Machine Teaching: uma ferramenta didática e de análise de dados para suporte a cursos introdutórios de programação

Laura O. Moraes
laura.moraes@coppe.ufrj.br
Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ e Colégio Pedro II

Carla A. D. M. Delgado
Instituto de Computação, UFRJ
carla@ic.ufrj.br

João Pedro Freire
joaofreire@poli.ufrj.br
Bacharelado em Ciências Matemáticas e da Terra, UFRJ

Carlos Eduardo Pedreira
pedreira56@gmail.com
Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ

RESUMO

O Machine Teaching é um ambiente de aprendizagem online de programação utilizado desde 2018 como ferramenta de apoio em cursos introdutórios. Além de apoiar os alunos em suas práticas e os professores na correção das tarefas, o Machine Teaching coleta dados enquanto os alunos programam. Esses dados são analisados e usados para apoiar a tomada de decisões. Em seu atual estágio de desenvolvimento, a meta é transformar as análises de dados implementadas em ações concretas, que implementem melhorias no curso e no processo de aprendizagem. Para isso, investigamos se a ferramenta projetada é útil para professores e alunos, quais dados devem ser coletados, e se sua apresentação é adequada aos processos de decisão dos diferentes atores no cenário de ensino introdutório de programação. Nossos resultados indicam que o sistema cumpre parcialmente com as metas estabelecidas, como sua usabilidade e utilidade, mas que há espaço para avanços no suporte à decisão.

CCS CONCEPTS

• **Social and professional topics** → *Computing education*; • **Applied computing** → **Interactive learning environments**;

PALAVRAS-CHAVE

ensino de programação, análise de dados educacionais, ambiente educacional

1 INTRODUÇÃO

Aprender a programar é reconhecidamente uma tarefa difícil, e muitos esforços têm sido despendidos para melhor entendê-la [23]. Em salas de aula presenciais, e turmas com um número razoável de alunos por professor, os professores realizam a "descoberta do conhecimento" (no sentido de mineração de dados educacionais) diariamente, observando o comportamento dos alunos e como eles reagem (tanto no sentido emotivo quanto em relação aos resultados obtidos) nas atividades didáticas e avaliativas. A partir destas observações, os professores identificam quais conceitos e habilidades os alunos já internalizaram e quais são mais complicados de incorporar, adaptando suas estratégias de ensino ao longo do processo. No caso de turmas com muitos alunos, e especificamente no caso do ensino remoto, a observação do professor fica bastante comprometida pelas limitações do ambiente, precisando de ferramentas para dar suporte a essa tarefa.

Uma alternativa para preencher essa lacuna da observação direta do professor é a análise do *log* de atividades do aluno. Análises sobre estes dados alimentam o professor com dados sobre as reações dos alunos às atividades propostas e sobre o processo cognitivo de criação de sentido pelo qual eles passam ao lidar com um determinado conteúdo. No entanto, para ser útil, essa quantidade de dados deve ser apresentada de forma que as partes interessadas entendam como estas informações podem influenciar não apenas suas percepções, como também suas ações. Relatórios recentes em análise de aprendizagem apontaram que precisamos aproximar a pesquisa publicada na área de análise de dados educacionais da aplicação em situações reais [11, 20, 24].

Neste artigo apresentamos a ferramenta Machine Teaching: um ambiente de aprendizado online usado desde 2018 como o principal recurso didático de apoio às aulas práticas presenciais dos cursos introdutórios de programação oferecidos pelo Instituto de Computação da Universidade Federal do Rio de Janeiro (UFRJ). Seu principal objetivo é coletar dados sobre o conhecimento dos alunos durante o processo de aprendizagem em programação para o apoio à tomada de decisões relativas ao curso e ao aprendizado de programação. Objetivos secundários deste ambiente são: fornecer *feedback* imediato durante as atividades, o que consideramos primordialmente um fator de estímulo aos alunos em suas práticas, e automatizar parcialmente a análise dos códigos dos alunos, facilitando o trabalho de acompanhamento e geração de *feedback* dos professores. Outra vantagem identificada é a conveniência para o aluno de ter o código produzido disponível na plataforma e a possibilidade de acessar as atividades e continuar ou rever sua prática de sua casa ou de outro lugar com acesso à internet. Além de uma ferramenta operacional de suporte às atividades didáticas dos cursos de programação, o Machine Teaching é uma ferramenta de coleta e análise de dados educacionais. Os impactos da análise dos dados coletados pelo Machine Teaching já eram sentidos no ambiente presencial, mas se tornaram ainda mais evidentes no ensino remoto. É através

EduComp'22, Abril 24-29, 2022, Feira de Santana, Bahia, Brasil (On-line)

da observação dos dados analisados que oportunidades para adaptações e reorientações importantes das estratégias de ensino, do material didático e do processo de condução do curso são feitas.

Em seu atual estágio de desenvolvimento, o projeto de pesquisa que engloba o desenvolvimento e uso do Machine Teaching[1] tem como meta transformar as análises de dados implementadas em ações concretas que implementem melhorias no curso e no processo de aprendizagem. Para isso, nossa pesquisa inclui investigar os requisitos de uma ferramenta de apoio, os dados que necessitam ser coletados para a construção das análises e se a forma de apresentação das análises são satisfatórias para a tomada de decisão. São essas as questões de pesquisa:

(1) Quais requisitos uma ferramenta computacional didática de apoio às atividades de prática em programação introdutória deve ter para ela ser útil para alunos e professores?
(2) Quais dados devem ser coletados para auxiliar alunos e professores na tomada de decisão sobre os processos de ensino-aprendizagem?
(3) A partir de informações coletadas sobre o processo de aprendizagem dos alunos, podemos levar os professores a adaptarem suas estratégias de ensino?
(4) A partir de informações coletadas sobre o processo de aprendizagem de um aluno, podemos levar esse aluno a adaptar suas estratégias de estudo?

No restante do artigo abordaremos a concepção e o estado atual da ferramenta Machine Teaching. A Seção 2 faz uma revisão da literatura sobre sistemas de correção automática de códigos em Python. Em seguida, a metodologia de ensino e a solução tecnológica proposta são descritas nas Seções 3 e 4 respectivamente. Na Seção 5, nosso caso de aplicação é apresentado: o curso introdutório de programação em Python ministrado pelo Instituto de Computação da UFRJ. A Seção 6 discute os resultados encontrados. Por fim, a Seção 7 descreve as conclusões e direções para trabalhos futuros.

2 TRABALHOS RELACIONADOS

Ambientes online que fazem correção automática de código-fonte são comumente utilizados como ferramentas de apoio pedagógico em disciplinas de programação, principalmente por sua rapidez em prover *feedback* aos alunos. A Tabela 1 reúne ambientes de correção automática para códigos em Python retirados das revisões sistemáticas da literatura feitas por Ihantola *et. al* [10] e Luxton-Reilly *et. al* [13], além de dois sistemas desenvolvidos por universidades brasileiras que não foram contemplados na revisão realizada, e do sistema Machine Teaching, proposto nesse artigo. Esses sistemas são analisados e comparados em relação a três critérios:

(1) O sistema foi desenvolvido baseando-se em alguma metodologia pedagógica ou de ensino? Qual?
(2) O sistema provê uma área dedicada aos professores com estatísticas e gráficos para que ele possa ter tanto uma visão geral da turma quanto de alunos específicos? O que está disponível?
(3) O sistema foi avaliado? Por qual método?

