INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS - *CAMPUS* OURO BRANCO
SISTEMAS DE INFORMAÇÃO

Luiz Filipe Ferreira Ramos

SHELLBLOCKS: UMA ABORDAGEM VISUAL PARA ENSINO DE LINHA DE COMANDO

Ouro Branco - MG
2026

LUIZ FILIPE FERREIRA RAMOS

SHELLBLOCKS: UMA ABORDAGEM VISUAL PARA ENSINO DE LINHA DE COMANDO

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais - *Campus* Ouro Branco para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof.ª Dra. Suelen Mapa de Paula

Ouro Branco - MG
2026

R175s Ramos, Luiz Filipe Ferreira.

Shellblocks: uma abordagem visual para ensino de linha de comando. / Luiz Filipe Ferreira Ramos. – 2026.

20f.il.col.

Orientadora: Suelen Mapa de Paula.

Trabalho de Conclusão de Curso (Sistemas de Informação) – Instituto Federal de Minas Gerais. *Campus* Ouro Branco, 2026.

1. Interface de linha de comando. 2. Programação em blocos. 3. Scaffolding instrucional. 4. Software Educativo. 5. Educação em computação. I. Paula, Suelen Mapa de. II. Instituto Federal de Minas Gerais. Campus Ouro Branco. III. Título.

CDU: 004:37

Catalogação: Márcia Margarida Vilaça - CRB-6/2235
Biblioteca do Instituto Federal de Minas Gerais, *Campus* Ouro Branco

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
CAMPUS OURO BRANCO
Av. Afonso Sardinha, nº 90, Bairro Pioneiros, CEP: 36.420-000, Ouro Branco - Minas Gerais
(31) 3742-2149 – gabinete.ourobranco@ifmg.edu.br

ANEXO II – ATA DE CONCLUSÃO DE TCC

Aos *19* dias do mês de *janeiro* de *2026*, às *21:05* horas, *Luiz Filipe Ferreira Ramos*, aluno(a) regularmente matriculado no Curso de Sistemas de Informação do Instituto Federal de Minas Gerais, campus Ouro Branco, matrícula *0100802*, concluiu o seu Trabalho de Conclusão de Curso por meio de:

( ) Publicação do artigo intitulado _____
na revista/conferência _____,
cujo comprovante de aceitação será anexado a esta ata, recebendo a nota _____ pelo trabalho. Eu, na qualidade de orientador do aluno, lavrei a presente ata atestando a conclusão do trabalho, a qual será assinada por mim e pelo aluno.

_____
Professor Orientador

_____
Aluno

(*X*) Defesa em sessão pública realizada às 21:05 horas, na sala auditório do Instituto Federal de Minas Gerais, campus Ouro Branco, na presença da banca examinadora composta pelos docentes:

1 - *Daniela Costa Terra*
2 - *Ederson Naves Fernandes Gonçalves Júnior*
3 - *Lucas Portela Costa da Silva*
4 - *Marcio Assis Miranda*

do artigo intitulado *ShellBlocks: Uma Abordagem Visual para Ensino de Linha de Comando*

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
***CAMPUS* OURO BRANCO**
Av. Afonso Sardinha, nº 90, Bairro Pioneiros, CEP: 36.420-000, Ouro Branco - Minas Gerais
(31) 3742-2149 – gabinete.ourobranco@ifmg.edu.br

A banca examinadora, após reunião em sessão reservada, deliberou pela *aprovação* do referido trabalho, atribuindo a nota *96,2*. Eu, na qualidade de presidente da banca examinadora, lavrei a presente ata que será assinada por mim, pelos demais examinadores e pelo aluno.
Observações pertinentes à defesa:

NOME E ASSINATURA DOS COMPONENTES DA BANCA E DO ORIENTADO

Documento assinado digitalmente
gov.br
SUELEN MAPA DE PAULA
Data: 27/01/2026 16:35:40-0300
Verifique em https://validar.iti.gov.br

Professor Orientador: *Suelen Mapa de Paula*

Documento assinado digitalmente
gov.br
DANIELA COSTA TERRA
Data: 28/01/2026 11:26:41-0300
Verifique em https://validar.iti.gov.br

Examinador 1: *Daniela Costa Terra*

Documento assinado digitalmente
gov.br
EDERSON NAVES FERNANDES GONCALVES JUN
Data: 28/01/2026 16:00:04-0300
Verifique em https://validar.iti.gov.br

Examinador 2: *Ederson Naves Fernandes Gonçalves Júnior*

Documento assinado digitalmente
gov.br
LUCAS PORTELA COSTA DA SILVA
Data: 29/01/2026 09:52:11-0300
Verifique em https://validar.iti.gov.br

Examinador 3: *Lucas Portela Costa da Silva*

Documento assinado digitalmente
gov.br
MARCIO ASSIS MIRANDA
Data: 29/01/2026 10:15:26-0300
Verifique em https://validar.iti.gov.br

Examinador 4: *Marcio Assis Miranda*

Documento assinado digitalmente
gov.br
LUIZ FILIPE FERREIRA RAMOS
Data: 29/01/2026 10:25:14-0300
Verifique em https://validar.iti.gov.br

Aluno(a): *Luiz Filipe Ferreira Ramos*

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
CAMPUS OURO BRANCO
Av. Afonso Sardinha, nº 90, Bairro Pioneiros, CEP: 36.420-000, Ouro Branco - Minas Gerais
(31) 3742-2149 – gabinete.ourobranco@ifmg.edu.br

DECLARAÇÃO ANTI-PLÁGIO

Eu, *Luiz Filipe Ferreira Ramos*, estudante do curso Bacharelado em Sistemas de Informação do IFMG – Campus Ouro Branco, declaro, para os devidos fins e efeitos, e para fazer prova junto ao IFMG – Campus Ouro Branco, que, **sob as penalidades previstas no art. 299 do Código Penal Brasileiro**, que é de minha criação o Trabalho de Conclusão de Curso que ora apresento.

Art. 299 do Código Penal Brasileiro, que dispõe sobre o crime de *Falsidade Ideológica*:

"Omitir, em documento público ou particular, declaração que dele devia constar, ou nele inserir ou fazer inserir declaração falsa ou diversa da que devia estar escrita, com o fim de prejudicar direito, criar obrigação ou alterar verdade sobre fato juridicamente relevante:

Pena – reclusão, de 1 (um) a 5 (cinco) anos, e multa, se o documento é público, e reclusão de 1 (um) a 3 (três) anos, e multa, se o documento é particular.

