Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Desenvolvimento de um ambiente virtual para aprendizagem de programação

Paulo Henrique Dias de Oliveira

JUIZ DE FORA
JULHO, 2018

Desenvolvimento de um ambiente virtual para aprendizagem de programação

Paulo Henrique Dias de Oliveira

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da computação

Bacharelado em Ciência da Computação

Orientador: Igor de Oliveira Knop

JUIZ DE FORA

JULHO, 2018

Desenvolvimento de um ambiente virtual para aprendizagem de programação

Paulo Henrique Dias de Oliveira

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Igor de Oliveira Knop
Doutor em Modelagem Computacional (UFJF)

Vânia de Oliveira Neves
Doutora em Ciências da Computação e Matemática Computacional (USP)

Eduardo Barrere
Doutor em Engenharia de Sistemas e Computação (UFRJ)

JUIZ DE FORA
09 DE JULHO, 2018

Resumo

Iniciar os estudos em programação tem sido uma barreira para um grande número de pessoas, o que reflete em um alto índice de reprovação nas disciplinas iniciais de programação. Além de exigir um pensamento lógico e uma abstração dos exercícios propostos, estas disciplinas possuem uma dinâmica bem diferente das tradicionais, vistas no ensino fundamental e médio: é exigido constantemente que o aluno mude a visão entre a análise e síntese, criando e evoluindo soluções a cada questão. Entre as propostas para estes problemas, está o uso de ambientes online de treinamento e competição de programação. Esses ambientes permitem que os alunos submetam suas soluções para um conjunto de questões e tenham, de imediato, a correção automática ou uma avaliação com os erros encontrados. Essa abordagem permite guiar o estudo ou treinamento e permitir que o aluno adquira experiência dentro de seu próprio tempo de estudo. Frequentemente, o engajamento é reforçado utilizando elementos de jogos na forma de competição ou ranqueamento das respostas. Ao propor objetivos, cumprimento de metas, superação de desafios, premiação por realizações, os sistemas aumentam a motivação dos usuários. Este trabalho faz uma revisão bibliográfica dos conceitos, metodologias e ferramentas existentes para avaliadores online de código e técnicas empregadas para engajamento dos usuários. É realizada uma análise sobre as características em comum e os diferenciais de cada abordagem. Adicionalmente, é proposta uma arquitetura e implementação de um protótipo de ambiente virtual de aprendizagem que analisa e estima um grau de habilidade para o aluno e realiza uma recomendação automática de questões para serem realizadas, fazendo um pareamento entre aluno e questão. Este protótipo é colocado em funcionamento e dados são colhidos e analisados para a evidenciar o funcionamento do método.

Palavras-chave: Ambiente virtual de aprendizagem, avaliação de *software*, pareamento.

Abstract

Starting studies in programming has been a barrier for a large amount of people, which reflects in a high rate of failure in the initial programming disciplines. In addition to requiring logical thinking and an abstraction of the proposed questions, these disciplines have a very different dynamic from the traditional ones seen in elementary and secondary education: the student is constantly required to change the vision between analysis and synthesis, creating and evolving solutions for each questions. Among the proposals for these problems are the usage of online training and competition environments for programming. These environments allow students to submit their solutions to a set of questions and have immediate, automatic correction or an evaluation of the encountered errors. This approach guides the study or training and enables the student to gain experience within their own study time. Often, engagement is reinforced using game elements in the form of competition or ranking responses. By proposing and meeting goals, overcoming challenges, rewarding achievements, these systems increases the user's motivation. This work reviews the existing concepts, methodologies and tools for online code evaluators and techniques used for user's engagement. An analysis is performed on the common characteristics and differentials of each approach. Additionally, is proposed an architecture and implementation of a virtual learning environment prototype that analyzes and estimates a degree of ability for the student and makes an automatic recommendation of questions to be performed, making a pairing between student and question. This prototype is put into operation and data are collected and analyzed to show how the method works.

Keywords: Virtual Learning Environment, Software Evaluation, Pairing.

Agradecimentos

A todos os meus parentes e amigos, pelo encorajamento e apoio.

Ao professor Igor de Oliveira Knop pela orientação, amizade e principalmente, pela paciência, sem a qual este trabalho não se realizaria.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários e alunos do curso, que durante esses anos, contribuíram de algum modo para o meu enriquecimento pessoal e profissional.

"Não há tecnologia que modernize uma mentalidade, mas uma mentalidade moderna não recusa uma nova tecnologia quando ela é necessária.".

Mário Sergio Cortella

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Listagens

Lista de Abreviações

AVA	Ambiente Virtual de Aprendizagem
BOCA	BOCA *Online Contest Administrator*
CSS	*Cascading Style Sheets*
DF	Data final
DI	Data inicial
E_a	Valor esperado
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
JSON	JavaScript *Object Notation*
LEC	Lista de códigos dos erros cometidos
MVC	*Model-View-controller*
NoSQL	*Not Only Structured Query Language*
P_f	Pontuação final
P_m	Pontuação de medidas
QDT	Quantidade de tentativas
QEC	Quantidade de erros cometidos
QRC	Quantidade de respostas corretas
QPI	Quantidade de soluções interrompidas
SF	Solução final da questão
SI	Soluções intermediárias das tentativas
TPS	Tempo para solução
URL	*Uniform Resource Locator*

1 Introdução

Neste Capítulo, é apresentada uma descrição do problema e sua justificativa, quais são os objetivos do trabalho e qual a metodologia utilizada para seu desenvolvimento.

1.1 Apresentação e contextualização do problema

Iniciar os estudos em programação tem sido uma barreira, mesmo para os alunos dos cursos na área de ciência da computação. Tais cursos possuem, logo no primeiro semestre, as disciplinas baseadas na análise e síntese de algoritmos, que possuem uma dinâmica bem diferente das tradicionais vistas no ensino fundamental e médio. Nessas disciplinas é exigido uma visão analítica e abstrata, conceitos matemáticos e raciocínio lógico, elementos aos quais os alunos podem não ter sido apresentados antes de ingressar na faculdade ou em um curso técnico.

Um outro problema, comumente observado, é o distanciamento entre alunos e professores, visto que em geral, um professor deve dividir sua atenção com vários alunos, já que essas disciplinas podem ser ofertadas para uma série de outros cursos da área de exatas. Dessa forma, fica difícil para o professor acompanhar pessoalmente o desempenho e rendimento de todos os alunos da turma até o momento da avaliação.

Além disso, muitas vezes as turmas são desniveladas, ou seja, existem alunos que já são proficientes na disciplina, provenientes de cursos técnicos no ensino médio ou cursaram previamente a disciplina em anos anteriores, enquanto outros nunca tiveram contato ou possuem uma grande dificuldade em acompanhar o andamento do curso. Com isso, é necessário por parte do professor definir um ponto comum para dar sequência e ritmo no andamento no curso com base na observação da turma. Entretanto, ao definir um ponto em comum para uma turma, da mesma forma que alguns alunos podem se sentir desmotivados por não ser fornecido desafio suficiente e outros perdem o interesse por não conseguir acompanhar o ritmo da turma.

As formas de avaliação também apresentam certas dificuldades. Durante uma

prova ou em exercícios propostos, alunos recebem questões que são arbitrariamente rotuladas com um certo nível de dificuldade (quando o são). Entretanto, uma questão que seja fácil para um aluno, não necessariamente, é fácil para outro. Propor uma questão muito difícil para um aluno pode desmotivá-lo, por nem sequer conseguir começar a resolvê-la. O contrário, propor apenas questões fáceis, pode fazer com que o aluno se sinta entediado ou pense que não há mais necessidade de estudar.

Geralmente observa-se um alto índice de reprovação nessas disciplinas, além de um grande número de evasão logo nos primeiros meses de curso. Apesar de não termos como relacionar diretamente os diversos fatores envolvidos, esses problemas citados podem vir a ser uma causa importante para que alunos se sintam desmotivados ou desinteressados à continuar com sua formação, podendo chegar até ao abandono de curso.

Portanto, este trabalho busca responder a questão mais geral "Como aumentar o rendimento dos alunos nas disciplinas iniciais de programação?". E leva à pesquisa de quais abordagens têm sido utilizadas, sejam em instituições de ensino ou ambientes privados de treinamento que levam os alunos a, voluntariamente, buscarem o conhecimento e melhorarem seu progresso.

Diante dos problemas apresentados, algumas das principais abordagens propostas para contribuir para a melhoria no aprendizado foi o uso de ambientes *online* de treinamento em programação. Esses ambientes permitem que os alunos pratiquem as soluções de problemas, recebendo de imediato, e forma automática, uma resposta ou avaliação de sua solução.

Para que os alunos se sintam motivados, pode ser oferecido um reforço no engajamento utilizando elementos de jogos (gamificação) na forma de competição ou ranqueamento das respostas. Ao propor objetivos, cumprimento de metas, superação de desafios e premiação por realizações, esses sistemas podem aumentar em muito a motivação e engajamento dos estudantes.

1.2 Objetivos do trabalho

É de maior interesse deste trabalho, os ambientes virtuais de aprendizagem (AVA) que fazem uso de ferramentas como a gamificação e que permitem dar um *feedbak* interativo

aos estudantes. Espera-se que essa abordagem vá aumentar o interesse dos alunos pela disciplina e aumentar a interatividade entre alunos e professores. Essa abordagem também permitirá aos professores terem uma maior facilidade de acompanhar o aprendizado dos alunos e dar indícios de quais conteúdos merecem reforço de forma mais precisa.

Assim, os objetivos gerais e específicos deste trabalho podem ser descritos como a revisão e comparação dos AVA's com conceitos de gamificação no ensino de algoritmos e linguagens de programação; formalização de uma arquitetura para um ambiente especializado em programação; implementação aberta de um protótipo desse ambiente, que combine os elementos mais importantes levantados na pesquisa; realizar uma proposta de um fluxo de treinamento, ou seja, um pareamento entre o nível de dificuldade das questões e o nível de habilidade do usuário; definir métricas e coletar dados de respostas ao longo do uso desse ambiente.

Dessa forma, ao propor um fluxo de treinamento em que dado o nível de habilidade de um usuário, questões que tenham um nível de dificuldade que se adéquam a essa habilidade serão sugeridas para ele, ou seja, os usuários terão acesso a questões que correspondem aos seus conhecimentos. Com isso, a progressão que um usuário realiza na plataforma se dá de acordo com sua capacidade individual, sem que ele tenha que tentar resolver questões que, no momento, não são adequadas.

Com a coleta de dados, estudos subsequentes poderão ser realizados para evidenciar se o uso de um ambiente especializado irá ou não contribuir para o acompanhamento do rendimento dos alunos. Através destes dados, o uso do ambiente poderá ser colocado em prática e sobre uma massa de dados maior, guiar possíveis alterações na condução das disciplinas.

1.3 Metodologia utilizada no trabalho

O desenvolvimento deste trabalho pode ser dividido cinco etapas. Inicialmente, foi realizada uma revisão bibliográfica dos conceitos, metodologias e ferramentas existentes para avaliadores *online* de código e técnicas empregadas para engajamento dos usuários, que serão abordadas com maior profundidade no Capítulo 2.

Em um segundo momento, foi realizada uma análise sobre as características em

comum e os diferenciais de cada abordagem, levando em consideração quais características estão mais presentes e quais são as mais importantes para obtenção de bons resultados. Esta análise pode ser vista em detalhes no Capítulo 3.

Após definir quais são os conceitos e as características mais relevantes, foi proposta uma arquitetura conceitual de projeto, descrita no Capítulo 4, que descreve tecnicamente como realizar a implementação de um ambiente que combine esses conceitos, como é a interação dos alunos com as atividades propostas, como os alunos são avaliados, quais as métricas de avaliação e uma definição do fluxo de treinamento.

Com base nessa arquitetura, foi criado um protótipo de ambiente, cuja implementação é apresentada no Capítulo 5 e por fim, é feita ama apresentação dos dados coletados.

Com a implementação de um ambiente para ensino de programação que permita criar, aplicar exercícios e acompanhar o rendimento dos alunos, espera-se realizar estudos subsequentes na busca por classificação de deficiências dos alunos no conteúdo e aplicar o reforço nos pontos necessários.

1.4 Justificativa das propostas de solução

Com seus estudos sendo guiados por um ambiente de treinamento, os alunos podem se sentir mais confortáveis e podem estudar e praticar com os exercícios em qualquer lugar e a qualquer hora, evitando que tenham contato com o conteúdo apenas no limitado horário para a disciplina em sala de aula ou laboratório. Com isso, os alunos poderão progredir no conteúdo da disciplina em seu próprio ritmo, com uma resposta imediata, independentemente do andamento das aulas.

Um ambiente destes pode aumentar o envolvimento dos alunos além da sala de aula ou laboratório, podendo fornecer um retorno instrutivo para os alunos, informando se estão no caminho certo ou especificando exatamente onde estão errando, o que serve de auxílio aos professores, que não precisam sanar cada dúvida individualmente.

Por fim, faz-se necessário frisar que a exploração de novos métodos complementares de ensino é de grande importância para as instituições de ensino, já que mesmo com a computação presente no cotidiano de todas as profissões, ainda registra-se um grande

número de reprovações nas disciplinas programação e algoritmos ofertadas a outros cursos.

