TesterDS: uma maneira fácil e estimulante para aprender Estruturas de Dados

Iago Fabiano S. de Souza, Helder L. Bertoldo dos Reis, Vânia de Oliveira Neves

[1]Departamento de Ciência da Computação – Universidade Federal de Juiz de Fora - UFJF
Juiz de Fora/MG

{helder.bertoldo, serpa, vania}@ice.ufjf.br

Abstract. *Some courses that are considered more difficult, such as Data Structures, have facing a high failure rate. Many students have difficulties for even starting to develop the proposed structure. Furthermore, an effort after class is required to achieve a better understanding of the content addressed in the classroom. However, to keep the students motivated to accomplish this task becomes a hard work for the teacher day by day. In this sense, this work presents TesterDS, a tool that has the proposal of facilitating and stimulating the learning of Data Structures in order to obtain a better understanding of the course by the students. For this, TesterDS uses concepts of Testing Driven Development and Gamification.*

Resumo. *Algumas disciplinas consideradas mais difíceis, como Estruturas de Dados, possuem altos índices de reprovação. Muitos estudantes possuem dificuldades para até para iniciar o desenvolvimento da estrutura proposta. Além disso, é necessário um reforço extra aula para conseguir uma melhor compreensão do conteúdo abordado em sala de aula, no entanto, manter os alunos motivados a realizar essa tarefa se torna a cada dia um trabalho árduo para o docente. Nesse sentido, este trabalho apresenta a ferramenta TesterDS que possui a proposta de facilitar e estimular o aprendizado de Estruturas de Dados a fim de que os alunos tenham um melhor aproveitamento da disciplina. Para tal, TesterDS utiliza conceitos de Desenvolvimento Dirigido a Testes e Gamificação.*

1. Introdução

Disciplinas voltadas a prática de programação, ofertados nos cursos relacionados a Ciência da Computação, possuem altos índices de reprovação e de evasão. Várias pesquisas já foram conduzidas a fim de identificar os principais problemas e desafios enfrentados pelos estudantes, principalmente os iniciantes. Segundo [Gomes et al. 2008], a principal dificuldade dos alunos em programar se deve à incapacidade de conceber algoritmos que, por sua vez, tem sua causa na incapacidade de resolver problemas. Outro aspecto apontado se deve a falta de motivação uma vez que disciplinas de programação, principalmente as mais avançadas, tem a reputação de serem difíceis e isso é passado de aluno em aluno. Com isso, estudantes que não tem uma motivação intrínseca, dificilmente serão bem sucedidos [Gomes et al. 2008].

Para amenizar tais problemas, várias abordagens têm sido propostas e pode-se observar uma tendência principalmente com relação a utilização de jogos e o desenvolvimento de ambientes pedagógicos para o ensino e aprendizagem de programação

DOI: 10.5753/cbie.sbie.2018.864

[Souza et al. 2016]. Todavia, a maior parte dessas propostas são direcionadas a alunos principiantes ou crianças e pouco se tem discutido para alunos de cursos mais avançados de programação, como os cursos de Estruturas de Dados. Nesse sentido, este artigo apresenta um jogo que consiste de fases derivadas a partir de técnicas de Desenvolvimento Dirigido a Testes (TDD, do inglês *Test Driven Development*). A fim de manter o *flow* do aluno, o jogo apresenta elementos como utilização de moedas, chamadas TDCoins.

Este trabalho está organizado de forma: na Seção 2 é apresentada a fundamentação teórica na qual o artigo foi baseado; na Seção 3 são apresentados os trabalhos relacionados; na Seção 4 é discutida a abordagem na qual a ferramenta se baseia; na Seção 5 o TesterDS é discutido em maiores detalhes e, por fim, na Seção 6 são feitas as considerações finais e trabalhos futuros.

2. Fundamentação Teórica

O objetivo desta seção é apresentar os conceitos teóricos que serão utilizados no desenvolvimento deste trabalho.