Parágrafo único. Se o agente é funcionário público, e comete o crime prevalecendo-se do cargo, ou se a falsificação ou alteração é de assentamento de registro civil, aumenta-se a pena de sexta parte".

Este crime engloba plágio e compra fraudulenta de documentos científicos. Por ser verdade, e por ter ciência do referido artigo, firmo a presente declaração.

Ouro Branco, 26 de janeiro de 2026

gov.br Documento assinado digitalmente
LUIZ FILIPE FERREIRA RAMOS
Data: 26/01/2026 21:59:41-0300
Verifique em https://validar.iti.gov.br

Assinatura do aluno: _____

ShellBlocks: Uma Abordagem Visual para Ensino de Linha de Comando

Luiz Filipe Ferreira Ramos[1], **Suelen Mapa de Paula**[1]

[1] Curso de Sistemas de Informação – Instituto Federal de Minas Gerais (IFMG)
Ouro Branco – MG – Brasil

0100802@academico.ifmg.edu.br, suelen.mapa@ifmg.edu.br

Resumo. *Interfaces de linha de comando (CLI) são poderosas, mas impõem uma curva de aprendizado íngreme devido à dependência da recordação de comando em vez do seu reconhecimento visual através de interface gráfica. Este trabalho apresenta o ShellBlocks, um ambiente visual projetado para mitigar essa barreira cognitiva por meio de estratégias de scaffolding, isto é, o fornecimento de apoio gradual para a construção da autonomia do aprendiz, unindo a interação por blocos à sintaxe textual. Essa abordagem de modalidade dual visa facilitar a transição para o uso da CLI sozinha, permitindo a correlação direta entre a lógica visual e o código gerado. A arquitetura do sistema integra um front-end baseado no Google Blockly a um back-end que utiliza contêineres Docker para garantir a execução segura de código em tempo real. A ferramenta foi avaliada por meio de um estudo experimental com 21 participantes, com foco em usabilidade e eficácia de aprendizado. Os resultados indicaram alta taxa de aceitação, com 90% dos usuários percebendo a abordagem proposta como mais eficaz do que tutoriais textuais tradicionais. Tais achados validam o modelo implementado no ShellBlocks como uma estratégia viável para a introdução de conceitos de CLI.*

Palavras-chave: *Interface de Linha de Comando. Programação em Blocos. Scaffolding Instrucional. Software Educativo. Educação em Computação.*

1. Introdução

A linha de comando (CLI, do inglês *Command-Line Interface*) permanece como uma ferramenta central em ambientes *Unix-like* (sistemas que operam sob os princípios de design do Unix original, como Linux e macOS). Sua relevância persiste principalmente entre administradores de sistemas e desenvolvedores, pois, embora apresente desafios de usabilidade, oferece alta eficiência operacional e permite a criação de rotinas complexas de automação via scripts (MURILLO; SÁNCHEZ, 2014). Sua expressividade, no entanto, contrasta com uma acentuada curva de aprendizado, que impõe barreiras significativas a usuários iniciantes. Este trabalho investiga a natureza dessa dificuldade e propõe uma nova visão computacional para mediar e facilitar o processo de aprendizagem da CLI.

A dificuldade intrínseca da linha de comando origina-se de duas características fundamentais de sua interação. Primeiramente, ela opera sob um paradigma cognitivo de recordação, exigindo que o usuário recupere de memória a sintaxe exata de comandos sem o auxílio de pistas visuais. Em segundo lugar, suas saídas são, em geral, texto não estruturado, desprovido de marcadores semânticos. Essa combinação impõe uma elevada

sobrecarga cognitiva, dificultando a construção de um modelo mental claro por parte do aprendiz.

Para mitigar esse cenário, esta pesquisa adota a abordagem de modalidade dual, concretizada pela visualização simultânea e sincronizada da lógica visual (blocos) e da sintaxe textual (código). Estruturada por estratégias de *scaffolding*[1], essa abordagem combina os processos cognitivos de reconhecimento, facilitado pelas representações visuais, e recordação, estimulada pela exposição contínua ao texto dos comandos. A lacuna preenchida por este trabalho é, portanto, a de uma mediação didática: uma ferramenta que torne a estrutura da CLI transparente sem eliminar o contato com a sintaxe formal.

Nesse contexto, o objetivo geral deste trabalho consiste no desenvolvimento e avaliação do ShellBlocks, um software educacional projetado para mitigar a barreira de entrada da CLI. A ferramenta permite que o usuário construa a lógica de scripts através de blocos visuais interativos, os quais são traduzidos instantaneamente para a sintaxe textual, facilitando a correlação cognitiva entre as representações.

Para viabilizar a experimentação prática, o sistema integra uma estrutura de desafios progressivos com feedback automático e utiliza contêineres Docker para a execução de código. Essa arquitetura de sandbox assegura que o aprendiz possa testar comandos reais e cometer erros construtivos em um ambiente isolado, sem risco de danos ao sistema anfitrião.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta o referencial teórico sobre os desafios cognitivos da CLI e as bases pedagógicas adotadas. A Seção 3 descreve a metodologia de pesquisa e os detalhes técnicos de arquitetura e desenvolvimento do ShellBlocks. A Seção 4 apresenta os resultados obtidos na avaliação com usuários e sua análise e discussão. Por fim, a Seção 5 apresenta as conclusões e propostas para trabalhos futuros.

2. Referencial Teórico

Este referencial teórico fornece os subsídios conceituais necessários para responder às três questões de pesquisa (QP) que norteiam este trabalho:

- **QP1** Qual o impacto de um ambiente de modalidade dual na eficácia do aprendizado e na dificuldade percebida por iniciantes na CLI?
- **QP2** De que maneira a sincronização em tempo real entre a construção visual de scripts (reconhecimento) e sua representação textual (recordação) facilita a transição do aprendiz?
- **QP3** Como a estruturação progressiva de tarefas (*scaffolding*) e o feedback de validação contribuem para a autonomia e a segurança do aprendiz frente à complexidade da CLI?

Para fundamentá-las, discute-se a seguir a carga cognitiva da CLI (QP1), as abordagens visuais correlatas (QP2) e, por fim, os pilares pedagógicos e a infraestrutura de virtualização do sistema (QP3).

[1]Um conceito que significa “andaime”, usado como metáfora para o apoio temporário dado a alguém enquanto aprende algo novo.