[1]O projeto de pesquisa envolve também o aprimoramento da metodologia de ensino de programação implementada nos cursos que usam o Machine Teaching, entre outras questões nas áreas de pesquisa em educação em computação

Segundo os próprios artigos, oito dos 10 sistemas avaliados foram construídos baseados em uma metodologia pedagógica principal. A existência de uma metodologia por trás de uma ferramenta influencia nas formas de uso da ferramenta previstas para o aluno, de maneira que as interações do aluno com a ferramenta serão pautadas pelos padrões de aprendizagem previstos da metodologia [14]. É interessante observar a pluralidade de metodologias consideradas, não sendo elas mutuamente exclusivas – um mesmo sistema pode incorporar várias. Por exemplo, o sistema apresentado neste artigo utiliza como principal pilar pedagógico o ensino modularizado de programação, colocando a construção de funções como a primeira etapa de aprendizado, conforme descrito na Seção 3. Mas o sistema também incorpora o aprendizado baseado em testes, conforme descrito na Seção 4.2. Portanto, nesta tabela, consideramos apenas a metodologia norteadora de cada ferramenta, conforme descritas nos artigos que as apresentam. Já as ferramentas apresentadas em artigos que não mencionam nenhuma metodologia são basicamente ambientes para escrita e correção de código, diferenciando-se em suas funcionalidades.

Todos os sistemas suportam o uso por alunos e professores. Para os alunos, os sistemas geralmente funcionam como um ambiente de desenvolvimento integrado (IDE), onde os alunos podem escrever código e receber *feedback* em tempo real. No entanto, as funcionalidades para professores variam bastante. No UUhistle, por exemplo, os professores usam o sistema para fazer simulações e exemplos. A maioria dos sistemas permite que os professores vejam se o aluno passou ou não uma questão individual. Além disso, os sistemas URI Online Judge e o brasileiro CodeBench fornecem uma análise aprofundada dos envios dos alunos através de painéis com gráficos ou estatísticas sobre os alunos e as turmas.

O Machine Teaching se diferencia desses sistemas, pois além de contemplar o uso por alunos da maneira usual, tem um foco na coleta de dados para análise e suporte às decisões dos professores e dos próprios alunos. Os painéis apresentados no sistema proposto não contêm somente informação para que um professor entenda ou avalie um aluno. As informações são disponibilizadas de maneira que o professor e outros atores envolvidos na administração do curso possam refletir sobre a ordem de apresentação da ementa, a dificuldade dos conteúdos e os prazos de entrega, estimulando a reflexão e apoiando a constante atualização do curso.

3 METODOLOGIA DE ENSINO

Essa seção apresenta a metodologia de ensino utilizada como guia para projetar e implementar o ambiente de aprendizado Machine Teaching.

O curso introdutório de programação oferecido pelo Instituto de Computação da UFRJ (IC/UFRJ) para estudantes de carreiras que não são de Computação possui como objetivo o desenvolvimento das competências para a construção de programas legíveis e modulares em Python. A descrição detalhada da metodologia pode ser vista em Delgado *et al.* [5].

Esta metodologia é baseada na abordagem imperativa estruturada, enfatizando a resolução de problemas, decomposição em funções e domínio de habilidades básicas. Esta proposta didática inverte a ordem usual de apresentação de conteúdos, que normalmente começa com uma estrutura completa do programa incluindo interação

Tabela 1: Ambientes online que fazem correção automática de código em Python

Ambiente	1. Metodologia	2. Área para professores	3. Avaliação
CloudCoder [9, 19]	Não	Notas individuais	Não avaliado
CodeWorkout [18]	Exercício e prática (*Drill-and-practice*)	Notas individuais, exercícios resolvidos	Questionário
PCRS [28]	Aprendizado por pares	Notas individuais, exercícios resolvidos	Não avaliado
UUhistle [26]	Simulação visual de programa	Não	Questionário e grupo de controle
Pythy [8]	Não	Não	Questionário
Web-CAT [7]	Aprendizado baseado em testes	Notas individuais, exercícios resolvidos	Grupo de controle
URI Online Judge [2, 3]	Gamificação	Notas individuais, exercícios resolvidos, estatísticas por problema, lista de exercícios e turma	Questionário
PEEF [1]	Aprendizado baseado em testes	Notas individuais, exercícios resolvidos	Não avaliado
CodeBench [6]	Ensino híbrido	Notas, exercícios resolvidos, estatísticas por problema, lista de exercícios e turma	Questionário e grupo de controle
Machine Teaching [15, 17]	Programação imperativa (funções primeiro)	Notas individuais, exercícios resolvidos, estatísticas por aluno, problema, lista de exercícios e turma	Questionário

do usuário que utiliza instruções como *"print"*, *"input"* e *"__main__"*. O diferencial da metodologia adotada no Machine Teaching é que a ênfase do curso consiste na construção de módulos de código concisos e independentes, deixando os mecanismos de interação com o usuário para o final do curso (quando o aluno já domina o básico e o pensamento modular). O conteúdo programático do curso é dividido em 12 semanas, conforme listado abaixo:

1. Introdução ao curso e Função
2. Funções e Tipos de dados
3. Tipos de dados, Strings, Estrutura Condicional
4. Variáveis e Atribuição, Strings e Tuplas
5. Manipulação de Strings, Tuplas e Listas
6. Listas e Dicionários
7. Funções para manipulação de iteráveis
8. Estrutura de Repetição — while
9. Estrutura de Repetição — for
10. Laços Aninhados e Matrizes
11. Interação com o usuário e programa principal
12. Modularização (incluindo o programa principal)

Na modalidade de ensino presencial, os estudantes têm aulas de duas horas duas vezes na semana, uma em sala de aula, e outra em laboratório. O tempo de laboratório é usado para a construção de código conforme listas de exercício organizadas por conteúdo e nível de proficiência preparadas para cada aula. No ensino remoto, o curso foi adaptado para o esquema de sala de aula invertida, onde roteiros de estudo são disponibilizados a cada semana, e devem ser lidos antes do primeiro encontro síncrono da semana, dedicado à discussão dos assuntos abordados no roteiro. Após o primeiro encontro síncrono da semana, são passadas as atividades práticas. Um segundo encontro síncrono na semana é dedicado a apoiar os estudantes nessas atividades.

4 METODOLOGIA PARA DESENVOLVIMENTO DO SISTEMA

Essa seção apresenta cada etapa da metodologia ágil utilizada para o projeto e desenvolvimento do sistema. No desenvolvimento ágil, a ideia principal é fornecer protótipos funcionais para serem testados pelos usuários em versões iterativas de quatro etapas [21]:

(1) Planejamento
(2) Projeto
(3) Implementação
(4) Validação do sistema

O Machine Teaching está sendo desenvolvido sob a licença GNU General Public License (GLP).

4.1 Planejamento

O planejamento começa com a elicitação de requisitos. É importante notar que, no desenvolvimento ágil, as funcionalidades resultantes desse levantamento são priorizadas e implementadas, não sendo disponibilizadas na mesma versão para os usuários. Ainda, elas estão em constante melhoria, aprimoradas conforme o uso e *feedback* dos usuários. Depois do levantamento dos requisitos e funcionalidades necessárias, é preciso determinar quais dados são necessários para realizar as análises propostas.