2.1. Gamificação

De acordo com [Deterding et al. 2011], gamificação é o uso de elementos de projeto de jogos em contextos não lúdicos. Essa abordagem tem se tornado tendência na educação pela sua maior capacidade de interatividade, em relação aos métodos tradicionais de ensino, possibilitando um envolvimento maior dos estudantes, semelhante ao que se é visto em jogadores de video-games e, com isso, pode-se ampliar ainda mais a criatividade desses alunos em busca de soluções. Mecânicas, dinâmicas e estratégias são elementos de jogos que, ao serem utilizados para resolver um determinado problema, provocam a aprendizagem. A mecânica envolve as regras e os benefícios compostos pelos jogos no sentido de torná-lo desafiador, gratificante, divertido, etc. [Kapp 2012, Deterding et al. 2011].

2.2. Teoria do *flow*

O envolvimento de elementos de jogos em ferramentas educacionais possibilita que estudantes abstraiam-se do tempo e do espaço, atingindo um estado chamado de *flow*, descrito por [Csikszentmihalyi 1991] em seu trabalho, cujo objetivo era identificar o que leva um indivíduo a alcançar o estado de felicidade. Quando a consciência desse indivíduo está em harmonia e realiza uma atividade por prazer e vontade própria, diz-se que esse indivíduo está em estado de *flow*. Três condições precisam ser satisfeitas para que esse estado seja atingido: i) metas claras a fim de direcionar o comportamento do indivíduo; ii) *feedback* rápido e direto, uma vez que o retorno rápido possibilita prosseguir com a atividade de forma prazerosa; iii) equilíbrio entre as habilidades do indivíduo e os desafios do jogo. Desafios e habilidades são as dimensões mais importantes, conforme mostra o diagrama da Figura 1. Se a habilidade de um indivíduo for maior que o desafio, pode provocar um estado de tédio. No entanto, se o desafio for superior às capacidades do indivíduo, tem-se a sensação de ansiedade. O *flow*, entretanto, deriva do equilíbrio dessa relação.

2.3. TDD - Desenvolvimento Dirigido a Testes

Test Driven Development (TDD), em português Desenvolvimento Guiado por Testes é uma abordagem de desenvolvimento em que o código é obtido através de uma especificação dos casos de teste. Uma funcionalidade é incrementada por meio se sucessivos

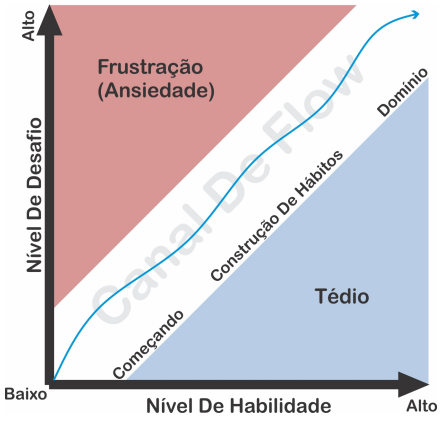

Figura 1. Diagrama de *flow*, adaptado de [Csikszentmihalyi 1991]

acréscimos, iniciando do mais simples. Assim, primeiramente é identificada um acréscimo do requisito que deverá ser codificada de maneira sucinta. Em seguida, é escrito um caso de teste para essa funcionalidade, que é implementado através de uma ferramenta automatizada. Dessa forma, a ferramenta é capaz de mostrar se o requisito pode ou não passar nos testes, informando o status. Em seguida, é realizada a execução desses casos de teste que, como o código referente a ele ainda não foi implementado, a ferramenta deverá acusar falha. Dessa maneira, o desenvolvedor deverá implementar um código simples, com a intenção apenas que esse código passe nos casos de teste. Em seguida, o código poderá ser refatorado e o ciclo se repete. A Figura 2 apresenta um diagrama do processo de desenvolvimento utilizando TDD.

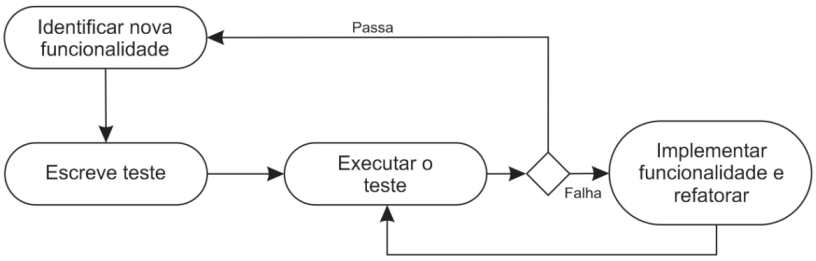

Figura 2. Desenvolvimento dirigido a testes, retirado de [?]

