ADHP: AMBIENTE PARA O DESENVOLVIMENTO DE HABILIDADES EM PROGRAMAÇÃO

Felipe Borges Tomaz[1], Ana Cláudia Garcia Barbosa[2]

Resumo: Este Trabalho de Conclusão de Curso (TCC) investigou como o reforço gradual do aprendizado, especificamente por meio da repetição espaçada, pode aprimorar a memorização e a velocidade na resolução de desafios de programação. O objetivo central foi desenvolver um ambiente web para a validação de código, incorporando *feedback* automatizado e testes unitários para a avaliação das soluções submetidas. O ambiente também explora a aplicação dessa metodologia de revisão otimizada para exercícios de desenvolvimento, conforme evidências bibliográficas que demonstram sua eficácia na retenção de conhecimento. O sistema conta com métricas para identificar pontos de dificuldade na resolução das questões, permitindo que os professores ajustem os enunciados ou melhorem o *feedback* fornecido. Além disso, ao automatizar a validação dos códigos e o fornecimento de *feedback* automatizado, o ambiente contribui para a redução da sobrecarga do instrutor. Essa abordagem de aprendizado, combinada com o *feedback* da validação do código e o automatizado, pode ajudar modestamente a capacidade dos estudantes de resolver problemas de programação, otimizando o processo de ensino tanto em modalidades presenciais quanto a distância e consolidando habilidades essenciais na área.

Palavras-chave: repetição espaçada; desenvolvimento de software; aplicação web; automação de testes; feedback automatizado.

[1]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), felipe.b.t@hotmail.com

[2]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), abg@unesc.net

ABSTRACT: This Final Undergraduate Project investigated how the gradual reinforcement of learning, specifically through spaced repetition, can enhance memorization and the speed of solving programming challenges. The main objective was to develop a web-based environment for code validation, incorporating automated feedback and unit testing to evaluate submitted solutions. The environment also explores the application of this optimized review methodology to programming exercises, supported by bibliographic evidence that demonstrates its effectiveness in knowledge retention. The system includes metrics to identify areas of difficulty in solving problems, allowing instructors to adjust question prompts or improve the feedback provided. Furthermore, by automating code validation and feedback delivery, the platform contributes to reducing the instructor's workload. This learning approach, combined with automated code validation and feedback, can modestly improve students' ability to solve programming problems, optimizing the teaching process in both in-person and remote learning contexts and reinforcing essential skills in the field.

Keywords: spaced repetition; software development; web application; test automation; automated feedback.

1 INTRODUÇÃO

Discentes de disciplinas de programação, tanto no ensino presencial quanto na Educação a Distância (EaD), enfrentam diversos desafios de natureza multifatorial. Entre eles, destacam-se a ausência de retorno orientativo no momento oportuno, deficiências na formação básica (fator importante para a resolução de problemas), além de dificuldades na compreensão dos conteúdos abordados em aula (Oliveira; Mota; Oliveira, 2015; Masola; Allevato, 2019; Queiroga et al., 2018).

Outro ponto de destaque é a sobrecarga do instrutor EaD, como citado por Veloso e Mill (2018). Em muitos casos, essa sobrecarga está relacionada ao elevado número de alunos matriculados (Mill; Santiago; Viana, 2008), o que dificulta o atendimento individualizado e a resolução das dúvidas dos estudantes. Como consequência, o tempo de resposta às solicitações aumenta, impactando negativamente o processo de aprendizagem.

No ambiente *online* voltado ao desenvolvimento de habilidades em programação, é fundamental que os erros do código executado sejam devidamente apresentados, pois esse *feedback* é essencial para a resolução do exercício proposto. Há diversas formas de fornecer esse retorno ao

discente, no qual é importante que ele seja apresentado por meio de um *design* instrucional adequado e, preferencialmente, acompanhado de dicas que o auxiliem na conclusão da tarefa. Nesse contexto, a qualidade do *feedback* torna-se um fator determinante para a eficácia da aprendizagem (Li et al., 2020; Hattie; Timperley, 2007; Hahn et al., 2021, tradução nossa).

Após a orientação adequada, é possível consolidar o conhecimento adquirido por meio da resolução dos exercícios propostos. Nesse sentido, a literatura acadêmica apresenta diversas técnicas voltadas à memorização e à manutenção do conteúdo assimilado (Weinstein; Madan; Sumeracki, 2018; Schimanke et al., 2015, tradução nossa). Dentre essas estratégias, destaca-se a repetição espaçada, que consiste na revisão periódica do material aprendido em intervalos definidos, favorecendo a retenção a longo prazo (Ronzani, 2022).

Pesquisas sobre repetição espaçada apresentam diversas aplicações em distintas áreas do conhecimento, como, por exemplo, no campo da aprendizagem de línguas. O estudo de Martins e Duarte (2021) teve como foco a memorização de palavras em inglês por meio da aplicação Anki, que utiliza a repetição espaçada para revisar conteúdos em intervalos definidos, o próprio sistema determina a data da próxima revisão com base no desempenho do usuário.

