui um módulo de pontuação, que registra o *ranking* de cada usuário, incentivando a competição entre os alunos. O ambiente virtual é estruturado a partir de dois perfis de usuários: professores e alunos. Os professores podem criar as questões e os questionários, enviar imagens e textos e definir as respostas certas. Os alunos podem responder as perguntas, interagir com outros e acompanhar suas respectivas pontuações.

É importante destacar que, apesar de o SEG ter sido desenvolvido inicialmente para atender às demandas específicas da FAODO, trata-se de um sistema abrangente, versátil e aplicável a diferentes áreas do ensino.

3.2 Versões do Sistema

Nesta seção, são descritas as duas interfaces de acesso do SEG, que estruturam o Sistema de Ensino Gamificado para diferentes contextos de uso:

Versão para Computador (*Web*): Acessível via navegador *web*, esta versão contempla todas as funcionalidades do sistema. É voltada, prioritariamente, para uma abordagem administrativa e para o uso do perfil docente (professores), embora também forneça acesso aos alunos, respeitando suas respectivas restrições de permissão.

Versão para Dispositivos Móveis (*Mobile*): Acessível por meio do aplicativo do navegador *web*, esta versão é direcionada principalmente ao uso dos discentes. Seu objetivo é disponibilizar as funcionalidades necessárias para que os alunos interajam com os recursos lúdicos criados pelos professores, podendo, da mesma forma, ser acessada pelo corpo docente.

3.3 Classes de usuários

Esta seção descreve as duas classes de usuários relevantes para o funcionamento do sistema (Ramos, 2026). A Figura 1 apresenta o Diagrama de Casos de Uso, no qual ficam ilustrados os perfis "Professor" e "Aluno", bem como o nível de acesso de cada um às funcionalidades da plataforma.

Professor: Possui permissões para criar questões, visualizar o *ranking*, gerenciar questionários e respondê-los, com o intuito de simular a experiência dos alunos. Como restrição de acesso, um docente não possui privilégios para editar as informações cadastrais de outro professor.
Aluno: Qualquer estudante de odontologia da UFMS pode ser cadastrado como um usuário do sistema. Suas permissões de acesso incluem responder às questões, comparar pontuação com a de outro aluno e visualizar seu *ranking* de pontos.

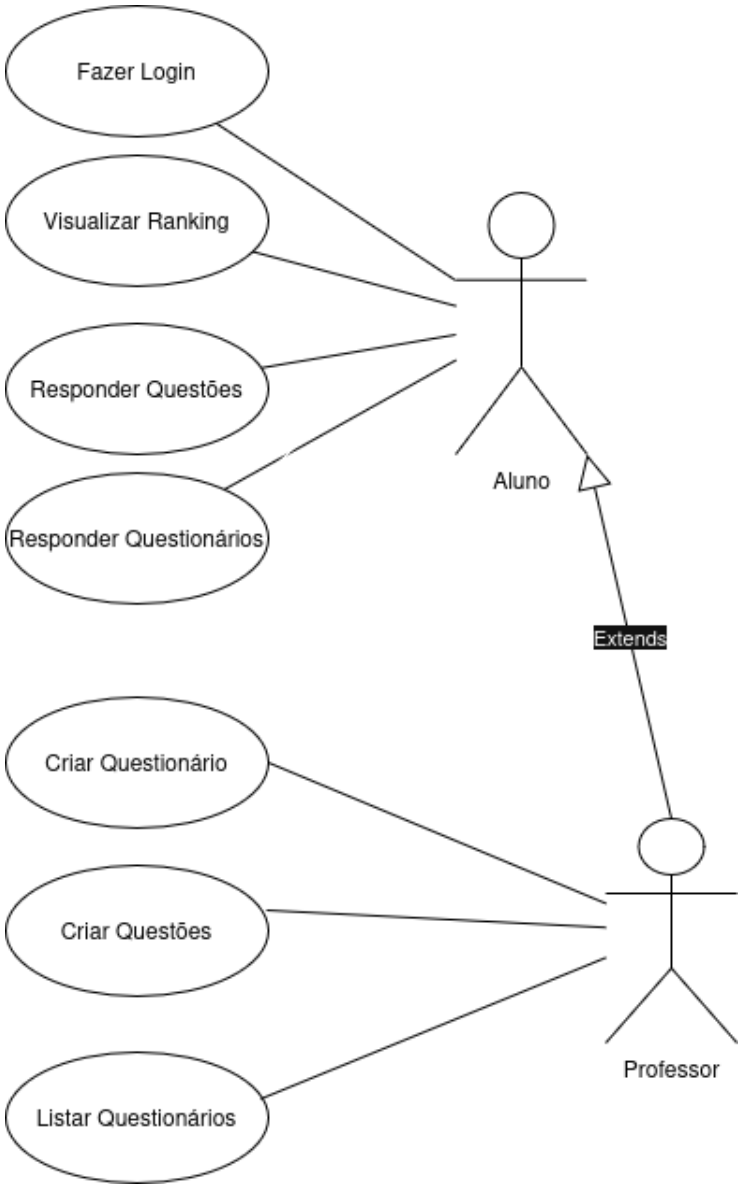


Figura 1. Diagrama de Caso de Uso

3.4 Requisitos de Software

Nesta seção são descritos os requisitos textuais do produto. Na Seção 3.4.1 são descritos os requisitos funcionais e na Seção 3.4.2 são descritos os requisitos não-funcionais.

3.4.1 Requisitos funcionais

Controle e nível de acesso para o Local:
RF-1. O sistema deve exigir que os usuários façam login.
RF-2. O sistema deve permitir que professores e alunos tenham fluxos de tela e uso geral do sistema completamente separados.
RF-3. O sistema deve permitir que professores criem questões de quatro tipos: múltipla escolha, análise de imagem, questão aberta e verdadeiro ou falso.
RF-4. O sistema deve permitir que professores criem questionários, sendo estes limitados a cinco questões cada.
RF-5. O sistema deve permitir que professores utilizem as funções de responder questionário e responder questões.
RF-6. O sistema deve emitir mensagem de erro em caso de acesso negado a usuários.
RF-7. O sistema deve permitir ao aluno responder questões (com ou sem questionário) e visualizar o ranking de pontuação.

Comprovantes e relatórios:
RF-8. O sistema deve permitir a listagem de todos os questionários já criados.
RF-9. O sistema deve permitir a listagem de todas as questões criadas pelo professor que está logado.
RF-10. O sistema deve possibilitar a geração de um ranking de pontuação dos usuários.

3.4.2. Requisitos não-funcionais

Desempenho:
RNF-11. O sistema deve salvar as informações no banco de dados imediatamente após a conclusão de questões e questionários.
RNF-12. O sistema deve permitir o acesso simultâneo de vários usuários.
RNF-13. O sistema deve permitir o armazenamento de imagens de alta qualidade, preferivelmente de até 50 *megabytes*.

Confiabilidade:
RNF-14. Deve haver um ou mais backups do sistema disponíveis em caso de falha.

Disponibilidade:
RNF-15. O sistema deve funcionar de forma estável, mantendo-se online sempre que possível e não ficando indisponível por mais de 30 minutos.

Segurança:
RNF-16 Todas as senhas do usuário devem ser armazenadas de forma segura usando funções hash.
RNF-17. O sistema deve ser capaz de diferenciar permissões de acesso e identificação para diferentes tipos de usuário.
RNF-18. O sistema deve funcionar apenas para alunos da UFMS.

Portabilidade:
RNF-19. O sistema deve ser capaz de ser executado em navegadores, como Chrome e Firefox, e em dispositivos móveis através dos sistemas operacionais Android e IOS.

4. Recursos e Bibliotecas

4.1 Recursos Utilizados

Para o desenvolvimento do Sistema de Ensino Gamificado (SEG), foram selecionadas tecnologias adequadas à implementação do *software*. É válido destacar que algumas dessas ferramentas também foram adotadas nos trabalhos relacionados que foram mencionados na seção 2. A seguir, detalham-se os recursos utilizados:

- *Front-end*: HTML5 (W3Schools, 2026) e CSS3 (W3Schools, 2026) para a estruturação e a estilização das interfaces gráficas. A linguagem *JavaScript* (Flanagan, 2021) foi empregada para dar dinamismo às páginas e tratar requisitos visuais, viabilizando interações que tornam a navegação mais intuitiva e amigável para os usuários.
- *Back-end* e Ambiente de Desenvolvimento: *JavaScript* com *Node.js* (2026) para a estruturação do servidor e o tratamento das requisições, utilizando o *framework Express.js* (2026) para a criação da *API REST*. Para o ambiente de desenvolvimento, fez-se uso da extensão *Live Server* (Dey, 2017), que fornece um servidor *web* local(Simulação de um servidor real, com protocolo HTTP) que funciona dentro do *Visual Studio Code* (Microsoft, 2015), hospedado na porta 5500. Essa ferramenta realiza a atualização automática sempre que são feitas mudanças no código e proporciona o uso de recursos extras do navegador, como local storage, visto que estes recursos normalmente seriam bloqueados no ambiente local. Esta extensão também oferece a possibilidade de acessar o site hospedado no servidor web simulado através de outros dispositivos que estejam conectados à mesma rede de internet.
- Editores de texto: Para a grande maioria do trabalho, foi utilizado o Visual Studio Code, por sua praticidade e enorme gama de recursos. O *Sublime Text* foi empregado secundariamente, para auxiliar em testes específicos e na realização de *backups* dos arquivos.
- Prototipação: O design da interface e dos elementos visuais da plataforma educacional foi prototipado utilizando uma ferramenta de design gráfico online e intuitiva (Canva, 2013).
- Criação de artefatos: Os diagramas e modelos do projeto foram elaborados utilizando o *software Astah*, da *Change Vision* (2009), e a plataforma *online Diagrams.net* (JGraph, 2012).

