Aurum
EDITORA

LABIA BLOCKS: PROGRAMAÇÃO EM BLOCOS, INTELIGÊNCIA ARTIFICIAL E GAMIFICAÇÃO NO ENSINO INTRODUTÓRIO DE JOGOS WEB

https://doi.org/10.63330/aurumpub.057-026

Tiago Saidelles
Mestre em Educação Profissional e Tecnológica
Universidade Federal de Santa Maria (UFSM)
E-mail: tiago-saidelles@redes.ufsm.br
Lattes: http://lattes.cnpq.br/786602013044509

Miguel Augusto Bauermann Brasil
Doutor em Extensão Rural
Universidade Federal de Santa Maria (UFSM)
E-mail: miguelbrasil@ctism.ufsm.br
Lattes: http://lattes.cnpq.br/9617974701932533

Resumo

A pesquisa apresenta e analisa a LabIA Blocks, uma plataforma educacional desenvolvida para apoiar a iniciação à programação e à criação de jogos digitais por meio da integração entre programação em blocos, HTML, CSS, JavaScript, gamificação e inteligência artificial. Trata-se de uma pesquisa aplicada, descritiva e de desenvolvimento tecnológico, cujo objetivo é descrever a construção da ferramenta, suas funcionalidades e suas possibilidades didáticas para professores e estudantes. A metodologia compreendeu análise funcional e documental da versão final do sistema, inspeção do catálogo de blocos e das áreas de professor e aluno, além da articulação das funcionalidades com princípios de metodologias ativas, pensamento computacional e aprendizagem baseada em projetos. Os resultados da análise evidenciam um ambiente que reúne editor visual com geração de código, Tutor IA contextual, planejamento em Portugol, desafios progressivos, jogos prontos, criação autoral e recursos docentes para organizar turmas, cadastrar estudantes, publicar atividades e acompanhar entregas. Como resultados esperados de sua utilização pedagógica, projeta-se favorecer a compreensão gradual da lógica de programação, a experimentação, a depuração, a autonomia e o protagonismo discente, aproximando conceitos iniciais de desenvolvimento web de situações práticas de criação de jogos. Conclui-se que a LabIA Blocks apresenta potencial como recurso didático integrado para experiências ativas e gamificadas de aprendizagem de programação.

Palavras-chave: Desenvolvimento de jogos; Gamificação; Inteligência artificial na educação; Metodologias ativas; Programação em blocos.

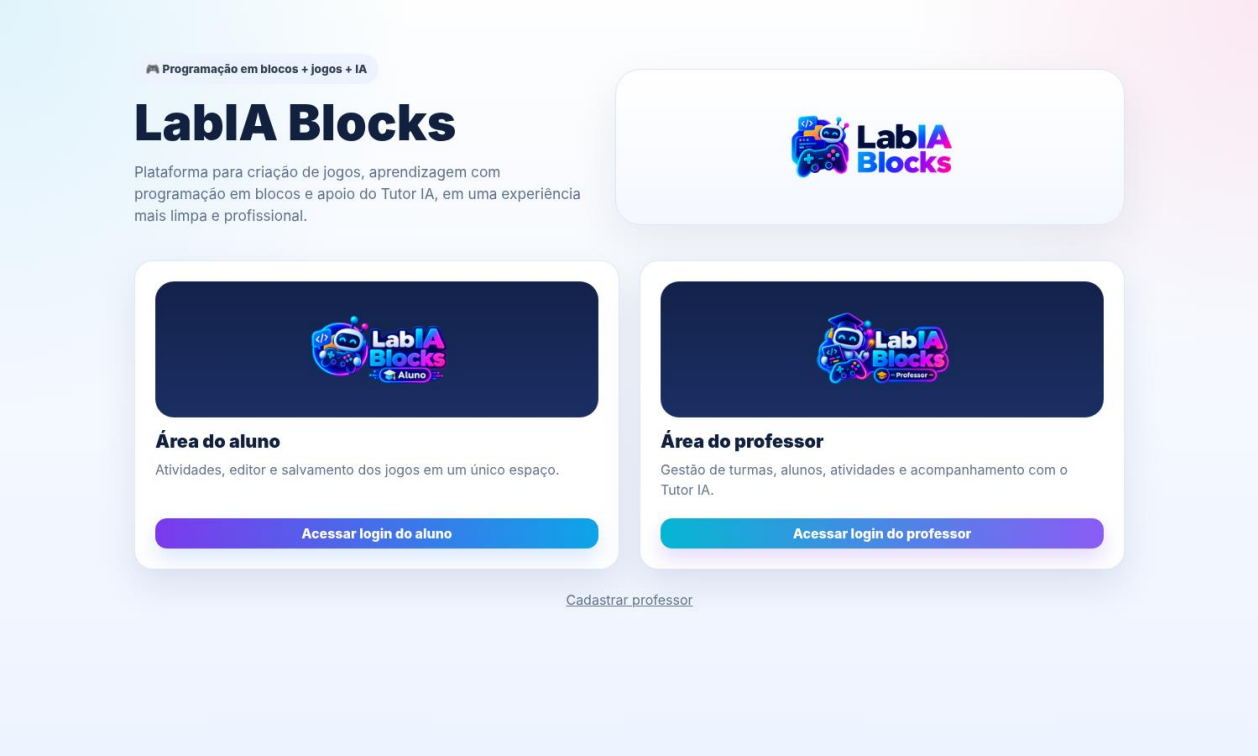
1 INTRODUÇÃO

A aprendizagem de programação nos anos iniciais de contato com a computação envolve uma dupla exigência: compreender conceitos abstratos, como variáveis, eventos, condições, repetição e decomposição de problemas, e dominar simultaneamente a sintaxe de uma linguagem. Essa combinação pode ampliar a carga cognitiva do estudante iniciante e deslocar sua atenção da lógica do problema para detalhes formais de escrita. Nesse contexto, ambientes de programação visual em blocos constituem uma alternativa para tornar a estrutura do programa mais visível e manipulável, sem eliminar a necessidade de compreender relações de causa, ordem, condição e estado. A literatura sobre pensamento computacional destaca justamente a importância de práticas de resolução de problemas, modelagem e construção de soluções como parte da formação em computação (Grover; Pea, 2013, p. 38-43). Ambientes visuais voltados à criação de histórias, animações e jogos ajudaram a consolidar a programação em blocos como estratégia de iniciação. O Scratch, por exemplo, foi concebido para favorecer criação, experimentação e remixagem de projetos, aproximando a programação de atividades autorais e significativas (Resnick et al., 2009, p. 60-67). Em estudo comparativo com estudantes do ensino médio, Weintrop e Wilensky (2018, p. 1-25) observaram ganhos de aprendizagem em ambientes tanto textuais quanto em blocos, com resultados favoráveis ao ambiente em blocos em determinados indicadores de aprendizagem e interesse futuro em computação. Tais evidências não autorizam afirmar superioridade universal dos blocos, mas sustentam seu uso como mediação inicial e como ponte para linguagens textuais.

No desenvolvimento web, a iniciação pode ser enriquecida quando os estudantes percebem funções distintas e complementares de HTML, CSS e JavaScript. O HTML organiza a estrutura e os elementos da página; o CSS define aparência, posicionamento e responsividade; e o JavaScript introduz comportamento, eventos, variáveis, condições e regras do jogo. Essa separação favorece uma leitura modular do sistema e permite relacionar cada bloco visual a um trecho textual equivalente. Ao criar um jogo, o estudante deixa de manipular comandos isolados e passa a articular objetivos, personagens, regras, estados, pontuação, colisões, vitória e derrota em um artefato executável. A criação de jogos também oferece condições para articular programação com metodologias ativas. Em vez de receber apenas uma sequência expositiva, o estudante pode formular um problema, projetar uma solução, testar hipóteses, depurar erros, revisar o projeto e compartilhar um produto. A literatura sobre aprendizagem ativa mostra que estratégias em que o estudante participa cognitivamente da resolução de tarefas podem produzir melhores resultados do que abordagens exclusivamente expositivas em contextos de ciência, tecnologia, engenharia e matemática (Freeman et al., 2014, p. 8410-8415). Ao mesmo tempo, elementos de jogos aplicados a contextos não lúdicos, como desafios, progressão, feedback, conquistas e metas, aproximam-se do conceito de gamificação proposto por Deterding et al. (2011, p. 9-15).