4.1.1 Elicitação de requisitos. Requisitos são um conjunto de descrições que determinam o que um sistema deve fazer e suas restrições operacionais. Sommerville [25] os divide em dois tipos: requisitos do sistema e requisitos do usuário. Os requisitos do sistema são descrições detalhadas com métricas objetivas que podem ser testadas posteriormente, enquanto os requisitos do usuário são declarações em linguagem natural que expressam os recursos desejados pelos usuários. Neste artigo, apresentamos somente os requisitos do

usuário, entendendo quais funcionalidades o sistema deve oferecer para cada um dos usuários. Para identificar os usuários, pensamos em todos os atores envolvidos nos processos de ensino aprendizagem que serão suportados pelo sistema, e que decisões cabem a cada um deles. Identificamos três tipos de usuários: estudantes, professores e coordenadores. No nosso cenário específico, há uma pequena equipe envolvida com a gestão dos cursos de programação (duas a quatro pessoas). Usamos o termo "coordenador"para nos referirmos aos membros dessa equipe. Para construir os requisitos de usuário, utilizamos o formato de história de usuários, onde cada funcionalidade desejada pode ser expressa como uma frase no formato: *"Como usuário <tipo de usuário>, eu gostaria de <objetivo>, para que eu possa <benefícios>"* [4, 12], respondendo às perguntas "para quem?", "o que?" e "por que?", respectivamente. A Tabela 2 apresenta os requisitos em formato de história do usuário.

4.1.2 Dados coletados. Uma vez definidos os requisitos, foram levantados os dados necessários para que as análises propostas pudessem ser realizadas. Os dados coletados e seu uso pretendido no sistema estão descritos abaixo:

(1) Dados relacionados ao resultado do aluno:
   (a) Porcentagem de casos de teste bem-sucedidos: é utilizado para calcular o resultado e pode ser usado para estimar a conclusão do curso pelo aluno.
   (b) Resultado da questão, pode ser aprovado, reprovado ou ignorado. Aprovado significa que o aluno respondeu à pergunta corretamente ou submeteu um código que passou em todos os casos de teste; reprovado significa que não foi aprovado em pelo menos um caso de teste e ignorado significa que o aluno pulou aquela pergunta e não respondeu. Essas informações podem ser usadas para entender as dificuldades dos alunos.
(2) Dados relacionados à resposta do aluno:
   (a) Solução do aluno: verificar os conceitos utilizados pelos alunos, descobrir erros e exceções de código.
   (b) Resultado do código do aluno: somente para questões do tipo código, esse resultado contém os valores de saída do código do aluno ou erros. É útil para entender se as saídas dos alunos apresentam os tipos e formatos esperados e em quais casos de teste a solução foi aprovada ou reprovada.
   (c) Tipo do erro: somente para questões do tipo código, armazena os tipos de erros e exceções dos códigos submetidos pelos alunos.
(3) Dados relacionados ao tempo:
   (a) Segundos no código, segundos na página e segundos para começar: esses três recursos combinados são usados para medir as dificuldades dos alunos.
   (b) Data e hora das submissões: utilizado para diferenciar o aprendizado de curto e longo prazo.

4.2 Projeto

O projeto começa com o desenho das interfaces e como será a interação do usuário com o sistema. Esta seção apresenta a versão atual das interfaces, que já estão em sua terceira versão. Cada interface visa completar um requisito de usuário definido na Tabela 2.

As perguntas definidas pelos professores podem ser de três tipos: resolução de um problema com código, múltipla escolha ou texto curto. Para resolução em código, foi criado um ambiente de desenvolvimento integrado (IDE), onde os alunos são apresentados ao problema e devem escrever o código em um espaço de texto livre. Para cada exercício, uma função geradora de casos de teste foi definida para corrigir as respostas. Os alunos recebem *feedback* sempre que enviam uma resposta e podem ver se foram aprovados ou reprovados em cada caso de teste. Se eles acertarem todos, a tarefa é considerada concluída e o aluno pode passar para outro problema. As informações são armazenadas na granularidade de submissão, gerando uma entrada no banco de dados para cada tentativa do aluno. O conjunto de submissões é utilizado para gerar estatísticas agregadas em relação a problemas, conteúdo, alunos, professores, curso, entre outras. Esta IDE com a avaliação automatizada atende aos requisitos 1 e 3, e sua interface pode ser vista na Fig. 1. As questões de múltipla escolha e texto curto somente consideram correta a resposta exatamente igual à resposta cadastrada pelo professor.

Os alunos têm acesso a dois painéis onde podem visualizar os exercícios da semana, o tempo esperado de conclusão, suas estatísticas de conclusão e se comparar com seus colegas, conforme mostrado nas Figs. 2 e 3. Estes painéis mostram quanto tempo o aluno levou para terminar a aula, o número de erros, erros mais comuns e o progresso total de cada aula. Estas interfaces foram projetadas para atender ao requisito 2.

O sistema oferece três interfaces para que os professores possam analisar as submissões dos alunos ao nível individual ou agregado por turma. Figs. 4 e 5 apresentam a interface com o detalhamento para cada aluno, incluindo os códigos, seus resultados finais e data e hora de cada submissão. Em especial, a interface da Fig. 5 foi solicitada pelos professores para poder comparar as submissões individuais dos alunos. Essa interface mostra todos os envios de alunos, lado a lado, com as respectivas data e hora de submissão e porcentagem de casos de teste para um único problema. Dessa forma, o professor pode detectar erros comuns entre os alunos. Essas duas interfaces foram projetadas para atender ao requisito 4.

Para apoiar a análise por turma, o painel da Fig. 6 foi proposto. Nesse painel, os professores conseguem verificar o conteúdo e os problemas que os alunos demoraram mais tempo para resolver ou que precisaram de mais tentativas, o que pode indicar a necessidade de ajuste na atividade didática. Informações sobre alunos individuais consolidadas também estão disponibilizadas. Portanto, esse painel foi projetado para atender aos requisitos 4 e 5.

O requisito 6 está planejado, mas ainda está sendo projetado. A priorização do desenvolvimento ágil favoreceu o projeto e implementação das funcionalidades para alunos e, posteriormente, professores. É preciso primeiro ter a adesão dos alunos ao sistema para que a coleta de dados possa ser feita. Como segunda etapa, é preciso a adesão dos professores para utilizar a ferramenta regularmente em suas aulas. A disciplina apoiada pela ferramenta é uma disciplina que conta com uma grande quantidade de professores, pois ela é oferecida para a maioria dos cursos da área de exatas. Portanto, é importante a adesão dos professores para que as análises oferecidas à coordenação abordem uma quantidade abrangente de turmas. Por fim, a adesão da coordenação para realizar a análise de diversas turmas e cursos é o próximo passo do sistema que será discutido em trabalhos futuros na seção 7.

Tabela 2: Requisitos do usuário

ID	Para quem?	O que?	Por que?
1	Estudante	Resolver exercícios	Para praticar o conteúdo
2	Estudante	Visualizar estatísticas de finalização de exercícios, taxa de erros e erros mais comuns	Para entender os pontos de atenção e direcionar os estudos
3	Professor	Prover exercícios com correção automática	Para diminuir a sobrecarga com tarefas manuais repetitivas e aumentar o tempo de qualidade com os alunos
4	Professor	Visualizar estatísticas de finalização de exercícios, taxa de erros e erros mais comuns	Para entender os pontos de atenção de cada aluno, diagnosticar equívocos e direcionar os estudos
5	Professor	Visualizar estatísticas de finalização e erros agregadas por turmas e problemas	Para entender os conteúdos que os alunos possuem maior dificuldade, analisar o prazo de entrega dos exercícios, a ordem de apresentação do conteúdo e a eficiência e eficácia das atividades didáticas propostas
6	Coordenador	Visualizar estatísticas de finalização e erros agregadas por professores, cursos, turmas e problemas	Para procurar padrões no abandono das disciplinas, a ordem de apresentação do conteúdo e a eficiência e eficácia das atividades didáticas propostas