3. Trabalhos Relacionados

Há vários trabalhos que relatam a utilização de ferramentas e jogos computacionais no ensino de programação, mas a maior parte deles está preocupada com alunos iniciantes ou crianças [Silva et al. 2015, Amaral et al. 2017, Nunes et al. 2017, Lopes et al. 2017]. O trabalho de [Gomes et al. 2008] apresenta uma descrição e análise das principais ferramentas nesse contexto, destacando-se o **CodeCombat** e o **CodinGame**. O **CodeCombat** considera perfis de jogadores predadores, realizadores, exploradores e socializadores a fim de manter os alunos motivados [Silva et al. 2015]. A proposta do jogo se baseia na abordagem *Flow* e oferece um ambiente imersivo e divertido apresentando elementos de motivação e iteração diferentes para cada perfil de jogador [Silva et al. 2015]. O **Codin-**

Game tem como objetivo treinar os conceitos básicos de programação e oferece recursos de *chat* entre os jogadores e *debug*.

Pelo melhor do conhecimento, não foi encontrado na literatura jogos que utilizam conceitos de TDD e Teste de Software para o desenvolvimento de algoritmos avançados. No entanto, há algumas propostas como o de [Camara and Silva 2016] e de [de Souza et al. 2015] em que a técnica de teste caixa branca é utilizada para apoiar o ensino de programação, mas não possui recursos de gamificação. [Camara and Silva 2016] adicionaram um passo a mais no passo de refatoração do TDD, para que além de realizar a refatoração do código, o estudante também crie casos de teste baseado na cobertura dos critérios do teste estrutural. No entanto, nesse trabalho os estudantes devem aprender os conceitos de teste de software e de TDD para conseguir aplicar a técnica e a inexperiência deles pode levar ao fracasso da abordagem. A ferramenta ProgTest proposta por [Silva et al. 2015] tem como princípio auxiliar tanto o ensino de programação quanto técnicas básicas de teste de software paralelamente [de Souza et al. 2015].

4. Método

A abordagem baseia-se em dois conceitos: TDD e Teste Estrutural de Software. Com a combinação dessas técnicas pretende facilitar a implementação de estruturas mais avançadas uma vez que possibilitará dividir o algoritmo em problemas menores e mais simples de implementar. A aplicação é realizada em duas etapas: na primeira, o professor segue o processo de TDD para implementar a estrutura proposta e os casos de teste. Ao seguir o processo, o professor identifica as fases do jogo que derivará os níveis e subníveis. Para cada um dos níveis, será informado o código esperado da solução e os casos de teste que o algoritmo do aluno deverá passar. Vale observar que como cabe ao professor a configuração das fases, ele poderá, dependendo do desempenho da turma, criar poucos níveis, tornando o jogo mais difícil ou vários níveis, deixando o jogo mais fácil tentando, dessa maneira, manter o estado de *flow* dos alunos.

Na segunda fase, o aluno segue os passos identificados pelo professor para escrever sua implementação e utiliza os casos de teste para verificar se ela está correta. Dessa forma, o aluno será capaz de desenvolver a estrutura proposta iniciando por subníveis simples e de fácil implementação e que vão se tornando mais complexos à medida que o estudante avança.

A Figura 3 apresenta o esquema de implementação na perspectiva do aluno. Um nível corresponde a uma estrutura de dados a ser implementada. Ao iniciar o desenvolvimento dessa estrutura, o aluno terá que passar por subníveis identificados pelo professor na fase anterior. Para cada subnível, o aluno deverá escrever um código que passe pelos casos de teste implementados pelo professor para aquele subnível. Terminado o desenvolvimento, ele submete o código para que seja feita a instrumentação deste e é verificado se passou ou não nos casos de testes previamente propostos pelo professor. Caso o retorno seja positivo, compara-se a complexidade do código enviado pelo aluno com o do código-base do professor, utilizando a complexidade ciclomática definida por [McCabe 1976]. Se o código do aluno for mais complexo que o do professor, o aluno permanecerá no subnível para refatorar o código até obter a qualidade adequada. Caso contrário, o aluno avança para o subnível seguinte. Alternativamente, o aluno poderá usar suas moedas, chamadas no jogo de TDCoins, para comprar dicas de como melhorar a qualidade de seu código ou