A incorporação recente de inteligência artificial generativa amplia as possibilidades de apoio individual durante atividades de programação, mas também introduz riscos de dependência, respostas incorretas e substituição do raciocínio do estudante. Em revisão sobre IA generativa na educação em computação, Humble (2024) identifica oportunidades relacionadas a apoio ao ensino, personalização, motivação e ampliação de conhecimento, simultaneamente a riscos de imprecisão, excesso de dependência e perda de habilidades. A UNESCO (2024) recomenda que a adoção educacional de IA seja orientada por critérios humanos, éticos e pedagógicos, com atenção à privacidade, à adequação do sistema e ao desenvolvimento da autonomia dos aprendizes. É nesse cenário que se insere a LabIA Blocks. A plataforma foi desenvolvida como ambiente educacional integrado para criação de jogos web por blocos, articulando editor visual, código textual, execução do jogo, recursos de planejamento, atividades estruturadas e Tutor IA. O problema que orienta este estudo pode ser formulado da seguinte maneira: de que modo uma plataforma que integra programação em blocos, desenvolvimento de jogos, gamificação e apoio de IA pode organizar experiências introdutórias de aprendizagem em HTML, CSS e JavaScript? O objetivo geral é descrever o processo de construção e as funcionalidades da LabIA Blocks e analisar suas possibilidades pedagógicas para o ensino inicial de programação e criação de jogos. Como objetivos específicos, busca-se: caracterizar sua arquitetura; identificar recursos destinados ao professor e ao aluno; mapear modalidades de atividade apoiadas pela ferramenta; e discutir qualidades, benefícios potenciais e limitações de seu uso educacional.

Figura 1 – Tela inicial da LabIA Blocks com acesso às áreas do aluno e do professor



Fonte: elaboração própria a partir da versão analisada da LabIA Blocks (2026).

2 METODOLOGIA

2.1 NATUREZA E DELINEAMENTO DA PESQUISA

O trabalho caracteriza-se como pesquisa aplicada, de caráter descritivo e de desenvolvimento tecnológico. O objeto de análise é a própria LabIA Blocks, considerada como artefato digital educacional. Não se realizou experimento com turmas, aplicação de questionários ou coleta de dados com seres humanos nesta etapa; por essa razão, os resultados apresentados referem-se à análise funcional e pedagógica do sistema, e não a evidências de eficácia de aprendizagem em uma população específica. A avaliação empírica com estudantes e professores é indicada como etapa posterior e, quando realizada, deverá observar os procedimentos éticos aplicáveis.

2.2 PROCESSO DE DESENVOLVIMENTO E ARQUITETURA

A versão analisada foi estruturada como aplicação web com PHP 8 ou superior, MySQL/MariaDB, Bootstrap, CSS e JavaScript. O sistema organiza dois perfis principais: professor e aluno. O professor administra turmas, estudantes, atividades, entregas e configurações do Tutor IA. O aluno acessa as atividades publicadas, abre o ambiente de criação, salva projetos e interage com o Tutor IA no contexto do trabalho em andamento. O editor de jogos é incorporado à área do aluno e mantém funcionamento local para a montagem e execução dos blocos, enquanto serviços como autenticação, projetos, atividades e entregas utilizam o banco de dados. A biblioteca de programação é organizada por um catálogo de blocos usado tanto pelo editor quanto pelo mecanismo de validação do Tutor IA. Na versão inspecionada, foram identificados 108 blocos: 16 de HTML, 15 de CSS e 77 de JavaScript. Os blocos HTML contemplam estrutura, textos, botões, campos, imagens, placar, mensagens, canvas, listas e rodapé. Os blocos CSS abrangem página, Flexbox, Grid, painéis, textos, botões, campos, placar, canvas, posicionamento, sombras, efeitos, animações, media queries e CSS personalizado. O conjunto JavaScript reúne eventos, movimento, aparência, som, controle, variáveis, listas, atualização da interface, pontuação, vidas, vitória e derrota, além de mecânicas específicas de plataforma e cenários de castelo.

O Tutor IA é executado no servidor. A chave de API configurada pelo professor não é exposta no HTML do aluno e é armazenada de forma criptografada nas configurações do docente. A lógica do Tutor utiliza o catálogo real de blocos para evitar nomes inventados e foi estruturada para interpretar a solicitação do estudante, associar entidades, eventos, condições, ações e variáveis, verificar ordem causal e produzir uma sequência de blocos com os campos que devem ser preenchidos e seus encaixes. Quando uma mecânica não pode ser comprovada com os blocos disponíveis, o sistema foi projetado para indicar a limitação em vez de apresentar uma montagem como correta.

2.3 PROCEDIMENTOS DE ANÁLISE FUNCIONAL

A análise foi realizada sobre o pacote de atualização disponibilizado em 13 de agosto de 2026. Foram examinados os arquivos de configuração, o catálogo de blocos, as páginas das áreas de professor e aluno, o editor visual, os módulos de atividades, projetos, entregas e Tutor IA, além das rotinas de planejamento em Portugol, desafios e jogos prontos. Em seguida, as principais interfaces foram executadas localmente para registro de telas e verificação das opções visíveis ao usuário. As funcionalidades foram classificadas segundo quatro dimensões: iniciação à programação, criação de jogos, gestão pedagógica e apoio/gamificação. A discussão pedagógica foi construída por aproximação entre as funcionalidades efetivamente presentes no sistema e literatura sobre pensamento computacional, programação em blocos, aprendizagem ativa, gamificação e IA generativa na educação. Por não se tratar de avaliação com usuários, expressões como benefício, qualidade e potencial são empregadas em sentido de possibilidade pedagógica fundamentada no desenho da ferramenta, e não como efeito causal já comprovado.

Quadro 1 – Componentes técnicos e funções educacionais da LabIA Blocks

Componente	Função na ferramenta	Possível contribuição pedagógica
Editor de blocos	Montagem visual de HTML, CSS e JavaScript	Reduzir barreiras sintáticas iniciais e tornar a lógica manipulável.
Código gerado	Exibir o código textual correspondente aos blocos	Apoiar transição entre representação visual e linguagem textual.
Preview do jogo	Executar e testar o projeto no próprio ambiente	Favorecer ciclo construir-testar-depurar-revisar.
Tutor IA	Responder dúvidas, explicar, dar pistas e analisar lógica	Oferecer apoio contextual e feedback durante a resolução de problemas.
Portugol	Planejar algoritmos e relacioná-los aos blocos	Estimular decomposição e organização lógica antes da implementação.
Desafios	Sequência progressiva com objetivos verificáveis	Estruturar progressão, feedback e metas de aprendizagem.
Jogos prontos	Modelos completos carregáveis e modificáveis	Permitir análise, remixagem e aprendizagem por exemplos.
Portal professor/aluno	Gerenciar atividades, projetos, entregas e feedback	Integrar autoria do aluno e acompanhamento docente.

Fonte: elaboração própria (2026).