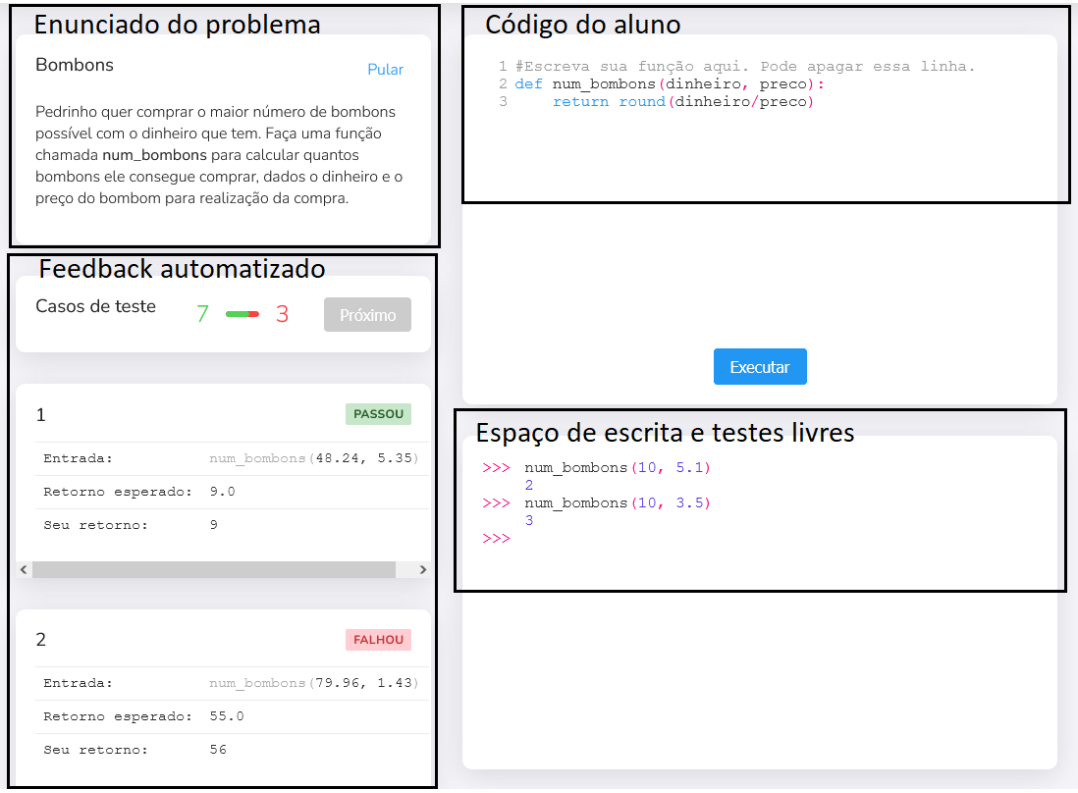

Figura 1: Ambiente de desenvolvimento integrado

4.3 Implementação

O sistema é construído com o *framework* Django, que utiliza a linguagem Python. O *framework* permite o gerenciamento do banco de dados, a geração de casos de teste dos exercícios e a utilização de bibliotecas gráficas. Um ponto importante no sistema é a segurança através do isolamento dos códigos dos alunos do servidor. Nos exercícios, os códigos e sua correção são executados no cliente, utilizando as bibliotecas Skulpt e Pyodide. Para a criação dos dashboards é utilizada a biblioteca gráfica Plotly, que permite a criação de visualizações interativas.

4.4 Validação do sistema

Ao final de cada período letivo, a usabilidade e impacto na tomada de decisões são avaliados pelos usuários através de questionários. Os questionários utilizam a escala Likert com cinco valores, onde

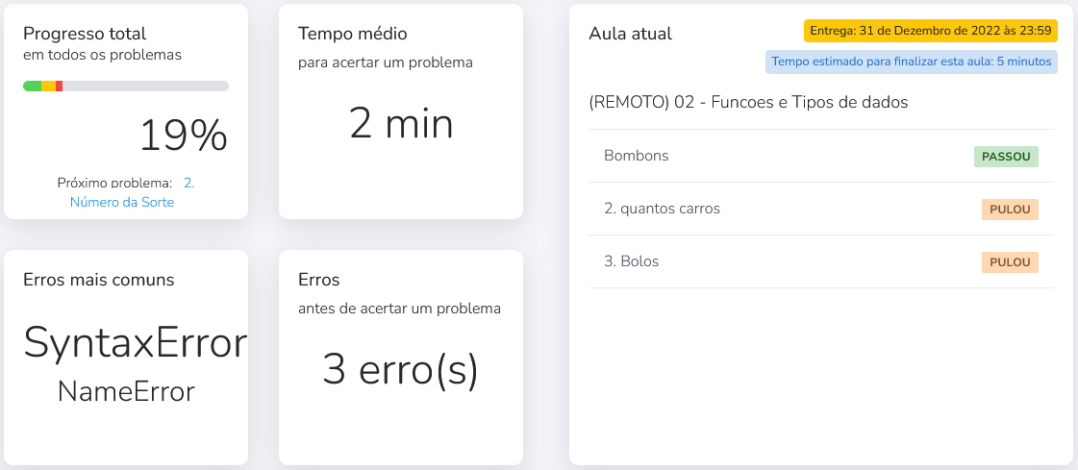

Figura 2: Painel inicial do aluno ao entrar no sistema

Figura 3: Painel com estatísticas do aluno

algumas afirmações são propostas e o usuário deve escolher um valor entre "Discordo completamente" (nota 1) e "Concordo completamente" (nota 5) para cada uma delas. Os questionários também possuem espaço de texto livre para sugestões e críticas que direcionem na criação de novas visualizações e melhorias no sistema. A participação é anônima e voluntária.

4.4.1 Pesquisa de Usabilidade do Sistema. Para avaliar o uso do Machine Teaching pelos usuários, fizemos perguntas sobre a usabilidade do sistema para a realização das tarefas semanais pelos alunos. Foram feitas as seguintes perguntas:

(1) O sistema possui uma interface de uso amigável.
(2) Eu tive facilidade em encontrar os exercícios da lista da semana para fazer.
(3) Eu tive facilidade em encontrar os exercícios das listas das semanas passadas para estudar.

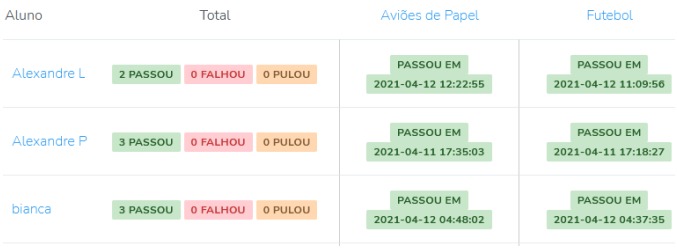

Figura 4: Interface com o resultado de cada aluno para um conteúdo específico.

4.4.2 Pesquisa de Apoio à Tomada de Decisão. Para responder às nossas perguntas e começar a investigar se é possível impactar a tomada de decisão de alunos e professores, perguntamos a eles se as informações exibidas nos dashboards os ajudaram na organização das atividades e na conscientização do tempo despendido em cada atividade. Foram feitas as seguintes perguntas:

(1) Sobre a organização do tempo:
   (a) **Para alunos:** o dashboard me ajudou a organizar meu tempo de estudo.
   (b) **Para professores:** o sistema me ajudou a ter uma melhor visão do esforço e/ou tempo que os alunos gastam fazendo os exercícios.
(2) Sobre organização do conteúdo:
   (a) **Para alunos:** o dashboard me ajudou a orientar o conteúdo que devo estudar, identificando os erros mais comuns (em relação ao dashboard da Fig. 2).
   (b) **Para professores:** o sistema me ajudou a identificar o conteúdo que é mais difícil para os alunos.