Dentro do contexto de repetição espaçada em desenvolvimento, o trabalho realizado por Jacinto, Medeiros e Sousa (2024, tradução nossa), tem como foco os conceitos introdutórios de programação utilizando linguagens de marcação *Hypertext Markup Language* (HTML) e *Cascading Style Sheets* (CSS). Além disso, o trabalho inclui o uso de uma linguagem de script amplamente adotada no desenvolvimento web: o JavaScript. A modalidade de ensino adotada nesse estudo foi a EaD. Um aspecto relevante a ser destacado é a oferta de sessões de monitoria em horários fixos, visando apoiar o processo de aprendizagem. A partir da verificação das hipóteses propostas, concluiu-se que a combinação da repetição espaçada com o estudo intercalado contribui significativamente para o desempenho dos discentes, favorecendo a retenção do conteúdo em médio e longo prazo.

Destaca-se ainda o estudo de YeckehZaare, Resnick e Ericson (2019, tradução nossa), que analisou algumas técnicas de memorização: repetição espaçada, intercalada e recuperação ativa. Em um curso introdutório de programação em Python. Teve como resultado que o uso dessas técnicas aumentou moderadamente o desempenho dos discentes no exame final.

O objetivo do presente trabalho é desenvolver um ambiente para a validação, organização, resolução e assimilação de exercícios de programação. Para tanto, utilizou-se o conceito de teste unitário para ser aplicado na validação do exercício, *feedback* pré-definido por questão, para ajudar na resolução mais ágil do exercício exposto, além propor o uso de repetição espaçada para a assimilação das questões já concluídas.

Os objetivos específicos deste trabalho consistem em compreender o uso de repetição espaçada; desenvolver um mecanismo de geração de *feedback* automatizado; construir um ambiente web usando contêineres para organização de exercícios e *feedback*; aplicar o uso do teste unitário para validação de exercícios; propor o uso de uma metodologia de repetição espaçada para exercícios concluídos.

2 MATERIAIS E MÉTODOS

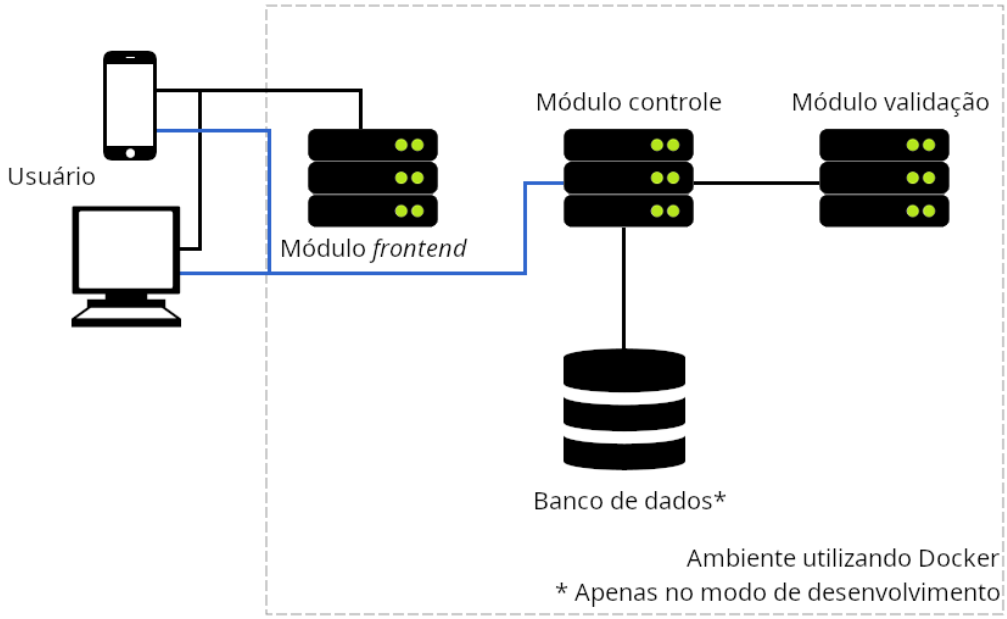
Essa é uma pesquisa exploratória de base tecnológica, em que foi desenvolvido um protótipo de Ambiente Virtual de Aprendizagem (AVA) com foco na resolução e validação de exercícios de programação, utilizando de *feedback* e repetição espaçada. O ambiente, representado pela Figura 1, é composto de 3 módulos[3] operam em paralelo por meio do uso de contêineres Docker: módulo *frontend*, módulo de controle e módulo de validação.

O módulo *Frontend* é direcionado ao controle de ações e funções que o usuário deve realizar no uso do ambiente, responsável por mostrar informações pertinentes ao nível de acesso, em que um usuário discente não pode acessar uma rota docente. O módulo controle é responsável pelo gerenciamento dos dados, atuando na persistência de informações relevantes para a solução de questões armazenadas no banco de dados, bem como realizar o controle de qual tipo de informação deve ser manipulada a depender do nível de acesso do usuário.