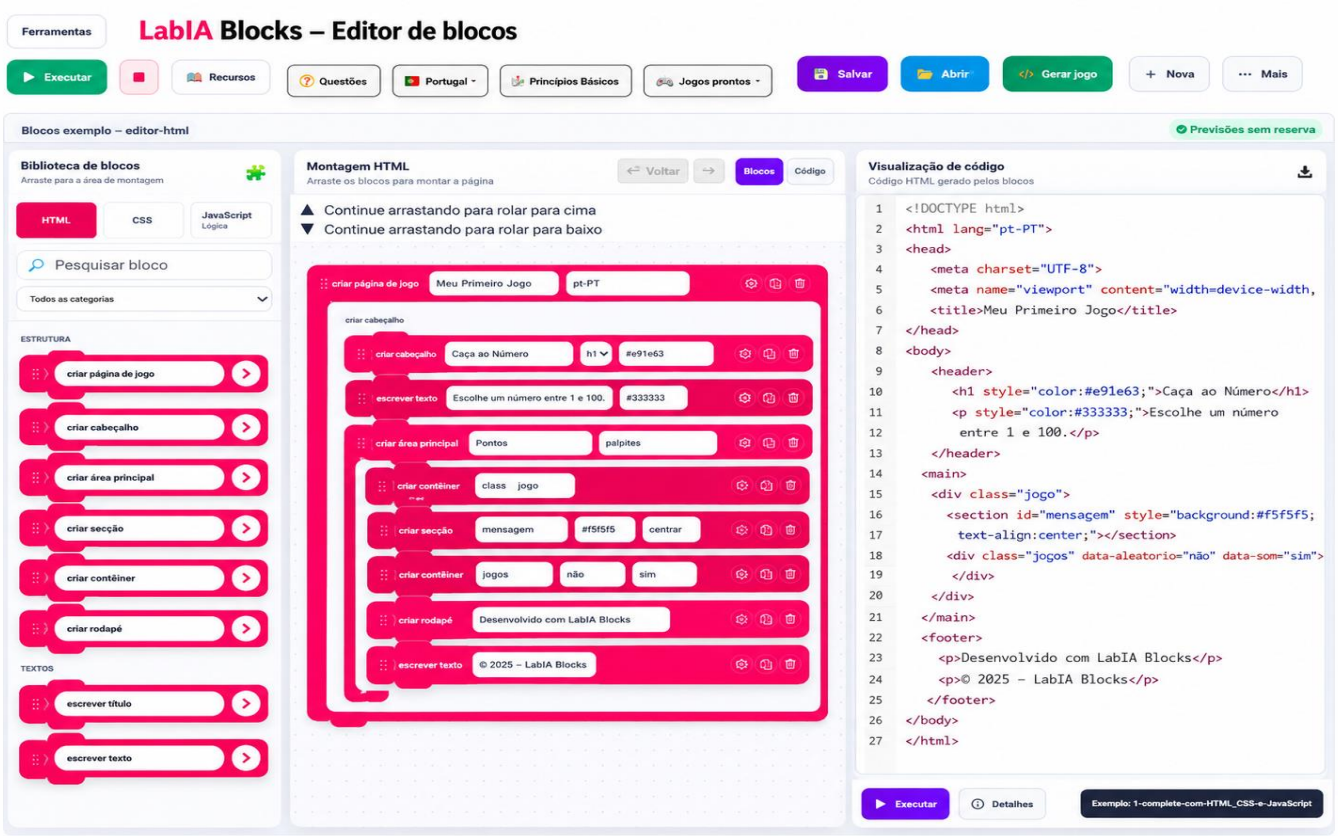
3 RESULTADOS E DISCUSSÃO

3.1 AMBIENTE DE CRIAÇÃO E ORGANIZAÇÃO DAS LINGUAGENS

A área de criação apresenta três espaços complementares: biblioteca de blocos, área de montagem e visualização do jogo. O estudante seleciona HTML, CSS ou JavaScript e trabalha com blocos diferenciados por linguagem e categoria. A separação visual reforça a função de cada camada do jogo web: estrutura em HTML, aparência em CSS e comportamento em JavaScript. A possibilidade de alternar entre a visualização por blocos e o código textual cria uma ponte didática particularmente relevante para turmas iniciantes, pois o aluno pode observar como uma decisão tomada visualmente se traduz em sintaxe. Os blocos possuem campos editáveis para valores como identificadores, classes, textos, teclas, coordenadas,

tamanhos, cores, nomes de variáveis, operadores e quantidades. Essa característica evita que a programação em blocos se reduza a encaixes mecânicos: para obter o comportamento esperado, o estudante precisa compreender o significado dos parâmetros. Em uma atividade de movimento, por exemplo, o evento de tecla deve ser associado a uma alteração coerente de X ou Y; em uma atividade de coleta, nomes de personagens, variáveis e condições de colisão precisam estar relacionados. O sistema também oferece desfazer/refazer, pesquisa de blocos, filtro por categorias, execução, reinício, geração do jogo em HTML e salvamento/abertura de projetos em JSON.

Figura 2 – Programação em blocos e visualização do código correspondente em HTML, CSS e JavaScript



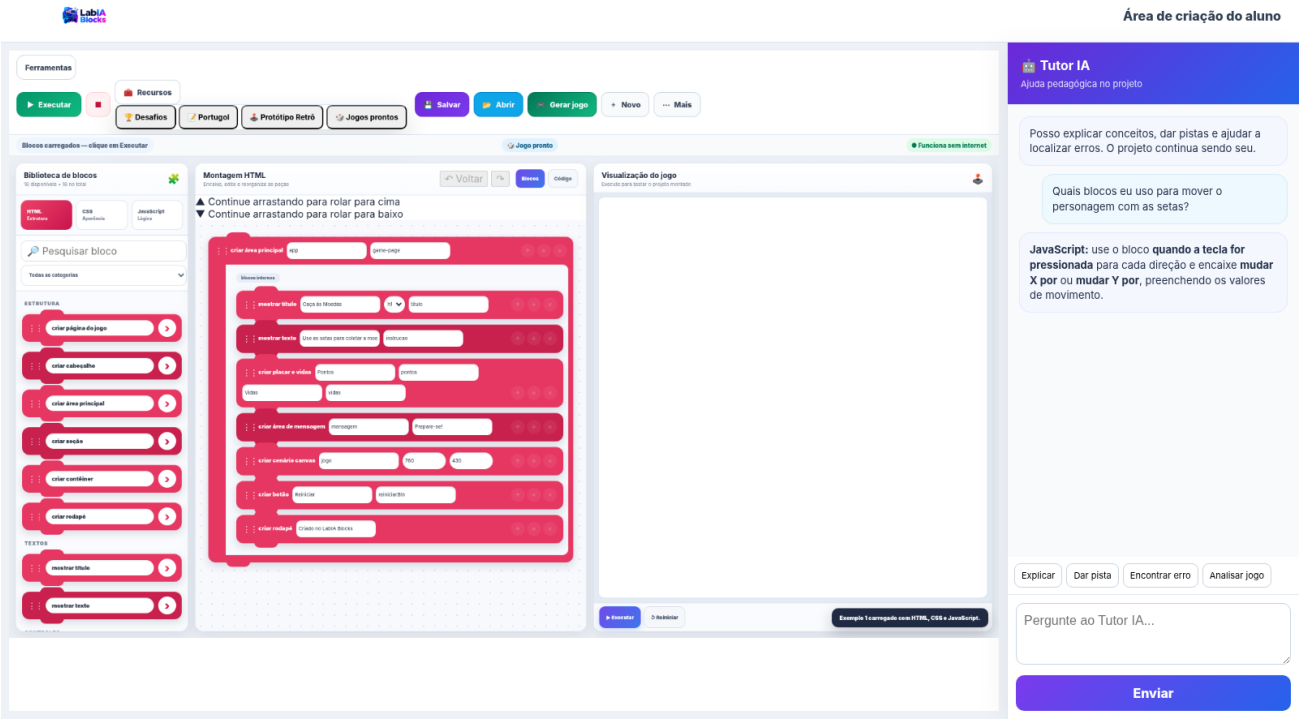
Fonte: elaboração própria a partir da versão analisada da LabIA Blocks (2026).

A presença simultânea de blocos, código e resultado visual pode apoiar uma abordagem de aprendizagem por experimentação. O estudante modifica um valor, executa novamente o jogo e observa a consequência. Esse ciclo aproxima-se das práticas de construção e depuração discutidas em ambientes criativos de programação (Resnick et al., 2009, p. 60-67) e evita que o código seja apresentado apenas como sequência abstrata de comandos. A ferramenta não elimina a complexidade da programação; ela reorganiza essa complexidade em etapas mais visíveis e manipuláveis.

3.2 TUTOR IA COMO APOIO PEDAGÓGICO CONTEXTUAL

O Tutor IA constitui um dos diferenciais da plataforma. Ele aparece integrado à área de criação e oferece ações rápidas para explicar o projeto, fornecer pistas, procurar possíveis erros e analisar a jogabilidade. O aluno também pode formular perguntas livres. O objetivo pedagógico declarado na interface é ajudar a compreender conceitos e localizar problemas sem retirar a autoria do estudante. Esse desenho é coerente com uma perspectiva de IA como mediadora e não como substituta da atividade intelectual do aprendiz. Para perguntas sobre construção de jogos, o Tutor utiliza o catálogo de blocos da ferramenta. A resposta esperada não se limita a recomendar genericamente HTML, CSS ou JavaScript: ela deve apresentar os nomes reais dos blocos, a ordem de uso, os campos que precisam ser preenchidos, valores sugeridos e relações de encaixe. O mecanismo de validação procura relacionar frases do pedido a um contrato lógico, identificando entidades, gatilhos, condições, ações, estados, vitória e derrota. Essa opção é importante em contexto escolar, pois reduz a discrepância entre a orientação textual da IA e aquilo que o estudante de fato encontra na interface.

Figura 3 – Exemplo da área de criação com Tutor IA integrado ao projeto do aluno



Fonte: elaboração própria a partir da versão analisada da LabIA Blocks (2026).

A literatura recente recomenda prudência nessa integração. Humble (2024) identifica a IA generativa como possível assistente de ensino e aprendizagem e como recurso para personalização e motivação, mas também chama atenção para imprecisão, dependência excessiva, dificuldade em tarefas complexas e perda de habilidades. A UNESCO (2024) enfatiza validação pedagógica, proteção de dados e

centralidade humana. Na LabIA Blocks, duas escolhas dialogam com essas preocupações: a chave da API permanece no servidor, e o professor pode controlar a disponibilidade e o nível de ajuda da IA em cada atividade. Ainda assim, a qualidade das respostas deve continuar sendo monitorada, sobretudo porque sistemas generativos não garantem correção absoluta em toda situação.

3.3 PLANEJAMENTO EM PORTUGOL E TRANSIÇÃO PARA OS BLOCOS

O módulo de Portugol introduz uma etapa anterior à montagem do jogo. O aluno pode escrever um algoritmo em linguagem estruturada, validar comandos, visualizar relações com os blocos e gerar um roteiro. O recurso contém exemplos de comandos como iniciar/fim, criar personagem, criar plataforma, criar estrela, aplicar gravidade, mover personagem e vencer o jogo. Também permite gerar PDF dos blocos relacionados e converter comandos reconhecidos em árvores de blocos do projeto.