4.5 Ciclos de atualização do sistema

Conforme descrito nesta seção, o sistema é desenvolvido utilizando metodologia ágil. Cada período letivo corresponde a um ciclo de iteração da metodologia: planejamento, projeto, implementação e validação. Ao final de cada validação, os novos requisitos ou melhorias nos requisitos antigos são determinados e priorizados para implementação ao longo do período. No entanto, grandes atualizações no sistema só são feitas ao final de cada período, quando o sistema já não está mais em uso e testes mais robustos de usabilidade e funcionamento podem ser realizados, garantindo sua estabilidade na utilização durante o período. A Tabela 3 descreve as principais modificações realizadas no sistema ao longo dos períodos letivos.

Tabela 3: Principais modificações em cada período letivo

Período letivo	Modificações
2018	Lançamento, utilização em aulas pontuais para revisão de conteúdo
2019/2	A partir desse semestre, o sistema começou a ser utilizado semanalmente como ambiente para submissão de exercícios
PLE	Atualização da interface, sem novas funcionalidades
2020/1	Atualização da interface, incorporação do espaço de testes livres, monitoramento das páginas visitadas, coleta de erros dos alunos e dashboards para alunos
2020/2	Incorporação de dashboard para professores
2021/1	Inclusão do protótipo da funcionalidade de previsão de conclusão da disciplina

5 RESULTADOS

O sistema Machine Teaching vem sendo adotado por cada vez mais professores nos cursos introdutórios de programação oferecido pelo Instituto de Computação da UFRJ para estudantes das carreiras de exatas. Isso pode ser visto através do número de turmas que adotaram o sistema ao longo dos anos: em 2019/2 foram 4 turmas, passando para 15 em 2020/1 e para 22 em 2021/1. Cada turma possui em média 30 alunos. Esses dados são corroborados pela Figura 7 que mostra a quantidade de alunos que se inscreveram, que ativamente utilizaram o sistema até o final do curso e que responderam o questionário em cada período letivo. Foi mantida uma média de 50% de resposta do questionário em relação aos alunos ativos, ou seja, aqueles que utilizaram o sistema até o final do curso.

No momento, o sistema construído está sendo usado por 22 turmas e 800 alunos. No total, mais de 2200 alunos usaram o sistema nos últimos três anos. Estamos trabalhando para disponibilizar o Machine Teaching para uso em outras instituições de ensino públicas. Isso requer, além da atualização operacional, um contato próximo com professores e coordenadores de curso dispostos a utilizá-lo. Nossa expectativa é ter mais uma instituição ainda este ano, e pelo menos duas no próximo ano.

Em relação à usabilidade do sistema, houve uma melhora na primeira atualização da interface, elevando a mediana do item "O sistema possui uma interface de uso amigável" da nota 4 para a nota 5. No entanto, essa atualização também piorou um pouco o item "Eu tive facilidade em encontrar os exercícios das listas das semanas passadas para estudar". Uma nova atualização de interface foi feita entre os períodos PLE e 2020/1, que elevou a mediana novamente de 4 para 5. O último item de usabilidade "Eu tive facilidade em encontrar os exercícios da lista da semana para estudar" já obteve mediana 5 desde a primeira versão do sistema. As Figs. 8, 9 e 10 apresentam os boxplots das notas para cada um dos períodos letivos.

As Figs. 11 e 12 mostram a percepção dos alunos e professores sobre o apoio à tomada de decisões. A maioria dos alunos não utiliza as informações fornecidas pelo painel da Fig. 3 para ajustar o tempo dedicado ao estudo da matéria e cerca de 40% deles utilizam os erros mais comuns para orientar seus estudos. Planejamos lançar uma versão do painel nova e simplificada para os alunos. A versão atual combina muitas informações em uma única tela. Estamos projetando o novo dashboard para fornecer dicas mais diretas aos alunos sobre como melhorar o aprendizado.

A percepção dos professores sobre a influência do sistema na tomada de decisão foi mais positiva que a dos alunos. Antes de sua adoção, os alunos costumavam enviar arquivos por e-mail anexando o código da resposta. Os professores não tinham como saber quanto tempo cada aluno despendia em cada exercício e quais tinham mais dificuldade em cada questão. Isso pode ser verificado pela avaliação dos professores sobre as questões de organização de tempo e conteúdo. Disponibilizamos o painel de educadores (Fig. 6) no

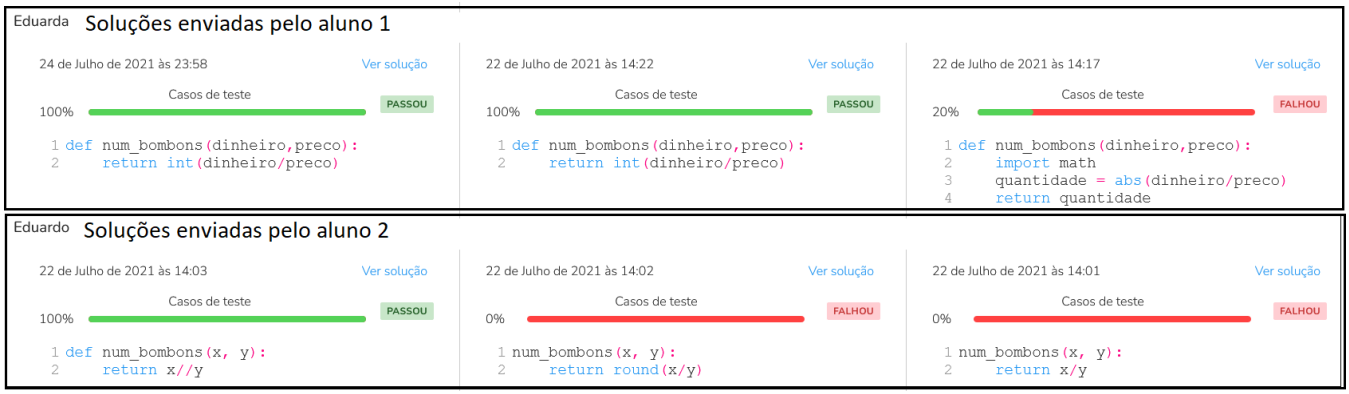

Figura 5: Interface com o detalhamento para cada aluno das soluções para um problema específico.