- Banco de dados: O sistema de gerenciamento de banco de dados relacional escolhido foi o *MySQL*, da Oracle (1995), em virtude de sua leveza e de seus recursos. Inicialmente, o *MySQL* foi operado via *Docker*, integrado à interface *phpMyAdmin*. Também foi realizada uma instalação local direta no sistema operacional *Ubuntu* (Canonical, 2004), sendo esta última a configuração escolhida em definitivo para hospedar o banco de dados do sistema e realizar a conexão via portas lógicas. A Figura 2 apresenta o Diagrama do Banco de Dados do projeto, gerado a partir do *MySQL* (Oracle, 1995). Nela, estão ilustradas as tabelas *aluno*, *professor*, *ranking*, *questoes*, *questionarios* e *alternativa_questao*, bem como seus respectivos campos e relacionamentos.

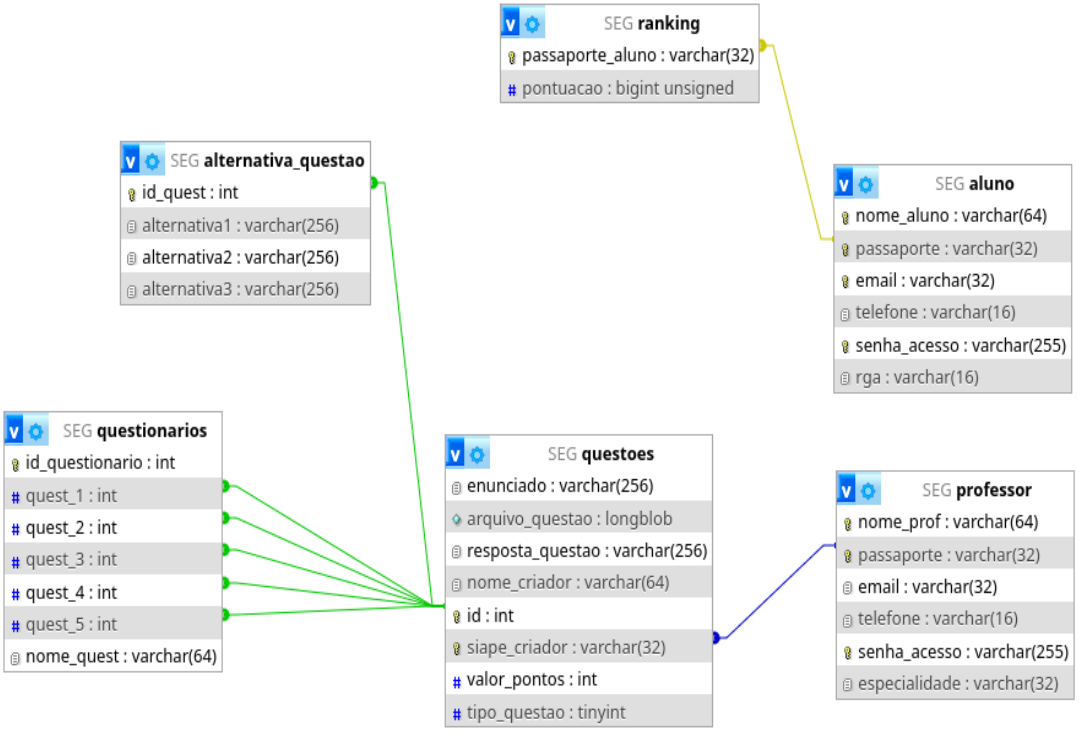


Figura 2. Diagrama do Banco de dados

4.2 Bibliotecas Utilizadas

Para o desenvolvimento do SEG, foi necessário utilizar algumas bibliotecas presentes no Node.js, de forma a atender requisitos do projeto e gerar maior praticidade, tanto para a construção do código em si quanto para os testes. A seguir, são apresentados trechos de código nos quais essas bibliotecas foram aplicadas, detalhando suas respectivas funcionalidades para o trabalho.

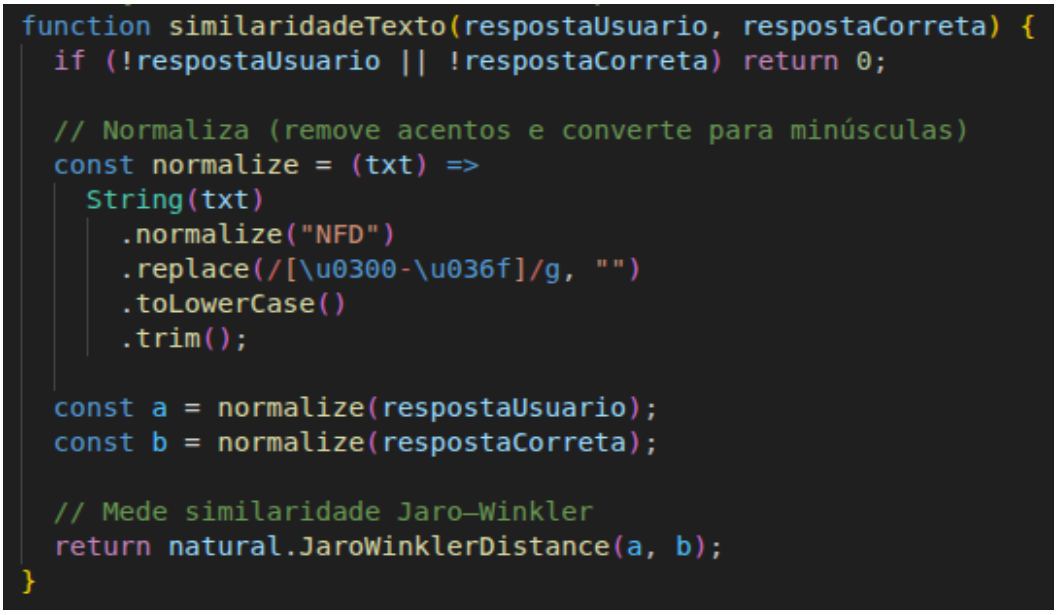
- Figura 3. *Express.js* (2026): *Framework* leve para o Node.js construído sobre os recursos HTTP nativos. Essa ferramenta é utilizada para a criação de *APIs* e serviços *web*, oferecendo maior facilidade para controle de requisições de clientes para *endpoints*. Além disso,

possibilita uma separação simples e clara na utilização dos métodos HTTP, mantendo o código sempre organizado e altamente escalável.

```
const express = require("express");
const db = require("./bd");
const natural = require("natural"); // Biblioteca para comparação de similaridade
const app = express();
const cors = require('cors');
const PORT = 3000;
const bcryptjs = require('bcryptjs');

app.use(cors());
app.use(express.json({ limit: '50mb' }));
app.use(express.urlencoded({ limit: '50mb', extended: true }));
```

Figura 3. Código do Express,js no servidor do sistema

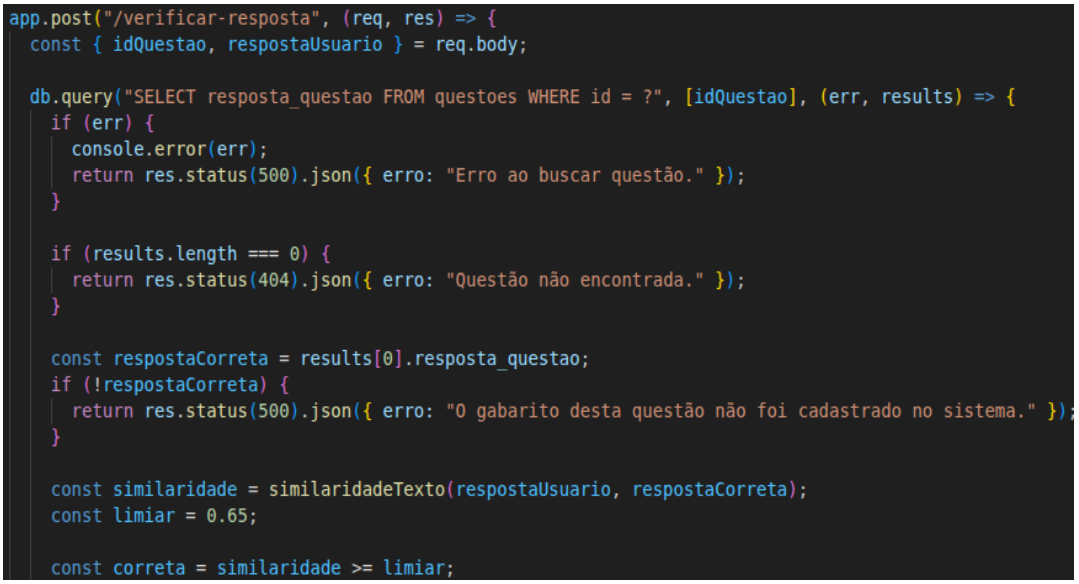
- CORS (*Cross-Origin Resource Sharing*, 2014*)*: É um *middleware* do *Express.js*, utilizado para habilitar e controlar a política de segurança dos navegadores em ambientes de desenvolvimento, permitindo a execução de arquivos da máquina em uso (*localhost*). Por padrão, os navegadores da *web* bloqueiam o recebimento de respostas de requisições provenientes de origens diferentes (domínios, protocolos ou portas lógicas distintas) como medida de precaução contra ataques cibernéticos. Nesse cenário, o CORS atua inserindo cabeçalhos HTTP específicos, os quais servem como uma forma de autorização de acesso legítimo para a *API*.
- Figura 4. Natural: Desenvolvida originalmente por Umbel (2011), é uma biblioteca de processamento de linguagem natural que realiza a comparação de proximidade entre duas *strings*, muito utilizado para que sistemas computacionais processem e compreendam textos redigidos por humanos. Oferece excelente nível de flexibilidade, possibilitando verificar a semelhança entre duas palavras ou duas frases através de algoritmos de cálculo de proximidade, como o Jaro-Winkler. Além de ser útil para a segmentação de textos em palavras-chave, a biblioteca *Natural* também fornece suporte para a classificação de dados visando ao treinamento de modelos de Inteligência Artificial (IA).



```
function similaridadeTexto(respostaUsuario, respostaCorreta) {
  if (!respostaUsuario || !respostaCorreta) return 0;

  // Normaliza (remove acentos e converte para minúsculas)
  const normalize = (txt) =>
    String(txt)
      .normalize("NFD")
      .replace(/[\u0300-\u036f]/g, "")
      .toLowerCase()
      .trim();

  const a = normalize(respostaUsuario);
  const b = normalize(respostaCorreta);

  // Mede similaridade Jaro–Winkler
  return natural.JaroWinklerDistance(a, b);
}
```