No módulo de validação, o código submetido é executado e, em seguida, testado para verificar sua conformidade com os critérios da questão. Os resultados dessa validação são então enviados ao módulo de controle, que inicia o processo de repetição espaçada. Todo esse fluxo de comunicação entre os módulos ocorre por meio do protocolo *Hypertext Transfer Protocol* (HTTP), com os dados estruturados no formato *JavaScript Object Notation* (JSON).

[3]Github com o projeto: https://github.com/alfuveam/adhp

Figura 1 - Fluxo do ambiente



Fonte: Elaborado pelo autor.

2.1 MÓDULO *FRONTEND*

Nesse módulo é realizada a interação com o usuário, responsável por comunicar de forma compreensível as informações recebidas do outros módulos, no qual foi desenvolvido utilizando o *Framework* Nuxt, versão 3.15.4 na linguagem de programação TypeScript, PrimeVue versão 4.0.0-rc.3 para componentes e estilos já definidos junto ao Tailwind CSS, versão 6.8.0 para completar com mais um conjunto de CSS pré-definidos. Para a elaboração do gráfico que representa à curva de esquecimento utilizou-se o chart.js, versão 4.4.8 e o Nuxt Monaco Editor, versão 1.3.1 para aplicar estilos de cores dependendo da linguagem configurada.

O controle das requisições foi realizado utilizando o Nginx, versão 1.21.6 e o gerenciamento dos pacotes é realizado por meio do Node Package Manager (NPM), na versão 10.8.2. A execução ocorre em um contêiner Docker, que utiliza o Node.js na versão 18.20.7. O desenvolvimento foi realizado no ambiente de edição de arquivos *Visual Studio Code*, versão 1.96.4, com a extensão *Codeium* (versão 1.42.3), em um sistema operacional *Manjaro Linux*, versão 24.2.1.

O uso do Framework Nuxt beneficia o desenvolvimento de aplicações *web* por conter o padrão algumas configurações já definidas, retirando entraves encontrados durante a configuração de um ambiente *web*. Uma delas, a possibilidade de verificar se o usuário tem permissão para

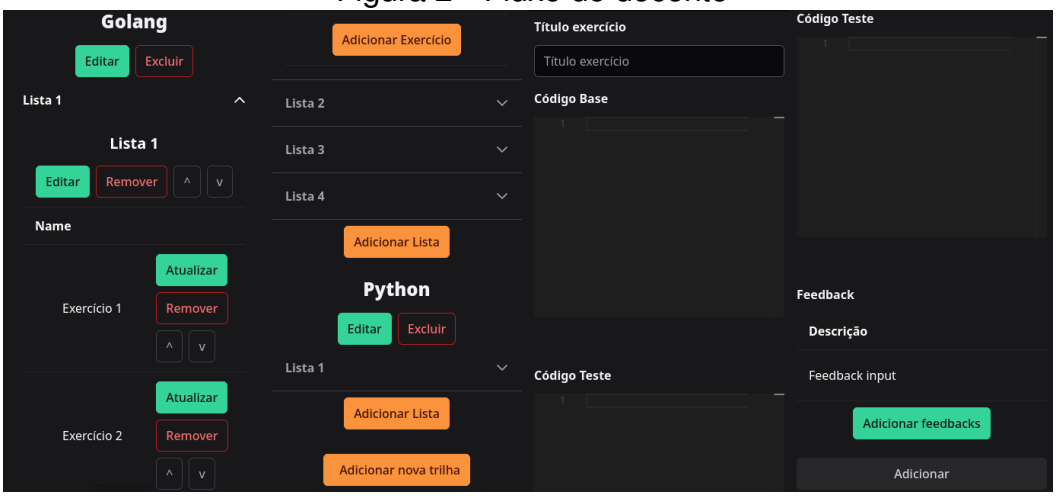
acessar uma página, extraindo do JSON Web Token (JWT) o nível de acesso público do ambiente, o que permite verificar se é um discente ou um docente.

A execução do Nuxt acontece em um contêiner Docker, utilizando como referência o ambiente disponibilizado pelo Oberlehner (2023), no qual acontece a divisão entre a execução do modo de produção e modo de desenvolvimento. Este último é útil para testar algumas funções de maneira mais rápida, com a possibilidade de utilizar o *Vue DevTools* quando necessário.

2.1.1 *Frontend* Docente

Na rota docente acontece as configurações básicas dos exercícios, no qual é de suma importância para tornar o ambiente acessível para o discente, em que é possível adicionar uma trilha, que irá conter as listas com os exercícios, para adicionar os *feedbacks* como mostra a Figura 2.

Figura 2 - Fluxo do docente



Fonte: Elaborado pelo autor.