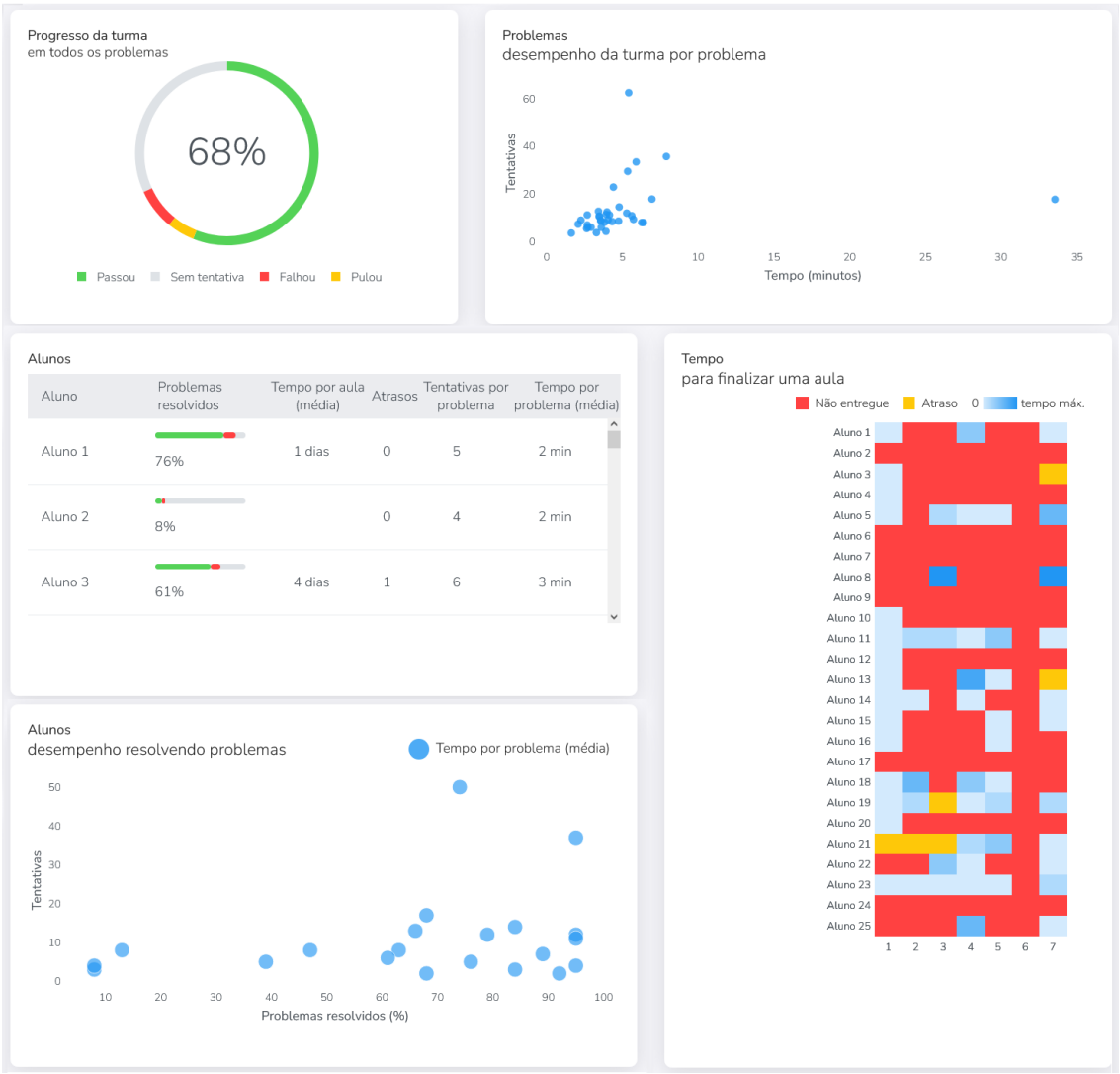

Figura 6: Painel para uma turma. Contém o progresso geral da turma, indicações dos problemas e conteúdos que precisaram de mais tempo ou mais tentativas e estatísticas de completude por aluno.

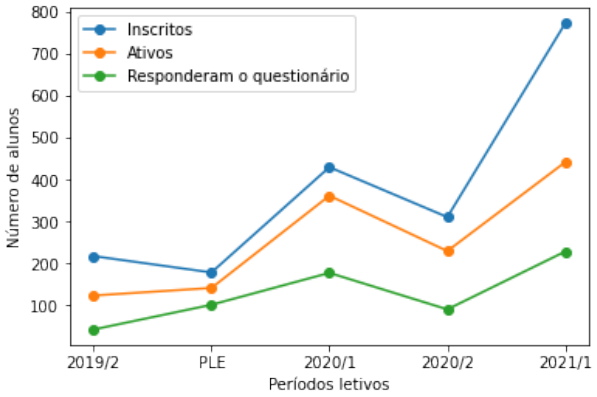

Figura 7: Quantidade de alunos inscritos, ativos e que responderam o questionário por período letivo.

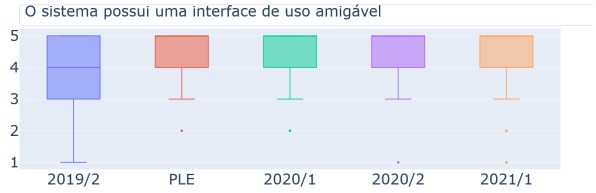

Figura 8: Notas por período letivo para a pergunta "O sistema possui uma interface de uso amigável"

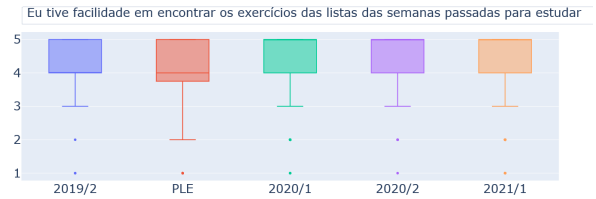

Figura 9: Notas por período letivo para a pergunta "Eu tive facilidade em encontrar os exercícios das listas das semanas passadas para estudar"

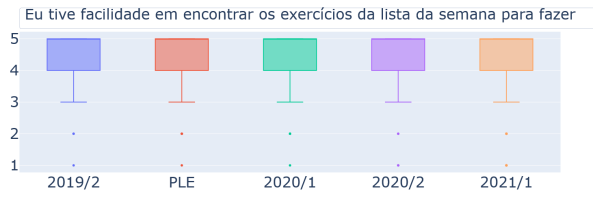

Figura 10: Notas por período letivo para a pergunta "Eu tive facilidade em encontrar os exercícios da lista da semana para estudar"

período letivo de 2020/2, aumentando a percepção positiva de 17% para 75%.

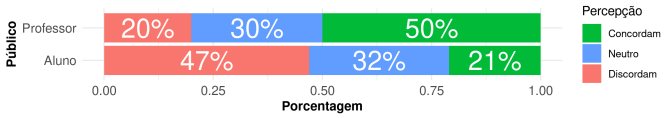

Figura 11: Percepção de alunos e professores sobre a tomada de decisão em relação ao tempo despendido nas atividades usando as informações fornecidas pelo painel.

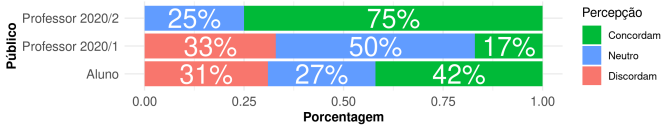

Figura 12: Percepção de alunos e professores sobre a tomada de decisão em relação ao conteúdo das atividades usando as informações fornecidas pelo painel.

Em relação às sugestões de texto livre, como o sistema sofreu grandes modificações de interface e funcionalidades entre os períodos 2019/2 e PLE e depois em PLE e 2020/1, analisamos aqui somente as sugestões a partir de 2020/1 como sugestões de melhoria no sistema. As sugestões anteriores a esse período já foram consideradas e implementadas. Dos 495 alunos que preencheram o formulário nesse período, 179 preencheram o campo de sugestões. Dessas sugestões, conseguimos identificar as prioridades para novas funcionalidades requisitadas pelos alunos. São elas:

(1) Maior clareza na indicação do erro, com explicações mais elaboradas.
(2) Biblioteca para salvar códigos passados que possam ser reutilizados, como um módulo.
(3) Indicação do passo-a-passo do código, como no Python Tutor [22].

Em relação aos professores, a priorização das funcionalidades seguindo suas sugestões são:

(1) Indicação de respostas semelhantes entre alunos.
(2) Melhorar a interface com apresentação dos códigos para facilitar a correção.