Figura 4 – Módulo de planejamento em Portugol da LabIA Blocks

Fonte: elaboração própria a partir da versão analisada da LabIA Blocks (2026).

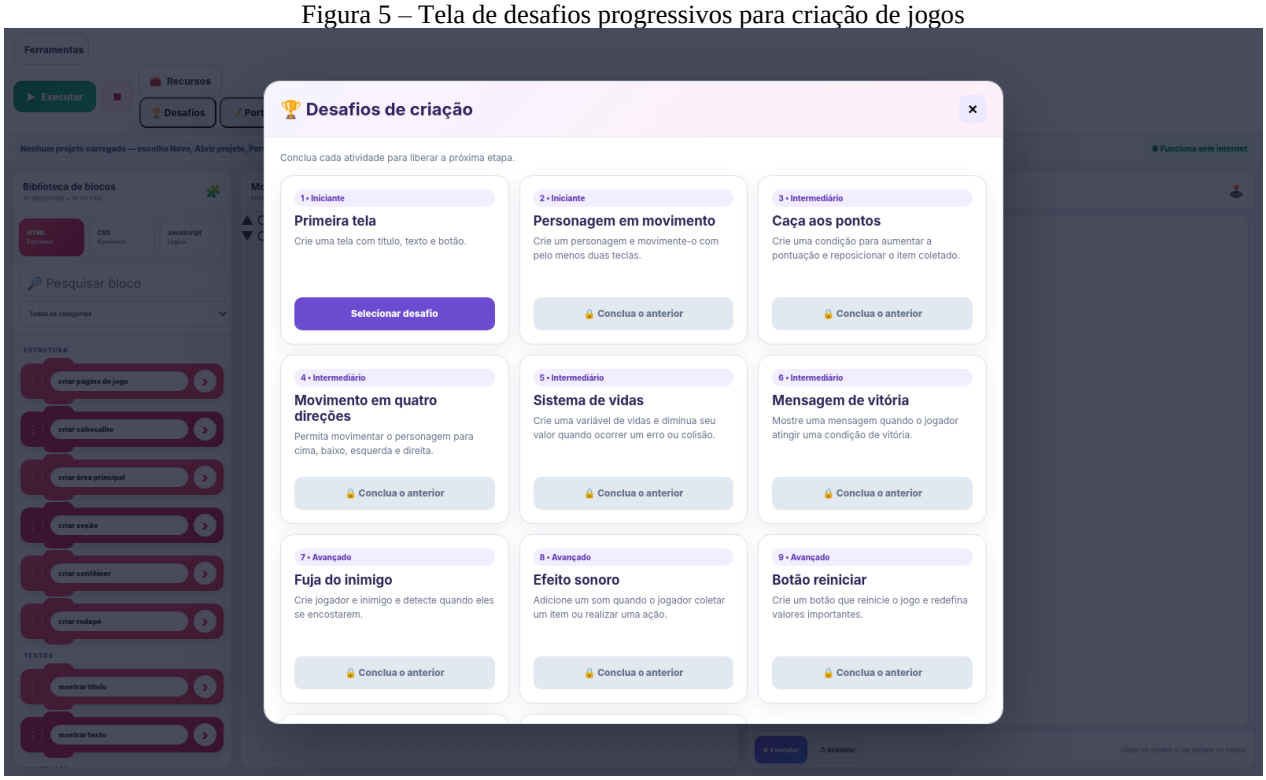
Pedagogicamente, o Portugol pode funcionar como camada intermediária entre a narrativa do problema e a implementação visual. Em vez de iniciar diretamente pela interface, o estudante é convidado a explicitar uma sequência lógica. Essa prática favorece decomposição do problema e antecipação de condições antes da implementação. Quando associada à alternância entre blocos e código, a ferramenta oferece três representações do mesmo raciocínio: descrição algorítmica, montagem visual e código textual.

Essa multiplicidade pode ser explorada pelo professor para comparar formas de representar uma solução e discutir equivalências e limitações.

3.4 DESAFIOS, PROGRESSÃO E GAMIFICAÇÃO

A LabIA Blocks apresenta uma sequência de desafios graduais. Na versão analisada, foram identificadas atividades como Primeira tela, Personagem em movimento, Caça aos pontos, Movimento em quatro direções, Sistema de vidas, Mensagem de vitória, Fuja do inimigo, Efeito sonoro, Botão reiniciar, Jogo completo e Aventura das Estrelas. Cada desafio contém objetivos verificáveis, e o próximo pode ser condicionado à conclusão do anterior. A plataforma registra conclusão e execução, apresenta feedback de objetivos atingidos ou pendentes e disponibiliza conquistas relacionadas ao uso do ambiente.

Figura 5 – Tela de desafios progressivos para criação de jogos



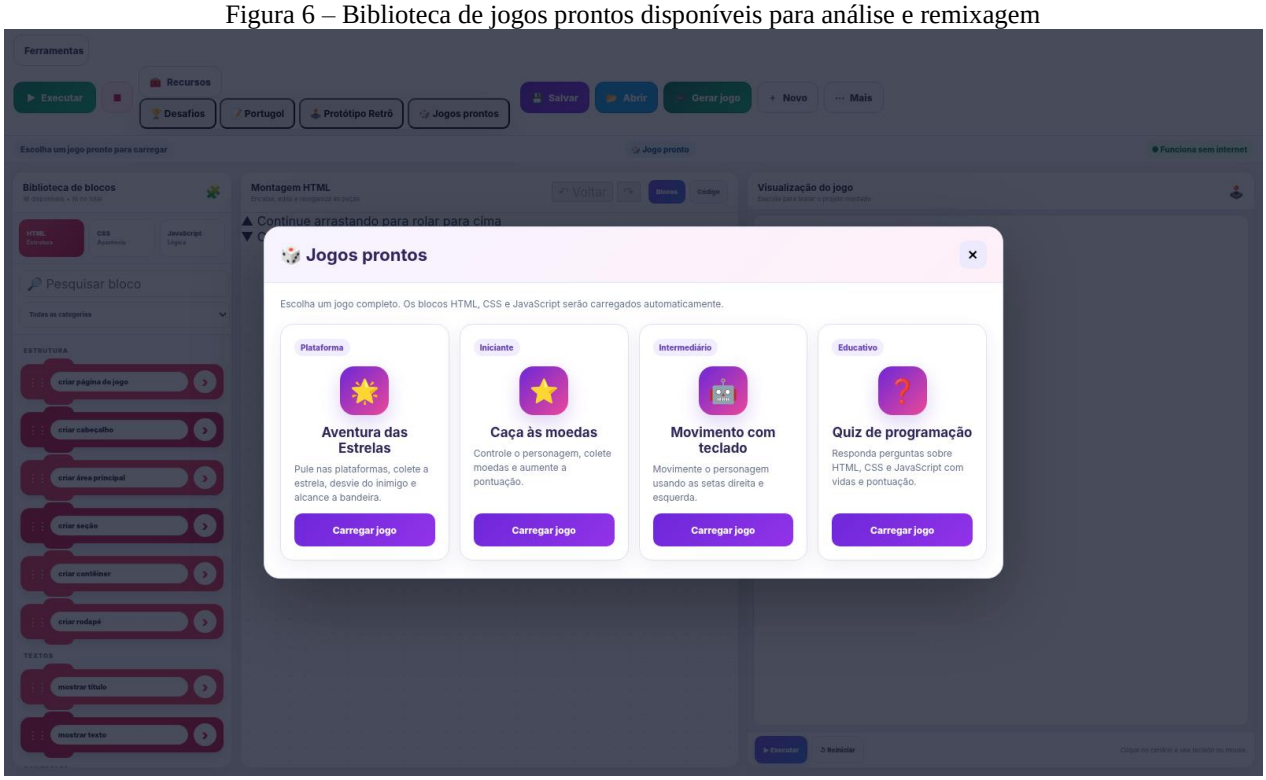
Fonte: elaboração própria a partir da versão analisada da LabIA Blocks (2026).

Esses elementos configuram uma aplicação de gamificação porque incorporam progressão, metas, feedback e conquistas a uma atividade educacional que não é, em si, apenas jogar, mas aprender a programar. Na definição de Deterding et al. (2011, p. 9-15), gamificação corresponde ao emprego de elementos de design de jogos em contextos não lúdicos. O valor pedagógico, entretanto, depende de como esses elementos são integrados ao objetivo de aprendizagem. Na LabIA Blocks, o desafio é considerado concluído pela presença e execução de elementos de programação, o que permite vincular a recompensa à construção efetiva de uma solução, e não somente à participação superficial.