2.1. Desafios Cognitivos e de Usabilidade da CLI

A dificuldade de aprendizado da CLI não é apenas uma percepção subjetiva, mas um desafio validado por pesquisas em Interação Humano-Computador. Evidências indicam que a CLI gera saídas de dados não estruturados, desprovidos de elementos semânticos que indiquem hierarquia (SAMPATH et al., 2021). No contexto de acessibilidade, isso obriga a uma varredura linear e exaustiva do conteúdo. Para um aprendiz iniciante, essa mesma falta de estrutura dificulta a identificação do que é relevante, elevando significativamente a sobrecarga cognitiva.

Reforçando essa visão, estudos empíricos compararam a usabilidade de interfaces gráficas (GUI) e de linha de comando. Demonstrou-se que tarefas realizadas via GUI tendem a ser mais eficazes e menos propensas a erros iniciais (FEIZI; WONG, 2012).

Essa distinção cognitiva é fundamental para este trabalho: interfaces gráficas favorecem o reconhecimento (escolher uma opção visível), enquanto a CLI exige recordação (recuperar da memória a sintaxe exata). Embora o reconhecimento reduza a barreira inicial, estudos apontam que a recordação é essencial para a retenção de conhecimento a longo prazo (DURHAM; EMURIAN, 1998).

2.2. Abordagens Visuais e Educacionais para CLI

Diante desses desafios, diversas iniciativas buscaram criar pontes entre a CLI e interfaces mais intuitivas. O Quadro 1 resume as principais características das ferramentas analisadas, que podem ser categorizadas em dois grupos: ferramentas utilitárias de automação e ambientes estritamente educacionais.

No contexto de ferramentas utilitárias, o foco reside na produtividade e na abstração da complexidade. O COLIMATE propõe um modelo onde a sintaxe é descrita por uma gramática formal para gerar interfaces automaticamente (SORZANO et al., 2002). De forma análoga, o gtkdialog utiliza uma linguagem declarativa para construir interfaces gráficas universais para interpretadores arbitrários (PERE; KONIORCZYK, 2005). Embora eficientes para a execução de tarefas, essas soluções atuam como "caixas pretas": elas ocultam deliberadamente a sintaxe do shell, impedindo que o usuário desenvolva a competência de operar diretamente no terminal.

No campo educacional, as propostas buscam expor a complexidade de forma controlada. A ferramenta GUISE foca no domínio de redes, utilizando a metáfora de *cockpits* para fornecer consciência situacional (ANDERSON, 2022). A plataforma Gitit representa um avanço significativo ao integrar gamificação e visualizações interativas com feedback em tempo real, validando o uso de *scaffolding* visual para o ensino de controle de versão (BUXTON, 2025). Outra abordagem relevante adota a técnica de *problem-posing* (MIHCI; SATICI, 2020). Embora eficaz para o engajamento, essa abordagem assume que o aluno já possua, ou seja capaz de buscar externamente, o conhecimento sintático necessário para formular os problemas, o que pode representar uma barreira para iniciantes absolutos.

2.3. Fundamentação Pedagógica

A concepção do ambiente ShellBlocks não se restringe à solução técnica, mas ancora-se em teorias consolidadas de aprendizagem.

Ferramenta	GUI	Foco Educacional	Blocos	Tempo Real	Extensível
COLIMATE	Sim	Não	Não	Não	Parcial
gtkdialog	Sim	Não	Não	Não	Sim
GUISE	Parcial	Sim	Não	Não	Sim
Gitit	Não	Sim	Não	Sim	Não
ShellBlocks	**Sim**	**Sim**	**Sim**	**Sim**	**Sim**

Quadro 1. Comparação entre iniciativas existentes e o ShellBlocks

2.3.1. Construcionismo e Aprendizagem por Criação

A abordagem de permitir que o aluno construa seus scripts remete ao Construcionismo. Define-se que o aprendizado é particularmente eficaz quando o aprendiz está engajado na construção de uma entidade pública e significativa (PAPERT; HAREL, 1991). O ShellBlocks atua como um micromundo onde o aluno externaliza seu entendimento sobre a CLI criando um artefato funcional (o script), testando hipóteses e observando resultados.

2.3.2. Scaffolding e a Zona de Desenvolvimento Proximal

A estratégia de desafios progressivos fundamenta-se no conceito de *Scaffolding* (andaime). O termo define o processo que permite a um novato resolver problemas além de sua capacidade individual através de suporte estruturado (WOOD et al., 1976). No ShellBlocks, os blocos visuais atuam como esse andaime, reduzindo a complexidade sintática inicial para que o aluno foque na lógica.

2.3.3. Ambientes de Modalidade Dual (*Dual-Modality*)

Para efetivar a transição para a autonomia textual, adota-se o modelo de modalidade dual. Alerta-se que o uso exclusivo de blocos pode impedir o desenvolvimento da "consciência sintática" (MOORS et al., 2018). Para resolver isso, indica-se que a visualização simultânea entre a representação gráfica e o código textual é crucial (PERERA et al., 2021).

2.4. Virtualização e Execução Segura

A execução de comandos de terminal em um ambiente educacional web exige um mecanismo que garanta simultaneamente a segurança do servidor e a consistência do ambiente para todos os alunos. Permitir a execução direta no sistema hospedeiro traria riscos de segurança e problemas de configuração residual.

Para mitigar esses riscos, utiliza-se a tecnologia de contêineres Docker. Esta tecnologia oferece um mecanismo de encapsulamento que garante a reprodutibilidade computacional (BOETTIGER, 2015). Ao empacotar todas as dependências em uma imagem imutável, assegura-se que o ambiente de execução seja idêntico para todos os usuários, eliminando inconsistências de configuração.

Do ponto de vista de desempenho e isolamento, a escolha por contêineres em vez de Máquinas Virtuais (VMs) é fundamentada na eficiência. Demonstra-se que, ao

compartilhar o *kernel* do sistema hospedeiro, o Docker atinge desempenho próximo ao nativo (*bare metal*) e tempos de inicialização drasticamente menores que os de VMs tradicionais (FELTER et al., 2015).

Essa baixa latência na inicialização, validada na literatura (FELTER et al., 2015), é o requisito técnico que torna viável o uso de instâncias efêmeras. Graças à velocidade de *deploy*, é possível instanciar, executar e destruir um contêiner isolado para cada submissão do aluno em tempo real, garantindo um ambiente seguro e sempre limpo (sandbox).