O exercício somente vai entrar para a lista depois que acontecer a verificação, no qual o código é enviado com o teste para o módulo de validação, se o resultado da execução for o esperado, então ele é adicionado à lista de exercícios. Também é possível fazer a alteração da ordem da lista quando necessário, útil para a organização, essa opção também está disponível para os exercícios presente nas listas, com isso formando parte importante para a organização da ordem de dificuldade das questões adicionadas.

Ainda sobre tema discutido no início do parágrafo anterior, ao submeter o exercício existe o retorno da execução da questão para assim

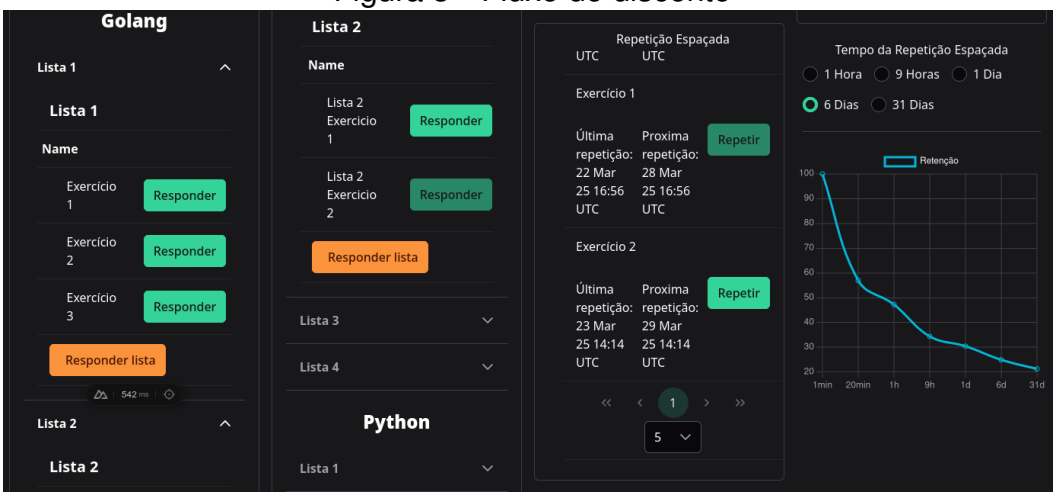
verificar o resultado do código e do teste enviado ao módulo de validação, isso mostra-se útil para a correção do código de referência, como também o código de teste que pode apresentar problemas em sua execução, se ambos funcionarem então serão armazenados no banco de dados para assim serem respondidas pelos discentes cadastrados no ambiente.

O uso do *feedback* é importante para a resolução de exercícios de forma mais rápida, assim tornando a obtenção de uma nova habilidade de programação mais eficiente pelos discentes (Gentrup et al., 2020, tradução nossa). Não há um limite de quantos *feedbacks* o usuário docente pode adicionar ao exercício, no qual é de sua responsabilidade escolher os mais adequados para a resolução mais ágil das questões propostas para a resolução.

2.1.2 *Frontend* Discente

Na rota discente acontece o uso dos dados adicionados pelo docente, no qual é realizado o carregamento das trilhas com as listas de exercícios. A escolha do tempo da repetição espaçada acontece no cadastro do usuário no qual é utilizado um componente do *Primevue* por adicionar um estilo no *radiobutton*. Por meio deste, é possível fazer uma amostra da curva de esquecimento, utilizando um gráfico para representar de forma visual a fórmula apresentada por Hermann Ebbinghaus (1913, p.77). A Figura 3 mostra as funcionalidades discutidas nesse parágrafo.

Figura 3 - Fluxo do discente



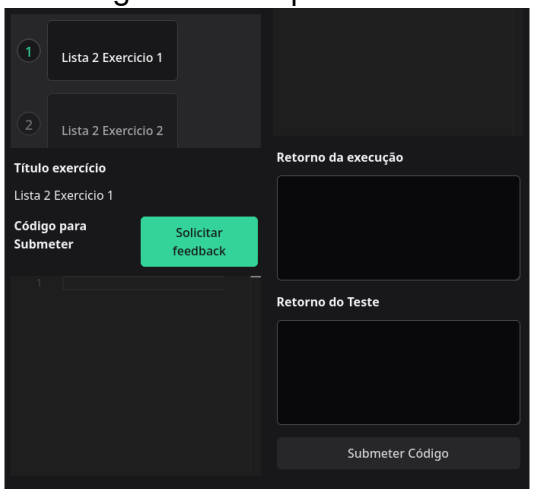
Fonte: Elaborado pelo autor.

Ao clicar no botão de responder, o discente é redirecionado para a página que contem os exercícios da lista e a questão selecionada.

Ao clicar no botão, basta usuário compreender que o exercício

está propondo e então preencher o campo que utiliza um componente Nuxt Monaco Editor para a inserção de textos como mostra a Figura 4.

Figura 4 - Responder lista



Fonte: Elaborado pelo autor.

O discente pode utilizar o *feedback* fornecido para a questão, que pode ser visualizado por meio do componente *Toast*. A mensagem será exibida por um determinado período em uma posição específica na tela, permitindo que o usuário a feche ou solicite uma nova mesmo enquanto a mensagem anterior ainda estiver ativa, esta última será carregada logo abaixo.