1.5 Organização deste trabalho

Este trabalho está organizado em 6 capítulos. O Capítulo 2 faz um revisão conceitual dos principais elementos discutidos ou utilizados ao longo do texto. O Capítulo 3 faz uma revisão e análise dos ambientes de treinamento de programação mais populares atualmente. O Capítulo 4 descreve uma arquitetura conceitual de projeto proposta para capturar os dados necessários e viabilizar a construção de ambientes em cima da proposta aqui apresentada. O Capítulo 5 relata o desenvolvimento e implantação da arquitetura na forma de um protótipo utilizado para evidenciar o método proposto. Por fim, o Capítulo 6 encerra este trabalho com considerações finais, limitações e sugestões para trabalhos futuros.

2 Fundamentação teórica

Neste Capítulo serão apresentados os principais conceitos utilizados direta ou indiretamente neste trabalho. Será feita uma revisão dos fundamentos do ensino de programação de computadores bem como os conceitos de ensino à distância. Serão listadas as abordagens mais populares para ensino e treinamento e suas características principais avaliadas.

2.1 Algoritmos e linguagens de programação

O princípio básico da computação é o desenvolvimento de algoritmos. Um algoritmo é um conjunto finito de instruções para alcançar determinado objetivo. Desenvolver um algoritmo é estabelecer uma norma de ações simples, sem ambiguidade e ordenadas, que em um tempo finito produz uma solução bem definida para um problema. Seguindo essa norma, sempre que executado, sobre as mesmas condições, um algoritmo produz o mesmo resultado (FORBELLONE; EBERSPÄCHER, 1993).

Dado um conjunto de instruções e um conjunto de valores (chamado entrada), um algoritmo deve descrever os passos que devem ser executados com tais valores a fim de gerar uma saída. Um algoritmo é dito correto quando para qualquer entrada ele sempre para com uma saída exata (CORMEN et al., 2002).

Algoritmos são capazes de realizar tarefas como ler e escrever dados, avaliar expressões matemáticas, avaliar resultados e tomar decisões de acordo com o resultado e repetir um conjunto de ações dependendo de uma condição (FERRARI; CECHINEL, 2008).

Quanto à sua representação, um algoritmo pode ser expresso em linguagem natural, fluxograma, linguagem de máquina, linguagem de programação e pseudocódigo (FERRARI; CECHINEL, 2008).

Para que se possa escrever um algoritmo para um computador é necessário utilizar-se de um conjunto de notações formais para descrever as ações ou operações que serão realizadas pela máquina (GUDWIN, 1997). Esse conjunto de notações é composto por

regras sintáticas, que dizem respeito à forma de escrita e regras semânticas que se referem ao significado. Seguindo essas regras, o computador é capaz de executar o algoritmo e gerar uma saída (LOUDEN et al., 2011).

Diante disso, o principal objetivo para os alunos que iniciam seus estudos em algoritmos é que não só aprendam os conceitos básicos como comandos e convenções. Mas sim que consigam desenvolver com uma opinião crítica se seu algoritmo criado está correto dentro de um conjunto de entradas. Analisando de forma iterativa, independente da sua forma de representação, quais erros ou defeitos o compilador ou interpretador retornou e atuando para corrigi-los.

2.2 Teste de software

Delamaro, Jino e Maldonado (2017) definem que testar um *software* é avaliar se um código ou modelo, seja de um sistema completo ou parte dele, se comporta da maneira esperada e identifica quais os motivos caso sua execução não atinja os resultados esperados.

De acordo com Rocha et al. (2001), existem vários níveis de teste de *software*, como os testes unitários, de integração, de sistema, de aceitação e de regressão. Os testes unitários avaliam a menor unidade de um programa, como funções, métodos ou pequenos trechos de código, com o objetivo de provocar falhas, que foram ocasionadas por defeitos na implementação.

Já os testes de integração visam as falhas que ocorrem quando partes distintas de um sistema são integradas. Os testes de sistema tentam buscar por falhas, simulando a utilização do software por um usuário final e assim verificar se o produto deixa de satisfazer algum de seus requisitos. Os testes de aceitação são realizados por um pequeno grupo de usuários finais, que validam se o *software* atende suas expectativas. Os testes de regressão são realizados em novas versões do *software* e consistem em realizar novamente os testes já aplicados nas versões anteriores (ROCHA et al., 2001).

Ainda no contexto de testes de *software*, se faz necessário definir os termos falha, defeito e erro. Um defeito é uma inconsistência no código ou uma implementação incorreta. Já um erro, é uma consequência do defeito, ou seja, um erro de execução. Por fim, uma falha é um comportamento não esperado de um *software*, que pode ser ocasionada

por um defeito ou erro (DELAMARO; JINO; MALDONADO, 2017).

Neste trabalho, o propósito é verificar se um algoritmo simples, implementado em uma linguagem de programação, apresenta defeitos ou erros. O modelo de teste a ser aplicado é uma adaptação do teste unitário, o qual verifica de forma isolada se aquele algoritmo atende ou não ao objetivo especificado em uma questão. Para tal, além da compilação ou interpretação que vai capturar defeitos de sintaxe ou erros de execução, serão aplicados alguns dados válidos e inválidos de entrada e feitas algumas afirmativas sobre os valores finais das variáveis e retornos de funções. As negativas em tais afirmações serão capturadas como erros.

2.3 Educação a distância e ambientes virtuais de aprendizagem

Educação a distância é uma forma de aprendizado que ocorre em um lugar diferente do local do ensino. Exige modelos diferentes de comunicação geralmente por meio da tecnologia de informação e tem por vantagens melhorar o sistema educacional, proporcionando aos alunos a chance de ampliarem suas aptidões, reduzir custos educacionais, direcionar a informação para público-alvos específicos, nivelar as desigualdades dos grupos etários (MOORE; KEARSLEY; DISTÂNCIA, 2007).

Aparecendo cada vez mais no contexto da sociedade contemporânea, a educação a distância se mostra extremamente adequada às demandas educacionais sendo que as tecnologias empregadas vão além dos sistemas de educação a distância, chegando a canais de televisão e produtos multimídia (BELLONI, 2003).

Visto que alunos e professores não estão, necessariamente, no mesmo lugar ou no mesmo horário, um método de comunicação a distância permite que possam interagir e trocar informações com uma dinâmica diferente daquela aplicada em sala de aula (MOORE; KEARSLEY; DISTÂNCIA, 2007).

Ao utilizar uma plataforma de educação a distância para o ensino de algoritmos, cria-se um maior dinamismo nas relações entre professores e alunos, tornando a disciplina mais interessante e sendo necessário poucas aulas presenciais para o aprendizado dos

conceitos iniciais (ROSA; GIRAFFA, 2011).

AVAs são ferramentas da tecnologia de comunicação e informação que se utilizam do meio eletrônico para disseminar informação e permitir uma maior interação entre discentes e docentes (PEREIRA; SCHMITT; DIAS, 2007). Para que o aprendizado seja controlado e o conteúdo disponibilizado corretamente, os ambientes devem oferecer ferramentas como controles de acesso, tempo e progresso dos alunos, avaliação, comunicação, ajuda e manutenção (MILLIGAN, 1999). Com isso, os AVA's utilizam a internet para aproximar os alunos dos professores e disponibilizarem o conteúdo e materiais de estudo de forma acessível a qualquer lugar e horário.

2.3.1 Sistemas online para aprendizado, prática e competição de programação

O uso de um ambiente virtual que visa o aprendizado e treinamento em programação e algoritmos, principalmente quando usado em disciplinas iniciais, é importante para a formação dos alunos. Ao usar estes ambientes, os alunos melhoram seu rendimento e desempenho além de ficarem melhor preparados para disciplinas mais complexas em períodos futuros (SANTOS; COSTA, 2006).

Nesses sistemas, os alunos podem por em prática o que foi aprendido, propondo soluções para os exercícios e recebendo uma resposta quase que imediata. Além disso, em alguns desses sistemas, os alunos podem interagir entre si, para discutir soluções, tirar dúvidas e aprimorar seus conhecimentos.

Uma outra abordagem utilizada para o engajamento dos alunos, é o uso de métodos de competição entre os participantes. De acordo com Duarte, Moreira e Mello (2010), tais métodos podem ser facilmente observados nas olimpíadas de matemática, física, computação e outra áreas do conhecimento, onde os alunos competem entre si com o objetivo de obter a maior pontuação. Dessa forma, ao participar de competições de programação, os alunos têm a oportunidade de aprender com entretenimento diversos conceitos sobre programação, algoritmos e desenvolvimento de software (CAMPOS; FERREIRA, 2004).

Em competições de programação, geralmente são utilizados sistemas para geren-

ciar as atividades do evento. Um desses sistemas é o BOCA (BOCA *Online Contest Administrator*), um sistema online para controle das atividades de uma competição de programação, desenvolvido para ser usado na Maratona de Programação da Sociedade Brasileira de Computação. Pode também ser utilizado em disciplinas em que seja necessário a submissão de possíveis soluções de problemas propostos durante a aula (CAMPOS; FERREIRA, 2004).

Nesse sistema os usuários podem montar uma equipe, que fará a submissão das soluções dos problemas. Após os participantes submeterem uma solução, um juiz irá determinar, através do sistema, se a solução está ou não correta. Usuários que não estão participando diretamente da competição podem acessar o sistema com o objetivo de ver as equipes participantes, placar, quantos e quais problemas foram resolvidos por quais equipes e quanto tempo foi gasto para resolvê-los.

As diversas abordagens para o uso de sistemas *online* para educação a distância se apresentam como ferramenta viáveis e permitem aos alunos interagir com o conteúdo, com as linguagens, com os professores e entre si para aprimorarem seus conhecimentos. O aprendizado é realizado no próprio ritmo do aluno e em qualquer lugar. O rendimento de um aluno em algoritmos pode ser melhorado com a prática constante em programação e quanto mais formas de expô-lo ao conteúdo maior a sua chance de progresso. Portanto, fornecer um ambiente virtual de aprendizagem de fácil acesso, sem necessitar a instalação de um ambiente de programação pode contribuir para que ele pratique mais e em qualquer lugar que tenha acesso via internet. Uma análise entre os ambientes de de ensino e treinamento em programação será feita no Capítulo 3.

2.4 Gamificação

Gamificação é vista como o processo de pensar em jogos e seus mecanismos para incentivar os usuários a resolverem problemas (ZICHERMANN; CUNNINGHAM, 2011), tendo como base o ato de pensar como se estivessem em um jogo, interagindo com um sistema e respeitando a dinâmica de jogos, sem necessariamente estar jogando (SILVA et al., 2014).

Nesse aspecto, elementos de jogos podem ser utilizados para estimular os alunos a estudar através de um ambiente virtual, visto que um conjunto de atividades podem

ser entendidas como um desafio a ser completado e cada atividade ser comparada a uma fase de um jogo, inclusive aumentando o nível de dificuldade a medida que se progride nas tarefas (ZICHERMANN; CUNNINGHAM, 2011).

Ao concluir certa etapa ou atender algum requisito proposto, o aluno recebe uma gratificação, pontuação ou classificação. Zichermann e Cunningham (2011) resumem que as pessoas são motivadas a jogar por quatro grandes razões: por entretenimento, quando querem se divertir; por socialização, quando desejam interagir com outros jogadores; por relaxamento, quando buscam uma forma de se distrair e passar o tempo; e por aprendizado, quando querem adquirir novas habilidades ou aprimorar seu conhecimento.

Além disso, para que um usuário fique imerso na atividade, a gamificação utiliza frequentemente a chamada teoria do *Flow*, para tentar manter o foco do usuário, prendendo sua atenção na atividade que está realizando para que não se desconcentre com o que acontece ao seu redor (SILVA et al., 2014). A teoria do *Flow*, representa o estado de emoção de um usuário em função da sua habilidade e da dificuldade proposta na atividade. A Figura 2.1, mostra um gráfico com destaque para as zonas de ansiedade, tédio e *Flow*, sobre um plano cartesiano relacionando o nível de dificuldade de uma atividade na ordenada com a habilidade de um usuário no eixo das abscissas.

Se o nível de dificuldade de uma atividade é muito alto em relação à habilidade do usuário, seu nível de ansiedade fica alto, possivelmente conduzindo à frustração e consequente desmotivação. Por outro lado, se o nível de dificuldade da atividade é baixa demais em relação à sua habilidade, ele pode ficar entediado e acabar desistindo por considerar trivial demais. Com isso, a teoria do *Flow*, sugere equilibrar a experiência do usuário controlando os níveis de dificuldade em função do progresso de habilidade para mantê-lo constantemente engajado na atividade (ZICHERMANN; CUNNINGHAM, 2011).

Ao combinar esses aspectos de jogos em um AVA, é construído um cenário no qual os alunos são estimulados a participar, socializar com os colegas de turma e professores. Entretanto é preciso planejar bem a forma na qual as atividades serão apresentadas para que o conteúdo seja consumido de uma forma ativa pelo usuário, que os desafios sejam páreos para as suas habilidades. A próxima seção apresenta algumas abordagens para se

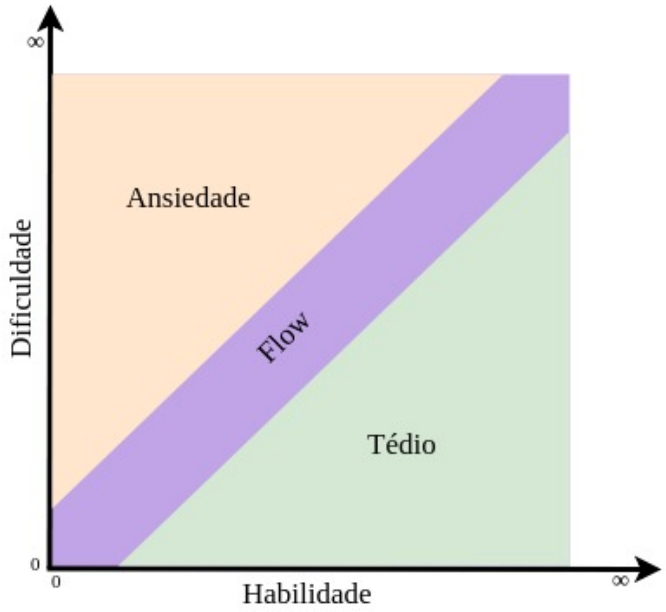

Figura 2.1: O estado de Flow é o equilibrio entre as zonas de ansiedade e tédio, considerando a dificuldade da atividade e a habilidade do usuário.

criar esse pareamento.