Figura 4. Uso da biblioteca Natural na prática

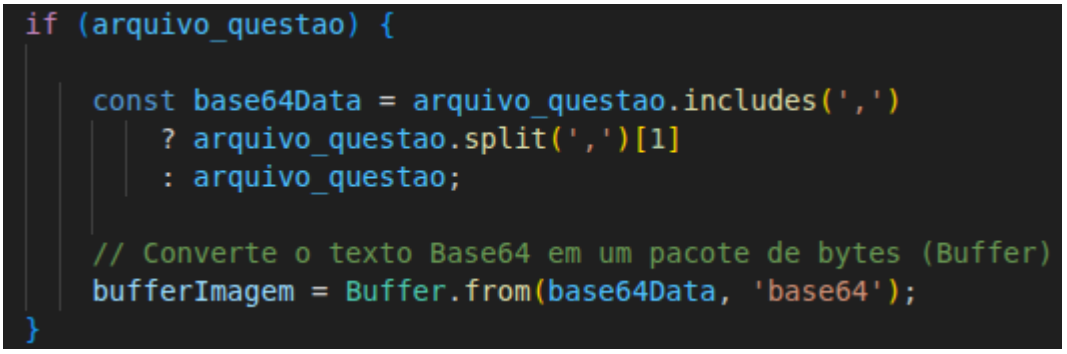
- Figura 5. Algoritmo Jaro-Winkler: usado para medir a similaridade de duas palavras ou strings, gera um valor entre 0 e 1, sendo próximo a 1 quando textos são muito parecidos, e próximo a zero quando são muito diferentes. Desenvolvido em 1990 pelo estatístico William E. Winkler, o cálculo aprimora a métrica anterior (conhecida como Distância de Jaro). O propósito é verificar se dois textos são semelhantes ou se fazem referência às mesmas coisas, ainda que apresentem variações de escrita. O algoritmo de Jaro-Winkler atribui maior peso ao prefixo das palavras, embasando-se em estudos que demonstram uma menor probabilidade de erros ortográficos no início dos vocábulos (Winkler, 1990). Por meio dessa métrica, o sistema consegue identificar equivalências de forma inteligente, como o nome “João” e a escrita sem acento “Joao”, como a frase “Ana Karina é professora” e outra frase “a professora é Ana Karina”. No escopo deste trabalho, o algoritmo permite aferir a proximidade entre a resposta inserida pelo aluno e a resposta correta registrada na plataforma, tendo sido estabelecido um limiar de aceitação de 65% de similaridade para considerar a questão como correta.



```
app.post("/verificar-resposta", (req, res) => {
  const { idQuestao, respostaUsuario } = req.body;

  db.query("SELECT resposta_questao FROM questoes WHERE id = ?", [idQuestao], (err, results) => {
    if (err) {
      console.error(err);
      return res.status(500).json({ erro: "Erro ao buscar questão." });
    }

    if (results.length === 0) {
      return res.status(404).json({ erro: "Questão não encontrada." });
    }

    const respostaCorreta = results[0].resposta_questao;
    if (!respostaCorreta) {
      return res.status(500).json({ erro: "O gabarito desta questão não foi cadastrado no sistema." });
    }

    const similaridade = similaridadeTexto(respostaUsuario, respostaCorreta);
    const limiar = 0.65;

    const correta = similaridade >= limiar;
```

Figura 5. Uso prático do algoritmo Jaro-Winkler

- Figura 6. Processamento de Imagens: Para viabilizar o armazenamento das mídias no banco de dados, utilizou-se a *API FileReader* no *front-end*, responsável por converter os arquivos originais em uma *string* codificada em *Base64* (W3C, 2009). No lado do servidor, a classe *Buffer* (nativa do *Node.js*) converte essa *string* em um pacote de *bytes*, permitindo a persistência segura dos dados na coluna de tipo *LongBlob* da tabela *questoes*.

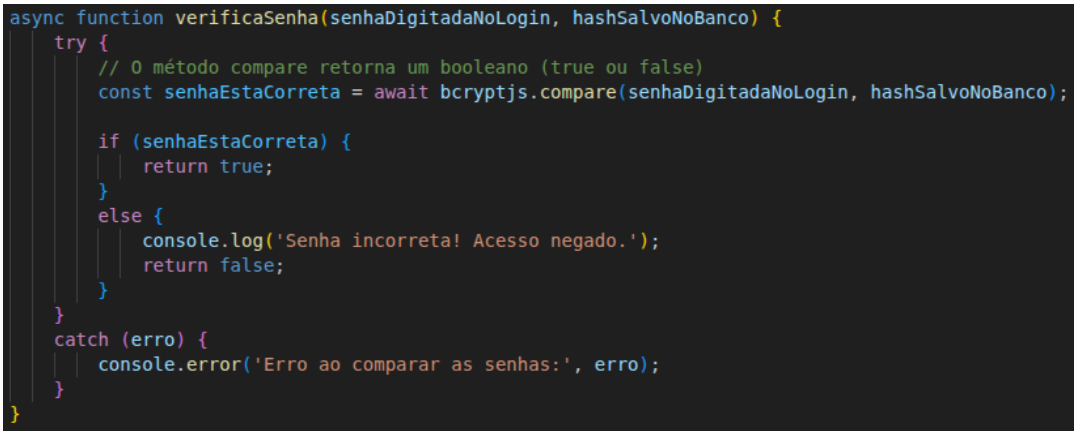


```
if (arquivo_questao) {

    const base64Data = arquivo_questao.includes(',')
        ? arquivo_questao.split(',')[1]
        : arquivo_questao;

    // Converte o texto Base64 em um pacote de bytes (Buffer)
    bufferImagem = Buffer.from(base64Data, 'base64');
}
```

Figura 6. Trecho de código da conversão de imagens

- Figura 7. Bcryptjs: Essencial para garantir a proteção dos usuários no banco de dados, a biblioteca bcryptjs foi adotada para converter as senhas de texto em claro em um *hash* irreversível (Wirtz, 2012). O algoritmo aplica um *salt* (um valor aleatório adicionado à senha original) antes da conversão matemática, o que garante que senhas idênticas gerem pacotes de dados completamente diferentes. Durante o processo de autenticação (*login*), o sistema gera o *hash* da credencial recém-digitada e o compara com o *hash* previamente armazenado no

banco de dados, validando o acesso com segurança sem nunca expor a senha original.



```
async function verificaSenha(senhaDigitadaNoLogin, hashSalvoNoBanco) {
    try {
        // O método compare retorna um booleano (true ou false)
        const senhaEstaCorreta = await bcryptjs.compare(senhaDigitadaNoLogin, hashSalvoNoBanco);

        if (senhaEstaCorreta) {
            return true;
        }
        else {
            console.log('Senha incorreta! Acesso negado.');
            return false;
        }
    }
    catch (erro) {
        console.error('Erro ao comparar as senhas:', erro);
    }
}
```

Figura 7. Uso da Bcryptjs para comparar senhas

5. Relatório de Desenvolvimento SEG

Com o objetivo de superar as limitações dos sistemas atuais de gamificação no ensino, este projeto foi idealizado a partir de uma demanda real. O professor Yuri Nejaim (FAODO) identificou a necessidade de uma ferramenta personalizada para otimizar as atividades de sua disciplina. Ele apresentou a situação do uso de sistemas de gamificação de ensino em aulas da faculdade, por meio da plataforma Kahoot!, que é a atual ferramenta de ensino gamificado adotada na instituição. Lançada em 2013, a plataforma foi fundada por Johan Brand, Jamie Brooker e Morten Versvik, com o auxílio de pesquisa do professor Alf Inge Wang (2015), em um projeto conjunto com a Universidade Norueguesa de Ciência e Tecnologia. O Kahoot!, assim como outras plataformas da mesma categoria, possui custos em dólar, sendo financeiramente inviável. Além disso, apresenta limitações técnicas para o uso no cotidiano universitário, como a restrição no tamanho de armazenamento de arquivos, cuja capacidade não satisfaz as necessidades práticas para o uso nas aulas. A partir desse diagnóstico, iniciaram-se as pesquisas para o desenvolvimento de um sistema gratuito e adaptado que pudesse suprir as necessidades do ensino na UFMS.