O uso da repetição ocorre após o exercício ser enviado ao módulo de validação e passar nos testes propostos, o que depende da linguagem da trilha escolhida: Go ou Python.

Para isso, são realizadas as ações necessárias para verificar o código selecionado pelo discente, retornando o resultado da execução. Caso o código seja válido, o exercício é incluído na lista de exercícios a serem repetidos.

2.2 MÓDULO DE CONTROLE

No desenvolvimento do módulo de controle, utilizaram-se tecnologias que visam o desenvolvimento mais ágil. O uso da linguagem Go, versão 1.23.4, contribui nesse sentido, já que conta em sua *standard library* com funções já implementas e de fácil utilização, para então ser executada usando um contêiner Docker.

Para a geração das instruções *Structured Query Language* (SQL) em Go é utilizado o SQLC, versão 1.28.0, por ser esse de grande importância para deixar o fluxo de desenvolvimento menos repetitivo. Após a etapa

mencionada, o código em Go contendo as instruções SQL conecta-se ao banco de dados PostgreSQL, versão 16.2. A escolha do banco deve-se à sua ampla adoção pela comunidade *open source*. Para o controle das *migrations*, foi utilizado o Goose.

O uso do banco de dados é indispensável para o desenvolvimento deste módulo. Nele são armazenadas todas as informações relacionadas aos exercícios e usuários, incluindo dados que indicam se o exercício já foi submetido e aprovado pelo módulo de validação. Dessa forma, o próximo exercício da lista do discente pode ser liberado, enquanto o exercício respondido permanece armazenado para eventual verificação.

O código para execução e o código teste para fazer a verificação de validação também fica armazenado no banco de dados, no qual é responsabilidade do usuário docente fazer a manutenção, alteração ou exclusão. Esse pode ser usado como contraprova se a posterior acontecer alguma contestação, por parte do usuário discente do ambiente. Ainda sobre responsabilidade do docente está o controle do *feedback* para o exercício, ao qual este não tem um limite de inserções no banco de dados.

A solicitação do *feedback* ocorre por meio do método GET do protocolo HTTP, no qual a requisição é enviada a este módulo contendo o identificador (ID) do exercício. Para ter acesso à possibilidade de fazer essa requisição, é necessário enviar junto ao seu cabeçalho o Token JWT no campo *Authorization*, é escolhido um *feedback* randômico a cada solicitação, usando uma opção no banco de dados para fazer essa escolha aleatória.

A informação referente ao tempo da próxima repetição também é armazenada no banco de dados como um valor inteiro, podendo ser recuperada por meio de uma instrução SQL. Esse valor é utilizado na soma de datas, resultando na data da próxima repetição do exercício. Além disso, ele é inserido em uma lista com as informações necessárias para a recuperação e alteração dos dados. Na tabela em questão, não é permitido que haja mais de um exercício com o mesmo ID para um mesmo usuário.

2.2.1 Gerenciamento de acesso ao módulo de controle

O gerenciamento do acesso às rotas privadas do módulo de controle é realizado por meio do Token JWT. Sua geração ocorre no momento do *login* do usuário, quando o *token* é criado com uma assinatura. Nesse *token*, é incluído o nível de acesso do usuário, obtido a partir do banco de dados, permitindo assim identificar se o usuário é do tipo docente ou

discente para controlar o acesso às funcionalidades.

O controle de acesso é fundamental para o gerenciamento de aplicações, por ser essencial para o funcionamento seguro de ambientes de uso compartilhado. Ele impede que usuários sem o nível de permissão adequado alterem dados importantes e permite controlar o acesso às páginas sem a necessidade de realizar uma requisição ao servidor. Assim, é possível verificar se o usuário possui autorização para acessar a página solicitada, efetuando o redirecionamento para uma *Uniform Resource Locator* (URL) compatível com seu nível de acesso.

2.2.2 Redirecionamento do fluxo

O redirecionamento desempenha um papel importante neste ambiente, pois permite criar ou encaminhar uma requisição HTTP, possibilitando o controle sobre qual módulo ou serviço externo deve ser acionado para obtenção dos dados necessários. Nesse contexto ele é útil devido a desacoplar o módulo de validação do módulo de controle, com isso resultando na possibilidade da criação de múltiplos módulos de validação quando necessário devido à demanda.

2.3 MÓDULO DE VALIDAÇÃO

No desenvolvimento do módulo de validação, utilizou-se a linguagem de programação Go, versão 1.23.4, executada em um contêiner Docker. O uso do Docker permite que componentes externos sejam instaladas de forma padronizada, como o Python, versão 3.12.9, a ferramenta Pytest, versão 8.3.4, além dos arquivos necessários para a execução dos códigos em Go.