6 DISCUSSÃO

Consideramos que em seu estado atual de desenvolvimento o sistema gerou resultados que levam a impactos no curto, médio e longo prazo. Os impactos a **curto prazo** estão ligados às questões operacionais do curso, como, por exemplo, acesso facilitado dos alunos às atividades práticas e agilidade para correção das atividades dos alunos. Esses são os impactos usuais esperados de uma ferramenta de apoio ao ensino. Além disso, o sistema gerou, no **médio prazo**, a possibilidade de transição de práticas presenciais para práticas remotas, onde o aluno está localizado em sua casa, e pode contar com o *feedback* imediato provido pela ferramenta. Horários para auxílio remoto às atividades práticas foram combinados com as turmas, o que se mostrou efetivo por ser a ferramenta um ambiente de prática controlado ao qual professores, alunos e monitores tem acesso às informações. No médio prazo também temos a possibilidade do professor visualizar o desempenho de alunos e turmas no cumprimento

de suas práticas, e consequentemente fazer adaptações e intervenções cabíveis em seu plano de curso. Impactos a **longo prazo** são consequência do uso efetivo de recursos de análise de dados para a melhoria do processo de ensino-aprendizagem, da metodologia de ensino, e da pesquisa sobre ensino de programação. Esses aspectos serão melhor discutidos na sessão 6.1, a seguir. Temos ainda, no longo prazo, o impacto social gerado pela aquisição da competência da programação, discutido na seção 6.2.

6.1 Análise de dados educacionais para melhoria do curso

Por parte dos alunos, a visualização de informações acerca de sua dedicação e desempenho para cumprir as tarefas os faz ter mais consciência sobre seu processo de aprendizagem e consequentemente, expectativas mais realistas. Com incentivo do professor, essa consciência será um incentivo para que o aluno organize melhor seu tempo para práticas e estudo. Os professores, por sua vez, conseguem visualizar sem nenhum esforço extra, dados consolidados por turma e por tarefa. A visão consolidada permite aprimorar a percepção acerca do impacto das tarefas nos grupos de alunos, insumo útil para decisões acerca de adaptações de prazos e atividades, bem como ordenação de conteúdos (opções didáticas que afetam todos os alunos) e que partes do curso são pontos de confusão que poderiam ser alvo de melhoria, tanto em relação ao planejamento, avaliação, e volume de tarefas, quanto a uma revisão da própria metodologia de ensino. Essa visão do todo também permite identificar os alunos que se destacam positiva e negativamente dos demais, como, por exemplo, alunos que precisam de maior atenção. A informação de como um aluno progride em uma atividade é um recurso valioso para acessar o processo cognitivo por trás do aprendizado de um conteúdo, o que alimenta pesquisas em didática da computação. Os dados coletados com essa ferramenta já foram disponibilizados [15] e utilizados para: testar recomendações de conteúdo para alunos [27] e clusterizar respostas de alunos em conceitos trabalhados em programação [16].

6.2 Impacto social

Devido à pandemia de COVID-19, uma transição abrupta da modalidade de ensino presencial para online teve que ser enfrentada por professores e alunos. O sistema foi de grande ajuda para apoiar essa transição, pois proporcionou aos alunos a possibilidade de realizarem suas práticas sozinhos e em seus próprios ambientes. Consideramos este um impacto social considerável, visto que os cursos introdutórios de programação são estratégicos para alunos que necessitaram iniciar prematuramente a carreira profissional devido aos prejuízos financeiros familiares sofridos como efeitos colaterais da crise em torno da pandemia de COVID-19.

7 CONSIDERAÇÕES FINAIS

A validação do sistema evidenciou que a usabilidade do sistema foi aprovada pelos dois tipos de usuários-alvo: alunos e professores. Acreditamos que a metodologia ágil de construção do sistema contribuiu para isso, permitindo que a usabilidade e os dados coletados fossem aprimorados a cada iteração e resultando na adoção do sistema por mais professores e turmas ao longo do tempo, respondendo às perguntas de pesquisa 1 e 2.

Considerando a meta de transformar as análises de dados em ações concretas para a melhoria do curso, como referenciado nas perguntas de pesquisa 3 e 4, a validação evidenciou que os dashboards apresentados conseguiram apoiar a decisão de professores em alguns aspectos como a organização do conteúdo de estudo e a percepção em relação ao tempo despendido pelos alunos nas atividades. Para os alunos, no entanto, apesar de o dashboard ter apoiado as decisões em relação ao conteúdo a ser estudado fornecendo informações sobre os erros mais frequentes, não observamos a tomada de decisão em relação ao tempo dedicado às atividades e ao estudo, sendo um ponto de melhoria em dashboards futuros.

Como trabalhos futuros, atualizaremos o sistema com um dashboard para apoiar a tomada de decisão dos coordenadores. Esse dashboard deve conter dados agregados dos cursos dos alunos, turmas e professores, permitindo adaptações na operacionalização dos cursos (oferta, quantidade de vagas, turmas por professor, etc.). Ainda, temos em vista expandir o uso da ferramenta para outros cursos, como, por exemplo, programação orientada a objetos. Porém, temos em mente que qualquer expansão deste tipo deve ser acompanhada de uma pesquisa sobre estratégias pedagógicas para realização de práticas em programação fortemente acopladas a uma metodologia de ensino própria.

Outro ponto a ser atacado na continuidade do projeto é a condução de uma proposta mais completa e fundamentada de validação do sistema, com o uso de métodos científicos específicos para este propósito. A validação atual foi baseada na percepção dos usuários acerca da usabilidade e da influência do sistema na tomada de decisão. Porém, outros indicadores podem ser usados no processo de validação, tais como as notas dos estudantes no curso antes e depois do uso da ferramenta. Além disso, temos como plano a condução de um estudo de caso contemplando o acompanhamento de uma turma para entender melhor a maneira como alunos e professores usam o sistema, como acessam as informações dos dashboards, e como as ações desses usuários relativas às decisões que já identificamos que o sistema pode influenciar evoluem, considerando também e se essas ações tem uma associação perceptível com uso do sistema. Acreditamentos que este seria um passo importante para entender melhor o impacto da ferramenta no processo de ensino-aprendizagem.

8 RECURSOS ONLINE

(1) Machine Teaching: www.machineteaching.tech
(2) Nossa apresentação no Festival do Conhecimento da UFRJ 2021: https://www.youtube.com/watch?v=7GCnZ_WyH6U
(3) Github: https://github.com/laura-moraes/machine-teaching

9 AGRADECIMENTOS

O presente trabalho foi realizado com o apoio do CNPq (financiamentos 141089/2016-4, 306258/2019-6 e bolsa PIBIC), FAPERJ (financiamento E26-200.840/2021-CNE) e CAPES (financiamento PROEX - 1201036).

REFERÊNCIAS

[1] Luis Gustavo Araujo, Roberto Bittencourt, and Christina Chavez. 2021. Python Enhanced Error Feedback: Uma IDE Online de Apoio ao Processo de Ensino-Aprendizagem em Programação. In *Anais do Simpósio Brasileiro de Educação em*

Computação (On-line). SBC, Porto Alegre, RS, Brasil, 326–333. https://doi.org/10.5753/educomp.2021.14500

[2] Jean Luca Bez, Carlos E. Ferreira, and Neilor Tonin. 2013. URI Online Judge Academic: A Tool for Professors. In *Proceedings of the 2013 International Conference on Advanced ICT and Education* (Hainan, China). 744–747. https://doi.org/10.2991/icaicte.2013.153

[3] Jean Luca Bez, Neilor Tonin, and Fabio Zanin. 2012. Enhancing traditional Algorithms classes using URI Online Judge. In *2012 International Conference on e-Learning and e-Technologies in Education (ICEEE)* (Lodz, Poland). 110–113. https://doi.org/10.1109/ICeLeTE.2012.6333402

[4] Mike Cohn. 2004. *User Stories Applied: For Agile Software Development* (1 ed.). Addison-Wesley Professional, IL, USA.