3.5 JOGOS PRONTOS, REMIXAGEM E CRIAÇÃO AUTORAL

O recurso Jogos prontos disponibiliza modelos completos que podem ser carregados automaticamente. A versão analisada apresenta Aventura das Estrelas, Caça às moedas, Movimento com teclado e Quiz de programação. Os modelos abrangem níveis e mecânicas diferentes, permitindo ao estudante partir de uma implementação funcional para observar blocos, alterar parâmetros, remover componentes e criar variações. Essa abordagem pode ser utilizada como atividade de leitura de código, remixagem, comparação de estratégias ou desafio de extensão.

Figura 6 – Biblioteca de jogos prontos disponíveis para análise e remixagem



Fonte: elaboração própria a partir da versão analisada da LabIA Blocks (2026).

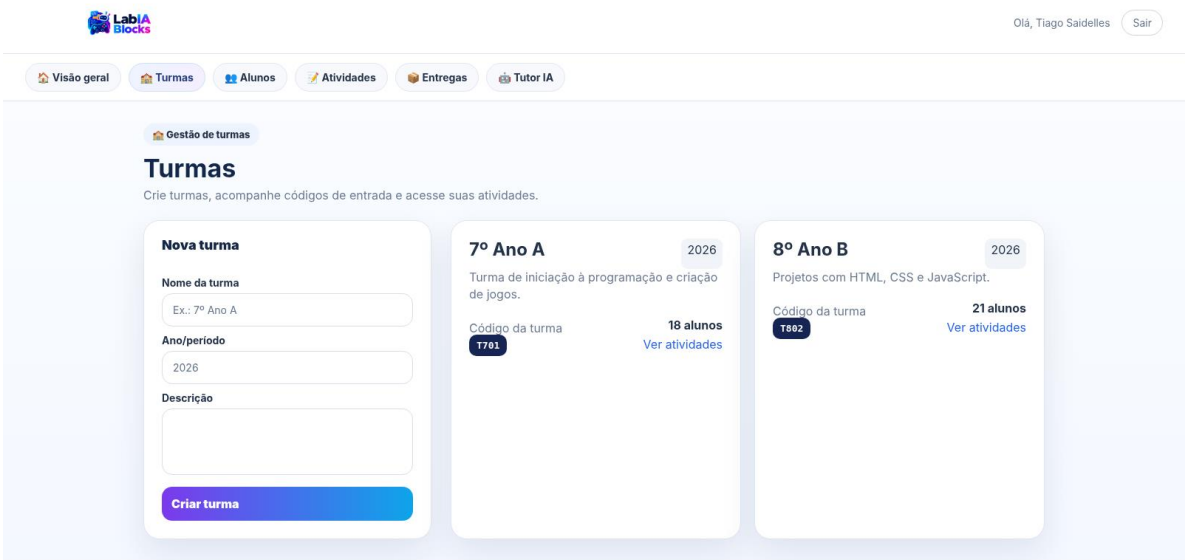
O uso de exemplos completos é particularmente relevante para estudantes que ainda não conseguem planejar um jogo do zero. Ao invés de transformar o modelo em resposta final, o professor pode convertê-lo em objeto de investigação: identificar quais blocos criam a interface, onde a pontuação é atualizada, como a colisão é detectada, quais condições determinam vitória e derrota ou como uma regra pode ser modificada. A lógica de criação e remixagem está alinhada à perspectiva de fluência digital que enfatiza produzir e transformar artefatos, e não apenas consumir tecnologia (Resnick et al., 2009, p. 60-67).

3.6 FUNCIONALIDADES DA ÁREA DO PROFESSOR

A área do professor organiza o uso da ferramenta como ambiente de aula. O docente pode criar turmas com código de acesso, cadastrar e gerenciar estudantes, publicar atividades e acompanhar entregas.

Na criação de uma atividade são informados turma, título, instruções e prazo. Também é possível selecionar o nível de ajuda da IA em cinco níveis: 0 – Sem IA; 1 – Perguntas; 2 – Pistas; 3 – Tutor; e 4 – Laboratório. Essa gradação permite adaptar o apoio ao objetivo da proposta, restringindo a assistência quando se deseja maior autonomia ou ampliando-a em situações de exploração orientada. As entregas dos estudantes são vinculadas a projetos e atividades. O professor pode registrar nota e comentário de feedback, alterando o status da submissão para revisada. Há ainda uma área específica para configuração da chave da OpenAI, seleção do modelo e habilitação geral do Tutor IA. A arquitetura mantém a chave fora do navegador do aluno, o que representa uma qualidade técnica importante para uso institucional, embora a escola ainda deva adotar políticas próprias de segurança, privacidade e governança de dados.Do ponto de vista didático, a gestão docente pode ser compreendida como um fluxo articulado em três movimentos: criar a turma, cadastrar os estudantes e, em seguida, publicar a atividade que orientará a experiência de aprendizagem. Essa organização permite que o professor transforme uma proposta de programação em uma sequência acompanhável, vinculando cada estudante a um grupo, cada desafio a uma turma e cada entrega a uma atividade específica. Em vez de utilizar o editor apenas como recurso isolado, a plataforma passa a funcionar como ambiente de planejamento, desenvolvimento e acompanhamento da aula.

Figura 7 – Cadastro e organização de turmas no painel do professor

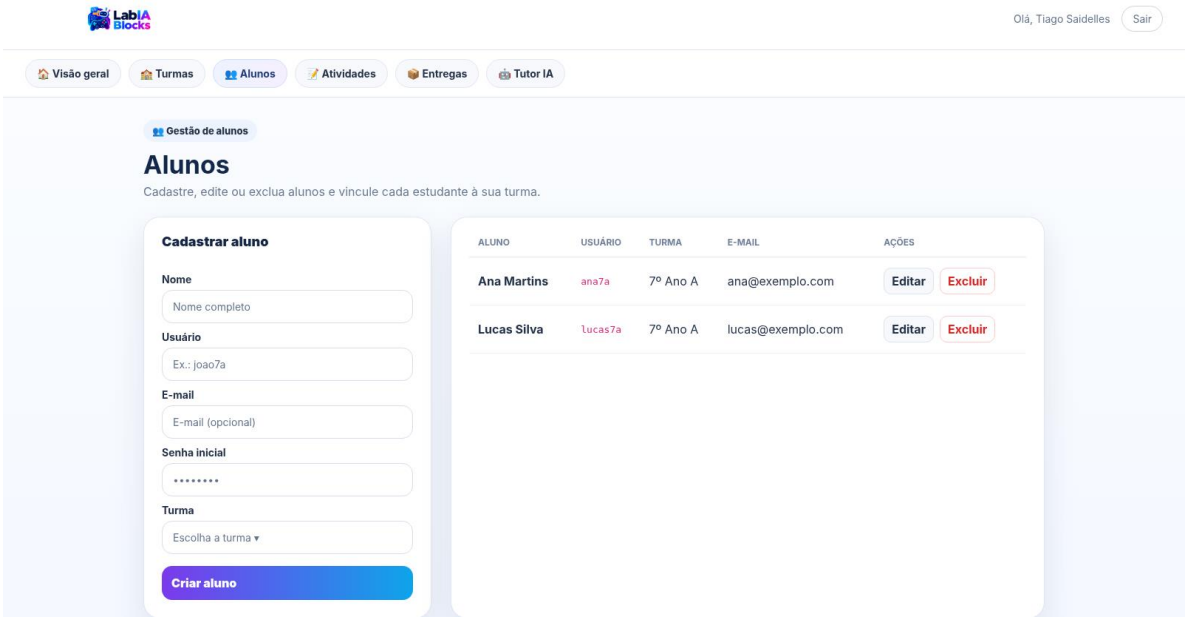


Fonte: elaboração própria a partir da versão analisada da LabIA Blocks (2026).

A criação de turma, apresentada na Figura 7, estabelece a unidade organizadora do trabalho. O professor informa nome, período e descrição e o sistema gera um código associado à turma. Em uma aplicação didática, esse recurso pode ser utilizado para separar grupos por ano, oficina, projeto ou nível de complexidade. A turma também funciona como referência para a distribuição de atividades e para o

acompanhamento das produções, permitindo que uma mesma ferramenta seja empregada em diferentes contextos sem misturar os registros dos estudantes.