2.5. Síntese e Implicações para a Solução Proposta

A revisão bibliográfica realizada permite consolidar os pilares que sustentam a proposta do ShellBlocks e oferece respostas teóricas para as questões de pesquisa levantadas. Concluise que a barreira de entrada da CLI não reside na incapacidade lógica do aprendiz, mas na sobrecarga cognitiva imposta pela exigência de recordação pura em um ambiente não estruturado.

Para responder à necessidade de facilitar a transição do aprendiz (QP1 e QP2), o referencial aponta que a solução deve atuar como uma ponte cognitiva. Isso justifica a escolha pela modalidade dual: a interface gráfica reduz a ansiedade inicial através do reconhecimento, enquanto a exposição simultânea do código treina a memória para a futura autonomia sintática.

No que tange à autonomia (QP3), a teoria do *scaffolding* valida a estruturação do sistema em níveis de dificuldade progressiva. O ShellBlocks não apresenta a CLI como um "muro" intransponível, mas fragmenta o aprendizado em etapas discretas, garantindo que o aluno receba feedback de validação específico para cada estágio de competência antes de avançar para comandos mais complexos.

Complementarmente, a opção arquitetural por um modelo orientado a dados (*data-driven*) responde à necessidade de extensibilidade identificada na análise das ferramentas existentes. Ao desacoplar a lógica do motor de execução das definições de conteúdo, sintaxe e visualização, o sistema permite que o design instrucional seja adaptado ou expandido organicamente, sem exigir refatoração do código-fonte.

Por fim, a infraestrutura de virtualização via Docker responde à exigência pedagógica do Construcionismo por um ambiente de experimentação seguro. O uso de contêineres efêmeros viabiliza um *sandbox* real, onde o erro construtivo é permitido. Isso possibilita que o aluno observe as consequências reais de suas ações no sistema de arquivos, fechando o ciclo de aprendizado sem oferecer riscos à segurança do servidor.

3. Metodologia e Desenvolvimento

Esta seção descreve o percurso metodológico adotado neste trabalho. Para atender ao objetivo de facilitar o ensino da CLI, a pesquisa foi estruturada em duas grandes etapas: (1) o desenvolvimento de um artefato de software, cuja arquitetura e decisões de projeto são detalhadas inicialmente; e (2) a condução da pesquisa sob o paradigma da *Design Science Research* (DSR), incluindo o protocolo experimental utilizado para validar a ferramenta junto aos usuários finais.

3.1. Apresentação da Ferramenta

A interface do ShellBlocks foi projetada para reduzir a carga cognitiva, organizando o fluxo em áreas funcionais, conforme a Figura 1.

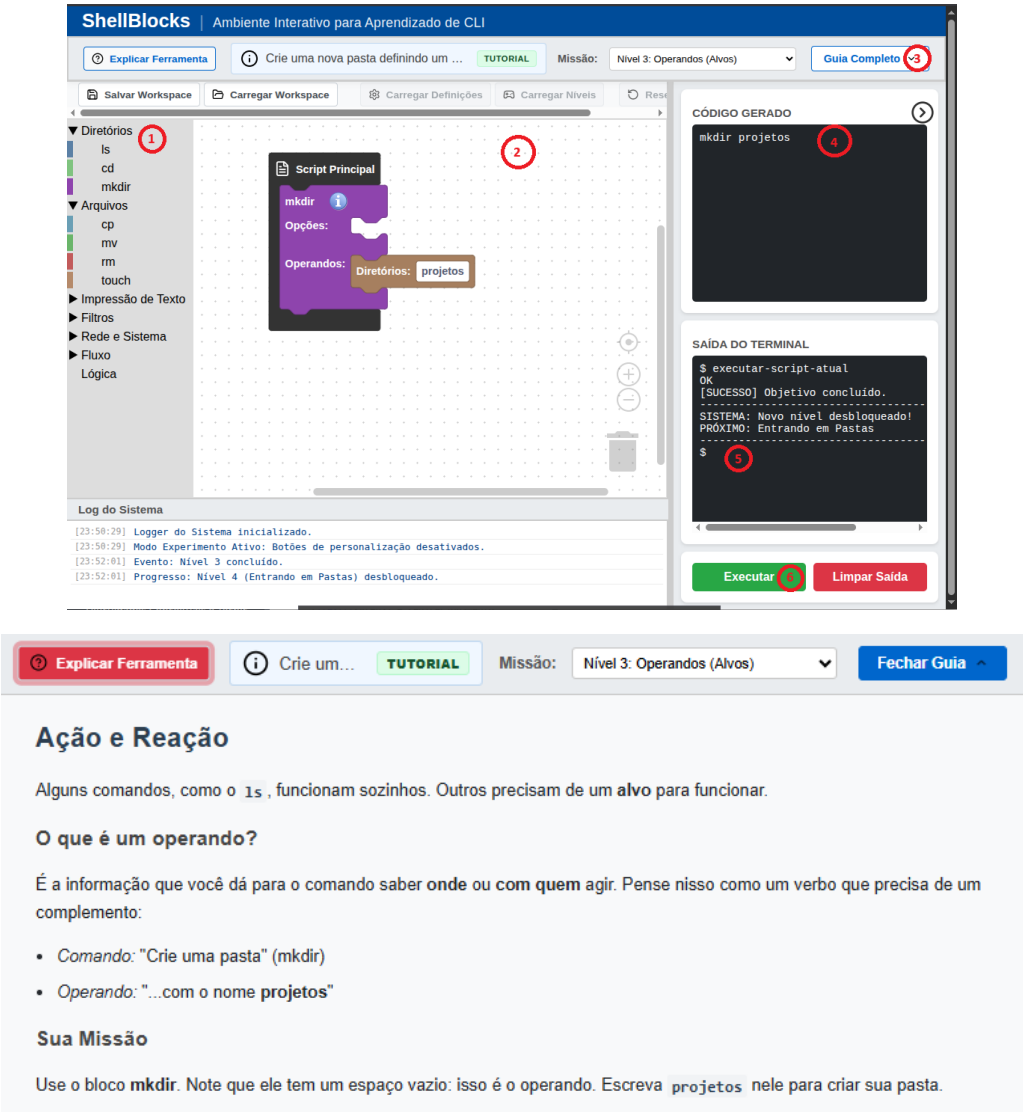

Figura 1. Interface do Usuário: (A) Visão geral das áreas funcionais; (B) Detalhe do scaffolding no Painel de Instruções.