2.5 Teoria de resposta ao item

A teoria de resposta ao item (TRI) consiste em modelos matemáticos que visam determinar a probabilidade de uma pessoa responder a uma questão corretamente de acordo com parâmetros relacionados ao item e as habilidades da pessoa. Dessa forma, quanto maior a habilidade de um indivíduo, maior a probabilidade dele acertar uma questão (ALEXANDRE et al., 2003).

De acordo com Valle (2000) os modelos matemáticos da TRI se fundamentam em três fatores: da natureza do item; do número de grupos de pessoas envolvidas e da quantidade de traços latentes que estão sendo medidos. Neste contexto, a natureza do item diz respeito a ele ser dicotômico, ou seja, possuir resposta correta ou resposta incorreta ou ser não dicotômico. Já a quantidade de grupos envolvidos podem variar entre um ou mais e a quantidade de traços latentes, que são características que não podem ser observadas diretamente, também podem variar entre um ou mais.

Uma das principais características da TRI é considerar principamente um item e não a prova toda, com isso, é possível comparar grupos que realizam provas seme-

lhantes ou comparar pessoas de um mesmo grupo que realizam provas totalmente distintas(ALEXANDRE et al., 2003). Ainda de acordo com ALEXANDRE et al. (2003) a TRI pode ser calculada de três formas diferentes e a equação 2.1 apresenta uma delas.

$$P(U_{ij} = 1|\theta_j) = \frac{1}{1 + e^{-D(\theta_j - b_i)}} \tag{2.1}$$

Nessa equação, i é um número que varia de 1 a n, o número de itens no teste. U_{ij} é uma variável dicotômica que assume os valores 1, quando o indivíduo j responde corretamente o item i, ou 0 quando o indivíduo j não responde corretamente o item i. O termo θ_j é o valor do traço latente de um indivíduo j. Já $P(U_{ij} = 1|\theta_j)$ é a probabilidade de um usuário j com habilidade θ responder corretamente à questão i. Já e é um número constante cujo valor é $2,718$, base natural dos logaritmos neperianos e D é um número constante cujo valor é $1,7$ para tornar a função o mais próximo de uma função normal. Por fim, b_i é a dificuldade relativa ao item.

Dessa forma, no contexto deste trabalho, um modelo semelhante ao da TRI, detalhado na Seção 4.4, será usado para calcular a probabilidade de um usuário, com certo nível de habilidade, acertar uma questão com determinado nível de dificuldade.

2.5.1 Elo rating

O *Elo rating* é um método estatístico criado inicialmente com o intuito de calcular, para cada jogador, uma classificação numérica em partidas de xadrez. Esta classificação, geralmente entre os valores 0 e 3000, é atribuída a cada jogador e varia com o tempo, de acordo com o seu desempenho (GLICKMAN; JONES, 1999).

Ainda de acordo com Glickman e Jones (1999), em uma partida entre dois jogadores A e B, a pontuação esperada para o jogador A é dada por:

$$E_a = \frac{1}{1 + 10^{-(R_a - R_b)/400}} \tag{2.2}$$

Onde E_a é o valor da pontuação e R_a e R_b são, respectivamente, as classificações dos jogadores A e B. Assim, a fórmula para atualizar a classificação do jogador A é dada por:

$$R'_a = R_a + K(S_a - E_a) \tag{2.3}$$

Onde, R'_a é o valor atualizado de sua classificação, R_a é a classificação antes do jogo, K, é uma constante que em geral, possui o valor 16, S_a é a pontuação obtida na partida e E_a é o valor esperado de pontuação, obtido conforme Equação 2.2.

Dessa forma, no contexto deste trabalho, um método semelhante ao *Elo Rating*, detalhado na Seção 4.6, será usado para calcular a pontuação de um usuário em relação a sua resposta a cada questão.

2.6 Considerações parciais

Este Capítulo apresentou os conceitos teóricos que são utilizados em diversas etapas deste trabalho. Através desses conceitos, será possível realizar uma avaliação das ferramentas disponíveis no Capítulo 3 e guiar uma proposta de arquitetura no Capítulo 4. Diante do exposto, será proposto um ambiente virtual onde estudantes dos cursos de tecnologia da informação poderão aprender e por em prática os ensinamentos da disciplina de algoritmos. Além disso, os alunos terão acesso à ferramentas que os incentivam e aumentam sua produtividade, rendimento e desempenho, uma vez que o uso da gamificação desperta o interesse dos estudantes a participarem das atividades e concluírem os desafios.

3 Trabalhos relacionados

Durante o desenvolvimento deste trabalho, diversos conceitos, tecnologias, abordagens e ferramentas foram analisados, dando importância aqueles que mais se assemelham à abordagem proposta. Este Capítulo apresenta os resultados das pesquisas realizadas.

3.1 Ambientes online para programação

A fim de detalhar as abordagens correntes para ensino de programação, foi realizada uma busca por plataformas e AVA's que têm por objetivo o ensino, treinamento ou competição de programação, dando atenção especial aqueles que fazem uso das ferramentas e abordagens descritas no Capítulo 2, como gamificação, competição e testes automatizados.

Esta busca mostrou a existência de vários ambientes *online* que permitem a pessoas aprender, treinar e aprimorar suas habilidades em programação de computadores. Muitas dessas plataformas possuem alunos de todas as partes do mundo e a utilização dos conceitos de execução *online* e técnicas de gamificação. Neste Capítulo, será feita uma revisão dessas plataformas e uma breve discussão de suas principais características.

3.1.1 Codewars

O Codewars é um ambiente de treinamento com o objetivo de alcançar o domínio de escrita de código através de desafios. Sua proposta é usar os *katas* ("forma", em uma tradução livre do japonês) como forma de treinamento de programação. Os uso de *katas* é uma técnica de treinamento individual de artes marciais utilizado para disseminação de várias técnicas eficientes. Assim se repetem as boas práticas de forma que estas sejam bem conhecidas e utilizadas ao invés de uma solução única, distribuída para várias pessoas (DEJARNETT et al., 2017). Neste ambiente, não há disponibilização de material didático, sendo que a forma de aprendizado se dá pura e exclusivamente pela resolução dos desafios e interação com outros membros.

Na forma de um ambiente colaborativo, o CodeWars permite que os desafios se-

jam construídos pelos próprios participantes, que opor sua vez, são classificados de acordo com uma pontuação acumulada. Os usuários podem ganhar pontos de reconhecimento ao resolver *katas* e completar objetivos propostos pela plataforma. Quanto maior a classificação de um membro, desafios mais complexos se tornam disponíveis.

Na plataforma é possível construir um perfil de usuário completo ao escolher as linguagens de programação nas quais se deseja treinar. É possível se tornar membro de um clã e entrar em contato com outros usuários para discutir as soluções dos desafios e fornecerem seu nível de satisfação ao resolvê-los. Ao propor uma solução, toda a comunidade pode realizar comentários, além de qualificar as soluções em categorias como boas práticas de codificação, solução inteligente ou comparar com a sua própria solução, que fica disponível com o intuito de inspirar outros usuários.

3.1.2 Khan Academy

A Khan Academy é uma organização sem fins lucrativos criada por desenvolvedores, professores, *designers*, estrategistas, cientistas e especialistas, cujo propósito é oferecer uma educação gratuita e de alta qualidade para todos, em qualquer lugar (O'SULLIVAN; SAL, 2005).

Ela oferece conteúdo na forma de vídeos instrucionais e exercícios, em um ambiente virtual personalizado que permite aos estudantes aprenderem no próprio ritmo, tanto dentro quanto fora de sala de aula, abordando conteúdos de matemática, ciências, economia, programação de computadores e outros. Em sua plataforma, além das ferramentas de ensino para estudantes, pais e professores podem ter acesso a ferramentas para acompanhar e monitorar o desempenho e rendimento dos filhos e alunos.

Para o ensino de programação, a proposta desse ambiente é fornecer conteúdo em forma de vídeo-aulas sobre determinado assunto e propor exercícios com um *feedback* imediato, apontando os erros do aluno, caso necessário.

3.1.3 Code School

A *Code School* é uma plataforma de aprendizado *online* destinada a quem deseja se tornar ou já é um programador. Seus cursos são construídos sobre uma narrativa temática,

usando elementos de jogos, vídeos instrutivos e ferramentas para codificação com o intuito de que os usuários aprendam fazendo e sintam que estão participando de um jogo com diversão e entretenimento (POLLACK, 2011).

É possível criar uma conta gratuita e ter acesso aos cursos grátis ou criar uma conta *premium*, paga mensalmente, para ter acesso ao conteúdo integral na plataforma. Seus cursos são categorizados de forma que o usuário siga um caminho de aprendizado, concluindo sequencialmente os cursos de determinada categoria.

Os cursos são divididos em módulos e seções. Cada seção possui um vídeo instrutivo e um conjunto de desafios, que quando concluídos, dão pontos ao usuário. Com esses pontos, o usuário pode comprar dicas e até mesmo soluções de exercícios mais complexos. Ao terminar todos os módulos de um curso, o usuário recebe uma medalha, que serve como certificado de conclusão.

Este ambiente se assemelha à uma sala de aula virtual e individual, onde o aluno aprende o conteúdo e pratica com os exercícios propostos, mas sem interagir com outros estudantes.

3.1.4 Code.org

A *Code.org* é uma organização sem fins lucrativos que se dedica a melhorar o ensino de ciência da computação, desmitificando que a ideia de programar seja algo difícil. Com a intenção de que pais, professores e alunos de todo o mundo possam ser incentivados a aprender programação, é oferecido um ambiente virtual onde todos podem iniciar seus estudos de forma descontraída e divertida (PARTOVI; PARTOVI, 2013).

Em sua plataforma, qualquer pessoa a partir dos quatro anos de idade pode começar a programar criando jogos eletrônicos simples. Usando uma ferramenta *online* de arrastar blocos com o *mouse*, os usuários podem facilmente criar uma sintaxe visual, em foma de fluxograma, para a execução de operações.

Após se cadastrar na plataforma, o usuário pode escolher entre os vários cursos disponíveis que abrangem desde atividades pré-alfabetização até os que já possuem conhecimento em ciência da computação.

Dessa forma, o ambiente fornece uma forma de ensino de programação, sem

necessariamente ter que escrever linhas de código, permitindo que os alunos aprendam os conceitos de algoritmo sem ter que conhecer uma linguagem de programação.

3.2 Propostas de ambientes de ensino de programação

No aspecto acadêmico, foi feito um levantamento de trabalhos que abrangem treinamento em programação e competição de programação, que se assemelham ou que utilizam os mesmos conceitos da abordagem proposta nesta monografia, além de um levantamento de suas conclusões obtidas e uma análise de seus objetivos futuros.

Esse levantamento mostra que foram realizados vários trabalhos relacionados ao treinamento e aprimoramento de habilidades em programação, utilizando AVA's e alguns dos conceitos já abordados. A seguir, será listada uma série histórica de alguns trabalhos científicos com uma breve análise dos resultados obtidos e quais objetivos ainda pretendem atingir.

No contexto do uso de AVAs, Ribeiro e Ishitani (2008) realizaram o levantamento e avaliação dos trabalhos existentes, elaboraram propostas de melhorias, propuseram o desenvolvimento de um AVA colaborativo, realizaram testes com os alunos do curso de Sistemas de Informação e compararam as notas finais dos alunos com as notas de turmas anteriores que não fizeram uso do AVA. Em sua análise dos resultados, os autores constataram uma melhora no aprendizado, sendo que os alunos foram capazes de escrever códigos corretos e mais elaborados.

Moreira e Favero (2009) realizaram um estudo sobre as técnicas utilizadas em um AVA e a utilização de avaliação automática de algoritmos, que possibilitam um *feedback* imediato ao aluno e ajuda o professor na avaliação dos exercícios propostos nas aulas práticas de laboratório. Adicionalmente, em um estudo de caso, realizaram uma comparação entre dois métodos de avaliação automática: um que faz uma regressão linear múltipla e outro que faz uso das métricas de engenharia de software, concluindo que ambas abordagens atendem bem o objetivo esperado na avaliação automática de programação. Como trabalho futuro, os autores pretendem realizar a criação de um módulo que permite que um AVA possa utilizá-lo em outras atividades, como resposta de questionários.

Já Rocha et al. (2010) desenvolveram uma ferramenta integrada a um AVA, se

baseando no Sistema Personalizado de Ensino. Além disso, realizaram uma análise de resultados de sua implantação em uma instituição de ensino e constataram que com o uso da ferramenta houve um aumento significativo na aprovação dos alunos e uma redução no abandono e desistência da disciplina. Como trabalhos futuros, Rocha et al. (2010) indicam a aplicação dessa metodologia em outras disciplinas de programação; o uso dessa e de outras ferramentas semelhantes em outras instituições de ensino para que seja feita uma análise mais profunda e a implementação de novas funcionalidades na ferramenta, como correção automática das submissões.