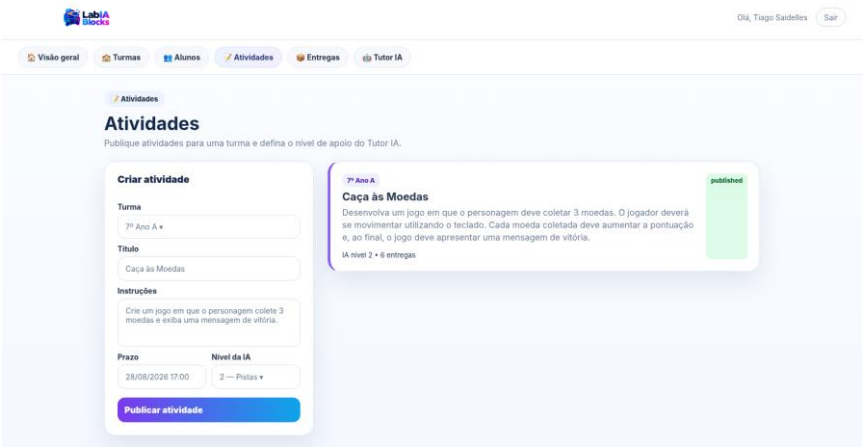
Figura 8 – Cadastro, vinculação e gerenciamento de estudantes



Fonte: elaboração própria a partir da versão analisada da LabIA Blocks (2026).

Na etapa seguinte, o docente cadastra o estudante, define usuário e senha inicial e seleciona a turma à qual ele será vinculado, conforme a Figura 8. A possibilidade de editar ou excluir registros facilita a manutenção da organização da turma ao longo do período letivo. Pedagogicamente, esse cadastro individual viabiliza percursos acompanhados: cada aluno acessa seu espaço, recebe a atividade correspondente e desenvolve seus projetos dentro de uma estrutura que permite ao professor observar entregas e oferecer devolutivas. Assim, a gestão de usuários deixa de ser apenas uma função administrativa e passa a sustentar a personalização e o acompanhamento formativo.

Figura 9 – Publicação de atividade com turma, instruções, prazo e nível de apoio da IA



Fonte: elaboração própria a partir da versão analisada da LabIA Blocks (2026).

A publicação de atividades completa esse fluxo. Como mostra a Figura 9, o professor seleciona a turma, define um título, descreve a missão, estabelece prazo e escolhe o nível de apoio do Tutor IA. Essa configuração permite adequar a mediação à intencionalidade pedagógica. Em uma atividade inicial, por exemplo, pode-se oferecer um nível maior de orientação para que os estudantes reconheçam a função dos blocos; em atividades posteriores, o apoio pode ser reduzido para ampliar a tomada de decisão e a autoria. O enunciado também pode explicitar critérios de sucesso, como movimentar um personagem, utilizar uma variável, construir uma condição de vitória e testar o jogo antes da entrega. Dessa forma, a própria criação da atividade já organiza objetivos de aprendizagem, critérios de realização e grau de suporte. Em sala de aula, uma estratégia possível consiste em o professor preparar previamente a turma e os acessos, publicar uma missão curta e iniciar a aula com a leitura coletiva do problema. Os estudantes podem então registrar a lógica em Portugol, construir a estrutura em HTML, trabalhar a apresentação em CSS e finalizar a interação em JavaScript. Durante o desenvolvimento, o docente acompanha dificuldades recorrentes e decide quando intervir diretamente ou quando orientar o estudante a consultar o Tutor IA. Ao final, a entrega pode ser acompanhada de uma explicação da solução adotada, permitindo avaliar não apenas se o jogo funciona, mas se o estudante compreende a relação entre os blocos utilizados, os valores configurados e o comportamento produzido no jogo.

Quadro 2 – Exemplos de atividades pedagógicas possíveis com a plataforma

Tipo de atividade	Ação do professor	Ação do estudante	Conceitos mobilizados
Montagem guiada	Fornecer objetivo e limitar o nível de ajuda da IA.	Montar a sequência indicada e testar o resultado.	Estrutura HTML, estilo CSS, eventos e variáveis.
Desafio de depuração	Entregar um projeto com erro ou regra incompleta.	Usar execução, código e Tutor IA para localizar a falha.	Depuração, condição, estado e causalidade.
Projeto de jogo	Definir tema, critérios e prazo.	Planejar, criar, testar, revisar e entregar um jogo.	Decomposição, lógica, autoria e design.
Portugol para blocos	Solicitar algoritmo antes da implementação.	Escrever a lógica e convertê-la para blocos.	Algoritmos, sequência, condição e repetição.
Remixagem	Selecionar um jogo pronto e propor alterações.	Ler a estrutura existente e modificar mecânicas.	Leitura de código, reutilização e criatividade.
Desafios gamificados	Definir sequência de desafios e critérios.	Cumprir metas progressivas e acompanhar feedback.	Progressão, autonomia e metacognição.
Quiz criado pelo aluno	Solicitar jogo de perguntas sobre conteúdo curricular.	Programar perguntas, alternativas, pontuação e vidas.	Eventos, condicionais, interface e avaliação formativa.
Apresentação de projeto	Solicitar explicação do funcionamento do jogo.	Exibir blocos, código e decisões de projeto.	Comunicação, justificativa e pensamento computacional.

Fonte: elaboração própria (2026).

3.7 QUALIDADES E BENEFÍCIOS PEDAGÓGICOS POTENCIAIS

A primeira qualidade identificada é a coerência entre representação visual e linguagem textual. O ambiente não apresenta programação em blocos como universo separado de HTML, CSS e JavaScript; ao contrário, organiza os blocos segundo essas linguagens e permite visualizar o código produzido. Essa

característica pode reduzir a ruptura normalmente percebida quando o estudante migra de um ambiente exclusivamente visual para uma linguagem de programação escrita. A literatura sobre ambientes em blocos sugere que representações visuais podem favorecer a aprendizagem inicial ao reduzir erros de sintaxe e apoiar reconhecimento de estruturas, desde que exista planejamento para a transição e para a compreensão conceitual (Weintrop; Wilensky, 2018, p. 1-25). A segunda qualidade é a aprendizagem orientada a produto. O estudante não monta sequências abstratas apenas para obter uma saída numérica; ele constrói uma experiência interativa que possui cenário, personagem, regras e objetivos. O resultado visível oferece significado imediato às alterações. Tal dinâmica pode favorecer motivação e persistência, especialmente quando o professor organiza o trabalho como projeto com metas intermediárias, revisão e apresentação final. Essa interpretação é compatível com abordagens de aprendizagem ativa nas quais o estudante participa da construção da solução e recebe feedback durante o processo (Freeman et al., 2014, p. 8410-8415).

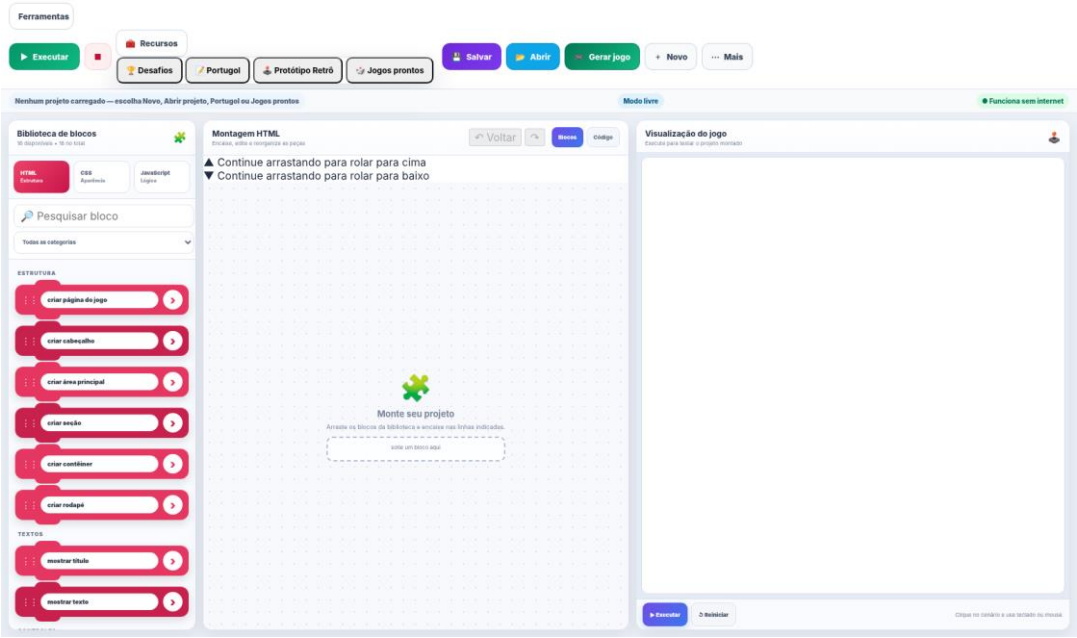
A terceira qualidade é a possibilidade de diferenciação pedagógica. O mesmo ambiente pode ser utilizado em modo livre, por desafios, por modelos prontos, por planejamento em Portugol ou dentro de uma atividade definida pelo professor. Além disso, o nível de auxílio da IA pode variar entre atividades. Em uma turma heterogênea, o docente pode propor o mesmo objetivo com diferentes graus de apoio: alguns estudantes trabalham a partir de um exemplo pronto, outros recebem apenas pistas e outros são desafiados a criar a solução desde o início. A quarta qualidade refere-se ao feedback. O aluno dispõe de execução imediata do jogo, mensagens de desafio, análise do Tutor IA e possibilidade de comparar blocos e código. O professor, por sua vez, acompanha entregas e registra avaliação e comentários. Essa combinação cria diferentes camadas de feedback: do próprio sistema, da IA e do docente. O potencial educacional da IA, contudo, depende de supervisão e de desenho de tarefas que exijam explicação, revisão e justificativa, evitando que a resposta automática substitua a compreensão. Essa cautela acompanha os riscos destacados por Humble (2024) e as recomendações de uso humano e pedagogicamente validado da UNESCO (2024). A quinta qualidade é a presença de elementos de gamificação integrados ao processo de aprendizagem. Desafios desbloqueáveis, progresso, conquistas e jogos prontos tornam a experiência mais próxima de um percurso de missão. O uso pedagógico desses elementos pode favorecer engajamento quando o progresso representa objetivos de aprendizagem claros. A ferramenta, portanto, oferece ao professor condições para empregar gamificação sem reduzir a atividade a pontos e medalhas, pois as metas verificadas estão associadas a ações concretas de programação.