O levantamento de dados acerca de sistemas gamificados em geral consistiu na primeira etapa do processo criativo. As informações obtidas foram registradas em documentos e, posteriormente, transformadas em requisitos iniciais e na definição das tecnologias adequadas para satisfazê-los. A orientação da professora Dra. Ana Karina Dourado Salina de Oliveira conduziu o início dos trabalhos visando à criação de um sistema de ensino gamificado que pudesse atender às exigências mapeadas.

O desenvolvimento do sistema foi iniciado com a prototipação de telas por meio da plataforma *Canva*, tendo como inspiração sistemas consolidados de questionários lúdicos, como o *Duolingo* e o próprio *Kahoot!*. Após a criação e validação dos protótipos pela orientadora e pelo cliente, iniciou-se a produção dos artefatos de modelagem para fundamentar e organizar a arquitetura do produto, utilizando o *software Astah* e a plataforma *web Draw.io*. Concluída esta etapa, realizou-se a validação desses artefatos. Nesse contexto, a professora Dra. Jane Dirce Alves Sandim Eleutério, que também prestou forte apoio ao desenvolvimento do sistema, sugeriu a incorporação do Modelo C4. Esse modelo consiste em uma abordagem baseada em diagramas que proporciona uma melhor visualização dos níveis de abstração de um *software*, orientando dinamicamente a apresentação e a organização das camadas do sistema (como regras de negócio, bancos de dados e servidores), ilustrando-as de forma separada, porém conectadas com clareza.

O C4 utiliza uma analogia de "zoom" para segmentar a arquitetura, organizando-a em quatro níveis de aprofundamento (origem do nome C4): Nível 1

(Contexto), que ilustra o escopo do sistema, os atores e os sistemas externos envolvidos; Nível 2 (Contêineres - Figura 8), que apresenta as aplicações individuais que compõem o *software*; Nível 3 (Componentes), que detalha os blocos lógicos dentro de um contêiner (como *controllers*, serviços e repositórios) e o nível 4 (Código), que aprofunda-se nas classes, interfaces ou diagramas UML específicos do sistema (Brown, 2011).

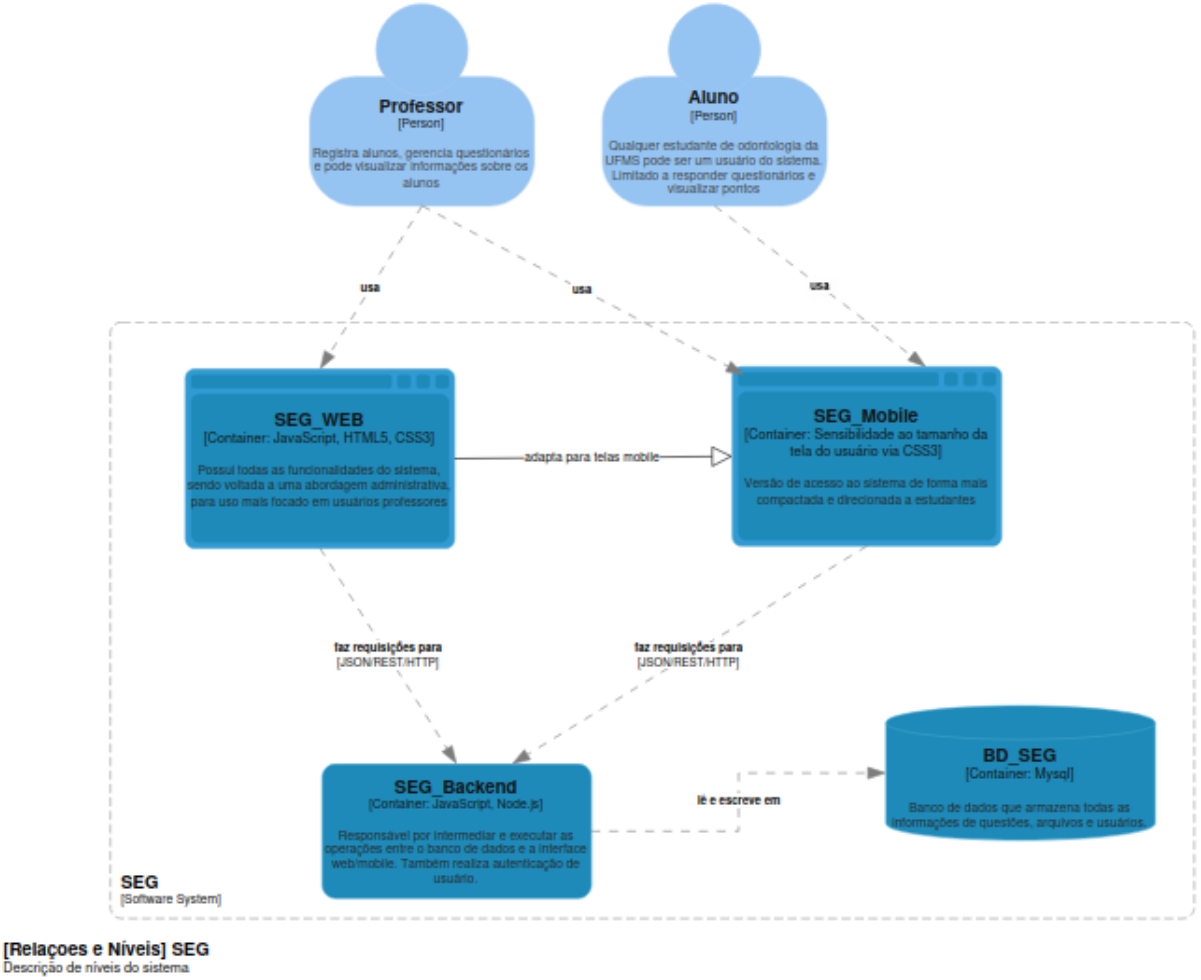


Figura 8. Nível 2 do diagrama C4 do sistema

5.1 *Front-end*

A codificação do *software* iniciou-se pelo desenvolvimento manual das interfaces gráficas utilizando HTML e CSS, com a aplicação da linguagem *JavaScript* para gerenciar o dinamismo necessário. Após a apresentação, aprovação pela orientadora e validação junto ao cliente, prosseguiu-se com o refinamento dessas interfaces. Nesse momento, definiu-se formalmente o nome da aplicação como SEG (Sistema de Ensino Gamificado). Ao todo, foram desenvolvidas 14 telas. O fluxo do sistema inicia-se por uma interface de boas-vindas (Figura 9), que oferece a opção de autenticação (*login*) como aluno ou professor, dependendo do perfil do usuário.

Figura 9. Tela inicial do SEG

Na visão do docente, o sistema conta com um menu inicial composto por cinco funcionalidades principais: criar questões, criar questionário (e listagem de questões na mesma tela), listar questionário, responder questões e responder questionário. Cada uma dessas telas foi aprimorada com recursos em *Javascript*, que também é responsável pelo roteamento e pela navegação geral entre as páginas do sistema. Dentre as funcionalidades dedicadas aos professores, destaca-se a tela de criação de perguntas (Figuras 10, 11, 12 e 13). Esse ambiente apresenta opções para a formulação de questões que serão respondidas pelos alunos, permitindo a inserção de imagens e a escolha de diferentes formatos: múltipla escolha (com alternativas a serem assinaladas), questões dissertativas (abertas) e questões de verdadeiro ou falso. Os campos exibidos na interface são habilitados ou desabilitados dinamicamente conforme o tipo de pergunta selecionado. Além disso, o sistema conta com validações de formulário que impedem o avanço e a gravação dos dados caso o docente não indique qual é a alternativa correta (no caso de questões fechadas).