Alves e Jaques (2014) propuseram um AVA com fácil interação, *feedback* personalizado e ferramentas que auxiliam os professores a acompanhar o aprendizado dos alunos. Um experimento foi realizado em uma turma da disciplina de programação, com 24 alunos, para avaliar a experiência no uso do ambiente proposto. Além disso, professores e alunos responderam a um questionário. Como resultado do experimento, os autores concluíram que o AVA foi útil para os alunos e como propostas de melhorias foi levantado a automatização dos cenários de teste e o uso de gamificação para aumentar o engajamento dos alunos.

Leite (2015) realizou uma revisão bibliográfica sobre a inclusão do ensino de programação na educação básica e propôs uma abordagem metodológica e simples com o objetivo de inserir o estudo de linguagens de programação no ensino básico. Essa abordagem cita o passo a passo para a utilização de uma ferramenta *online* para criação de códigos, que possui uma interface simples e pode ser usada de forma multidisciplinar. Com isso, é esperado que professores possam utilizar essa e outras ferramentas para o ensino de programação nas escolas, sendo necessário apenas um laboratório de informática com acesso à internet.

3.3 Propostas de ambientes de competição de programação

Ainda no campo acadêmico, vários trabalhos relacionados à competição de programação e utilização de juízes online já foram abordados. A seguir, será listada uma série histórica

de alguns desses trabalhos e realizada uma breve análise dos resultados obtidos.

Figueiredo et al. (2008) propuseram uma abordagem gamificada para o ensino de programação orientada a objetos a fim de motivar alunos em seus estudos. Para verificar os resultados da abordagem proposta, os dados das notas dos alunos referentes aos resultados das avaliações da disciplina foram comparados com os dados de alunos de uma turma que não participou da abordagem gamificada, onde ambas turmas possuíam a mesma ementa, conteúdos programáticos, carga horária, material didático e instrumentos de avaliações, diferenciando-se apenas pelo uso da gamificação. Os autores concluíram que as turmas que participaram da abordagem gamificada tiveram um aproveitamento maior do que a turma que não fez uso de tal abordagem.

França e Soares (2011) apresentaram um ambiente que integra um AVA a uma ferramenta usada no apoio à competições de programação. Dessa forma, foi possível registrar o resultado da execução de problemas propostos aos alunos, rastreando suas submissões e possibilitando a correção do professor via interface do AVA. Com base nos resultados obtidos, os autores observaram que a maioria dos alunos afirmam que foi possível utilizar a ferramenta para acompanhar a disciplina em uma abordagem onde existe pouca proximidade entre alunos e professores.

Santos e Ribeiro (2011) propuseram o desenvolvimento de um AVA integrado a um juiz online didático, com o intuito de apoiar as disciplinas de programação, ajudar alunos a aprimorarem seus conhecimentos e auxiliar os professores no processo de automatização da avaliação dos códigos. Dessa forma, os autores esperam contribuir significativamente na formação acadêmica de alunos da computação, e que estes sejam incentivados a participar de maratonas de programação e desenvolver programas em grupo e de forma colaborativa.

Sales, Jr e Sales (2016) realizaram uma análise do uso de estratégias baseadas em maratonas de programação e juízes eletrônicos e da influência dessas estratégias no desempenho de alunos. Sua metodologia foi realizar uma pesquisa qualitativa e elaborar um estudo de caso. Com os resultados encontrados, os autores constataram que os alunos participantes tiveram um aumento em seu desempenho individual quando comparado com alunos que não participaram. Além disso, foi observado que após cursar uma disciplina

que faz das estratégias citadas, os alunos também obtiveram melhor desempenho nas disciplinas seguintes.

3.4 Comparação entre os trabalhos

A fim de facilitar uma análise por comparação, as características observadas nos trabalhos foram organizadas em tabelas. Esta seção discute rapidamente os trabalhos.

3.4.1 Comparação entre ambientes online para programação

Para os ambientes *online*, os critérios de comparação foram: se disponibilizam material didático; se fazem uso de testes automatizados; se usam abordagens gamificadas; se disponibilizam interação entre os usuários; se as atividades disponíveis seguem uma progressão de dificuldade de acordo com o desempenho do usuário e se o ambiente fornece um *feedback* instrutivo para o usuário. Na Tabela 3.1, a letra "X" representa se a plataforma atende ao critério.

Tabela 3.1: Comparação entre ambientes online para programação.

Plataforma	Material	Testes	Gamificação	Interação	Progressão	Feedback
Codewars		X	X	X	X	
Khan Academy	X		X			
Code School	X	X	X		X	X
Code.org	X	X			X	X

Para os trabalhos acadêmicos, os critérios de comparação foram: se fazem uso de testes automatizados; se usam abordagens gamificadas e se a proposta fornece um *feedback* instrutivo para o usuário. Na Tabela 3.2, a letra "X" representa se a proposta atende ao critério.

3.5 Considerações parciais

Neste Capítulo foram apresentados os trabalhos utilizados como referência para este estudo. Os trabalhos comerciais servem para guiar a construção de uma arquitetura que possa ser interessante para os alunos, visto que suas plataformas, ferramentas e métodos

Tabela 3.2: Comparação entre ambientes online de treinamento e competição de programação.

Trabalho	Testes automatizados	Gamificação	Feedback instrutivo
Ribeiro e Ishitani (2008)	X		
Moreira e Favero (2009)	X		X
Rocha et al. (2010)			
Alves e Jaques (2014)			X
Leite (2015)			
Figueiredo et al. (2008)		X	X
França e Soares (2011)	X		X
Santos e Ribeiro (2011)	X		X
Sales, Jr e Sales (2016)	X		

de ensino atraem pessoas de todo o mundo. Os trabalhos científicos podem ser usados como indicadores para a aplicação dessa abordagem no contexto do ensino de algoritmos e linguagens de programação dentro das universidades.

Foi observado nos trabalhos acadêmicos, que o uso de AVA's aliado a ferramentas como a gamificação e um *feedbak* interativo, aumenta o interesse dos alunos pela disciplina, aumenta a interatividade dos alunos entre si e com os professores e melhora o desempenho dos alunos. Além disso, professores passam a ter uma maior facilidade de acompanhar o aprendizado dos alunos.

Através da análise dos trabalhos relacionados será descrita uma arquitetura e a implementação de um ambiente que permita combinar características que se destacaram em projetos diferentes e obter métricas em um possível estudo de campo. Essa arquitetura conta com o uso de testes para validar as soluções, abordagens gamificadas, especificamente no pareamento de questões e usuários, que também funciona como progressão e por fim um *feedbak* sobre solução. Como esta arquitetura não considera, necessariamente, uma turma ou disciplina, o fornecimento de material didático não está no escopo deste trabalho. O Capítulo 4 apresenta a arquitetura conceitual da proposta relacionando essas características.

4 Arquitetura de treinamento

Este Capítulo tem por objetivo propor uma arquitetura conceitual para um ambiente *online* de treinamento, no qual os usuários receberão um conjunto de questões e poderão submeter soluções de exercícios e receber de imediato uma resposta com seu progresso. Adicionalmente, esse ambiente vai usar do sistema de pareamento como elemento de gamificação na tentativa de aumentar o engajamento os alunos.

4.1 Definição do ambiente

Foi construído um ambiente *online*, que pode ser divido em três camadas: as interfaces com usuário para sua operação (o *front-end*, em inglês) e o acesso aos bancos de dados e código que deve ser executado em segurança fora da máquina do cliente (o *back-end*, em inglês), por fim, uma camada intermediária, que faz o controle dos dados apresentados e a comunicação entre o *front-end* e o *back-end*.

Para que se tenha maior agilidade no desenvolvimento, serão utilizados *frameworks* que vão auxiliar na construção de ambas as partes. Esses *frameworks* tendem por facilitar o desenvolvimento, diminuir o tempo gasto para implementação e garantir boas práticas de programação durante o desenvolvimento. Adicionalmente, por ser uma plataforma aberta, outros desenvolvedores poderão, no futuro, se sentir incentivados à colaborar, adaptar ou realizar uma implementação independente.

Para o controle, a interface com usuário será acessada através de um navegador de internet, em qualquer sistema operacional, inclusive em dispositivos móveis. Entretanto, é aconselhável que o acesso seja feito através de um computador, para que o usuário tenha uma melhor experiência de uso pois a digitação de código fonte sem um teclado é um limitador. Uma conta deverá ser criada, com identificação e senha, que irá garantir o acesso seguro do usuário à plataforma. Através da sua conta, o usuário poderá ter acesso aos exercícios propostos e ao acompanhamento das questões já respondidas.

Para o código a ser executado no servidor, será criado um sistema capaz de prover

as interfaces para usuário, receber as suas ações e registrar o estado em bancos de dados. Como cada submissão feita pelos usuários deve ser executada, será utilizada uma técnica chamada *sandboxing*. Essa técnica permite executar programas, códigos ou trechos de código em um computador que não seja o do usuário. Dessa forma, a máquina que executa o código pode limitar o uso de memória, uso de processador e tempo de execução, garantindo assim que a integridade do servidor no qual o sistema está implantado não seja prejudicada.

Cada usuário verá a lista de atividades propostas, previamente cadastradas na aplicação e poderá enviar a solução para cada uma delas. Todo o processo é registrado e medido. Uma avaliação sobre a execução de um código submetido será fornecida ao usuário, que poderá seguir ao próximo exercício ou refazer uma solução, caso algum erro ou falha tenham sido encontrados.

Em um primeiro momento, a plataforma irá oferecer apenas atividades em uma única linguagem de programação, o JavaScript. Entretanto, futuramente, espera-se que quaisquer outras linguagens possam ser executadas de modo seguro (*sandboxing*) no sistema operacional do servidor poderão ser oferecidas.

4.2 Processo de treinamento

Todo o processo de escolha, apresentação e avaliação das questões disponíveis para treinamento precisa ter definidos os termos utilizados neste trabalho. Esta seção, apresenta todo o fluxo de trabalho referente ao processo de treinamento proposto dentro do ambiente.

4.2.1 Processo de resolução

Um Processo de Resolução se inicia no momento em que um usuário inicia a solução de uma questão e termina com a avaliação de sua solução. A Figura 4.1 mostra um diagrama de atividades que representa todo o processo de resolução iniciado por um usuário.

4.2.2 Questão

Uma questão é uma tarefa, atividade ou desafio que será proposto ao usuário. Possui um título, uma descrição textual, um nível de dificuldade e um conjunto de casos de teste. O nível de dificuldade é inicialmente igual para todas as questões e recalculado automaticamente através das medidas que são colhidas durante o Processo de Resolução, conforme serão detalhadas na Seção 4.3.

4.2.3 Solução

Uma solução é um texto que o usuário submete para solucionar uma questão. Esta solução pode ser um texto simples mas dentro de deste contexto de aplicação, ela é um código em uma linguagem de programação. Essa solução sempre é enviada ao Executor. A cada submissão, medidas são colhidas para futuramente serem analisadas pelo sistema.

4.2.4 Executor

O Executor recebe uma Solução e determina se ela é executável dentro dos parâmetros do sistema, gerando uma Resposta. O executor irá definir, por exemplo, se uma solução contém algum erro, ou seja, se há erro de sintaxe, uso excessivo de memória ou extrapola os limites de tempo de execução.

4.2.5 Resposta

Uma Resposta é a definição do Executor sobre a Solução. Caso o executor detecte algum defeito, ele informa ao usuário qual foi o defeito encontrado e o Processo de Resolução deve ser reiniciado. Caso o executor não encontre defeitos, a Resposta é enviada ao Avaliador.

4.2.6 Avaliador

O Avaliador recebe uma Resposta, aprovada pelo Executor e aplica a técnica de teste unitário, usando os casos definidos na criação da Questão. O Avaliador vai buscar por falhas, gerando uma Avaliação. Esta Avaliação é enviada ao usuário. Assim, o Avaliador irá definir, por exemplo, se a execução dessa Respostas leva a uma falha.

4.2.7 Avaliação

A Avaliação é a informação gerada pelo Avaliador. Caso a Solução seja aprovada nos testes, a Avaliação é dada como correta e o Processo de Resolução termina. Caso a Solução não seja aprovada, o usuário reinicia o Processo de Resolução.