A execução do código submetido para validação e do código de teste é realizada por meio de uma função da linguagem Go. Essa execução retorna os resultados de ambos os códigos, que são então organizados em formato JSON e enviados via HTTP para o módulo de controle. Como resultado, tornou-se viável a construção do ambiente, uma vez que toda a execução ocorre de maneira padronizada no módulo de validação por meio do uso do contêiner Docker.

2.4 CRITÉRIOS PARA ESCOLHA DAS LINGUAGENS DE PROGRAMAÇÃO

Foram escolhidas duas linguagens de programação, considerando a facilidade de instalação em um contêiner de forma prática. Essas linguagens também contribuem para a execução simplificada tanto do

código submetido quanto dos testes correspondentes.

2.4.1 Go

Go é uma linguagem de programação que começou a ser desenvolvida no ano de 2007 pelos programadores Robert Griesemer, Ken Thompson e Rob Pike no Google, projetada para melhorar o desempenho de sistemas distribuídos (Google; Pike, 2020, tradução nossa). Além dessa característica, a linguagem também é reconhecida por tornar à escrita de código mais eficiente, buscando simplificar seu uso para novos usuários. O código desenvolvido é compilado em um formato eficiente para a sua execução nos sistemas operacionais suportados (Pereira, 2021).

2.4.2 Python

Python teve seu desenvolvimento iniciado pelo programador Guido van Rossum, com a sua primeira versão pública no ano de 1991 (Sanner et al., 1999, tradução nossa). Além disso, ele tem suporte a programação orientada a objeto, possibilitando a criação das mais diversas soluções, além de utilizar uma sintaxe considerada de fácil interpretação para os desenvolvedores de software (Kumar; Panda, 2019, tradução nossa).

2.5 IMPLANTAÇÃO

O uso de ferramentas externas contribui para tornar a implantação de ambientes de desenvolvimento menos onerosa. Entre elas, destaca-se o *GitHub Actions*, uma solução de *Continuous Integration/Continuous Delivery* (CI/CD), que permite a realização de diversas verificações por meio de um arquivo de definição de ações. Entre os resultados obtidos, estão a criação de arquivos *.env* e a geração de contêineres de forma automatizada.

A implantação do ambiente foi realizada em uma *Virtual Private Server* (VPS) localizada na região Sudeste do Brasil, com as seguintes configurações: 1 vCPU, 1 GB de RAM, 10 GB de disco (NVMe) e sistema operacional Ubuntu 24.04 LTS. O acesso seguro à VPS foi feito por meio do *Secure Shell Protocol* (SSH), utilizando um arquivo com a *private key*. Após o acesso, foram realizados o *download* dos contêineres e sua execução.

2.6 MÉTRICAS

Para a obtenção de dados relevantes, foram definidas métricas específicas, permitindo a análise dos valores gerados durante a utilização do ambiente pelos usuários. Essas métricas são fundamentais para identi-

ficar eventuais dificuldades enfrentadas pelos discentes, possibilitando melhorias contínuas na plataforma. Como resultado, espera-se que os alunos consigam concluir as questões da trilha escolhida de forma mais eficiente.

Adicionalmente, foi incluída uma métrica voltada para a análise do tempo de execução do código submetido ao módulo de validação — tanto o código principal quanto o de teste. Essa informação é valiosa para avaliar a qualidade do código enviado, permitindo calcular a média de tempo de validação e identificar quais questões demandam maior tempo de processamento.

A obtenção desses dados também é importante para configurar melhor o servidor que será utilizado, o que pode melhorar o tempo de resposta dos códigos enviados para validação. Isso é relevante especialmente em cursos na modalidade EaD, no qual normalmente não há um limite de usuários acessando o ambiente ao mesmo tempo. Com muitos usuários utilizando o sistema simultaneamente, é possível que ocorram sobrecargas, o que pode acabar causando falhas na hora de validar as questões enviadas pelos estudantes.

3 DISCUSSÃO E RESULTADOS

O uso de *frameworks* com componentes pré-configurados ajudou na construção mais ágil da interface de interação com o usuário, resultando em componentes que se podem adaptar nos mais diferentes formatos de tela. Mas, é necessário atentar-se a necessidade de configurações adicionais para a sua utilização, resultando em esforços extras, a fim de operar de maneira correta.

Com o ambiente online em uma VPS, foram realizados alguns testes para verificar o seu comportamento. Simulando o uso do Docente como adicionar, alterar e remover questões e *feedback*. E do Discente em responder às questões e fazer uso *feedback*. Para essas funções o ambiente demostrou-se funcional.

O uso Nginx ajudou a configurar o redirecionamento para o contêiner correto, mas isso adicionou um camada extra de complexidade para a configuração do ambiente, sendo então necessário a alteração para o *Internet Protocol* (IP) correto.

Ao submeter o exercício, fazer uso do *feedback* e usar a repetição espaçada são obtidas métricas de uso. Esses dados são armazenados em um banco de dados, o que possibilita a realização de análises sobre o uso do ambiente pelos usuários. A verificação de possíveis entraves na re-

solução das questões, assim como a avaliação da eficácia da metodologia na memorização dos exercícios, é fundamental para o aprimoramento do processo de aprendizagem.