até mesmo para avançar para o próximo nível. Nessa etapa também é gerado um Grafo de Fluxo de Controle (GCF) do código do estudante, que poderá observar quais requisitos do teste estrutural foram cobertos pelos casos de teste fornecidos. O ciclo segue dessa forma até concluir todos os subníveis e, consequentemente, tendo desenvolvido seu algoritmo de forma correta e com qualidade adequada. A cada novo subnível, são realizados testes de regressão e, se falharem, aluno não poderá avançar de fase.

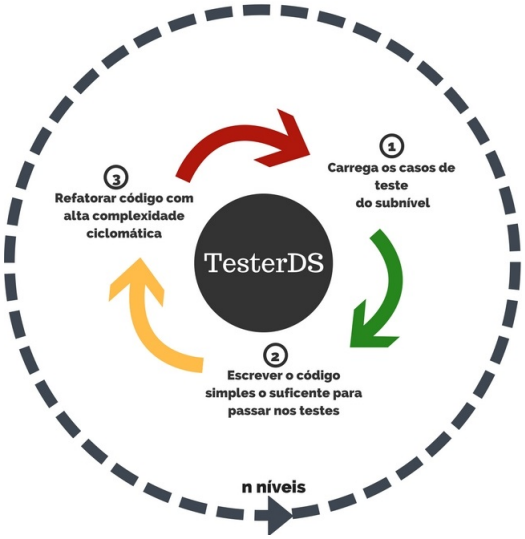

Figura 3. Esquema de TDD na perspectiva do aluno

5. TesterDS

A fim de oferecer suporte a abordagem proposta na Seção 4, está sendo desenvolvido o jogo TesterDS que tem como objetivo simplificar e estimular o aprendizado de disciplinas de programação, principalmente as mais avançadas como Estruturas de Dados.

Os *stakeholders* dos jogos são o professor e o aluno. O **professor** é capaz de efetuar o login no jogo; gerenciar turmas; gerenciar níveis, referente a uma determinada estrutura de dados, e subníveis, referente as fases necessárias para a completa implementação dessa estrutura. O gerenciamento dos subníveis ainda incluem o envio de código-fonte base utilizado na comparação da solução proposta, envio de casos de teste que serão utilizados para verificar se o programa do aluno está correto, configuração de recompensas como pontuação, TDCoins, compra de dicas, etc. O **aluno** é capaz de efetuar o login no jogo, verificar sua pontuação e as fases disponíveis para ele, de acordo com a turma que ele pertence. Com isso, ele poderá participar da implementação de estruturas definidas nos níveis e habilitar os próximos níveis de acordo com a pontuação obtida. As Figuras 4 e 5 apresentam, respectivamente a tela de registro e a tela de implementação do jogo.

Ao iniciar cada fase, uma breve descrição da estrutura de dados a ser implementada é apresentada. Os nomes de cada classe e métodos a serem implementados também são descritos para que os casos de teste sejam executados com sucesso, conforme pode ser visto na Figura 5, que apresenta o exemplo de implementação de uma árvore vermelha e preto. Com essas informações o aluno deve desenvolver um código, no qual atenda aos requisitos propostos. Ao clicar no botão enviar, o jogo verificar se o código do aluno

