Marcio Franklin Monteiro Souza

Aplicação de Gamificação no Ensino-Aprendizagem de Programação com Sistema de Ranqueamento e Feedback

São Luís/MA

2019

Marcio Franklin Monteiro Souza

Aplicação de Gamificação no Ensino-Aprendizagem de Programação com Sistema de Ranqueamento e Feedback

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Universidade Federal do Maranhão

Orientador: Prof. Dr. Carlos de Salles Soares Neto

São Luís/MA

2019

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Monteiro Souza, Marcio Franklin.
Aplicação de Gamificação no ensino-aprendizagem de programação com sistema de ranqueamento e feedback / Marcio Franklin Monteiro Souza. - 2019.
62 f.

Orientador(a): Prof. Dr. Carlos de Salles Soares Neto.
Monografia (Graduação) - Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís/MA, 2019.

1. Análise automática de código. 2. Feedback. 3. Gamificação. 4. Online Judge. 5. Ranqueamento. I. Soares Neto, Carlos de Salles. II. Título.

Marcio Franklin Monteiro Souza

Aplicação de Gamificação no Ensino-Aprendizagem de Programação com Sistema de Ranqueamento e Feedback

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Trabalho aprovado. São Luís/MA, 16 de julho de 2019:

Prof. Dr. Carlos de Salles Soares Neto
Universidade Federal do Maranhão
Orientador

Prof. Dr. Mário Antonio Meireles Teixeira
Universidade Federal do Maranhão
Membro da banca

Profª. M.ª Alana Oliveira Meireles Teixeira
Universidade Federal do Maranhão
Membro da banca

Prof. M.e Marcelo Henrique Monier Alves Júnior
Instituto Federal do Maranhão
Membro da banca

São Luís/MA
2019

Agradecimentos

Agradeço a Universidade Federal do Maranhão (UFMA) e a todos seus professores e colaboradores por me proporcionarem a oportunidade de realizar o curso na área de maior afinidade. Foram longos anos, mas posso dizer que cada momento vivido e compartilhado foi único e incrível.

Ao meu orientador, Prof. Carlos de Salles Soares Neto, principalmente pela paciência e sua bondade, pela oportunidade de desenvolver este projeto desafiador, por não desistir de mim e por sempre estar disposto a ajudar, não só durante este projeto, mas durante todo o curso.

A minha família por estar sempre ao meu lado nessa caminhada, por ter me ensinado a trilhar o caminho certo e ser uma pessoa do bem. Em especial à minha mãe Gracenildes por ser o maior exemplo de mãe que pode existir, por todos os esforços que fizera para que seus filhos estudassem. Igualmente ao meu pai Ribamar, agradeço-os do fundo do coração por tudo.

A minha namorada Amélia Castro por ser uma pessoa incrível e por ter me suportado por tanto tempo. Por não desacreditar de mim e por me ajudar na correção deste trabalho. Por ser uma pessoa alegre, apesar dos momentos difíceis, por ser alguém que eu admiro muito e que quero sempre do meu lado.

Ao meu amigo Alysson Cirilo que sempre me ajudou desde o começo desta caminhada. Por ter desenvolvido comigo um jogo com centenas de Ifs no primeiro período. Em especial neste projeto por ter me ajudado na formatação e nas discussões a cerca do trabalho. Por ser meu melhor amigo e por ser uma pessoa querida por todos.

Ao meu amigo Francisco que tirou dúvidas durante o desenvolvimento do sistema, que se mostrou uma pessoa humilde e de bom coração.

A minha amiga Júlia Manayra por ter me ajudado durante todo o curso, por sempre ser meu par durante as atividades. Por ser um pessoa que transborda bondade. Em especial, neste trabalho, por ter me ajudado com dicas, apontando onde eu poderia melhorar.

A todos meus amigos da UFMA, em especial o pessoal do grupo PETA (Malheiros, Gabriel, Gineton, Gilberlan), ao grupo Sem Diplomas - atual nome do grupo da turma de Ciência da Computação 2013.1 - e do grupo Programadores de Elite de onde sai os maiores programadores desse Brasil. André, Garcês, Tarcio, Marcos, Alex, Ricardo, Dilsy, Alane, Iago, Pedro, Rui, Wagner, obrigado por suas amizades.