Por se tratar de um ambiente *online* de desenvolvimento, existem algumas limitações para utilizar as funcionalidades de cada linguagem, como a não possibilidade de utilizar *breakpoints* para a verificação do estado da execução do código num ponto específico. Outro ponto é a limitação de interação do usuário com a execução do *script*. O que torna necessária a inserção de valores ao iniciar o uso do código de teste para assim simular essa interação como um usuário.

O artigo publicado por Martins e Duarte (2021) utilizou um Software de Aprendizagem por Revisão Espaçada (SARE) chamado Anki, que tem suporte para dispositivos móveis e computadores, para com isso investigar os seus efeitos no aprendizado da língua inglesa. Para a obtenção de dados foi utilizado questionário eletrônico, para assim fazer uma análise qualitativa. Os resultados da pesquisa indicaram o uso do SARE aprimorou a memorização de conteúdos em inglês, como também demostrou para os discentes sobre aplicabilidade do uso da SARE para outras disciplinas. Comparando com esse trabalho, o uso do Anki é mais amplo, diferente desse ambiente que o foco é o uso da repetição espaçada na programação.

O trabalho de dissertação elaborado por Jacinto, Medeiros e Sousa (2024, tradução nossa) teve como foco o uso de linguagens de marcação como o HTML e CSS, e de *script* como o Javascript para a verificação das hipóteses da pesquisa, no qual foi utilizado a repetição espaçada para a análise da metodologia no ensino dessas linguagens. Também foi utilizado o Google Forms para os discentes responderem às questões, que eram de múltipla escolha, verdadeiro ou falso, com alguns exercícios sendo necessária a revisão manual, as questões discursivas que necessitam à escrita de código era necessário fazer a correção manual. Além de ter um monitor para o acompanhamento semanal, diferente do ambiente que foi desenvolvido nesse trabalho, em que é utilizado o *feedback* automatizado e a validação do código com o uso do teste unitário para as questões, tudo isso acontecendo de forma autônoma.

A pesquisa desenvolvida por YeckehZaare, Resnick e Ericson (2019, tradução nossa) teve como foco a aplicação das técnicas de repetição espaçada, recuperação ativa e intercalada em um curso de introdução à programação de Python. Foi utilizado o Runestone Interactive junto a

um e-book interativo, essa ferramenta permite o fornecimento de *feedback* e o uso de teste unitário para a validação do código, sendo então disponibilizada para 193 discentes. Os resultados indicam que o uso dessas técnicas melhorou moderadamente o desempenho dos discentes na prova final. Comparando a este ambiente, onde o foco é na repetição espaçada, o trabalho de YeckehZaare, Resnick e Ericson (2019) teve um foco mais amplo das técnicas de memorização.

4 CONCLUSÃO

Este trabalho teve como foco o desenvolvimento de um protótipo para o aprimoramento na resolução de exercícios de programação, com enfoque na memorização usando uma metodologia para avaliar o uso da repetição espaçada, tendo a possibilidade de soluções mais ágeis com o uso de *feedback* para a conclusão mais rápida de questões.

Com o levantamento bibliográfico, foi possível compreender o uso da repetição espaçada, não somente na área da programação, mas nas diversas áreas da educação. Com isso foi possível aplicar o seu uso em um ambiente online de desenvolvimento. Contudo, é importante considerar algumas limitações: a repetição espaçada, em sua essência, foi desenvolvida utilizando sílabas e consoantes aleatórias, diferentemente da programação, que segue uma lógica na sequência das palavras. Portanto, é necessário um aprofundamento maior para aplicar essa técnica adequadamente nesse contexto.

Com o protótipo desenvolvido, foi possível aplicar uma metodologia de repetição espaçada com o uso de *feedback* automatizado. O ambiente demonstrou-se capaz de realizar as suas funções básicas. Deve-se tomar cuidado com o uso do *feedback* para não acabar respondendo à questão por completo.

As etapas escolhidas para a organização e execução dos códigos no módulo de validação, demostrou-se funcional para a execução das linguagens de programação escolhidas, Go e Python. Qualquer outra linguagem de programação que possua a mesma estrutura de execução do código e dos testes pode garantir a reprodutibilidade, sem a necessidade de desenvolver etapas adicionais.

O uso de ferramentas que ajudem na implantação demostrou-se ter sido uma boa escolha, devido a remover uma parte entediante do processo de implantação ou atualização, o qual é a repetição de comandos sempre iguais no terminal. Com isso, o uso do *GitHub Actions*, uma ferra-

menta de integração e entrega contínuas (CI/CD), foi fundamental para o *release* do ambiente, pois automatiza os procedimentos necessários para gerar, em cada módulo, um contêiner com o código compilado em modo otimizado. Com o seu uso a execução dos módulos torna-se mais rápida para ser disponibilizada ao acesso dos usuários.