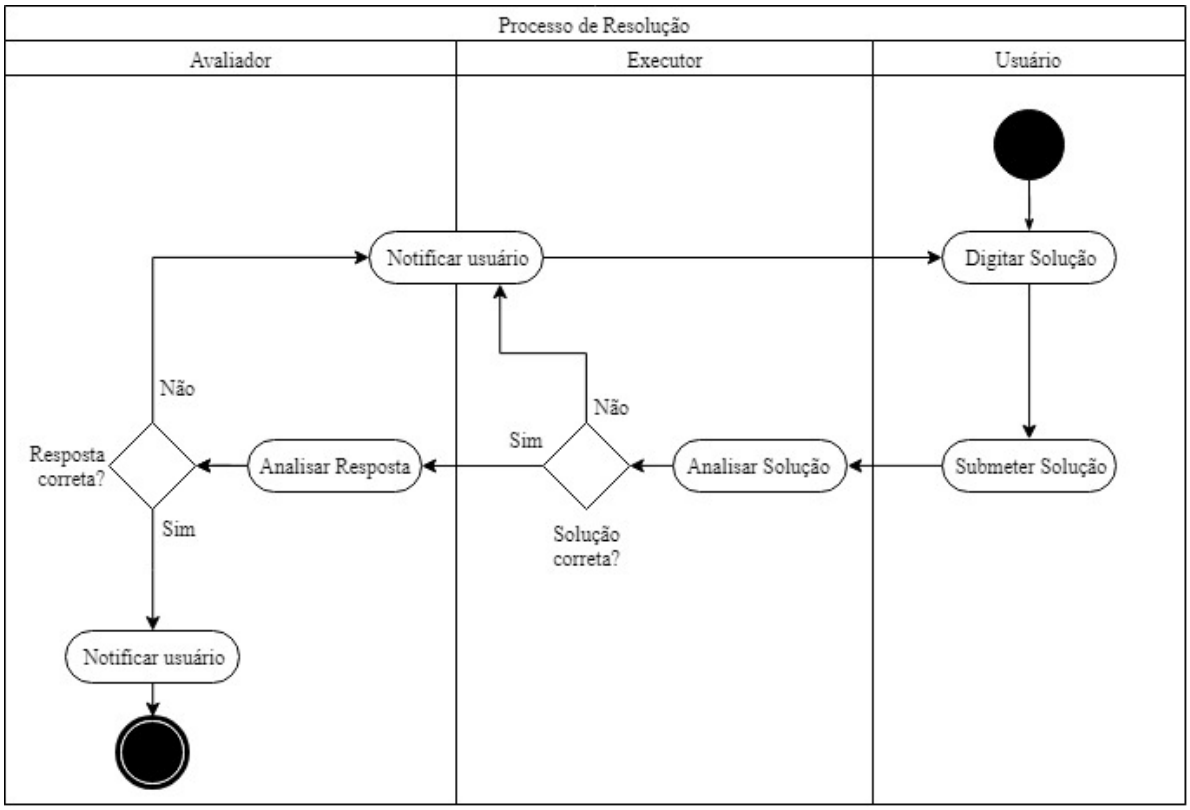

Figura 4.1: O Processo de Resolução tem início quando um usuário tenta resolver uma questão. A solução enviada é analisada pelo Executor e, caso seja executável, é enviada ao Avaliador. O Avaliador analisa a Resposta e, caso correta, o processo termina. Caso a solução ou resposta sejam incorretas, o processo se reinicia.

4.3 Medidas e Métricas

As Medidas são observações realizadas durante todo o Processo de Resolução. Elas servem para que o sistema possa, posteriormente, definir métricas sobre as questões e sobre o progresso dos usuários.

Inicialmente, as medidas realizadas durante o Processo de Resolução são o tempo para solução (TPS), quantidade de tentativas (QDT), quantidade de enganos cometidos (QEC), quantidade de respostas corretas (QRC), quantidade de processos de solução interrompidos (QPI), a lista de códigos dos enganos cometidos (LEC) e as soluções, tanto

intermediárias (SI), que são as soluções em cada tentativa, quanto a solução final (SF). As datas de início (DI) e fim (DF) da solução, o valor esperado (E_a), valor de medidas (P_m) e pontuação final (P_f) também são registrados.

O tempo de solução é o tempo que o usuário leva para interpretar e submeter uma solução que seja avaliada como correta, que é a solução final da questão. Já a quantidade de tentativas é o número de soluções que tiveram uma resposta ou uma avaliação incorreta e as soluções intermediárias são os códigos submetidos em cada tentativa. A quantidade de respostas corretas é o número de vezes que uma solução foi avaliada como correta e a quantidade de soluções interrompidas é o número de vezes em que o processo de resolução foi iniciado, mas não se chegou em uma solução correta. Por fim, a quantidade de enganos cometidos é a soma de todos os erros encontrados pelo executor e avaliador nas soluções incorretas e a lista de códigos é o conjunto dos valores textuais, em geral com os códigos de erro de execução, geradas pelo avaliador ou executor.

Tabela 4.1: Sigla, nomenclatura e unidade de cada medida.

Sigla	Nomenclatura	Unidade
DF	Data final	Data
DI	Data inicial	Data
E_a	Valor esperado	Número
LEC	Lista com os enganos cometidos	Lista textual
P_f	Pontuação final	Número
P_m	Pontuação de medidas	Número
QDT	Quantidade de tentativas	Número
QEC	Quantidade de enganos cometidos	Número
QRC	Quantidade de respostas corretas	Número
QPI	Quantidade de soluções interrompidas	Número
SF	Solução final da questão	Texto
SI	Soluções intermediárias das tentativas	Texto
TPS	Tempo para solução	Segundos

4.4 Pareamento de questões com as habilidades dos usuários

Com o objetivo de melhorar o fluxo de treinamento, um pareamento entre o nível de dificuldade das questões e o nível de habilidade do usuário pode ser realizado. Dessa

forma, os usuários terão acesso a questões com um nível de dificuldade que se adéquam a sua habilidade.

Em um momento inicial, quando um usuário tem acesso pela primeira vez ao AVA, baseando-se no modelo Elo *rating*, seu nível de habilidade é estipulado em 1500 e de acordo com sua interação com o ambiente, o valor de sua habilidade varia entre 0 e 3000. A mesma regra é aplicada para as questões. Ou seja, quando se tornam disponíveis para serem resolvidas, seu nível de dificuldade tem o valor 1500 e se ajusta à medida que usuários a respondem, variando também entre 0 e 3000.

No contexto deste trabalho, para que se utilize as métricas da TRI, alterações nas constantes são necessárias, visto que há diferenças nas ordens de grandeza da habilidade e dificuldade. Além disso, as probabilidades de acerto são calculadas levando em consideração uma única questão e a habilidade de um único usuário.

Dessa forma, a função que calcula a probabilidade de um usuário acertar uma questão é definida como:

$$P(U = 1|\theta) = \frac{1}{1 + e^{-D(\theta - b)}} \tag{4.1}$$

Onde U é uma variável dicotômica que assume os valores 1, quando o usuário responde corretamente a questão, ou 0 quando o usuário a responde incorretamente. O termo θ é valor do traço latente, ou seja, da habilidade de um usuário. Já $P(U = 1|\theta)$ é a probabilidade de um usuário com habilidade θ responder corretamente à questão. E e é um número constante cujo valor é $2,718$, base natural dos logaritmos neperianos. D é um número constante cujo valor é $0,0051$ para tornar a função o mais próximo de uma função normal. Por fim, b é a dificuldade relativa ao item.

Tomando como exemplo um usuário com habilidade $\theta = 1693,67$ e uma questão com nível de dificuldade $b = 1797,50$, $P(U = 1|\theta)$ assume o valor $0,37$. Considerando o mesmo usuário e uma outra questão com $b = 1516,72$, a probabilidade deste usuário acertá-la é $0,71$. Caso $\theta = b$, temos $P(U = 1|\theta) = 0,50$. Ou seja, através desse cálculo, é possível classificar as questões para cada usuário.

4.5 Listagem de questões

Na listagem de questões, é realizado o cálculo conforme a Equação 4.1 para cada uma das questões listadas. As questões são então agrupadas em questões adequadas, difíceis e fáceis. Em cada grupo, são ordenadas em ordem decrescente da probabilidade de aquele usuário acertar a questão.

O Agrupamento das questões questões se dá da seguinte maneira: se $P(U = 1|\theta)$ = 0,35 ou menos, a questão é considerada difícil para esse usuário, caso $P(U = 1|\theta)$ esteja entre 0,36 a 0,65, a questão é considerada adequada e se $P(U = 1|\theta)$ entre 0,66 a 1,00, a questão é considerada fácil.

A plataforma deve orientar o usuário a escolher entre as questões adequadas, porém é permitido que ele tente responder a qualquer uma, se assim desejar, visto que ele já sabe o quão difícil será respondê-la.

4.6 Pontuação dos usuários e questões

Quando o Processo de Resolução se completa, é possível realizar avaliações do usuário em relação a sua resposta. Essa avaliação se dá através de um método baseado no *Elo Rating* que, no contexto deste trabalho, considera que um usuário está competindo com uma questão, ou seja, a pontuação do adversário é vista como o nível de dificuldade da questão. A pontuação esperada do Usuário, E_a, é calculada de acordo com a Equação 2.2, reproduzida abaixo:

$$E_a = \frac{1}{1 + 10^{-(R_a - R_b)/400}}$$

Além disso, os dados coletados também são atribuídos na pontuação, considerando que o tempo de solução e a quantidade de tentativas é inversamente proporcional à habilidade do usuário e diretamente proporcional à dificuldade da questão. Dessa forma, considerando esses fatores, a fórmula para se obter a pontuação de medidas do usuário é dada por

$$P_m = \left(\frac{t+s}{\theta}\right) \cdot b \tag{4.2}$$

Onde P_m é o valor da pontuação de medidas obtida, θ é a habilidade do usuário e b é a dificuldade relativa à questão. Já o termo t é o tempo medido e s é o número de tentativas, ambos normalizados entre 0 e 1, para manter a ordem de grandeza dos valores.

Com esses valores, é possível recalcular a habilidade do Usuário e o nível de dificuldade da Questão. Considerando as modificações realizadas no método, as fórmulas de atualização são dadas por:

$$\theta' = \theta + k[1 - (P_m + E_a)] \tag{4.3}$$

$$b' = b - k[1 - (P_m + E_a)] \tag{4.4}$$

Onde b' e θ' são, respectivamente, os valores atualizados de θ e b, k mantêm-se com o valor 16. Dessa forma, quando $k[1-(P_h+P_e)] < 1$ significa que o usuário superou a questão e estes pontos são somados a sua habilidade e subtraídos da dificuldade da questão. Caso contrário, o processo inverso ocorre.

Tomando como exemplo um usuário com habilidade $\theta = 1693,67$ e uma questão com nível de dificuldade $b = 1797,50$, a pontuação esperada do Usuário E_a assume o valor $0,354$, calculada de acordo com a Equação 2.2. A pontuação de medidas P_m, calculada de acordo com a equação 4.2, assume o valor $0,137$. Assim, a pontuação final calculada é $k[1 - (P_m + E_a) = 8.11$, fazendo com que os valores atualizados sejam $\theta = 1701.79$ e $b = 1789.38$, de acordo com as equações 4.3 e 4.4, respectivamente.

Dessa forma, os níveis de dificuldade das questões vão sendo moldados de acordo com as soluções dos usuários, sendo que questões realmente difíceis são aquelas pontuação mais alta e questões fáceis são as com pontuação mais baixa.

5 Implementação

Neste Capítulo será feito um detalhamento da implementação do protótipo da plataforma *online*, chamado Coding Training, especificando tecnicamente as tecnologias que foram utilizadas em seu desenvolvimento.

5.1 Arquitetura do protótipo

O Coding Training foi implementado seguindo uma estrutura *full-stack* em JavaScript, de forma que a linguagem é utilizada tanto no *front-end* e *back-end* quanto na persistência com um banco de documentos.

Ambas partes se comunicam através de requisições *Hypertext Transfer Protocol* (HTTP), também chamadas de serviços. Esses serviços foram construídos para terem um formato em JavaScript *Object Notation* (JSON), que é uma linguagem comum às duas partes. Dessa forma, ambas partes são independentes uma da outra, sendo que uma delas pode ser substituída sem que a outra precise de modificações.

Para o *front-end*, foram utilizadas as linguagens Javascript, *HyperText Markup Language* (HTML) e *Cascading Style Sheets* (CSS), além de *frameworks* e bibliotecas como AngularJS[1] versão 1.6, Bootstrap[2] versão 3.3.7, JQuery[3] versão 3.2.1, entre outras. O HTML é utilizado para criar as estruturas das páginas enquanto o CSS é o responsável por fornecer os estilos de *layout*. O JavaScript fornece o controle e comportamento da aplicação.

Utilizando o *framework* AngularJS, que combina essas tecnologias e fornece meios de aplicar os conceitos de *Model-view-controller*(MVC), o *front-end* está divido em três partes, as *views*, que são arquivos HTML, responsáveis pela renderização das telas; os *models*, que tratam os dados da aplicação; e por fim os *controllers*, que são os responsáveis pelo comportamento da aplicação e fazem a interação entre as *views* e os dados.

[1] https://angularjs.org/
[2] http://getbootstrap.com.br/
[3] https://jquery.com/

Para o *back-end*, será utilizado a linguagem Javascript, mais especificamente, a ferramenta Node.js[4] versão 8.10, com o auxílio de *frameworks* e bibliotecas. O Node.js é um ambiente que executa a linguagem JavaScript no servidor que permite a criação de *web servers*.

As principais bibliotecas e *frameworks* que foram utilizados são o Express[5] versão 4.13.3, Passport[6] versão 0.4 e o GF3-Sandbox[7] versão 0.8.6. O Express realiza o roteamento, ou seja, uma soliticação do cliente à um caminho específico, através de uma solicitação HTTP. O Passport realiza o intermédio da autenticação do usuário no servidor e o GF3-Sandbox é o responsável por criar um *sandbox* para a execução segura dos códigos submetidos pelos usuários.

Um exemplo de um roteamento pode ser visto na Listagem 5.1, que define uma *Uniform Resource Locator* (URL) para a listagem de usuários.