A interação inicia-se na Toolbox (A-1), de onde os blocos são arrastados para o Workspace (A-2). O código gerado é exibido em tempo real (A-4). O suporte pedagógico é provido pelo Painel de Instruções (A-3), detalhado na imagem inferior (B), que contextualiza a missão com formatação rica antes de o usuário executar o código no Terminal (A-5).

3.2. Desenvolvimento e Arquitetura do ShellBlocks

O ShellBlocks foi desenvolvido como uma aplicação distribuída, adotando uma estratégia de desacoplamento entre a representação visual (no cliente) e a geração e execução de código (no servidor). A solução é composta por dois módulos principais que se comunicam via protocolo HTTP (REST), conforme ilustrado na Figura 2.

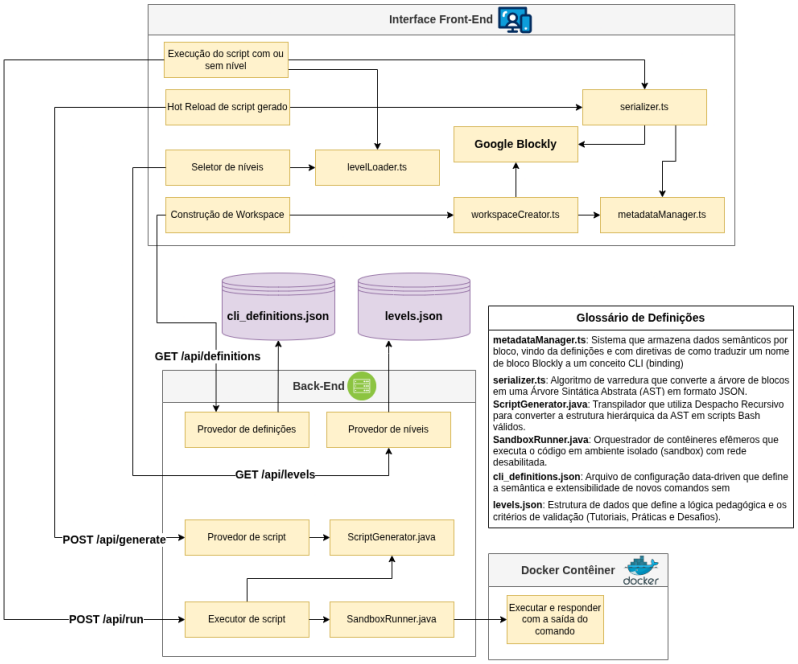

Figura 2. Fluxo de dados e arquitetura: da interação visual à execução isolada.

O front-end (TypeScript/Blockly) transcende a mera exibição gráfica ao implementar um serializador customizado. Diferente de implementações tradicionais que geram strings, o sistema constrói uma Árvore Sintática Abstrata (AST) em JSON, desacoplando a lógica visual da sintaxe exata do Bash e garantindo independência tecnológica.

No back-end (Java/Javalin), o componente `ScriptGenerator` atua como um transpilador baseado em despacho recursivo, convertendo a AST recebida em comandos de shell fiéis. A segurança crítica é garantida pelo gerenciamento de contêineres Docker: para cada requisição, uma instância efêmera e isolada (*sandbox*) é criada e imediatamente destruída após a execução, mitigando riscos ao sistema hospedeiro.

3.3. Estratégia Pedagógica e Sistema de Níveis

Um dos diferenciais da ferramenta reside na sua concepção orientada a dados (*data-driven*). A lógica pedagógica não está "cimentada" no código-fonte, mas definida em arquivos externos de configuração (JSON), permitindo a criação dinâmica de trilhas de aprendizado. O arquivo mestre `levels.json` orquestra a experiência do usuário através de uma sequência progressiva de desafios.

A estrutura dos níveis foi desenhada para guiar o aluno da dependência visual para a autonomia conceitual, categorizada em três fases de dificuldade crescente, conforme detalhado no Quadro 2.

Cada nível implementa um mecanismo de validação automática baseado em teste de caixa-preta (*black-box testing*). O sistema não avalia apenas se o aluno usou o bloco "correto", mas verifica se o estado final do ambiente (arquivos criados, conteúdo alterado, processos rodando) corresponde ao objetivo do nível. Isso é realizado através da injeção de scripts de verificação ocultos no contêiner Docker após a execução do código do aluno, garantindo que o aprendizado seja avaliado pelo resultado prático e não apenas pela sintaxe.

Níveis	Foco Temático	Objetivo de Aprendizagem
01–05	Comandos Básicos	Navegação e manipulação simples (`ls`, `cd`, `mkdir`, `cp`). Introdução ao conceito de argumentos e *flags*.
06–09	Fluxo e Texto	Manipulação de conteúdo (`cat`, `grep`) e redirecionamento de I/O (Pipe `|` e Saída >). Início da composição de comandos.
10–12	Rede e Processos	Comandos de longa duração e interação com o sistema (`ping`, `curl`, `ps`).
13–17	Treinamento	Missões compostas sem guia passo-a-passo. Exige combinar conceitos anteriores (ex: baixar arquivo e mover para pasta criada).
18–19	Desafios Finais	Cenários complexos de segurança e *deploy* (simulação de *keylogger* e servidor web). Foco em resolução de problemas reais.

Quadro 2. Progressão Pedagógica dos Níveis do Experimento

3.4. Abordagem de Pesquisa

A pesquisa foi conduzida sob o paradigma da *Design Science Research* (DSR), voltada à criação de artefatos para solucionar problemas práticos. O projeto envolveu ciclos iterativos de **Construção** (engenharia do artefato detalhada acima) e **Avaliação** (validação empírica com usuários), conforme o protocolo a seguir.

3.5. Protocolo de Avaliação Experimental

A atividade, realizada em ambiente laboratorial (60 min), seguiu três etapas: (1) Ambientação sobre o objetivo da ferramenta; (2) Interação guiada, onde os participantes resolveram a sequência progressiva de níveis no ShellBlocks; e (3) Coleta de Dados via formulário pós-teste.

Os instrumentos de coleta incluíram questões em Escala Likert (1-5) para avaliar usabilidade e percepção de aprendizado, questões comparativas (visual *versus* textual) e campos abertos para registro qualitativo de dificuldades e sugestões.

Para garantir a integridade, transparência e reprodutibilidade da pesquisa, definiu-se um ponto de congelamento (*snapshot*) do projeto. O pacote experimental completo, compreendendo o código-fonte (`v1.0.0`), o binário executável, o instrumento de coleta original (PDF do formulário), os dados brutos (CSV) e o relatório estatístico consolidado, encontra-se centralizado e disponível publicamente na Release Oficial do Experimento[2].