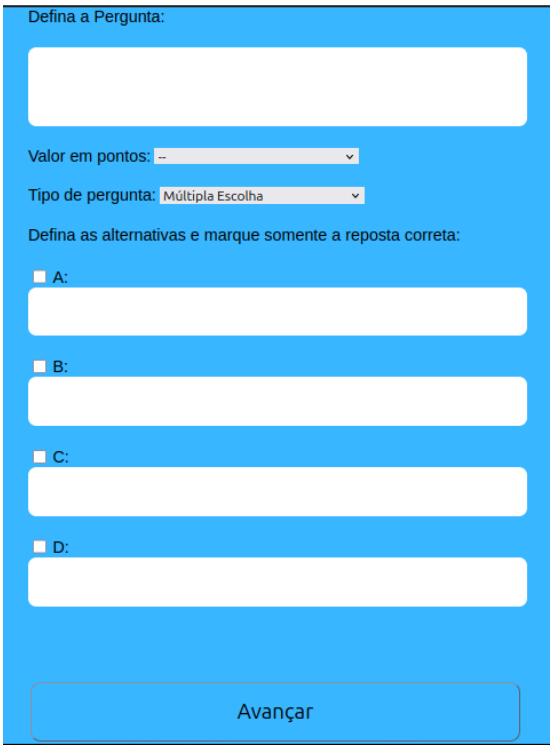


Figura 10. Tela de criação de múltipla escolha

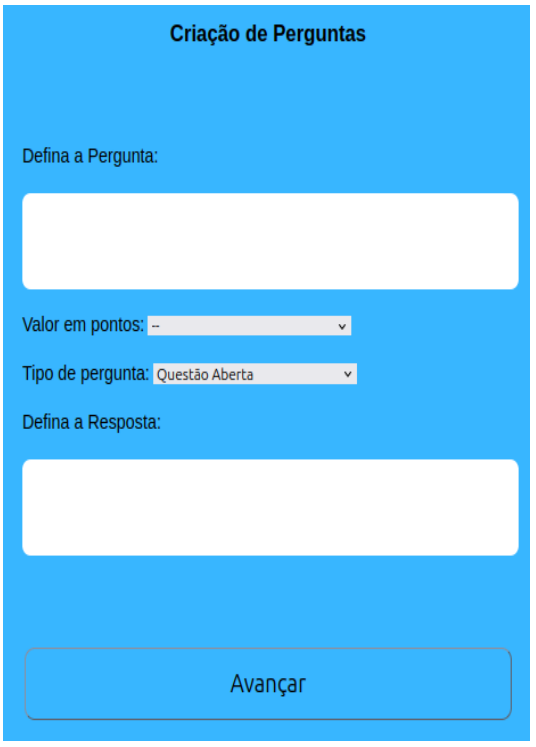


Figura 11. Tela de criação de questão aberta

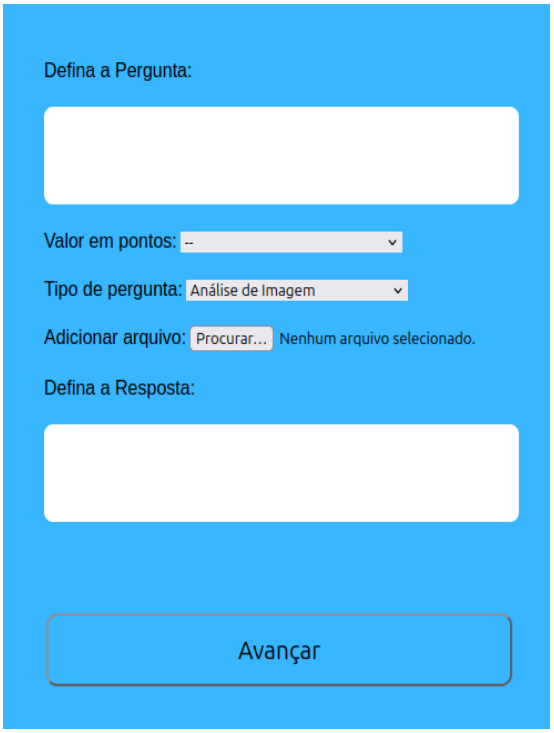


Figura 12.Tela de criação de análise de imagem

Figura 13. Tela de criação de verdadeiro ou falso

Ainda no módulo docente, o sistema apresenta a funcionalidade de criação de questionários (Figura 14). Nessa interface, é possível formar um grupo de cinco questões sob um mesmo título. Esse agrupamento permite que os alunos busquem o questionário e respondam às perguntas de forma sequencial e ininterrupta, obtendo os resultados somente após concluir toda a resolução. Ao final, a pontuação referente ao questionário (somatória dos acertos), é registrada.

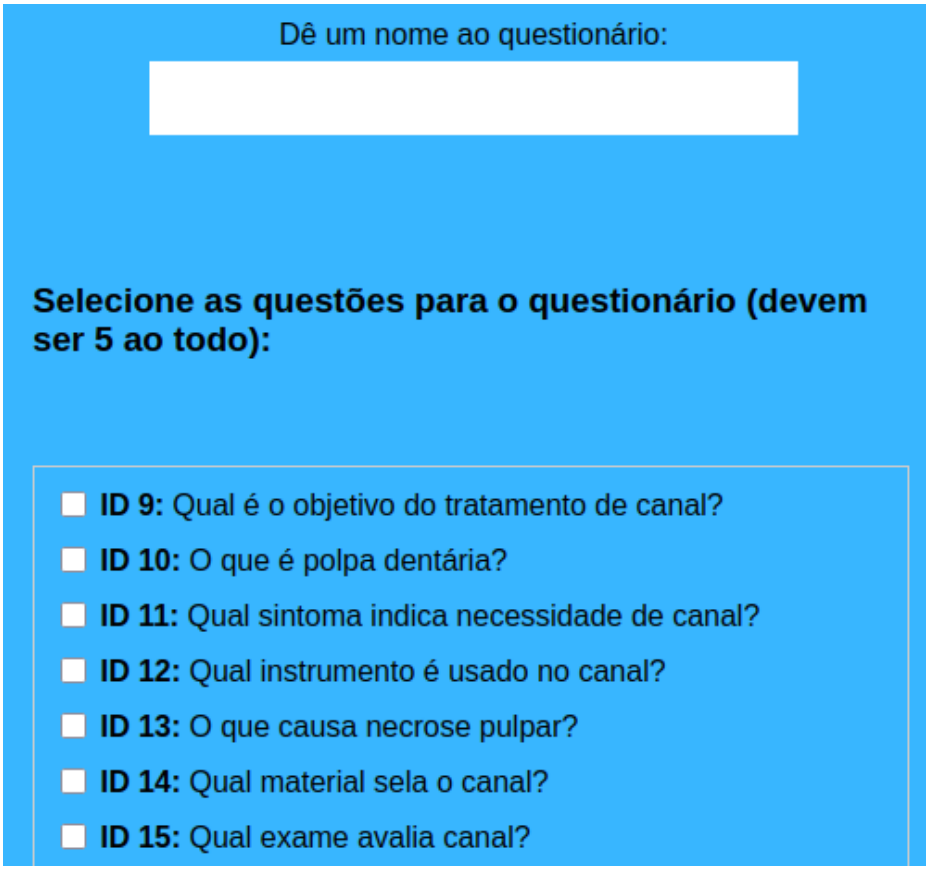


Figura 14. Tela de criação de questionário

A interface voltada ao aluno, por sua vez, conta com um menu principal estruturado em três funcionalidades: buscar questionário, buscar questão e visualizar *ranking*. A primeira opção direciona o usuário aos questionários elaborados pelos professores, os quais podem contemplar os quatro diferentes tipos de questões (Figuras 15, 16, 17 e 18), sendo cada formato renderizado em sua respectiva tela.

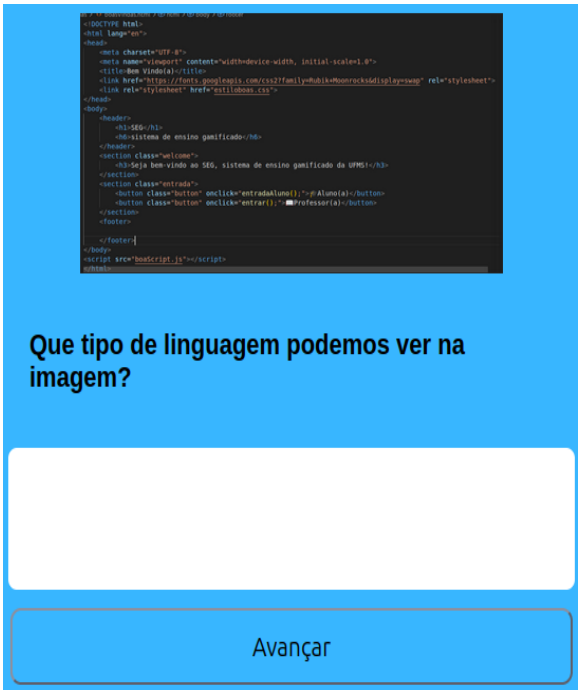


Figura 15. Tela de análise de imagem

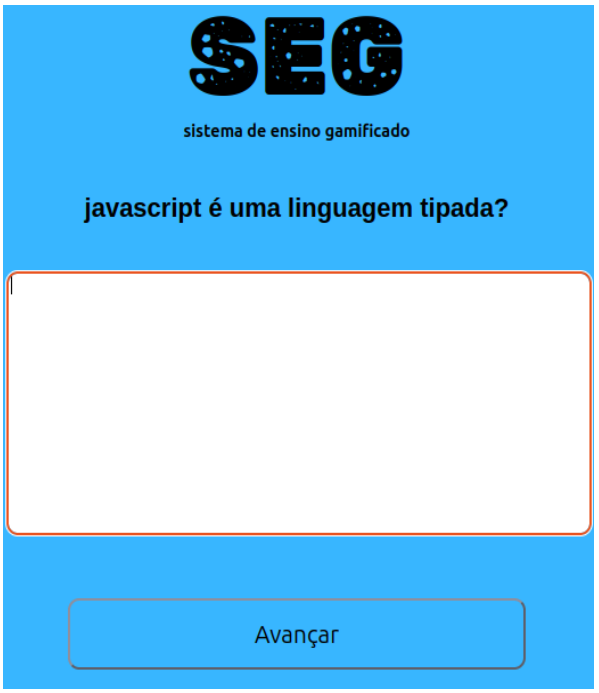


Figura 16. Tela de questão aberta

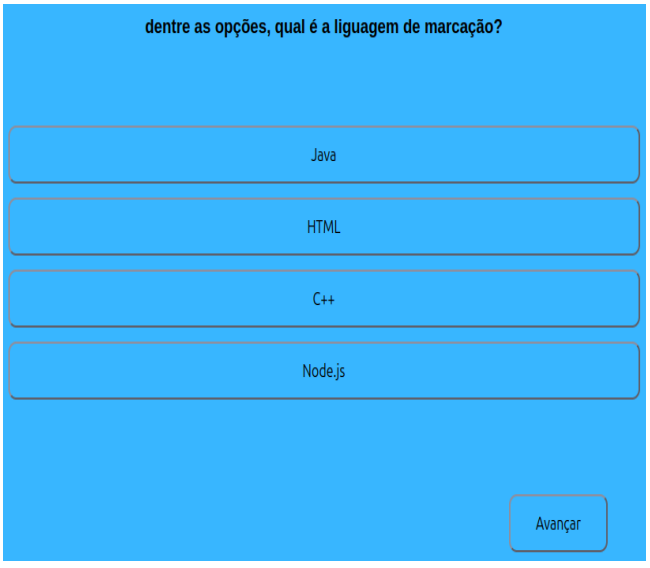


Figura 17. Tela de múltipla escolha

Figura 18. Tela de verdadeiro ou falso

Por fim, o discente tem acesso à aba de *ranking* (Figura 19). Essa funcionalidade contabiliza a pontuação total obtida por cada aluno na resolução das atividades e gera uma lista classificatória com todos os usuários pontuadores. O painel é estruturado em ordem decrescente, destacando nas primeiras posições os estudantes com o maior desempenho acumulado.