Listagem 5.1: Roteamento e função de mediação do express (*express middleware function*) na forma de serviço que trata uma requisição de listagem de usuários e a retorna como um JSON.

```
1 router.get('/read',(req,res,next) => {
2    User.find({}, (err,users)=>{
3       if(err) {
4          res.status(500).json({errMsg: err})
5       }
6       res.status(200).json(users);
7    });
8 });
```

O MongoDB é um banco de dados *Not Only Structured Query Language* (NoSQL) que utiliza esquemas no formato JSON para armazenar os dados. Dessa forma as informações persistidas podem ser facilmente trabalhadas, pois podem ser construídas de forma a facilitar a indexação ou a recuperação de dados.

Foi utilizado o Mongoose[8] versão 5.2.1, que é um *framework* para o Node.js, que fornece uma solução na criação dos modelos de dados, facilitando a interação com a aplicação, como validação e construção de consultas. Um exemplo de um modelo de dados no formato JSON pode ser visto na Listagem 5.2, que define a tabela de Questões.

[4]https://nodejs.org
[5]https://www.express.com/
[6]http://www.passportjs.org/
[7]https://github.com/gf3/sandbox
[8]http://mongoosejs.com/

Listagem 5.2: Modelo de dados no mongoose para persistir itens na coleção Questao.

```
1 mongoose.Schema({
2    titulo: {type: String},
3    descricao: {type: String},
4    dicas: {type: String},
5    nivelDificuldade: {type: Number},
6    casosTeste: {type: String}
7 });
```

Uma visão mais geral do modelo de dados pode ser observado no diagrama de classes da Figura 5.1. Nela é possível ver o relacionamento entre Usuário, Resposta e Questão. Cada resposta registrada guarda a identificação do usuário e da Questão resolvida para serem recuperadas posteriormente, além dos dados de medidas que são coletados durante o Processo de Resolução.

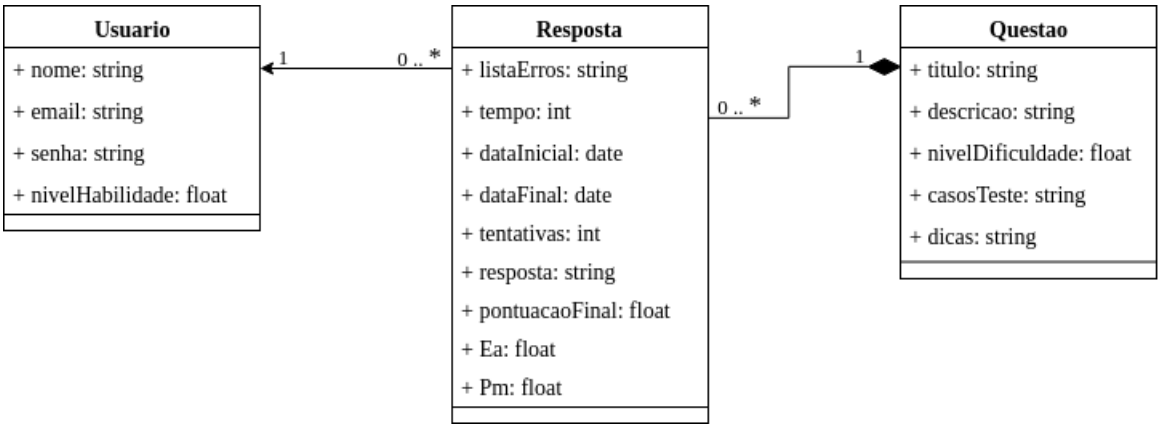

Figura 5.1: Diagrama de classes do Processo de Resolução: Cada Resposta registrada é relacionado a um Usuário e a uma Questão.

5.2 Coleta de dados e resultados

No início do Processo de Resolução, o tempo de resolução começa a ser medido, o usuário então envia uma solução, que é guardada e a primeira tentativa é realizada. Nesse momento o usuário recebe do Avaliador um *feefback*, indicando quais foram os pontos avaliados correta e incorretamente. Caso esteja correto, o usuário submete essa solução como resposta final e o processo termina. Caso sua solução não esteja correta, o *feedback* mostra quais casos de teste não foram aprovados. Um exemplo dessa situação é mostrado na Figura 5.2, onde o ponto de exclamação indica uma falha ou um defeito no código e o marcador verde indica que sua solução foi aprovada naquele caso de teste.

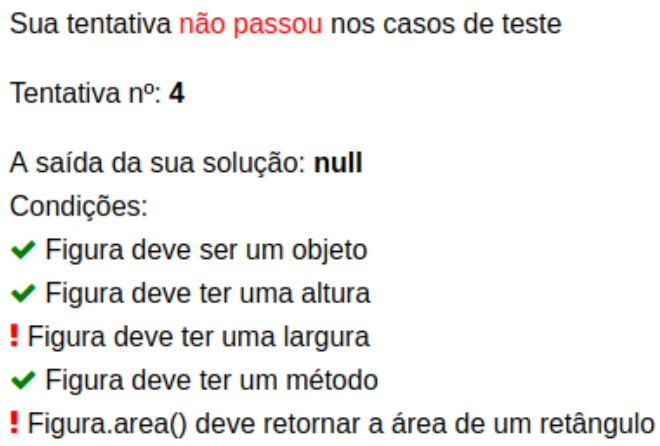

Figura 5.2: Visualização do *feedback* feito pelo Avaliador.

Os dados coletados são armazenados em um modelo de dados conforme Listagem 5.3, e os valores por questão podem ser vistos na Figura 5.3, que apresenta para cada questão o seu título, seu nível de dificuldade atual, a média da pontuação esperada, a média da pontuação de medidas, a média da pontuação final, a média de tempo gasto até a resposta correta, a média de tentativas, a quantidade de respostas corretas e a quantidade de processos interrompidos. As colunas de média apresentam também o desvio padrão calculado.

Listagem 5.3: Modelo de dados no mongoose para persistir itens na coleção Resposta.

```
1  mongoose.Schema({
2     idUsuario: {type: String},
3     idExercicio: {type: String},
4     resposta: {type: String},
5     listaErros: {type: String},
6     tempo: {type: Number},
7     tentativas: {type: Number},
8     dataInicial: {type: Date},
9     dataFinal: {type: Date},
10    Ea: {type: Number},
11    Pm: {type: Number},
12    pontuacaoFinal: {type: Number}
13 });
```

Os dados coletados também podem ser vistos para cada usuário junto com seu nível de habilidade atual, entretanto para preservar a identidade dos usuários, apenas sua identificação é informada. Os dados vistos por usuário podem ser vistos da Figura 5.4, que apresenta para todas as soluções de cada usuário, a média da pontuação esperada, a média da pontuação de medidas, a média da pontuação final, a média de tempo gasto até a

Título	Nvl Dif.	MEa	MPtm	Mpf	Mtempo	Mtent	Nresp	NCinc
Variáveis numéricas	1,375.40	0.56 (1.08)	0.13 (1.51)	4.98 (3.90)	24.08 (32.83)	1.64 (1.49)	25	11
Operação: Soma	1,457.66	0.49 (1.57)	0.32 (1.83)	3.08 (8.95)	75.50 (167.33)	2.06 (1.75)	16	10
Objetos e funções	1,474.36	0.42 (1.47)	0.41 (1.51)	2.85 (4.89)	98.00 (123.51)	1.89 (2.18)	9	9
Cálculo de Conta de Energia	1,478.59	0.40 (2.00)	0.33 (2.08)	4.28 (5.34)	72.60 (96.57)	2.40 (1.74)	5	21
Soma de vetor itens em um vetor	1,495.05	0.43 (2.07)	0.54 (1.99)	0.49 (5.31)	130.00 (150.39)	2.50 (1.57)	10	13
Operação: Soma em função	1,498.11	0.48 (1.96)	0.51 (2.07)	0.12 (12.57)	129.63 (247.94)	2.44 (2.24)	16	14
Busca pelo maior no vetor retornando objeto	1,520.54	0.40 (2.03)	0.81 (1.63)	-3.36 (6.55)	206.14 (210.75)	2.43 (1.76)	7	6
Objetos e Vetores	1,521.69	0.39 (4.11)	1.29 (3.21)	-10.85 (15.46)	321.50 (322.72)	4.50 (3.50)	2	9
Objetos e métodos	1,536.14	0.42 (6.15)	0.90 (5.69)	-5.16 (14.73)	197.00 (214.76)	6.57 (5.70)	7	17
Busca pelo maior no vetor	1,538.58	0.42 (0.91)	0.84 (0.81)	-4.29 (12.03)	232.11 (297.86)	1.33 (0.94)	9	14
Objetos e propriedades	1,550.00	0.42 (5.25)	0.93 (4.82)	-5.56 (17.49)	208.11 (269.51)	5.67 (6.46)	9	10
Variáveis texto	1,558.51	0.45 (5.24)	0.83 (4.91)	-4.50 (15.03)	183.62 (237.66)	5.69 (4.98)	13	12
Soma de vetor itens em um vetor com função	1,582.42	0.44 (3.10)	1.05 (2.69)	-7.96 (19.90)	271.91 (381.43)	3.55 (3.17)	11	12
Invertendo um vetor	1,627.40	0.37 (3.40)	1.14 (2.73)	-8.23 (16.81)	284.38 (340.01)	3.77 (3.35)	13	20
Contando itens em um vetor	1,631.84	0.38 (4.12)	1.61 (2.98)	-15.88 (23.45)	409.30 (442.93)	4.50 (4.50)	10	13
Condicionais	1,635.01	0.40 (3.24)	1.37 (2.45)	-12.27 (22.22)	358.45 (437.01)	3.64 (3.23)	11	27
Quadrado Mágico	1,923.47	0.25 (4.42)	5.16 (3.25)	-70.58 (91.71)	1,339.83 (1,580.90)	4.67 (2.56)	6	13

Figura 5.3: Visualização dos dados coletados, listados por questão.

resposta correta, a média de tentativas, a quantidade de respostas corretas e a quantidade de processos interrompidos. As colunas de média apresentam também o desvio padrão calculado.

A ordenação e classificação das questões por usuário pode ser vista na tela de listagem de questões. A Figura 5.5 mostra a listagem de questões para um usuário com nível de habilidade $\theta = 1506,70$. Já a Figura 5.6 mostra as mesmas questões para um usuário com habilidade $\theta = 1689,61$. Os dados presentes nessas imagens foram simulados para melhor apresentação do funcionamento da interface. Na frente de cada título das questões, a cor do marcador corresponde ao agrupamento em que a questão se encontra, sendo que o marcador de cor verde indica uma Questão adequada à habilidade daquele usuário; um marcador de cor vermelha indica uma Questão que é mais difícil para aquele usuário e um marcador na cor cinza indica que aquela Questão é mais fácil para aquele usuário.

Id do Usuario	NvlUser	MEa	MPtm	Mpf	Mtempo	Mtent	Nresp	NCinc
5b3230dc1bb44e0004fccd19	1,317.20	0.40 (3.41)	1.32 (3.55)	-22.85 (42.71)	325.38 (731.64)	3.81 (3.69)	16	4
5b30e33b8a047000044acc53	1,364.44	0.38 (5.62)	1.20 (4.90)	-18.50 (22.04)	278.23 (354.24)	6.00 (7.36)	13	5
5b31ae36e51c43000455c3fd	1,375.06	0.34 (4.84)	1.12 (4.21)	-14.70 (22.28)	259.94 (384.07)	5.18 (5.17)	17	11
5b313fe719a4f00004eea586	1,402.38	0.43 (3.30)	1.13 (2.72)	-17.75 (17.99)	290.73 (360.40)	3.73 (2.14)	11	29
5b2ee0613ebaa50004e95fe2	1,434.16	0.45 (1.80)	0.22 (2.04)	10.60 (4.14)	41.25 (50.27)	2.25 (1.09)	4	7
5b31a559e51c43000455c3ed	1,444.08	0.48 (2.92)	0.87 (2.57)	-11.18 (11.00)	222.40 (250.32)	3.40 (2.87)	10	17
5b2c08f92dfdde000441407a	1,450.07	0.48 (2.82)	0.72 (2.87)	-6.41 (21.43)	181.47 (418.66)	3.29 (2.11)	17	15
5b313cef19a4f00004eea584	1,454.34	0.56 (8.44)	3.29 (5.71)	-91.31 (54.66)	927.00 (918.00)	9.00 (0.00)	1	2
5b31ac69e51c43000455c3f8	1,465.59	0.48 (1.70)	0.65 (1.66)	-4.05 (10.19)	163.47 (229.80)	2.18 (1.46)	17	9
5b328b179761d60004387dd6	1,468.16	0.38 (0.62)	2.61 (1.61)	-63.68 (32.84)	731.00 (730.00)	1.00 (0.00)	1	0
5b31423319a4f00004eea589	1,468.34	0.48 (3.36)	0.85 (3.13)	-10.55 (17.39)	212.33 (318.66)	3.83 (3.93)	6	6
5b31a934e51c43000455c3f5	1,478.10	0.43 (2.19)	0.74 (2.00)	-5.47 (11.41)	186.00 (253.06)	2.63 (1.93)	8	3
5b3a31c150ade000045f5885	1,481.27	0.34 (2.66)	1.83 (1.17)	-37.45 (21.73)	480.00 (477.00)	3.00 (0.00)	1	0
5b32cab930824f0004279ac6	1,488.97	0.38 (0.62)	0.96 (0.82)	-11.03 (14.79)	265.00 (352.78)	1.00 (0.00)	2	2
5b2ee09b3ebaa50004e95fe5	1,495.78	0.49 (1.71)	0.56 (1.66)	-1.69 (5.22)	145.60 (159.83)	2.20 (1.47)	5	9
5b31d2c5bf5ed30004a9f8e8	1,500.00	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0	3
5b32eae430824f0004279ada	1,500.00	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0	1
5b2ee1323ebaa50004e95fe7	1,500.00	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0	1
5b32c52830824f0004279ab8	1,500.00	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0	9
5b2fe4a029f6c40004838c8e	1,500.00	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0	14

Figura 5.4: Visualização dos dados coletados, listados por usuário.