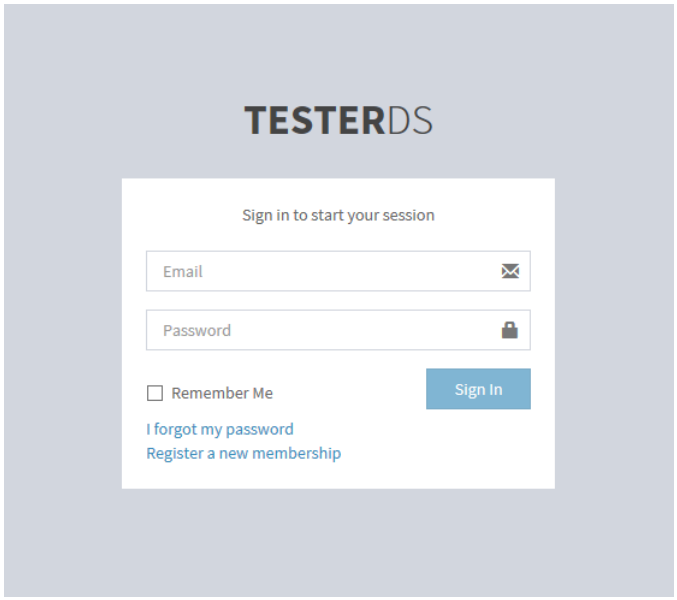

Figura 4. Tela inicial

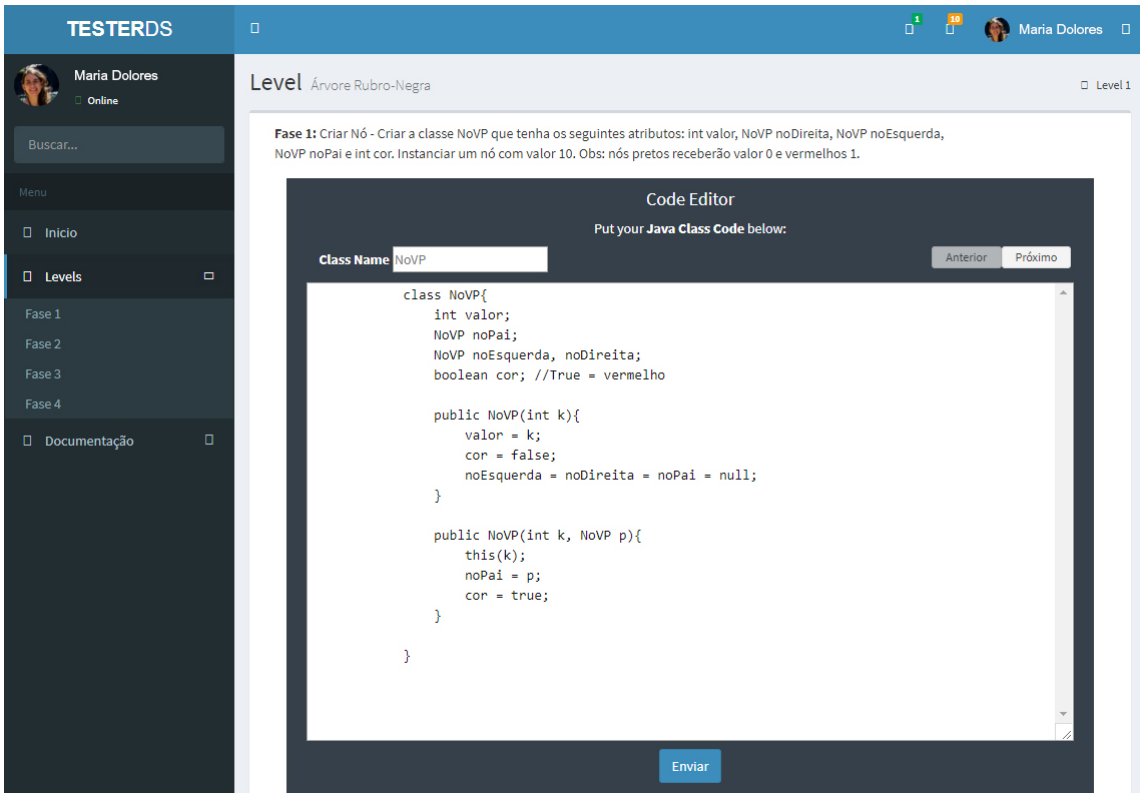

Figura 5. Tela de implementação

passa nos casos de teste definidos previamente para esse subnível pelo professor. Caso ele tenha dificuldades, ele poderá comprar ajudas, usando as TDcoins (moeda do jogo), que deverão ser configuradas previamente pelo professor para aquele subnível. O código também é avaliado com relação a sua complexidade. Caso a complexidade seja menor ou igual ao código base, o aluno segue para próxima etapa, e recebe uma compensação em

pontos e TDCoins proporcional a porcentagem da diferença da complexidade ciclomática do seu código.