4. Discussão dos Resultados

Esta seção apresenta os dados coletados durante a avaliação experimental da ferramenta ShellBlocks. O estudo foi conduzido conforme o protocolo descrito na Metodologia, contando com a participação de 21 alunos de graduação. Embora todos os presentes tenham realizado a atividade, foram obtidos 21 envios de formulários, dos quais um foi descartado por inconsistência nos dados (caracterizado como envio de teste), resultando em uma amostra final de 20 respostas válidas para análise.

[2]Pacote Experimental Completo (Artefato, Instrumento de Coleta e Dados): https://github.com/Luiz-Filipe26/ShellBlocks/releases/tag/v1.0.0-experiment

4.1. Perfil dos Participantes

A caracterização da amostra revela um público alinhado ao alvo da ferramenta: usuários com pouco ou nenhum conhecimento prévio sobre linha de comando. Conforme detalhado na Tabela 1, 90% dos participantes se autodeclararam iniciantes ou sem nenhuma experiência prévia, o que valida os resultados como representativos para o cenário de introdução à computação.

Tabela 1. Distribuição do Conhecimento Prévio dos Participantes

Nível de Conhecimento Declarado	Participantes	%
Iniciante (Básico: ls, cd, mkdir)	13	65%
Nível 0 (Nunca digitou um comando)	5	25%
Intermediário (Entende pipes, redirecionamento)	2	10%
Total	**20**	**100%**

Em relação ao engajamento, observou-se uma variação na progressão dos níveis. Um participante completou todos os desafios (Nível 19), zerando o jogo, enquanto a maioria concentrou-se nos níveis intermediários, interrompendo a sessão entre os desafios de filtros (`grep`) e redirecionamento, frequentemente devido a limitações de tempo da sessão experimental ou dificuldades pontuais que serão discutidas na Análise.

4.2. Avaliação Quantitativa da Usabilidade

A usabilidade e a eficácia percebida da ferramenta foram avaliadas por meio de uma escala Likert de 5 pontos (1 = Péssimo, 5 = Excelente). A Figura 3 apresenta as médias aritméticas obtidas nas seis dimensões avaliadas.

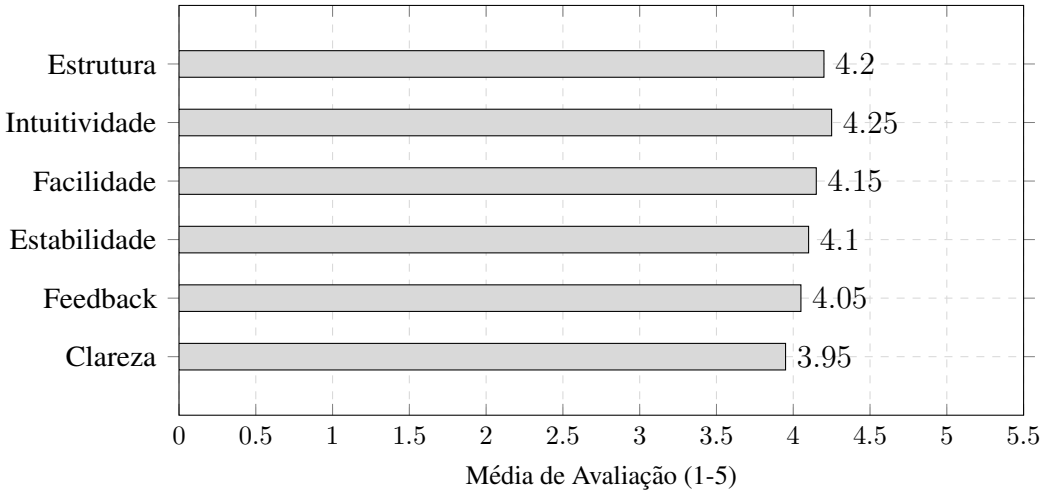

Figura 3. Médias de avaliação por dimensão de usabilidade

Legenda: Clareza (Missões), Feedback (Erro), Estabilidade (Técnica), Facilidade (Entendimento), Intuitividade (Interface), Estrutura (Sintaxe).

Os dados indicam uma recepção positiva consistente. O aspecto mais bem avaliado ($\mu = 4,25$) foi a intuitividade da interface, seguida de perto pela capacidade dos blocos visuais em auxiliar no entendimento da estrutura dos comandos ($\mu = 4,20$). A estabilidade

técnica obteve média de 4,10, apesar de problemas isolados reportados qualitativamente. A clareza das missões obteve a menor média ($\mu = 3,95$), indicando um ponto de atenção no *design* instrucional dos textos de apoio.

4.3. Comparação com Métodos Tradicionais

Quando questionados sobre a eficácia do ShellBlocks em comparação à leitura de tutoriais em texto tradicionais, a resposta foi expressivamente favorável à abordagem visual. Conforme ilustrado na Figura 4, 18 dos 20 participantes (90%) classificaram a ferramenta como "Mais eficaz".

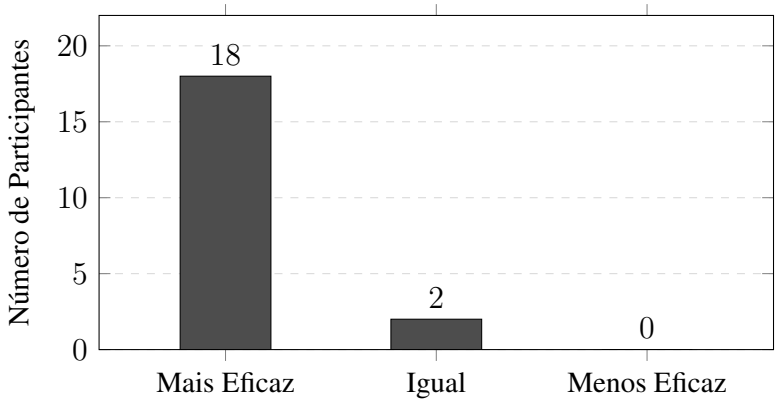

Figura 4. Percepção de eficácia do ShellBlocks versus Tutoriais em Texto

4.4. Dificuldades Conceituais e Técnicas

A análise dos erros e do feedback qualitativo permitiu mapear os conceitos que ofereceram maior resistência ao aprendizado. O tópico de **Redirecionamento de Saída** (operador >) foi citado como o mais difícil por 14 participantes (70%). Em seguida, aparecem os conceitos de **Filtros e Pipes** (`grep`, `|`) com 6 citações.