5.3 Coleta e análise de dados reais

O Coding Training foi implantado no Heroku[9], uma plataforma na nuvem que fornece *conteiners* para aplicações como serviço. A plataforma pode ser acessada ⟨https://guarded-refuge-80906.herokuapp.com⟩, onde é possível se cadastrar e responder as questões.

Para que fosse possível coletar dados reais, 18 questões foram disponibilizadas na plataforma e alunos das disciplinas DCC121 - Laboratório de Programação Web (segundo período), DCC192 - Laboratório de Programação de Sistemas Web (sétimo período) e DCC104 - Seminários em Computação (eletiva, períodos mistos), foram convidados a utilizar a plataforma no período de 22/06/2018 à 03/07/2018. Neste tempo, 28 usuários se cadastraram, 158 respostas corretas e 513 tentativas foram registradas.

Além da forma tabular já apresentada, também é possível visualizar os dados por questões e usuários através de gráficos, como na figuras 5.7 e 5.8 e uma breve análise dos dados coletados mostra que a medida em que os usuários respondem as questões, suas habilidades começam a ser traçadas.

[9]https://www.heroku.com

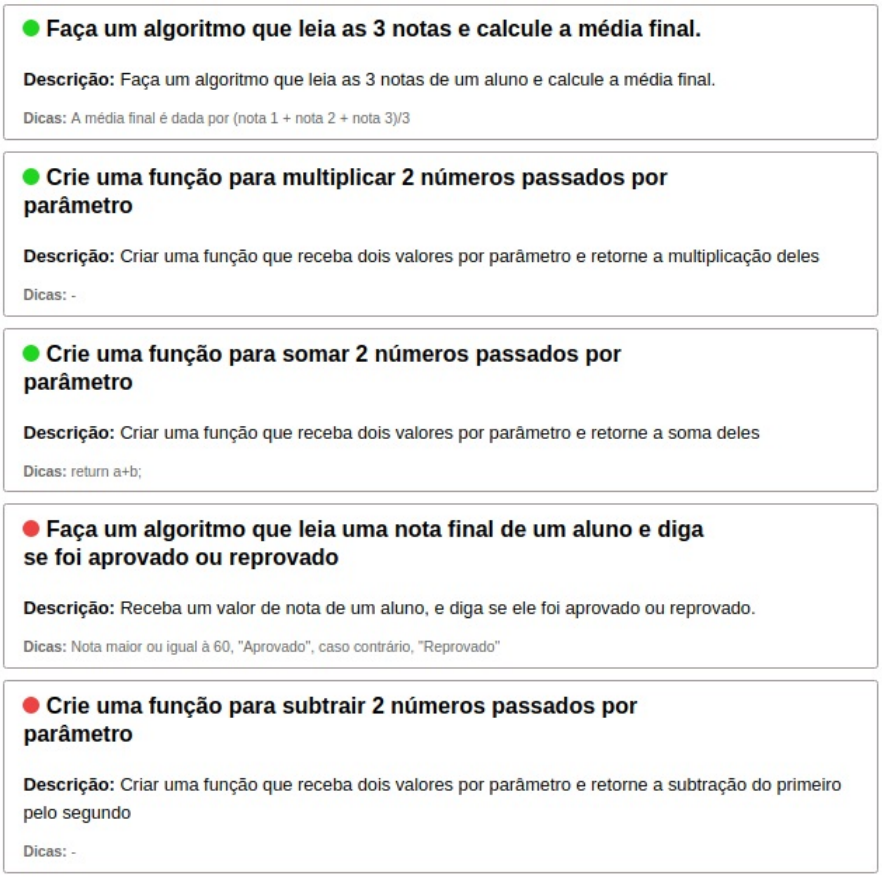

Figura 5.5: Listagem de Questões de um usuário com habilidade $\theta = 1506,70$. As Questões são agrupadas de acordo com nível de dificuldade, e em cada grupo, em ordem decrescente pela probabilidade do usuário acertá-la.

Tomando como exemplo 2 usuários, a Figura 5.9 mostra o seu desempenho após responderem mais de 15 questões corretas. A linha na cor azul representa um usuário que inicialmente teve certo grau de dificuldade, mas a medida que foi respondendo mais questões, sua habilidade aumentou. Já a linha na cor laranja, representa um cenário diferente, um usuário que inicialmente conseguiu aumentar um pouco sua habilidade, porém começou a sentir mais dificuldade ao responder o restante das questões.

Uma análise semelhante pode ser feita para algumas questões. Como pode ser visto na Figura 5.10, a questão representada pela linha azul, teve sua dificuldade reduzida quase que constantemente. Já a questão representada pela cor lilás, se manteve, aproximadamente com o nível inicial. A questão representada em amarelo, se destacou por subir seu nível de dificuldade rapidamente.

Através desses dados, é possível concluir que algumas questões, que exigem muito tempo ou que tiveram várias tentativas de resposta, aumentaram seu nível de dificuldade,

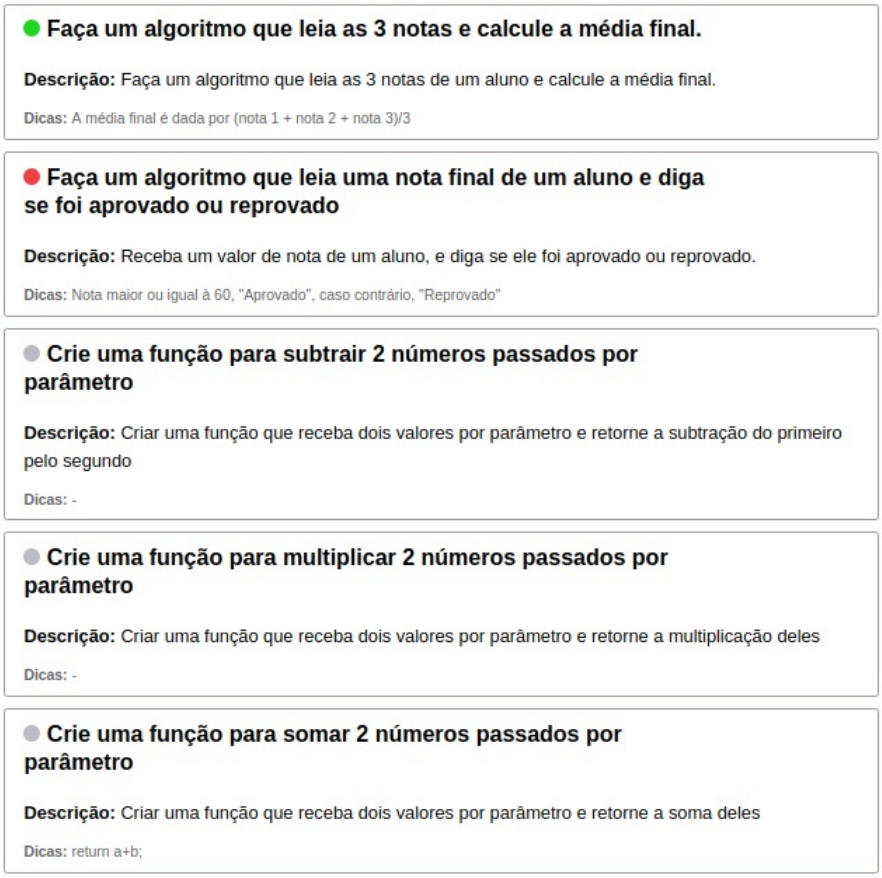

Figura 5.6: Listagem de Questões de um usuário com habilidade $\theta = 1689,61$. As Questões são agrupadas de acordo com nível de dificuldade, e em cada grupo, ordenadas de forma decrescente pela probabilidade do usuário acertá-la.

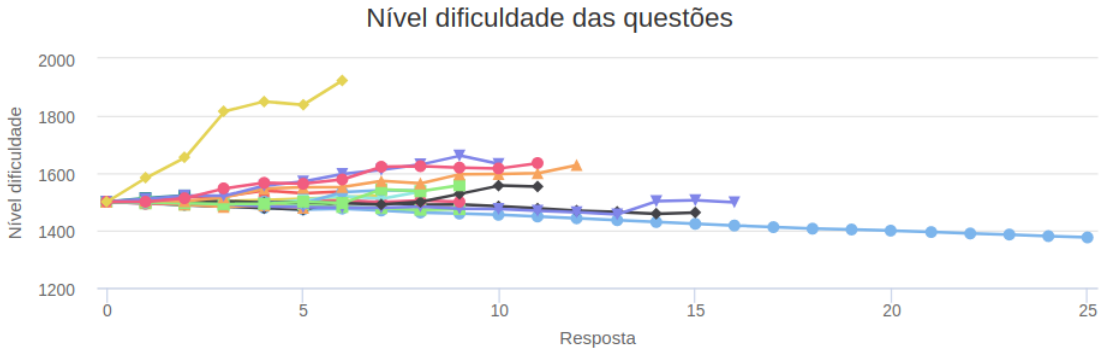

Figura 5.7: Gráfico dos dados coletados, mostrados por questão.

enquanto questões que foram resolvidas rapidamente e em poucas tentativas, têm o seu nível de dificuldade reduzido. Dessa forma, é possível ver que na listagem de questões, as informações de dificuldade para cada usuário começam a se estabilizar, ou seja, as questões são classificadas de acordo com o nível mais adequado para cada usuário.

Embora os dados evidenciem o funcionamento da arquitetura proposta, estudos com maior rigor e mais dados precisariam ser realizados para que se possa extrair in-

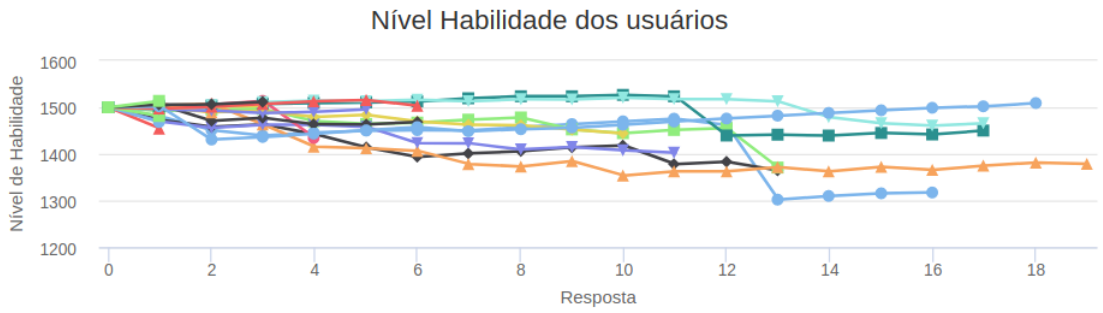

Figura 5.8: Gráfico dos dados coletados, mostrados por usuário.

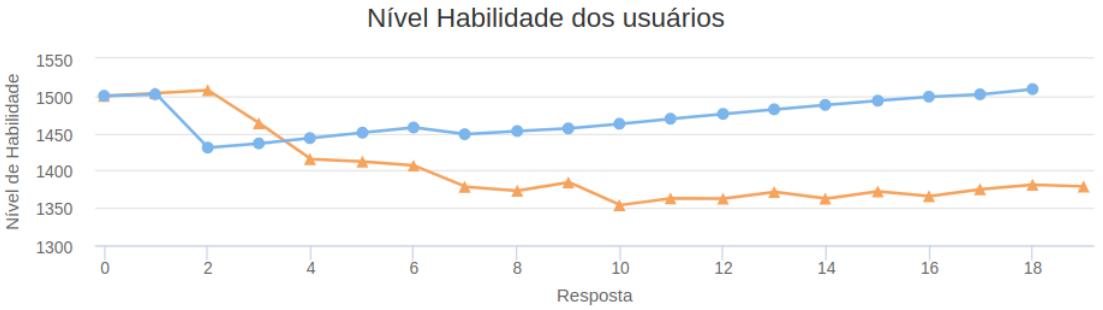

Figura 5.9: Gráfico que compara o rendimento de dois usuários.

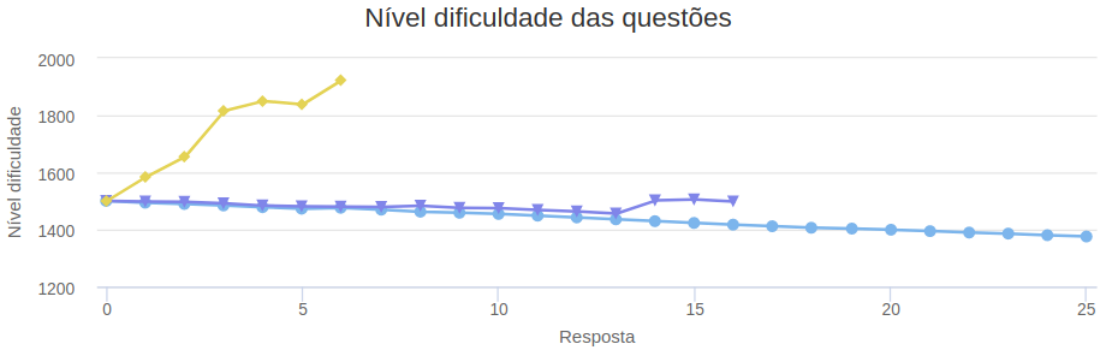

Figura 5.10: Gráfico que compara o comportamento do nível de dificuldade de algumas questões.

formações mais precisas, visto que, enquanto esses dados foram coletados, mudanças de interface e funcionamento foram realizadas. Além disso, as questões precisaram ser validadas por usuários reais, para garantir que as medidas fossem coletadas corretamente. Entretanto, o escopo deste trabalho não contempla uma análise profunda dos dados coletados, mas sim propor uma arquitetura e garantir o funcionamento do AVA.