[5] Delgado, C. et al. 2016. The teaching of functions as the first step to learn imperative programming. In *Anais do Workshop sobre Educação em Computação (WEI)*. Sociedade Brasileira de Computação - SBC, 388–397. https://doi.org/10.5753/wei.2016.9683

[6] Leandro Galvão e David Fernandes e Bruno Gadelha. 2016. Juiz online como ferramenta de apoio a uma metodologia de ensino híbrido em programação. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)* 27, 1, 140–149. https://doi.org/10.5753/cbie.sbie.2016.140

[7] Stephen H. Edwards and Manuel A. Perez-Quinones. 2008. Web-CAT: automatically grading programming assignments. In *Proc. 13th Annu. Conf. on Innovation and Technology Computer Science Education*. Madrid, Spain, 328. https://doi.org/10.1145/1384271.1384371

[8] Stephen H. Edwards, Daniel S. Tilden, and Anthony Allevato. 2014. Pythy: improving the introductory python programming experience. In *Proc. 45th ACM technical symposium on Computer science education*. Atlanta, GA, USA, 641–646. https://doi.org/10.1145/2538862.2538977

[9] David Hovemeyer and Jaime Spacco. 2013. CloudCoder: A web-based programming exercise system. *J. Comput. Sci. in Colleges* 28, 3, 30.

[10] Petri Ihantola, Arto Vihavainen, Alireza Ahadi, Matthew Butler, Jürgen Börstler, Stephen H. Edwards, Essi Isohanni, Ari Korhonen, Andrew Petersen, Kelly Rivers, Miguel Ángel Rubio, Judy Sheard, Bronius Skupas, Jaime Spacco, Claudia Szabo, and Daniel Toll. 2015. Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies. In *Proc. 2015 ITiCSE Working Group Report*. Vilnius, Lithuania, 41–63. https://doi.org/10.1145/2858796.2858798

[11] Jelena Jovanović, Shane Dawson, Srećko Joksimović, and George Siemens. 2020. Supporting actionable intelligence: reframing the analysis of observed study strategies. In *Proceedings of the 10th International Conference on Learning Analytics & Knowledge (LAK '20)*. Association for Computing Machinery, New York, NY, USA, 161–170. https://doi.org/10.1145/3375462.3375474

[12] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn E. M. van der Werf, and Sjaak Brinkkemper. 2016. The Use and Effectiveness of User Stories in Practice. In *International working conference on requirements engineering: Foundation for software quality (LNCS, Vol. 9619)*. 205–222. https://doi.org/10.1007/978-3-319-30282-9_14

[13] Andrew Luxton-Reilly, Simon, Ibrahim Albluwi, Brett A. Becker, Michail Giannakos, Amruth N. Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. 2018. Introductory Programming: A Systematic Literature Review. In *Proc. Companion 23rd Annu. ACM Conf. Innovation and Technology in Computer Science Education (ITiCSE 2018 Companion)*. Larnaca, Cyprus, 55–106. https://doi.org/10.1145/3293881.3295779

[14] Carlos Monroy, Virginia Snodgrass Rangel, and Reid Whitaker. 2013. STEMscopes: contextualizing learning analytics in a K-12 science curriculum. In *Proceedings of the Third International Conference on Learning Analytics and Knowledge (LAK '13)*. Association for Computing Machinery, New York, NY, USA, 210–219. https://doi.org/10.1145/2460296.2460339

[15] Laura O. Moraes and Carlos Eduardo Pedreira. 2020. Designing an Intelligent Tutoring System Across Multiple Classes. In *4th Educational Data Mining in Computer Science Education Workshop* (Virtual).

[16] Laura O. Moraes and Carlos Eduardo Pedreira. 2021. Clustering Introductory Computer Science Exercises Using Topic Modeling Methods. *IEEE Trans. Learn. Technol.* 14, 1, 42–54. https://doi.org/10.1109/TLT.2021.3056907

[17] Laura O. Moraes, Carlos Eduardo Pedreira, Carla Delgado, and João Pedro Freire. 2021. Supporting Decisions Using Educational Data Analysis. In *Anais Estendidos do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia)*. SBC, 99–102. https://doi.org/10.5753/webmedia_estendido.2021.17622

[18] Krishnan Panamalai Murali. 2016. *CodeWorkout: Design and implementation of an online drill-and-practice system for introductory programming*. Thesis. Virginia Tech. https://vtechworks.lib.vt.edu/handle/10919/81072

[19] Andrei Papancea, Jaime Spacco, and David Hovemeyer. 2013. An open platform for managing short programming exercises. In *Proc. 2013 ACM Conf. Int. Computing Education Research*. San Diego, CA, USA, 47–52. https://doi.org/10.1145/2493394.2493401

[20] Abelardo Pardo, Kathryn Bartimote, Simon Buckingham Shum, Shane Dawson, Jing Gao, Dragan Gašević, Steve Leichtweis, Danny Liu, Roberto Martínez-Maldonado, Negin Mirriahi, Adon Christian Michael Moskal, Jurgen Schulte, George Siemens, and Lorenzo Vigentini. 2018. OnTask: Delivering Data-Informed, Personalized Learning Support Actions. *Learning Analytics* 5, 3, 235–249. https://doi.org/10.18608/jla.2018.53.15

[21] Roger S. Pressman and Bruce Maxim. 2014. *Software Engineering: A Practitioner's Approach* (8ª edição ed.). McGraw-Hill Science/Engineering/Math, New York, NY.

[22] Python Tutor. [n.d.]. Python Tutor. https://pythontutor.com/. Online; accessed 28-October-2021.

[23] Yizhou Qian and James Lehman. 2017. Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review. *ACM Trans. Comput. Educ.* 18, 1, Article 1, 24 pages. https://doi.org/10.1145/3077618

[24] Carolyn P. Rosé, Elizabeth A. McLaughlin, Ran Liu, and Kenneth R. Koedinger. 2019. Explanatory learner models: Why machine learning (alone) is not the answer. *British Journal of Educational Technology* 50, 6, 2943–2958. https://doi.org/10.1111/bjet.12858

[25] I. Sommerville. 2011. *Engenharia de software* (9 ed.). Pearson Prentice Hall, São Paulo, Brasil. 57–80 pages.

[26] Juha Sorva and Teemu Sirkiä. 2010. UUhistle: a software tool for visual program simulation. In *Proc. 10th Koli Calling Int. Conf. Computing Education Research*. Koli, Finland, 49–54. https://doi.org/10.1145/1930464.1930471

[27] Chunpai Wang, Shaghayegh Sahebi, Siqian Zhao, Peter Brusilovsky, and Laura O. Moraes. 2021. Knowledge Tracing for Complex Problem Solving: Granular Rank-Based Tensor Factorization. In *Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization* (Utrecht, Netherlands) *(UMAP '21)*. Association for Computing Machinery, New York, NY, USA, 179–188. https://doi.org/10.1145/3450613.3456831

[28] Daniel Zingaro, Yuliya Cherenkova, Olessia Karpova, and Andrew Petersen. 2013. Facilitating code-writing in PI classes. In *Proc. 44th ACM Technical Symp. Computer Science Education*. Denver, CO, USA, 585–590. https://doi.org/10.1145/2445196.2445369