Com base no conhecimento adquirido no desenvolvimento desse ambiente de programação, é recomendado para trabalhos futuros o uso de Inteligência Artificial (IA) para a geração de *feedback*, possibilitando respostas mais completas para as dúvidas dos discentes. Todavia, é necessário ter cautela quanto ao tipo de *prompt* enviado à IA. Ao formular a pergunta, deve-se deixar claro que a resposta tem como objetivo apenas auxiliar na resolução da questão, evitando que a IA forneça diretamente a solução completa.

REFERÊNCIAS

EBBINGHAUS, H. **Memory; a contribution to experimental psychology**. [S.l.]: New York city, Teachers college, Columbia university, 1913.

GENTRUP, S. et al. **Self-fulfilling prophecies in the classroom: Teacher expectations, teacher feedback and student achievement**. Learning and Instruction, 2020, v. 66, p. 101296. ISSN 09594752.

GOOGLE; PIKE, R. **Using Go at Google, The Go Programming Language**. 2020. Acesso em: 12 Abril 2025. Disponível em: <https://go.dev/solutions/google/>.

HAHN, M. G. et al. **A systematic review of the effects of automatic scoring and automatic feedback in educational settings**. IEEE Access, 2021, v. 9, p. 108190–108198. ISSN 2169-3536.

HATTIE, J.; TIMPERLEY, H. **The power of feedback**. Review of Educational Research, 2007, SAGE Publications Inc., v. 77, p. 81–112. ISSN 00346543.

JACINTO, A. S.; MEDEIROS, F. P. A. de; SOUSA, M. P. de. **An approach to spaced repetition methodology for teaching markup and scripting programming languages**. 2024, IEEE, p. 1–9.

KUMAR, A.; PANDA, S. **A survey: How python pitches in it-world**. 2019, IEEE, p. 248–251.

LI, J. et al. **Using feedback to promote student participation in online learning programs: evidence from a quasi-experimental study**. Educational Technology Research and Development, 2020, Springer, v. 68, p. 485–510. ISSN 15566501.

MARTINS, E. D.; DUARTE, A. F. C. **Como aprimorar a memorização em língua inglesa? os efeitos de um software de aprendizagem por revisão espaçada**. Revista Agulhas Negras, 2021, v. 4, p. 136–153. ISSN 2595-1084.

MASOLA, W.; ALLEVATO, N. **Dificuldades de aprendizagem matemática: algumas reflexões**. Educação Matemática Debate, 2019, v. 3, p. 52–67. ISSN 2526-6136.

MILL, D. R.; SANTIAGO, C. F.; VIANA, I. de S. **Trabalho docente na educação a distância: condições de trabalho e implicações trabalhistas**. Revista extra-classe, 2008, v. 1.

OBERLEHNER, M. **Running Nuxt 3 in a Docker Container - Markus Oberlehner — markus.oberlehner.net**. 2023. Acesso em: 02 Setembro 2024. Disponível em: <https://markus.oberlehner.net/blog/running-nuxt-3-in-a-docker-container/>.

OLIVEIRA, A. S.; MOTA, L. D. C.; OLIVEIRA, A. A. D. **Uso de ambientes virtuais de aprendizagem como suporte ao ensino de programação: uma revisão sistemática**. Interfaces Científicas - Exatas e Tecnológicas, 2015, Universidade Tiradentes, v. 1, p. 9. ISSN 2359-4934.

PEREIRA, F. Z. **Avaliação de concorrência e sincronização de diferentes linguagens de programação populares**. Monografia (Graduação) — Universidade Federal do Rio Grande do Sul, 2021. Disponível em: <http://hdl.handle.net/10183/223221>.

QUEIROGA, E. M. et al. **Modelo de predição da evasão de estudantes em cursos técnicos a distância a partir da contagem de interações**. Revista Thema, 2018, v. 15, p. 425–438. ISSN 21772894.

RONZANI, F. G. **COMO NÃO ESQUECER? O uso da repetição espaçada na manutenção do conhecimento médico**. Monografia (Graduação) — Universidade Federal de Santa Catarina, Centro de Ciências da Saúde, Medicina, 2022.

SANNER, M. F. et al. **Python: a programming language for software integration and development**. J Mol Graph Model, 1999, v. 17, p. 57–61.

SCHIMANKE, F. et al. **Implications of short term memory research for the design of spaced repetition based mobile learning games**. 2015, IEEE, p. 571–576.

VELOSO, B. G.; MILL, D. **Precarização do trabalho docente na educação a distância: elementos para pensar a valorização da docência virtual**. Educação em Foco, 2018, p. 111–132. ISSN 2447-5246.

WEINSTEIN, Y.; MADAN, C. R.; SUMERACKI, M. A. **Teaching the science of learning**. Cognitive Research: Principles and Implications, 2018, v. 3, p. 2. ISSN 2365-7464.

YECKEHZAARE, I.; RESNICK, P.; ERICSON, B. **A spaced, interleaved retrieval practice tool that is motivating and effective**. 2019, ACM, p. 71–79.