Nos campos de resposta aberta, foram identificados dois problemas recorrentes que impactaram a experiência:

1. **Timeout no comando Ping:** Usuários que chegaram aos níveis 10 e 11 relataram erros de "Timeout Exception" ao tentar executar o comando `ping`. Isso ocorreu devido à configuração restritiva de tempo de execução no Sandbox (Docker), que encerrava o processo antes do retorno da rede simulada.
2. **Visibilidade da aba Fluxo:** Alguns participantes relataram confusão ao tentar encontrar os blocos de controle (como o Pipe), pois não perceberam que estes estavam agrupados em uma categoria separada na caixa de ferramentas (*toolbox*), denominada "Fluxo".

Apesar desses obstáculos pontuais, os comentários finais incluíram elogios à interface ("bonita e intuitiva") e solicitações para a inclusão de mais comandos avançados, como `ssh` e `wget`.

4.5. Considerações Acerca das Perguntas de Pesquisa

Para responder as perguntas de pesquisas formuladas no Referencial Teórico, as subseções seguintes as responderá a partir dos resultados obtidos.

4.5.1. QP1 e QP2: Eficácia do Modelo Dual e Barreiras de Interface

As questões de pesquisa QP1 e QP2 investigavam o impacto da modalidade dual (visual + textual) na curva de aprendizado. Os dados confirmam que a visualização dos comandos reduz a abstração inicial, mas revelaram que a própria interface gráfica pode introduzir novas barreiras cognitivas se não projetada com *affordances* claras.

O obstáculo mais notável ocorreu no **Nível 8 (Redirecionamento)**, onde a maioria dos participantes travou. A análise *post-hoc* indicou que a dificuldade não era conceitual (entender o que o > faz), mas de interface: o bloco `redirect_out` exigia o encaixe de um bloco auxiliar `file_target` para definir o destino. Visualmente, essa dependência não estava explícita para os iniciantes. Contudo, após uma intervenção explicativa no quadro, a turma superou o nível rapidamente. Isso sugere que o ShellBlocks é eficaz na transição do reconhecimento para a recordação, desde que o vocabulário visual seja autoexplicativo.

Outro ponto de fricção foi a descoberta das categorias. Alguns alunos não perceberam inicialmente que as opções (*flags*) estavam aninhadas na mesma categoria do comando principal, ou tiveram dificuldade em localizar a aba "Fluxo". Esses achados indicam que, embora a metáfora de blocos seja poderosa, a arquitetura da informação dentro da *toolbox* (caixa de ferramentas) precisa de refinamentos para evitar que a busca pelo bloco se torne uma tarefa cognitiva concorrente ao aprendizado da lógica.

4.5.2. QP3: Scaffolding, Autonomia e Modelos Mentais

A QP3 questionava como o *scaffolding* contribui para a autonomia. Observou-se que a estrutura progressiva fomentou um fenômeno de aprendizado colaborativo (*peer learning*), onde alunos utilizavam a interface de blocos como uma linguagem comum para explicar a lógica do Linux aos colegas.

Um detalhe qualitativo revelador sobre a usabilidade surgiu quando um participante demonstrou dúvida sobre a necessidade de limpar o editor para iniciar um novo desafio. O aluno acumulou blocos de níveis anteriores no *workspace*, o que resultou na geração de scripts inválidos ou redundantes. Esse comportamento evidencia que o modelo mental de fluxo de trabalho por sessão não é imediato; o usuário esperava que a ferramenta realizasse a limpeza automática do ambiente visual a cada mudança de nível. Esse incidente validou a necessidade de melhorias na automação da interface para garantir que o estado visual do editor reflita estritamente o desafio atual.

Ironicamente, uma das primeiras barreiras de autonomia foi a própria execução do artefato. A necessidade de utilizar o comando `java -jar` no terminal para abrir a ferramenta educacional gerou fricção inicial, reforçando o argumento central deste trabalho sobre a hostilidade da CLI para usuários sem experiência prévia.

4.5.3. A Resiliência do Engajamento frente a Falhas Técnicas

Um dos achados mais significativos do estudo emergiu não do sucesso da ferramenta, mas de suas falhas. Durante o experimento, limitações na configuração do Sandbox Docker

causaram exceções de tempo limite (*timeout*) em comandos de rede (`ping`, `curl`) em diversas máquinas, devido ao limite restritivo de 5 segundos configurado no *back-end*.

Paradoxalmente, essas falhas técnicas serviram para validar a hipótese de engajamento. Em um cenário de ensino tradicional, erros técnicos graves frequentemente levam à dispersão e abandono da atividade. No entanto, observou-se uma alta resiliência motivacional:

1. Os alunos buscaram soluções adaptativas para contornar o tempo limite; embora o exercício solicitasse o envio de quatro pacotes (`-c 4`), a orientação para reduzir a contagem para apenas um (`-c 1`) permitiu que os participantes compreendessem a lógica do comando e avançassem, demonstrando flexibilidade diante de restrições de infraestrutura;
2. Houve solicitações ativas para que o instrutor desbloqueasse níveis manualmente (via manipulação do *localStorage*), permitindo o acesso aos desafios finais mesmo sem a validação automática do sistema;
3. O feedback qualitativo final permaneceu extremamente positivo, com participantes recorrendo a expressões coloquiais de grande entusiasmo para descrever a experiência, indicando que a proposta de valor da ferramenta superou a frustração com a estabilidade do ambiente.

Vale ressaltar que os níveis finais (18 e 19) haviam sido previamente anunciados como experimentais e não obrigatórios. O fato de os alunos terem persistido em tentar acessá-los, recorrendo até a "hacks" de navegador para superar os bloqueios técnicos, demonstra que a abordagem visual gerou uma recompensa intrínseca suficiente para transformar a atividade avaliativa em um desafio pessoal de superação.

5. Conclusão

O presente trabalho investigou as barreiras cognitivas associadas ao aprendizado da linha de comando e propôs, como solução, o desenvolvimento e a avaliação da ferramenta ShellBlocks. A partir da implementação de um ambiente de modalidade dual, foi possível demonstrar que a integração entre a representação visual (blocos) e a textual (código) atua como um facilitador significativo na transição de novatos para a proficiência em ambientes *Unix-like*.