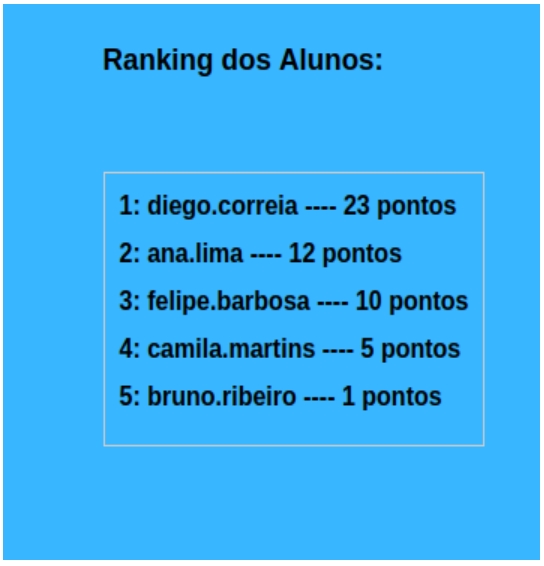


Figura 19. Tela de ranking

5.2 Banco de Dados

Na etapa subsequente do desenvolvimento, o banco de dados (Figura 20) do sistema SEG foi inteiramente estruturado. Inicialmente, utilizaram-se os recursos de conteinerização do *Docker* (2013) associados ao *MySQL* (Oracle, 1995) e à interface *phpMyAdmin*. Em seguida, optou-se também pela instalação local do *MySQL* no sistema operacional *Ubuntu* (Canonical, 2004). O banco relacional do sistema é composto por seis tabelas principais: *aluno*, *professor*, *questoes*, *ranking*, *questionarios* e *alternativa_questao*. Vale destacar Cumpre destacar o papel da tabela *alternativa_questao*, que atua como uma entidade auxiliar para a tabela *questoes*. A lógica de modelagem adotada estabelece que as alternativas incorretas de uma pergunta sejam registradas nessa tabela, enquanto a alternativa correta é armazenada diretamente na tabela *questoes*. Essa estratégia garante uma inserção padronizada para todos os formatos de perguntas aceitos pelo sistema. Durante a execução, ao carregar uma questão de múltipla escolha, todas as alternativas são resgatadas simultaneamente por meio de uma cláusula *LEFT JOIN* na consulta *SQL (Zhao*, 2022*)*.

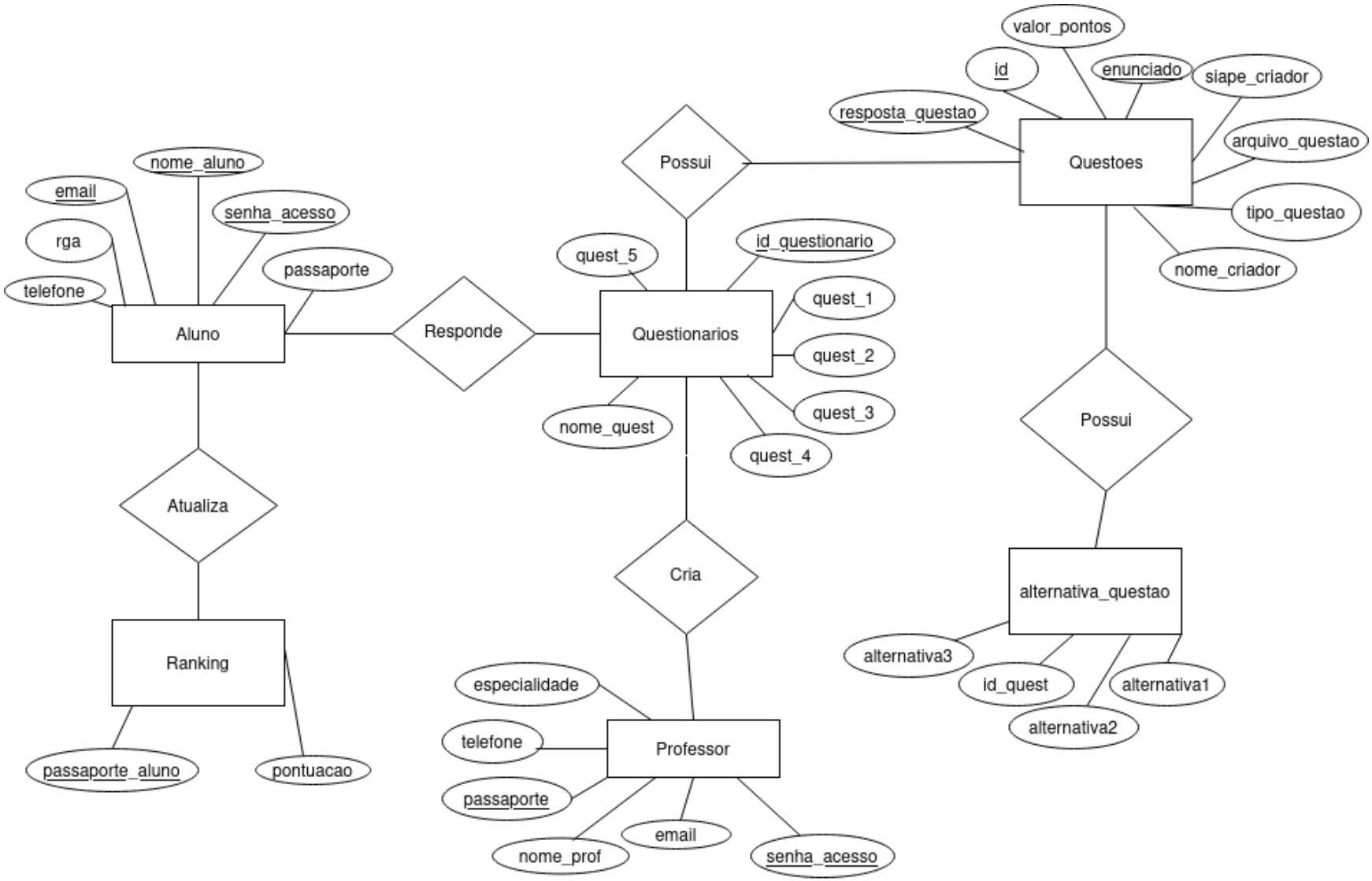


Figura 20. Diagrama entidade-relacionamento do banco de dados

Destaca-se também a tabela *ranking*, que se relaciona com a tabela *aluno* pelo identificador de usuário (Passaporte). Para exibir a classificação consolidada e de forma decrescente, a consulta utiliza um comando *ORDER BY DESC (Zhao*, 2022*)*.

Os dados utilizados para o preenchimento inicial das tabelas (massa de testes) foram gerados de forma híbrida. Visando a simplificar e agilizar o processo, empregaram-se ferramentas de Inteligência Artificial Generativa (*ChatGPT* e *Gemini*). Paralelamente, inseriram-se dados provenientes de fontes reais, elaborados manualmente a partir de conhecimentos específicos da área de computação. A utilização de ferramentas de IA para a geração de massa de dados apresenta desafios práticos. Exige-se uma validação rigorosa para combater a criação de informações demasiadamente genéricas ou destoantes da realidade do sistema. Ademais, existem limitações técnicas intrínsecas a essas plataformas, como as restrições diárias de uso para a análise e o processamento de grandes volumes de arquivos e dados.

5.3 Backend

Para o desenvolvimento do *back-end* do SEG, optou-se pela utilização do *Node.js* (2026). Essa escolha justifica-se pelo fato de a tecnologia já ser abordada nas disciplinas do curso de Sistemas de Informação da UFMS, além de dispor de um amplo ecossistema de bibliotecas capazes de atender plenamente às mais diversas necessidades estruturais de um sistema web. A partir disso, estruturou-se um arquivo de configuração exclusivo para estabelecer a conexão com o banco de dados *MySQL* (hospedado no ambiente local *Ubuntu*), o qual é importado e utilizado pelo código principal do servidor.