Entre os elementos de gamificação considerados a fim de manter a experiência do jogador mais atrativa e manter o estudante no estado de *flow* estão:

- **Pontos:** que vão ser distribuídos de acordo com o progresso no jogo, onde pode se obter pontuação máxima ao alcançar uma complexidade ciclomática igual ou menor de acordo com o código base, ou pontuação mínima, que seria uma complexidade ciclomática maior do que o código base, porém abaixo do mínimo definido pelo professor.
- **Moedas:** apelidadas de TDCoins, são responsáveis por trazer elementos de competição aos jogadores. Essas moedas poderão ser utilizadas para comprar ajudas dentro da plataforma, customizar seus avatares ou desbloquear conteúdos.
- **Conteúdo desbloqueável:** a fim de trazer uma compensação pelos méritos obtidos, será possível comprar ou obter conteúdos extras.
- **Avatares:** dentro da plataforma será possível a customização dos avatares por meio da compra de *skins*, dessa forma, promoverá uma compensação social dentro das turmas, pois alunos que completam as fases mais rapidamente poderão obter os melhores *skins* e, dessa forma, poderá motivar a turma a implementar o algoritmo de uma maneira mais rápida.
- **Conquistas:** o jogo será implementado no meio acadêmico, sendo uma forma de avaliação dos alunos portanto o professor será capaz de recompensar o aluno fora da plataforma com uma nota na respectiva disciplina.
- **Itens:** a cada fase o aluno poderá usar comprar ajudas e adiamento de prazos de entrega de trabalho por meio da compra usando os TDCoins.
- ***Rankings:*** os jogadores serão classificados de acordo com a razão das fases jogadas pelas pontuações obtidas.
- **Dificuldade:** a dificuldade do jogo aumenta gradativamente a medida que o aluno avança de nível e subnível.
- **Desafios:** são propostos pelos professores, como uma forma de incentivar os alunos como por exemplo, o primeiro que finalizar as fases propostas para turma, terá mais X dias para entregar o trabalho final da disciplina.
- ***Feedback:*** a pontuação do aluno, a cada fase, é informada na hora. Além disso, a ferramenta realiza o *feedback* do código implementado pelo jogador por meio tanto da execução dos casos de teste, que pode afirmar que o código está correto, quanto da análise da complexidade ciclomática, verificando se o código informado poderia ser melhorado. Com isso, as próximas fases de refatoração podem se tornar intuitivas.
- **Regras:** o jogo dá ao jogador um breve resumo sobre o que deverá ser feito, qual o nome da classe, qual o método a ser implementado, o valor a ser inserido e o resultado a ser obtido. O jogador poderá desenvolver o código da maneira que julgar melhor, porém, sua pontuação será definida pela complexidade ciclomática da fase. Dessa forma ele só passará para os próximos níveis caso atenda a esse quesito, fazendo com que ele se preocupe em implementar um código com maior qualidade e espera-se que essa preocupação torna-se intrínseca do jogador a medida que ele utiliza o jogo.

- **Restrições:** a complexidade ciclomática é calculadas a cada subnível. Dessa forma, o plágio será dificultado pois ainda que o aluno obtenha um código final implementado por outra pessoa, ele não conseguirá passar para o próximo subnível. As respostas serão salvas no banco de dados, e atrelados ao usuário, dessa maneira caso a mesma resposta apareça novamente, será possível, a penalização dos mesmos.
- **Emoções:** usando elementos de jogos convencionais, tais como animação, efeitos sonoros, destaque social por meio do rank, espera-se um engajamento emocional dos jogadores, fazendo que joguem cada vez mais.
- **Narrativa:** Com uma narrativa implícita e explícita, será possível, expor o que espera dos jogadores e, dessa forma, mesmo que ele tenha fracasso naquela fase, ele saiba como resolver o execícios de outra forma ou com a ajuda da plataforma ou com ajuda pela comunidade do jogo. Deixando de uma forma clara na narrativa que não esperamos pontuação máxima dos jogadores, não causamos desapontamento por parte do aluno, assim ele volta a utilizar a plataforma
- **Relações:**Pelo ambiente competitivo proporcionado, queremos que os alunos da mesma turma disputem pelo rank a melhor colocação e busquem ou alunos das outras turmas ou turmas que já concluíram aquela etapa para tirar dúvidas e pedirem ajudas