Os resultados obtidos permitem concluir que o objetivo geral foi alcançado. A ferramenta não apenas removeu a fricção inicial causada pela sintaxe rígida da CLI, como também promoveu um ambiente seguro para experimentação. A validação empírica, com uma aprovação de 90% em comparação a métodos tradicionais, sugere que o modelo de *scaffolding* visual é uma estratégia pedagógica eficaz para reduzir a carga cognitiva intrínseca a este domínio.

Do ponto de vista técnico, o projeto validou uma arquitetura híbrida robusta. A decisão de manter a lógica de serialização no cliente e a execução isolada no servidor (Docker) provou-se acertada, garantindo uma experiência de usuário fluida sem comprometer a segurança do sistema hospedeiro.

5.1. Contribuições

As principais contribuições deste trabalho para a área de Informática na Educação e Engenharia de Software incluem:

- **O Artefato ShellBlocks:** Uma ferramenta funcional, extensível e de código aberto, capaz de auxiliar o ensino introdutório de Sistemas Operacionais.
- **Validação da Modalidade Dual:** Evidências empíricas que reforçam a teoria de que a visualização da estrutura (reconhecimento) deve preceder ou acompanhar a memorização da sintaxe (recordação).
- **Resiliência do Engajamento:** A constatação de que a abordagem gamificada e visual gera motivação intrínseca suficiente para manter os alunos engajados mesmo diante de instabilidades técnicas.
- **Modelo Arquitetural Seguro:** A demonstração de uma arquitetura de sandbox viável para execução de código de alunos em ambientes web, utilizando contêineres efêmeros.

5.2. Limitações do Estudo

Apesar dos resultados positivos, o estudo apresenta limitações que devem ser consideradas:

- **Tempo de Exposição:** A avaliação foi realizada em sessão única. Não foi possível medir a retenção de conhecimento a longo prazo, ou seja, se os alunos conseguiram transferir o aprendizado dos blocos para o terminal real dias ou semanas depois.
- **Fricção de Instalação:** A dependência do ambiente Java (execução via arquivo JAR) gerou dificuldades iniciais para usuários não técnicos, representando uma barreira de entrada antes mesmo do uso da ferramenta.
- **Usabilidade da Interface:** A organização da categoria *Fluxo* na interface não foi intuitiva para todos os usuários, criando um obstáculo artificial para o uso de pipes e redirecionamentos.
- **Restrições do Ambiente de Execução:** O tempo limite rígido configurado no Docker impediu a conclusão de missões envolvendo comandos de rede mais lentos (como o `ping`), gerando frustração pontual.

5.3. Trabalhos Futuros

Como desdobramentos desta pesquisa, sugerem-se as seguintes melhorias e investigações:

1. **Refinamento de UX:** Reestruturação da caixa de ferramentas para dar maior visibilidade a operadores de fluxo e feedback visual mais claro sobre o estado de execução do contêiner.
2. **Expansão do Vocabulário:** Implementação de novos blocos sugeridos pelos participantes, como `ssh`, `wget` e editores de texto simples (`nano`), ampliando o escopo para tarefas de administração remota.
3. **Sistema de Dicas Contextuais:** Desenvolvimento de um analisador de erros baseado em heurísticas (ou alternativamente via modelos de linguagem - LLMs) para identificar padrões comuns de falha e oferecer orientações pedagógicas precisas (ex: "Você esqueceu de conectar a saída..."), superando a cripticidade das mensagens de erro padrão do terminal.
4. **Estudo Longitudinal:** Realização de um experimento de longa duração, comparando o desempenho de alunos que usaram o ShellBlocks versus métodos tradicionais ao longo de um semestre letivo completo.

Referências

ANDERSON, C. J. (2022). *Developing an Expandable GUI Tool to Enhance Networking Education: Graphical User Interface for Shell Entry (GUISE)*. PhD thesis, Monterey, CA; Naval Postgraduate School.

BOETTIGER, C. (2015). An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79.

BUXTON, T. P. (2025). *GitIt: An Interactive Online Platform for Teaching Command Line Based Skills Through Visualizations*. PhD thesis, Virginia Tech.

DURHAM, A. G.; EMURIAN, H. H. (1998). Learning and retention with a menu and a command line interface. *Computers in human behavior*, 14(4):597–620.

FEIZI, A.; WONG, C. Y. (2012). Usability of user interface styles for learning a graphical software application. In *2012 International Conference on Computer & Information Science (ICCIS)*, volume 2, pages 1089–1094. IEEE.

FELTER, W.; FERREIRA, A.; RAJAMONY, R.; RUBIO, J. (2015). An updated performance comparison of virtual machines and linux containers. In *2015 IEEE international symposium on performance analysis of systems and software (ISPASS)*, pages 171–172. IEEE.

MIHCI, C.; SATICI, A. (2020). Developing a web-based environment for learning to solve problems with the linux command line: The problem-posing approach. *Online Submission*, 8(3):142–157.

MOORS, L.; LUXTON-REILLY, A.; DENNY, P. (2018). Transitioning from block-based to text-based programming languages. In *2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*, pages 57–64. IEEE.

MURILLO, S. R.; SÁNCHEZ, J. A. (2014). Empowering interfaces for system administrators: Keeping the command line in mind when designing guis. In *Proceedings of the XV International Conference on Human Computer Interaction*, pages 1–4.

PAPERT, S.; HAREL, I. (1991). Situating constructionism. *constructionism*, 36(2):1–11.

PERE, L.; KONIORCZYK, M. (2005). A universal fast graphical user interface building tool for arbitrary interpreters. *Journal of Visual Languages & Computing*, 16(3):231–244.

PERERA, P.; TENNAKOON, G.; AHANGAMA, S.; PANDITHARATHNA, R.; CHATHURANGA, B. (2021). A systematic mapping of introductory programming languages for novice learners. *IEEE Access*, 9:88121–88136.

SAMPATH, H.; MERRICK, A.; MACVEAN, A. (2021). Accessibility of command line interfaces. In *Proceedings of the 2021 CHI conference on human factors in computing systems*, pages 1–10.

SORZANO, C. O. S.; CARAZO, J. M.; TRELLES, O. (2002). Command-line interfaces can be efficiently brought to graphics: Colimate (the command line mate). *Software: Practice and Experience*, 32(9):873–887.

WOOD, D.; BRUNER, J. S.; ROSS, G. (1976). The role of tutoring in problem solving. *Journal of child psychology and psychiatry*, 17(2):89–100.