O arquivo *server.js* atua como o *Controller* (Controlador) da aplicação, baseando-se no padrão de arquitetura MVC (*Model-View-Controller*). Ele é responsável por interceptar todas as requisições, processá-las, encaminhar os comandos adequados ao banco de dados e, por fim, retornar as respostas no formato *JSON*. Esses dados são então consumidos e renderizados pelas interfaces gráficas (*Views*) por meio de funções estruturadas em *JavaScript*. O *server* também concentra funções importantes, como: Interpretação e conversão de Strings de imagens, normalização de textos de respostas de questões, utilização da biblioteca Natural com o algoritmo Jaro-Winkler e definição do valor do limiar que corresponde ao acerto ou ao erro de uma resposta do usuário.

5.4 Análise e Implementação das Questões

Em relação ao funcionamento das questões, embora todas sejam armazenadas em uma tabela unificada no banco de dados, cada formato possui uma lógica de

processamento e renderização distinta. A seguir, detalham-se a análise e a implementação de cada tipo de questão:

5.4.1 Questões de Múltipla Escolha

Como as questões de múltipla escolha utilizam a tabela auxiliar *alternativa_questao* para armazenar as opções incorretas, o resgate desses dados exige uma requisição (método *GET*) específica, que realize a junção das informações provenientes de ambas as tabelas. Esse processo gera um *array* contendo todas as alternativas, as quais são sempre embaralhadas aleatoriamente antes de compor o objeto *JSON* enviado para a interface da aplicação.

No processo de correção desse modelo de questão, apenas o texto interno do botão selecionado pelo aluno é enviado ao servidor para validação. A comparação é realizada de forma exata, utilizando o operador de igualdade estrita (`===`). Como o usuário apenas seleciona o botão correspondente à resposta, não há incidência de erros ortográficos, dispensando o uso de algoritmos avançados de aproximação lexical para validar o acerto.

5.4.2 Questões Escritas

As questões de formato escrito (dissertativas), por sua vez, fazem uso da requisição *GET* padrão, cuja finalidade é apenas recuperar o enunciado para exibição na interface. No entanto, o método de correção desse formato possui uma complexidade maior. O sistema utiliza o algoritmo de Jaro-Winkler atrelado a uma etapa prévia de normalização de texto (que realiza a conversão para letras minúsculas e a remoção de caracteres diacríticos, como acentos) (Winkler, 1990). Essa preparação visa adequar o texto digitado pelo usuário para a comparação com o gabarito armazenado no banco de dados. A comparação retorna um limiar (entre 0 e 1), considerando como correta toda aquela que atinge 0.65 (65%) ou mais de proximidade. Após a validação, o servidor retorna um valor booleano (*true* para acerto ou *false* para erro) para a interface gráfica.

5.4.3 Questões com Imagem

Para as questões de análise de imagem, a lógica de funcionamento é semelhante à das questões escritas, com a adição do processamento de mídias. Como os arquivos de imagem são inseridos e persistidos no banco de dados na forma de *bytes*, a requisição *GET* padrão aciona uma estrutura condicional específica. Esse recurso é responsável por interceptar a informação e

convertê-la de volta para uma *string* codificada em *Base64*, viabilizando o seu envio na rede e a consequente renderização da imagem na tela do usuário.

5.4.4 Questões Verdadeiro ou Falso

Nas questões do tipo verdadeiro ou falso, a manipulação das respostas difere das demais. As opções são convertidas em valores numéricos para o processamento no *back-end*, em que 1 representa "verdadeiro" e 2 representa "falso", sendo armazenadas dessa mesma forma no banco de dados. A requisição *GET* utilizada é a padrão, mas a correção exige a captura do valor correspondente ao botão selecionado pelo usuário na interface. Esse dado é enviado ao servidor, onde ocorre a comparação. Como as respostas estão armazenadas no banco com o tipo de dado *VARCHAR*, faz-se necessária a conversão prévia por meio da função *parseInt()* antes de aplicar o operador de igualdade estrita (===). O resultado dessa validação retorna um booleano (*true* ou *false*) para o *front-end*.

5.5 Descrição de Funcionalidades Especiais

A seguir, detalham-se as funcionalidades especiais que foram criadas para o sistema.

5.5.1 Consultas no Sistema SEG

As consultas no sistema SEG são divididas em duas modalidades: busca individual por questão e busca por questionário. As buscas individuais são executadas utilizando, simultaneamente, o identificador (*id*) e o tipo da questão, sendo essa lógica aplicada a todos os formatos de perguntas. Os parâmetros são enviados ao servidor, que realiza a consulta no banco de dados para retornar o conteúdo solicitado (ou indicar a sua inexistência). O retorno da requisição ocorre de forma objetiva, entregando à interface gráfica um objeto *JSON* com todas as informações consolidadas.

5.5.2 Questionário

Quanto ao funcionamento dos questionários, concebidos como ferramentas de preparação prévia, o servidor utiliza o nome da lista como parâmetro de localização. A busca inicial ocorre na tabela *questionarios*, que identifica os cinco identificadores (*ids*) das questões vinculadas àquele grupo. Posteriormente, uma nova consulta (*query*) é executada na tabela *questoes* para recuperar os dados detalhados de cada item. Nesse estágio, o sistema valida o tipo de cada pergunta: se for identificada uma questão de múltipla escolha, realiza-se uma busca adicional para resgatar as alternativas incorretas, as quais

são integradas ao array original na posição correspondente da respectiva pergunta.

Todo esse conjunto de informações é transmitido ao front-end em um único pacote de dados consolidado. Para gerenciar a exibição e a navegação sequencial e ininterrupta entre as perguntas, a interface utiliza um script JavaScript denominado *carregaQuestionario*.

5.5.3 Ranking de Pontos

A pontuação dos alunos no *ranking* é atualizada após a resolução de cada questão individual ou ao término de um questionário. Caso o discente ainda não possua registro na tabela correspondente (ou seja, caso ainda não tenha pontuado no sistema), o servidor realiza a inserção de seus dados assim que os primeiros acertos forem contabilizados. Para garantir a eficiência dessa operação de inserção ou atualização no banco de dados, utiliza-se a instrução *ON DUPLICATE KEY UPDATE pontuacao (Zhao,* 2022*)*. Dessa forma, o painel de classificação exibe os usuários de maneira contínua e atualizada, sempre ordenados de forma decrescente (do maior para o menor pontuador).

5.6 Segurança do Sistema

O módulo de autenticação do sistema foi desenvolvido com mecanismos voltados à proteção das credenciais dos usuários. As senhas não são armazenadas em texto puro no banco de dados; em seu lugar, é utilizada uma função criptográfica de hash para gerar um valor derivado da senha original, que é então armazenado de forma segura. Durante o processo de autenticação, o usuário informa seu passaporte institucional e sua senha. Essas informações são encaminhadas ao servidor, que realiza a consulta do usuário correspondente no banco de dados. Após a identificação do registro, uma função de verificação compara a senha informada com o valor hash armazenado, validando a autenticidade das credenciais sem a necessidade de recuperar ou expor a senha original. Quando a autenticação é bem-sucedida, o usuário é direcionado às funcionalidades correspondentes ao seu perfil de acesso.

Além da autenticação, o sistema implementa mecanismos de controle de acesso para garantir que cada usuário tenha acesso apenas aos recursos compatíveis com suas permissões. No módulo de gerenciamento de questões, por exemplo, a consulta é realizada considerando a identificação do professor autenticado, permitindo que sejam exibidas apenas as questões previamente cadastradas por esse usuário. Essa abordagem contribui para a preservação da integridade dos dados e impede o acesso indevido a conteúdos pertencentes a outros docentes.

No módulo de questionários, o sistema disponibiliza uma funcionalidade de listagem que recupera os questionários armazenados no banco de dados e apresenta informações essenciais para sua identificação e utilização. A implementação prioriza a exibição apenas dos dados necessários para a operação, reduzindo a exposição de informações internas da aplicação e contribuindo para a organização e usabilidade da interface. Dessa forma, os módulos desenvolvidos buscam atender simultaneamente aos requisitos funcionais da aplicação e às boas práticas de segura.

5.7 Testes

Os testes do SEG foram realizados por meio da hospedagem local fornecida pelo *Live Server* (Dey, 2017). O acesso à aplicação ocorreu via *web*, a partir da própria máquina de desenvolvimento, e também em ambiente *mobile*, por meio de *smartphones*. Adotou-se uma abordagem de testes funcionais manuais, iterando diretamente sobre as funcionalidades implementadas para a sua validação.

O método de teste empregado foi direto, utilizando as funcionalidades desenvolvidas para verificação. Testes de estresse foram usados para encontrar erros e vulnerabilidades que pudessem inviabilizar o funcionamento adequado ou até mesmo causar a queda do servidor (Pressman; Maxim, 2016). Como contramedida aos problemas encontrados, o código-fonte foi refatorado com a inclusão de estruturas condicionais (*if*, *else*) voltadas estritamente ao tratamento de exceções.

Os diferentes tipos de questões foram inseridos um após o outro, então os conteúdos dessas inserções foram verificados nas tabelas do banco de dados através do terminal do *MySQL* (Oracle, 1995). Os pacotes de informações enviados e recebidos pelo servidor foram analisados por meio de ferramentas de desenvolvimento nativas dos navegadores de internet, através das quais também foi possível encontrar mensagens de erros que ainda não haviam sido tratados.

A inteligência artificial (*Gemini*) serviu como auxílio para descobrir as causas de problemas até então desconhecidos e para sugerir soluções a serem empregadas para solucioná-los.