O TesterDS está sendo desenvolvido utilizando a linguagem Java e consiste de uma aplicação web desenvolvida utilizando serviços RESTful como API. A interface de usuário foi desenvolvida em Angular 4, um framework web de código-fonte aberto e *front-end* baseado em TypeScript liderado pela Equipe Angular do Google. A Figura 6 apresenta o diagrama de classes da parte já desenvolvida do TesterDS.

6. Conclusões e Trabalhos Futuros

Este artigo apresentou o jogo TesterDS voltado ao ensino de Estruturas de Dados que utiliza técnicas de TDD e Teste Estrutural de Software. O jogo tem foco em estudantes de disciplinas de programação mais avançadas, como Estruturas de Dados, uma vez que parte desses alunos iniciam a disciplina ainda com dificuldades de obter uma visão abrangente do problema e, consequentemente, em como resolvê-lo. No entanto, o jogo também poderá ser aplicada para alunos iniciantes em programação.

Uma vez que muitos dos alunos já iniciam a disciplina desmotivados e pensando que não são capazes ou que ela é muito difícil, espera-se que a aplicação desse jogo engaje os alunos nessas disciplinas, diminuindo o número de evasão e de reprovação.

Um trabalho futuro já em andamento é prosseguir com o desenvolvimento do jogo. Além disso, pretende-se difundir e disponibilizar essa ferramenta para a comunidade acadêmica e, com isso, obter mais dados de alunos de outros cursos e de outras instituição para serem avaliados. Entre essas avaliações, considera-se medir o tempo que alunos iniciam o desenvolvimento de determinada estrutura e se esse tempo vai melhorando ou aumentando à medida que ele vai avançando no jogo. Poderá ser observados também os pontos em que os alunos têm mais dificuldades e pensar em estratégias para minimizá-las e, além disso, traçar um perfil da turma com as notas obtidas e compará-las com as demais turmas.