3.8 ESTRATÉGIA DIDÁTICA PARA UTILIZAÇÃO DA LABIA BLOCKS EM SALA DE AULA

A utilização pedagógica da LabIA Blocks pode ser organizada como um percurso de construção progressiva, no qual o estudante passa da compreensão do problema para a representação algorítmica e,

posteriormente, para a implementação do jogo. O ponto de partida recomendado não é a escolha aleatória de blocos, mas a definição de uma missão clara: o que o jogador precisa fazer, quais objetos existem, quais regras determinam sucesso ou fracasso e quais ações devem ocorrer diante de cada evento. Essa etapa inicial favorece a decomposição do problema e aproxima a atividade de práticas de pensamento computacional, nas quais problemas são organizados em partes menores, relações e procedimentos antes da implementação (Grover; Pea, 2013, p. 38-43). Na preparação da aula, o professor pode organizar previamente a turma, cadastrar os estudantes e publicar uma atividade vinculada a esse grupo. A atividade pode apresentar o desafio, o produto esperado, o prazo, os critérios de entrega e o nível de apoio do Tutor IA. Essa etapa é importante porque explicita a intencionalidade pedagógica antes que o aluno abra o editor. Em vez de iniciar pela escolha aleatória de blocos, o estudante recebe uma missão com objetivos definidos e pode planejar como representar a solução em HTML, CSS e JavaScript. O professor, por sua vez, pode graduar a complexidade das tarefas e acompanhar a evolução das entregas ao longo de uma sequência didática. Para o estudante, uma sequência didática possível inicia-se pela leitura da missão e pelo planejamento em Portugol. Nessa fase, busca-se registrar ações em linguagem próxima da fala, como iniciar o jogo, criar personagem, mover ao pressionar uma tecla, verificar colisão, alterar pontuação e testar a condição de vitória. Em seguida, o aluno acessa o editor e começa pela estrutura HTML, selecionando blocos que organizam a página, o cenário, títulos, textos, placar e demais elementos necessários. O passo posterior concentra-se no CSS, quando são discutidos aparência, organização espacial, dimensões e identidade visual. Somente depois dessa base o estudante implementa, em JavaScript, eventos, variáveis, condições, repetições, colisões e regras do jogo.

Figura 10 – Editor em estado inicial: biblioteca de blocos, área de montagem e visualização do jogo



Fonte: elaboração própria a partir da versão analisada da LabIA Blocks (2026).

A Figura 10 evidencia uma característica importante para a organização da aula: biblioteca de blocos, área de montagem e espaço de visualização permanecem simultaneamente disponíveis. Essa disposição favorece um ciclo curto de ação e feedback. O estudante escolhe um bloco, configura seus valores, encaixa-o na sequência, executa o projeto e observa a consequência. Caso o comportamento seja diferente do esperado, pode retornar à montagem e revisar a estrutura. O erro, nesse contexto, pode ser trabalhado como parte do processo de aprendizagem e não apenas como resultado negativo. O professor pode explorar esse ciclo pedindo que o estudante formule hipóteses antes de cada alteração e explique posteriormente por que determinada mudança produziu o resultado observado. O Tutor IA pode ser incorporado principalmente nos momentos de dúvida, depuração e revisão. Um uso didaticamente produtivo consiste em solicitar explicações sobre a função de determinado bloco, pedir pistas para localizar um erro ou comparar duas formas de implementar uma mecânica. A resposta recebida deve ser testada no próprio editor e confrontada com o comportamento do jogo. Desse modo, a IA integra um processo de investigação e não substitui a execução. A UNESCO (2024) recomenda que o emprego de IA generativa na educação seja orientado por finalidade pedagógica, supervisão humana e desenvolvimento de capacidades críticas; na LabIA Blocks, essas condições podem ser operacionalizadas por meio do controle docente sobre os níveis de assistência e pela exigência de validação prática das sugestões.

Ao final de uma atividade, recomenda-se que o estudante apresente não apenas o jogo funcionando, mas também a lógica empregada. Ele pode explicar quais blocos estruturam a interface, quais eventos recebem comandos do teclado, quais variáveis armazenam estados, como as condições controlam vitória ou derrota e quais alterações foram realizadas durante a depuração. Essa apresentação transforma o produto digital em evidência do processo de aprendizagem e permite ao professor avaliar compreensão, autoria e capacidade de justificar decisões. A prática também ajuda a reduzir uma dificuldade recorrente em tarefas com inteligência artificial: distinguir uma solução apenas copiada de uma solução compreendida e modificada pelo estudante.

3.9 PROPOSTA DE SEQUÊNCIA DE ATIVIDADES E AVALIAÇÃO FORMATIVA

A LabIA Blocks permite estruturar uma sequência de atividades que parte de tarefas curtas e avança para projetos autorais. Em uma etapa inicial, o professor pode propor reconhecimento da interface e exploração orientada de blocos HTML, CSS e JavaScript. O objetivo não é construir imediatamente um jogo complexo, mas compreender a função de cada grupo de comandos. Uma atividade simples pode solicitar a criação de um título, uma mensagem e um cenário; outra pode pedir a alteração de cores e dimensões; uma terceira pode introduzir movimento por teclado. Cada tarefa acrescenta uma camada de complexidade e fornece repertório para o projeto seguinte. Em uma segunda etapa, desafios de lógica podem trabalhar eventos, variáveis e condições de forma isolada. O estudante pode criar um contador, fazer

um objeto desaparecer após uma colisão, limitar o número de vidas ou encerrar o jogo quando uma variável atingir determinado valor. A progressão por pequenas metas tende a tornar visíveis relações de causa e efeito que mais tarde serão combinadas em mecânicas maiores. Essa organização é compatível com metodologias ativas porque o aluno atua sobre problemas concretos, testa soluções e recebe feedback durante o percurso, características associadas a melhores resultados em contextos de aprendizagem ativa quando comparadas à exposição exclusivamente passiva (Freeman et al., 2014, p. 8410-8415). A Figura 9 exemplifica como o professor pode transformar uma competência de programação em missão de aprendizagem. Em vez de solicitar genericamente que o aluno "faça um jogo", a atividade pode explicitar mecânica, critérios de funcionamento e elementos obrigatórios. Um desafio de coleta, por exemplo, pode exigir movimentação por teclado, três itens independentes, atualização de pontuação e mensagem de vitória após a última coleta. A descrição objetiva favorece a avaliação, pois os mesmos requisitos utilizados para orientar a construção podem compor uma rubrica de verificação. O nível de apoio da IA, definido no momento da publicação, torna-se parte do planejamento didático e pode variar conforme a complexidade e o objetivo de autonomia.

Quadro 3 – Sequência didática sugerida para iniciação à programação e criação de jogos

Etapa	Objetivo didático	Ação do estudante	Evidência para avaliação
1. Planejar	Decompor o jogo em entidades, regras e objetivos.	Descrever a missão e representar a lógica em Portugol.	Sequência coerente de ações e condições.
2. Estruturar com HTML	Compreender a organização dos elementos da interface.	Montar cenário, textos, placar e componentes necessários.	Estrutura compatível com o jogo proposto.
3. Estilizar com CSS	Relacionar propriedades visuais à experiência do jogo.	Configurar cores, dimensões, disposição e aparência.	Interface legível e coerente com a proposta.
4. Programar com JavaScript	Implementar comportamento, eventos e estados.	Criar movimento, variáveis, colisões e regras.	Mecânicas executadas conforme os requisitos.
5. Testar e depurar	Desenvolver análise de erros e revisão.	Executar, localizar falhas, solicitar pistas e corrigir.	Registro ou explicação das alterações realizadas.
6. Apresentar e remixar	Consolidar autoria e capacidade de explicação.	Demonstrar o jogo, justificar blocos e propor melhorias.	Apresentação, funcionamento e reflexão sobre o projeto.