6. Considerações Finais

A aplicação da gamificação no contexto educacional representa uma mudança de paradigma significativa na forma como o conhecimento é transmitido e absorvido. Ao incorporar alguns elementos de jogos, como quizzes, pontuações e desafios graduais, o ambiente de aprendizagem vai além do ensino tradicional, colocando o aluno como protagonista ativo do seu desenvolvimento. Essa estratégia pedagógica não apenas eleva os níveis de engajamento e motivação, mas também reduz a ansiedade atrelada ao erro, transformando-o em uma etapa natural e iterativa da descoberta. Dessa forma, a tecnologia atua como uma ponte facilitadora, permitindo que a fixação do conteúdo ocorra de maneira dinâmica, imersiva e altamente interativa.

As demandas recebidas da Faculdade de Odontologia (FAODO) foram atendidas: Um sistema sem custos para utilização, com um limite de 50 *megabytes* para garantir a utilização de imagens de alta resolução para a criação de perguntas e com várias opções de questões interativas.

Para viabilizar essa experiência educacional, este projeto apoiou-se em um ecossistema tecnológico robusto, desde a sua concepção até a sua implantação. A fase de documentação e engenharia de software foi fundamentada no Modelo C4 (Brown, 2011), utilizando ferramentas especializadas de diagramação, como o *Astah*, da *Change Vision* (2009) e o *Diagrams.net* (JGraph, 2012), para mapear o contexto e os contêineres do sistema. Na camada de aplicação e interação com o usuário, o tráfego seguro e a manipulação de dados em tempo real foram garantidos pela adoção de padrões web consolidados, como as políticas *CORS* (*Cross-Origin Resource Sharing*, 2014*)* e a interface nativa FileReader API (W3C, 2009).

O núcleo inteligente de correção do quiz foi viabilizado no servidor através da biblioteca de Natural (Umbel, 2011), aplicando o rigor matemático do algoritmo de Jaro-Winkler para avaliar a similaridade e mitigar erros ortográficos nas respostas (Winkler, 1990). Por sua vez, a persistência e a integridade das informações foram estruturadas em um banco de dados relacional *MySQL*, da Oracle (1995), operado via *SQL (Zhao,* 2022*)*, enquanto a proteção das credenciais dos usuários foi assegurada pelo uso de hash criptográfico com a biblioteca bcryptjs (Wirtz, 2012). Por fim, a confiabilidade e a escalabilidade do ambiente de produção foram alcançadas através da conteinerização da infraestrutura via *Docker* (2013), sempre apoiado sobre a estabilidade de um servidor executado diretamente no *Ubuntu* (Canonical, 2004).

Após um longo período de desenvolvimento, o SEG consegue entregar recursos valiosos para o ensino. Sua implementação permite uso de questões de forma estimulante, com margem para uma variabilidade de abordagens de uma resposta de usuário, ao mesmo tempo possui potencial para grande crescimento, seja de suas opções oferecidas ou de sua forma de apresentação e interação com o usuário.

A criação do Sistema de Ensino Gamificado elucidou muitas possibilidades que agregam fortemente ao ensino da UFMS, mas em especial a Faculdade de Computação, pois deixa registrado o uso de tecnologias e bibliotecas que podem pavimentar o caminho dos estudantes que virão, mostrando-se como um exemplo de criatividade e exercício das habilidades de desenvolvimento de software.

7. Trabalhos Futuros

Para a continuação do desenvolvimento do Sistema de Ensino Gamificado, existem muitas melhorias que podem ser implementadas, tais como:

- Versão mobile aprimorada em um aplicativo mais avançado, com funcionalidades únicas para esta plataforma.
- Adição de novos tipos de questões a serem respondidas, como completar texto, correção de código automatizada, etc.
- Aderir ao padrão de layout da Agetic, para que o SEG possa ser disponibilizado para toda a UFMS e que seja cuidado pela agência.
- Usar o componente de autenticação de usuário da UFMS.
- Implementar uma classificação de questões por área de estudo, como computação, odontologia, medicina, administração, entre outras.
- Implementar subtipos de questões, como dentística, direito penal, eficiência de algoritmos, *SCRUM*, entre outros.
- Inserir um impeditivo que previne a resolução de um mesmo questionário mais de uma vez.

Referências

MADUREIRA, J. S.; SCHNEIDER, H. N. Gamificação no ensino de programação de computadores em turmas do ensino médio: uma experiência com o software Kahoot!. RENOTE - Revista Novas Tecnologias na Educação, Porto Alegre, v. 19, n. 2, p. 306–315, dez. 2021. DOI: 10.22456/1679-1916.121191. Disponível em: <https://seer.ufrgs.br/index.php/renote/article/view/121191/65830>

GOMES, L. K. de O. Gamificação no processo de aprendizagem: desenvolvimento e aplicação de uma plataforma lúdica educacional. 2025. Trabalho de Conclusão de Curso (Graduação) – Universidade Federal de Sergipe, São Cristóvão, 2025. Disponível em: <https://ri.ufs.br/bitstream/riufs/23435/2/Luan_Kyrtinem_Oliveira_Gomes.pdf>

SOUSA, K. H. F. de. Quizzes gamificados como recursos tecnológicos no ensino-aprendizagem de programação. 2023. Dissertação (Mestrado) – Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, João Pessoa, 2023. Disponível em: <https://repositorio.ifpb.edu.br/bitstream/177683/3001/1/Disserta%C3%A7%C3%A3o%20-%20Karine%20-%20Assinada.pdf>

WANG, Alf Inge. The wear out effect of a game-based student response system. Computers & Education, v. 82, p. 217-227, 2015. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0360131514002516?casa_token=knZus9941GQAAAAA:XOPOGv968PkDoOzczX2O0cxFl8f7kiHWoP0vUXHmLMKUicNf-uFK3JZgWE76HXnGmZYg-gCwj58>

W3SCHOOLS. HTML Tutorial. [S. l.], 2026. Disponível em: <https://www.w3schools.com/html/>

W3SCHOOLS. CSS Tutorial. [S. l.], 2026. Disponível em: <https://www.w3schools.com/css/>

FLANAGAN, David. JavaScript: o guia definitivo. 7. ed. Porto Alegre: Bookman, 2021.

NODE.JS. Node.js Documentation. [S. l.], 2026. Disponível em: <https://nodejs.org/>

EXPRESS.JS. Express.js Documentation. [S. l.], 2026. Disponível em: <https://expressjs.com/>

DEY, Ritwick. vscode-live-server. [S. l.], 2017. Disponível em: <https://github.com/ritwickdey/vscode-live-server>

MICROSOFT. Visual Studio Code. [S. l.], 2015. Disponível em: <https://code.visualstudio.com/>

CANVA. Canva. [S. l.], 2013. Disponível em: <https://www.canva.com/pt_br/>

CHANGE VISION. Astah. [S. l.], 2009. Disponível em: <https://astah.net/>

JGRAPH. diagrams.net (antigo draw.io). [S. l.], 2012. Disponível em: <https://www.diagrams.net/>

ORACLE. MySQL. [S. l.], 1995. Disponível em: <https://www.mysql.com/>

W3C. Cross-Origin Resource Sharing. [S. l.], 2014. Disponível em: <https://www.w3.org/TR/cors/>

UMBEL, Chris. natural. [S. l.], 2011. Disponível em: <https://github.com/NaturalNode/natural>

WINKLER, William E. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. Proceedings of the Section on Survey Research Methods, American Statistical Association, [S. l.], p. 354-359, 1990.

W3C. File API. [S. l.], 2009. Disponível em: <https://www.w3.org/TR/FileAPI/>

WIRTZ, Daniel. bcrypt.js. [S. l.], 2012. Disponível em: <https://github.com/dcodeIO/bcrypt.js>

DOCKER. Docker. [S. l.], 2013. Disponível em: <https://www.docker.com/>

CANONICAL. Ubuntu. [S. l.], 2004. Disponível em: <https://ubuntu.com/>

ZHAO, Alice. SQL Guia Prático: um guia para o uso de SQL. 4. ed. São Paulo: Novatec, 2022.

BROWN, Simon. The C4 model for visualising software architecture. [S. l.], 2011. Disponível em: <https://c4model.com/>

LAUDON, Kenneth C.; LAUDON, Jane Price. Sistemas de informação gerenciais. 11. ed. São Paulo: Pearson Prentice Hall, 2014.

BACICH, Lilian; MORAN, José (Org.). Metodologias ativas para uma educação inovadora: uma abordagem teórico-prática. Porto Alegre: Penso, 2018.

BUSARELLO, Raul Inácio. Gamificação na educação. São Paulo: Pimenta Cultural, 2016.

KAPP, Karl M. The gamification of learning and instruction: game-based methods and strategies for training and education. San Francisco: Pfeiffer, 2012.

DETERDING, S. et al. From game design elements to gamefulness: defining gamification. In: Proceedings of the 15th International Academic MindTrek Conference. ACM, 2011. p. 9-15.

RAMOS, Leonardo de Souza e Silva. Especificação de Requisitos do Sistema de Ensino Gamificado. Campo Grande, 2026. Documento técnico não publicado. Disponível em : <https://docs.google.com/document/d/11kvo4CZFr7IBeOww_seWwqCpyLt6G_A5/edit?usp=sharing&ouid=105395958099576821957&rtpof=true&sd=true>

PRESSMAN, Roger S.; MAXIM, Bruce R. Engenharia de software: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.