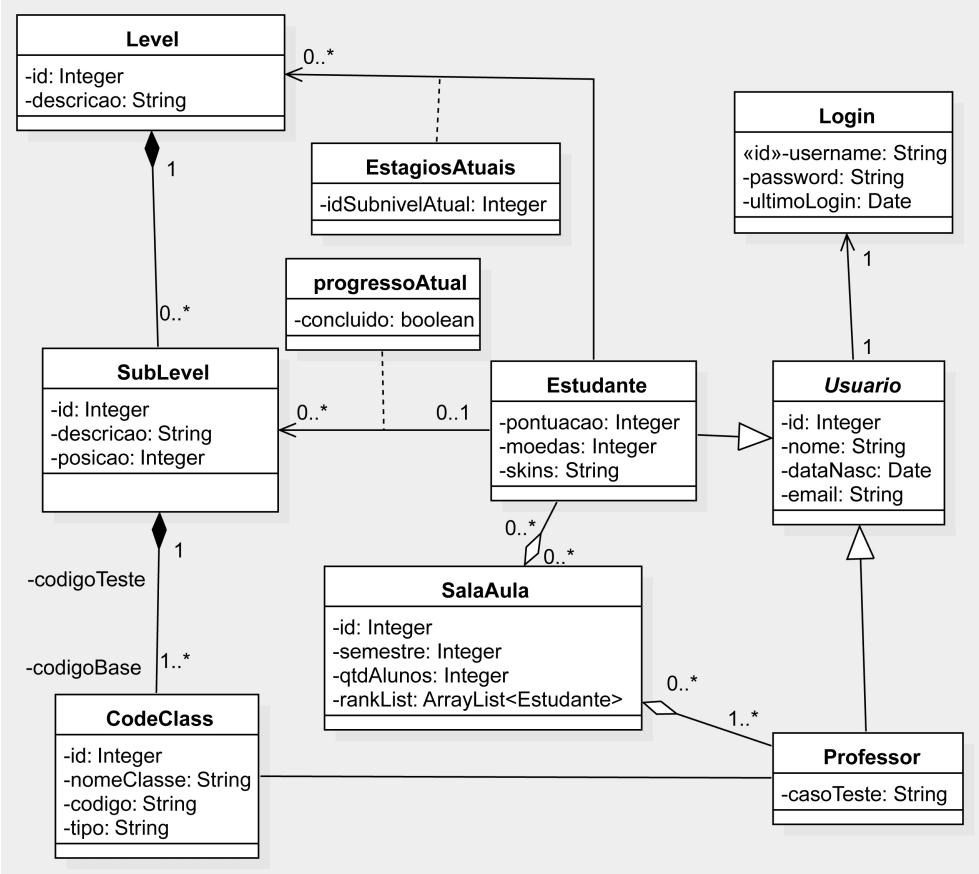

Figura 6. Diagrama de classes do TesterDS

Também pretende-se oferecer suporta a outras linguagens de programação e, para isso, pretende-se que o jogo seja integrado a plataforma ElasTest, que realizará a instrumentação e execução dos casos de teste. Outro ponto a ser considerado em pesquisas futuras é a utilização de técnicas mais sofisticadas de teste de software como, por exemplo, análise de mutantes para a geração de casos de testes automáticos para o código desenvolvido pelo aluno.

Referências

Amaral, E., Camargo, A., Gomes, M., Richa, C. H., and Becker, L. (2017). Algo+ uma ferramenta para o apoio ao ensino de algoritmos e programação para alunos iniciantes. In *Anais do XXVIII Simpósio Brasileiro de Informática na Educação*, SBIE 2017, pages 1677–1686, Porto Alegre, RS. SBC.

Camara, B. H. P. and Silva, M. A. G. (2016). A strategy to combine test-driven development and test criteria to improve learning of programming skills. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pages 443–448, New York, NY, USA. ACM.

Csikszentmihalyi, M. (1991). *Flow: The Psychology of Optimal Experience*. Harper Perennial, New York, NY.

de Souza, D. M., Isotani, S., and Barbosa, E. F. (2015). Teaching novice programmers using progtest. *IJKL*, 10(1):60–77.

Deterding, S., Dixon, D., Khaled, R., and Nacke, L. (2011). From game design elements to gamefulness: Defining "gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, MindTrek '11, pages 9–15, New York, NY, USA. ACM.

Gomes, A., Areias, C., Henriques, J., and Mendes, A. J. (2008). Aprendizagem de programação de computadores: dificuldades e ferramentas de suporte. *Revista Portuguesa de Pedagogia*, pages 161–179.

Kapp, K. M. (2012). *The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education*. Pfeiffer & Company, 1st edition.

Lopes, P. P., Gomes, M. S., Dantas, T. F., and Amaral, M. H. (2017). Proposta de um Sistema para o Monitoramento das Atividades de Programação Para Alunos Iniciantes. In *Anais dos Workshops do VI Congresso Brasileiro de Informática na Educação (CBIE 2017)*, pages 942–951.

McCabe, T. J. (1976). A complexity measure. In *Proceedings of the 2Nd International Conference on Software Engineering*, ICSE '76, pages 407–, Los Alamitos, CA, USA. IEEE Computer Society Press.

Nunes, R., Pedrosa, D., Morgado, L., Martins, P., Paredes, H., Cravino, J., and Barreira, C. (2017). SimProgramming: uma abordagem motivacional para a aprendizagem de alunos intermediários de programação. In *Anais dos Workshops do VI Congresso Brasileiro de Informática na Educação (CBIE 2017)*, pages 1099–1110.

Silva, T. S. C. d. S., Melo, J. C. B., and Tedesco, P. C. d. A. R. (2015). Teoria do *Flow* na contribuição do engajamento estudantil para apoiar a escolha de jogos no ensino de programação. In *Anais do XXVI Simpósio Brasileiro de Informática na Educação*, SBIE 2015, pages 607–616, Porto Alegre, RS. SBC.

Souza, D. M. d., Batista, M. H. d. S., and Barbosa, E. F. (2016). Problemas e dificuldades no ensino e na aprendizagem de programação: um mapeamento sistemático. *Revista Brasileira de Informática na Educação*.