5.4 Considerações parciais

Este Capítulo apresentou uma breve descrição das tecnologias e ferramentas utilizadas para o desenvolvimento do Coding Training. Além disso, foi evidenciado alguns dos resultados obtidos, como medidas coletadas para algumas questões e usuários.

Observou-se que o uso de uma única tecnologia (JavaScript) em todo o projeto, conhecido como como *full stack* JavaScript, facilita o processo de desenvolvimento pois o fato de se utilizar uma única linguagem no *front-end*, *back-end* e na notação do banco de dados, torna a curva de aprendizado mais curta e o código mais legível e compreensível. Além disso, a comunicação entre ambas as partes é mais eficiente, sem a necessidade de conversão de objetos JSON, visto que os objetos de dados são os mesmos em todos os nível da aplicação.

Como desvantagem, o uso linguagem JavaScript no servidor, embora com uma comunidade grande, ainda é recente e pode ser mais verbosa comparada a outras, ou seja, é necessário escrever mais código para realizar uma mesma funcionalidade.

Além disso, uma breve análise dos dados reais coletados é apresentada mostrando que o Coding Training apresenta os conceitos propostos pela arquitetura e realiza a coleta de medidas. Ainda não é possível tirar conclusões mais profundas sobre esses dados, pois além da possibilidade dos dados coletados possuírem ruídos, a quantidade de questões e usuários ainda é pequena.

6 Considerações finais

Este Capítulo apresenta uma visão geral deste trabalho composta da avaliação dos objetivos, um conjunto de considerações finais que foram obtidas durante todo o desenvolvimento e sugestões para trabalhos futuros.

6.1 Avaliação dos objetivos

As dificuldades encontradas por alunos ao iniciar os estudos em programação, como pensamento lógico e uma abstração dos problemas, distanciamento entre professores e alunos, desnivelamento da turma e dificuldade na forma de avaliação, fazem com que, geralmente, se observe um alto índice de reprovação nas disciplinas iniciais de programação.

Entre as propostas para amenizar esses problemas, está o uso AVA's especializados em programação em conjunto com o ensino tradicional. Neste trabalho, foi feito um levantamento de conceitos e ferramentas que são utilizadas para aprendizagem de linguagens de programação. Adicionalmente, uma breve comparação foi realizada entre essas ferramentas, destacando as principais características de cada uma.

Após essa comparação, uma proposta de arquitetura foi apresentada, utilizando os conceitos pesquisados e combinando-os, para a criação de um ambiente específico para treinamento em programação. Essa arquitetura foi implementada em um protótipo, o Coding Training, e divulgada para um grupo de alunos para evidenciar o seu funcionamento. Os detalhes de implementação e dados colhidos são descritos neste trabalho.

O ponto central desse ambiente é realizar um pareamento automático, no qual os alunos são incentivados a responder questões que condizem com seu nível de habilidade e não definidas arbitrariamente por outra pessoa. À medida que os usuários interagem com a plataforma e tentam resolver as questões, sua habilidade e a dificuldade da questão convergem para um valor específico. Dessa forma, o ambiente se adapta à realidade individual de cada usuário, fazendo com que se sintam mais confortáveis em continuar o seu progresso em seu próprio ritmo.

Outro aspecto é a capacidade de coleta de dados durante o processo de resolução ao longo do uso do AVA. Com esses dados, é possível obter informações que podem permitir aos professores ou coordenadores de disciplinas de massa terem uma maior facilidade de acompanhar o aprendizado de cada aluno associado a cada conteúdo. O protótipo evidencia isso realizando as medidas de tempo, quantidade de tentativas de solução e erros cometidos, por cada usuário e para cada questão.

Estudos subsequentes poderão ser realizados com uma massa maior de dados, a fim de se obter indícios de quais conteúdos merecem mais atenção ou serem abordados de outra forma.

6.2 Limitações e trabalhos futuros

O Coding Training foi capaz de evidenciar que foram coletados, para cada usuário, dados como tempo de solução, quantidade de tentativas de solução e quais erros cometidos para cada questão respondida. O Coding Training ainda é capaz de identificar quais são as questões mais indicadas para um usuário resolver de acordo com a sua habilidade.

Contudo, com os dados coletados, ainda não é possível determinar qual a dificuldade específica de cada aluno, como por exemplo, dificuldade em estruturas condicionais ou de repetição, nem características individuais, como alunos que demoram muito para resolver uma questão e são muito assertivos ou alunos que façam várias tentativas até chegar à resposta correta, ou seja, não é possível identificar se um aluno costuma resolver um exercício atráves de tentativa e erro ou se ele realmente tem dificuldade naquela questão. Além disso, também não é possível impedir que o aluno trapaceie durante uma solução.

Como trabalhos futuros, espera-se poder utilizar o Coding Training em disciplinas reais que envolvam o ensino de linguagens de programação, em especial, JavaScript para obter resultados em maior escala e compará-los a uma turma que não tenha utilizado a plataforma.

Adicionalmente, espera-se acrescentar outras linguagens de programação para treinamento. Na Universidade Federal de Juiz de Fora (UFJF), uma das primeiras linguagens de programação a ser ensinada é a linguagem C. Com isso, seria interessante

que esta fosse a próxima linguagem para treinamento a ser introduzida no Coding Training, pois vários alunos poderiam se beneficiar do uso, e informações mais precisas seriam coletadas.

Também é possível aplicar outras formas de elementos de jogos além do pareamento automático, como a premiação por selos para desafios propostos ou ranqueamento de usuários. Entretanto, espera-se ainda focar no progresso individual.

O Coding Training ainda carece de melhorias na geração de relatórios específicos dos alunos, que descrevem o seu treinamento e quantificam o seu aprendizado ao longo do uso do sistema. Para isso, seria necessário a criação de categorias e indicadores das questões, além de um banco de questões maior, para buscar em quais assuntos eles possuem mais dificuldade ou onde o ensino precisa ser mais focado. Para tal é necessário que o sistema de habilidades seja separado por cada grupo de questões ou categorias que se deseja avaliar.

A arquitetura apresentada, pode ainda ser adaptada para outras áreas de ensino, além da tecnologia da informação, como matemática, física ou ciências biológicas. Dado à característica do processo de resolução ser modular, ao invés de escrever um algoritmo, o aluno usaria as notações específicas da disciplina para resolver questões que seriam tratadas por avaliadores específicos e receberiam *feedbacks* mais interativos.

Além disso, com uma base de dados mais extensa, técnicas de aprendizado de máquina poderiam ser aplicadas a fim de reconhecer padrões de comportamento dos usuários ou das questões e realizar previsões, ajustar melhor o fluxo de treinamento e recomendação, ou deixar o ambiente mais adaptativo a cada usuário ou classificando automaticamente as questões em mais categorias além de “Fácil”,“Adequado” ou “Difícil”.

Bibliografia

ALEXANDRE, J. W. C. et al. Análise do número de categorias da escala de likert aplicada à gestão pela qualidade total através da teoria da resposta ao item. *Encontro Nacional De Engenharia De Produçao*, v. 23, p. 1–20, 2003.

ALVES, F. P.; JAQUES, P. Um ambiente virtual com feedback personalizado para apoio a disciplinas de programação. In: *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*. [S.l.: s.n.], 2014. v. 3, n. 1, p. 51.

BELLONI, M. L. *Educação a distância*. [S.l.]: Autores Associados, 2003.

CAMPOS, C. P. D.; FERREIRA, C. E. Boca: um sistema de apoio a competições de programação. In: *Workshop de Educação em Computação*. [S.l.: s.n.], 2004. p. 885–895.

CORMEN, T. H. et al. Algoritmos: teoria e prática. *Editora Campus*, v. 2, 2002.

DEJARNETT, P. et al. *Code Wars*. 2017. ⟨https://www.codewars.com/⟩. [Online; Acessado em 26 de junho de 2018].

DELAMARO, M.; JINO, M.; MALDONADO, J. *Introdução ao teste de software*. [S.l.]: Elsevier Brasil, 2017.

DUARTE, A.; MOREIRA, H.; MELLO, T. S. Competitividade como fator motivacional para o estudo de computaçao. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2010. v. 1, n. 1.

FERRARI, F.; CECHINEL, C. Introdução a algoritmos e programação. *Bagé: Universidade Federal do Pampa*, 2008.

FIGUEIREDO, K. da S. et al. Uma abordagem gamificada para o ensino de programação orientada a objetos. 2008.

FORBELLONE, A. L. V.; EBERSPÄCHER, H. F. *Lógica de programação: a construção de algoritmos e estruturas de dados*. [S.l.]: Makron Books, 1993. v. 3.

FRANÇA, A. B.; SOARES, J. M. Sistema de apoio a atividades de laboratório de programação via moodle com suporte ao balanceamento de carga. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2011. v. 1, n. 1.

GLICKMAN, M. E.; JONES, A. C. Rating the chess rating system. *CHANCE-BERLIN THEN NEW YORK-*, SPRINGER INTERNATIONAL, v. 12, p. 21–28, 1999.

GUDWIN, R. R. Linguagens de programação. *Campinas: DCA/FEEC/UNICAMP*, 1997.

LEITE, R. M. Uma proposta para o ensino de programação de computadores na educação básica. 2015.

LOUDEN, K. C. et al. *Programming languages: principles and practices*. [S.l.]: Cengage Learning, 2011.

MILLIGAN, C. Delivering staff and professional development using virtual learning environments. *The Role of Virtual Learning Environments in the Online Delivery of Staff Development. Institute for Computer Based Learning, Heriot-Watt University, Riccarton, Edinburgh EH14-4AS*, 1999.

MOORE, M.; KEARSLEY, G.; DISTÂNCIA, E. a. Uma visão integrada. *Tradução por Roberto Galman. São Paulo: Thomson Learning*, 2007.

MOREIRA, M. P.; FAVERO, E. L. Um ambiente para ensino de programação com feedback automático de exercícios. In: *Workshop sobre Educação em Computação (WEI 2009)*. [S.l.: s.n.], 2009. v. 17.

O'SULLIVAN, S.; SAL. *Khan Academy*. 2005. ⟨https://pt.khanacademy.org/⟩. [Online; Acessado em 26 de junho de 2018].

PARTOVI, A.; PARTOVI, H. *Code.org*. 2013. ⟨https://code.org/⟩. [Online; Acessado em 26 de junho de 2018].

PEREIRA, A. T. C.; SCHMITT, V.; DIAS, M. Ambientes virtuais de aprendizagem. *AVA-Ambientes Virtuais de Aprendizagem em Diferentes Contextos. Rio de Janeiro: Editora Ciência Moderna Ltda*, p. 4–22, 2007.

POLLACK, G. *Code School*. 2011. ⟨https://www.codeschool.com/⟩. [Online; Acessado em 26 de junho de 2018].

RIBEIRO, U. E.; ISHITANI, L. Um ambiente virtual de aprendizagem colaborativo para o ensino de algoritmos. *Anais do XIX Simpósio Brasileiro de Informática na Educaç ao*, 2008.

ROCHA, A. R. C. d. et al. Qualidade de software: teoria e prática. *São Paulo: Prenttice Hall*, 2001.

ROCHA, P. S. et al. Ensino e aprendizagem de programação: análise da aplicação de proposta metodológica baseada no sistema personalizado de ensino. *RENOTE*, v. 8, n. 3, 2010.

ROSA, M. M. da; GIRAFFA, L. M. M. O ensino de programação de computadores e ead: uma parceria possível. In: *17 Congresso Internacional da ABED, 2011, Brasil*. [S.l.: s.n.], 2011.

SALES, A. B. de; JR, E. C.; SALES, M. B. de. Utilização de problemas da maratona de competição de programação e juízes eletrônicos como estratégia de ensino em um curso de graduação em engenharia de software. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2016. v. 27, n. 1, p. 210.

SANTOS, J. C.; RIBEIRO, A. R. Jonline: proposta preliminar de um juiz online didático para o ensino de programação. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2011. v. 1, n. 1.

SANTOS, R. P. dos; COSTA, H. A. X. Análise de metodologias e ambientes de ensino para algoritmos, estruturas de dados e programação aos iniciantes em computação e informática. *INFOCOMP Journal of Computer Science*, v. 5, n. 1, p. 41–50, 2006.

SILVA, A. R. L. da et al. *Gamificação na Educação*. [S.l.]: Pimenta Cultural, 2014.

VALLE, R. da C. Teoria de resposta ao item. *Estudos em avaliação educacional*, n. 21, p. 7–92, 2000.

ZICHERMANN, G.; CUNNINGHAM, C. *Gamification by design: Implementing game mechanics in web and mobile apps*. [S.l.]: "O'Reilly Media, Inc.", 2011.