Fonte: elaboração própria (2026).

O Quadro 3 pode ser empregado em uma sequência de encontros ou adaptado a uma oficina concentrada. Em turmas iniciantes, as três linguagens podem ser introduzidas de modo progressivo, evitando que a quantidade de blocos disponíveis se transforme em sobrecarga. Em turmas com maior experiência, o professor pode propor um único projeto e utilizar as etapas como marcos de acompanhamento. A flexibilidade é especialmente útil para atividades em grupos: integrantes podem discutir regras do jogo, testar diferentes configurações e comparar soluções, mas a avaliação deve preservar

evidências individuais de compreensão, como uma explicação oral, uma breve justificativa escrita ou a modificação de uma regra diante do professor.

A avaliação formativa pode combinar funcionamento do jogo e processo de desenvolvimento. Recomenda-se observar, pelo menos, cinco dimensões: compreensão do problema; escolha e organização dos blocos; coerência da lógica; capacidade de testar e depurar; e explicação das decisões. Elementos estéticos podem ser valorizados, mas não devem ocultar o objetivo central de compreender estruturas de programação. Da mesma forma, o uso do Tutor IA pode ser considerado parte do processo quando o estudante demonstra que analisou a sugestão, executou a solução e realizou ajustes próprios. A ênfase recai, portanto, sobre a autoria apoiada, e não sobre a simples presença ou ausência da inteligência artificial. Os elementos de gamificação podem ser articulados a essa avaliação sem transformar a aprendizagem em competição exclusivamente por pontos. Desafios, conquistas, progressão e missões podem sinalizar etapas alcançadas, fornecer objetivos próximos e reconhecer avanço. Deterding et al. (2011, p. 9-15) definem gamificação a partir do uso de elementos de design de jogos em contextos que não são jogos; no contexto educacional analisado, seu valor depende de a progressão representar aprendizagem efetiva. Assim, desbloquear um desafio pode significar que o estudante demonstrou uma competência, e não apenas que permaneceu mais tempo conectado à plataforma.

3.10 LIMITAÇÕES E CONDIÇÕES PARA USO EDUCACIONAL

Embora a análise mostre um conjunto amplo de recursos, a ferramenta não deve ser interpretada como substituta do professor nem como garantia automática de aprendizagem. A primeira limitação decorre do caráter do presente estudo: não foram coletados dados de desempenho, engajamento ou aprendizagem de estudantes. Assim, não é possível afirmar, com base apenas na análise funcional, que a LabIA Blocks aumenta notas, reduz evasão ou produz ganhos mensuráveis de pensamento computacional. Tais hipóteses precisam ser investigadas em estudos de aplicação. A segunda limitação relaciona-se ao Tutor IA. Mesmo com validação por catálogo de blocos e regras semânticas, modelos generativos podem produzir respostas inadequadas, interpretar ambiguidades de maneira diferente da intenção do aluno ou sugerir uma solução tecnicamente possível, porém pedagogicamente indesejada. O professor deve manter critérios de acompanhamento e incentivar o estudante a executar, testar e justificar a resposta recebida. A ferramenta contribui mais quando a IA é tratada como fonte de pistas e explicações sujeitas a verificação do que como autoridade final.

A terceira condição diz respeito à infraestrutura. O editor foi concebido para funcionar localmente, mas o portal utiliza PHP/MySQL e componentes do Bootstrap podem depender de CDN conforme a instalação. O Tutor IA também exige conectividade e uma API configurada quando estiver habilitado. Em escolas com conexão restrita, uma implantação totalmente local deve incluir os recursos front-end

necessários e prever atividades que continuem funcionais sem o Tutor. A possibilidade de operar o editor e salvar projetos em JSON oferece uma alternativa importante para esses contextos.

4 CONCLUSÃO

O estudo descreveu a construção e as funcionalidades da LabIA Blocks e analisou suas possibilidades para o ensino introdutório de HTML, CSS, JavaScript e desenvolvimento de jogos. A plataforma reúne, em um mesmo ambiente, programação visual em blocos, visualização de código, execução imediata, planejamento em Portugol, desafios progressivos, modelos de jogos, conquistas, gestão de atividades e Tutor IA. A análise do catálogo evidenciou 108 blocos, abrangendo desde elementos básicos de estrutura e estilo até eventos, variáveis, listas, colisões, pontuação, vidas e mecânicas de jogos. Para o professor, a ferramenta oferece organização de turmas e estudantes, criação de atividades com prazo e níveis diferenciados de apoio da IA, acompanhamento de entregas e registro de avaliação e feedback. Para o estudante, oferece um percurso que pode começar por exemplos e desafios e avançar para projetos autorais, permitindo observar a relação entre blocos e código textual. Essa combinação cria condições para propostas de aprendizagem baseada em projetos, resolução de problemas, depuração e remixagem, ao mesmo tempo em que utiliza elementos de gamificação como progressão, desafios e conquistas. A principal contribuição pedagógica potencial da LabIA Blocks está na articulação entre diferentes formas de representação e apoio: planejar em linguagem algorítmica, montar visualmente, ler o código, executar o jogo, receber feedback e revisar a solução. O Tutor IA pode ampliar essa mediação quando empregado de forma contextual, com níveis definidos pelo professor e preservação da autoria do aluno. Entretanto, a IA não elimina a necessidade de verificação, mediação docente e desenvolvimento de pensamento crítico.

Como continuidade, recomenda-se realizar estudos de aplicação em turmas de educação básica e formação inicial em programação, comparando diferentes estratégias de uso da ferramenta. Pesquisas futuras podem investigar aprendizagem de conceitos específicos, qualidade dos projetos, padrões de depuração, percepção de professores e estudantes, níveis de autonomia e influência dos diferentes níveis do Tutor IA. Essa etapa permitirá passar da análise de potencial pedagógico para evidências empíricas sobre condições, limites e resultados de uso da LabIA Blocks em contextos reais de ensino.

REFERÊNCIAS

DETERDING, Sebastian; DIXON, Dan; KHALED, Rilla; NACKE, Lennart. From game design elements to gamefulness: defining gamification. In: **PROCEEDINGS OF THE 15TH INTERNATIONAL ACADEMIC MINDTREK CONFERENCE**. Tampere: ACM, 2011. p. 9-15. DOI: https://doi.org/10.1145/2181037.2181040. Acesso em: 13 ago. 2026.

FREEMAN, Scott; EDDY, Sarah L.; MCDONOUGH, Miles; SMITH, Michelle K.; OKOROAFOR, Nnadozie; JORDT, Hannah; WENDEROTH, Mary Pat. Active learning increases student performance in science, engineering, and mathematics. **Proceedings of the National Academy of Sciences**, v. 111, n. 23, p. 8410-8415, 2014. DOI: https://doi.org/10.1073/pnas.1319030111. Acesso em: 13 ago. 2026.

GROVER, Shuchi; PEA, Roy. Computational thinking in K-12: a review of the state of the field. **Educational Researcher**, v. 42, n. 1, p. 38-43, 2013. DOI: https://doi.org/10.3102/0013189X12463051. Acesso em: 13 ago. 2026.

HUMBLE, Niklas. Risk management strategy for generative AI in computing education: how to handle the strengths, weaknesses, opportunities, and threats? **International Journal of Educational Technology in Higher Education**, v. 21, art. 61, 2024. DOI: https://doi.org/10.1186/s41239-024-00494-x. Acesso em: 13 ago. 2026.

RESNICK, Mitchel; MALONEY, John; MONROY-HERNÁNDEZ, Andrés; RUSK, Natalie; EASTMOND, Evelyn; BRENNAN, Karen; MILLNER, Amon; ROSENBAUM, Eric; SILVER, Jay; SILVERMAN, Brian; KAFAI, Yasmin. Scratch: programming for all. **Communications of the ACM**, v. 52, n. 11, p. 60-67, 2009. DOI: https://doi.org/10.1145/1592761.1592779. Acesso em: 13 ago. 2026.

UNESCO. **Guia para a IA generativa na educação e na pesquisa**. Paris: UNESCO, 2024. ISBN 978-92-3-700028-1. Disponível em: https://unesdoc.unesco.org/ark:/48223/pf0000390241. Acesso em: 13 ago. 2026.

WEINTROP, David; WILENSKY, Uri. Comparing block-based and text-based programming in high school computer science classrooms. **ACM Transactions on Computing Education**, v. 18, n. 1, art. 3, p. 1-25, 2018. DOI: https://doi.org/10.1145/3089799. Acesso em: 13 ago. 2026.