Aos meus amigos em geral, os de Apicum-Açu e os de São Luís, em especial a Neil

Wagner por sempre estender a mão quando eu preciso. Agradeço também aos amigos do trabalho por me proporcionarem dias agradáveis e engraçados. Agradeço a todos por tudo que já fizeram e fazem por mim.

Por fim, queria agradecer a Geralt de Rívia por ter vencido a Caçada Selvagem e por sempre nos proteger.

"O medo é a oportunidade
de ter coragem"
(Franklin)

Resumo

O índice de desistência de cursos das áreas de de ciência da computação está entre os mais altos nas universidades brasileiras, e um dos fatores que mais contribui para isso é a dificuldade que muitos alunos têm em solucionar algoritmos. Este projeto vem propor a aplicação de uma prática didática que está se tornando cada vez mais comum quando se precisa repassar um conteúdo difícil e desafiador, a Gamificação, que é a utilização de mecânicas e dinâmicas de jogos como forma de motivar, desafiar e entreter pessoas fora do contexto de jogos. Com a utilização da gamificação e ranqueamento no aprendizado de algoritmos, visa-se criar um ambiente mais amigável e dinâmico, favorecendo uma melhor evolução do aluno. E para que isso seja possível, é desenvolvido um sistema online de análise automática de código intitulado Posh Code - Online Judge.

Palavras-chave: Gamificação, Ranqueamento, Feedback, Análise automática de código, Online Judge, Programação.

Abstract

The dropout rate of IT courses is among the highest in Brazilian universities, and one of the factors that contributes most to this is the difficulty that many students have in solving algorithms. This project proposes the application of a didactic practice that is becoming more and more common when it is necessary to pass on difficult and challenging content, Gamification, which is the use of game mechanics and dynamics as a way to motivate, challenge and entertain people outside the context of games. With the use of gamification and ranking system in algorithms learning, this work aims to create a more friendly and dynamic environment, favoring a better evolution of the student. And for that to be possible, it was developed an online judge system with the named Posh Code - Online Judge.

Keywords: Gamification, Rankings, Feedback, Automatic Code Analysis, Online Judge, Programming.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Nos períodos iniciais dos cursos da área de ciência da computação é indispensável disciplinas com foco no aprendizado de algoritmos e que abordem os princípios da lógica de programação, para que os estudantes desenvolvam a capacidade de análise e resolução de problemas na forma de algoritmos. Entretanto, tais disciplinas costumam ter um alto índice de evasão e reprovação, o que resulta na dificuldade ou até mesmo impedimento da continuidade dos estudantes no curso (RAPKIEWICZ et al., 2007).

Segundo o Censo da Educação Superior de 2016, levantado pelo INEP (Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira), pode-se observar que os cursos da área de computação no Brasil nesse ano tiveram 37.760 novos ingressos, 19.820 matrículas trancadas e 2.008 mudanças de curso, totalizando 21.828 abandonos nesse ano, conforme Tabela 1. O que significa que um número maior que a metade de novos ingressos abandonou o curso.

Tabela 1 – Dados gerais dos cursos superiores da área de computação - 2016

Curso	Vagas	Ingresso	Trancamento	Transferência	Abandono Total
Administração de redes	33.917	9.561	5.373	463	5.836
Banco de dados	3.648	836	344	57	401
Ciência da computação	45.686	23.001	11.484	1.322	12.806
Informática	17	15	1	0	1
Tecnologia da informação	1.285	1.301	718	15	733
Desenvolvimento de softwares	7.449	3.046	1.900	151	2.051
Total	92.002	37.760	19.820	2.008	21.828

Fonte: INEP (2017)

Muitas são as razões que levam ao baixo rendimento do estudante nessas disciplinas. Conforme Rodrigues (2002) dentre os principais motivos estão: (1) dificuldade em desenvolver raciocínio lógico, quando estão acostumados a decorar; (2) dificuldade em relacionar teoria e prática; (3) baixa motivação; e (4) baixa assimilação de abstrações.

A partir do experimento de Costa (2013) conforme resumo demonstrado na Tabela 2, observa-se que outros fatores também podem influenciar no rendimento do estudante, tais como: (1) possuir vínculo empregatício; (2) a falta de prática de exercício de programação; (3) verificação da corretude da solução por tentativa e erro; (4) desconcentração durante as aulas; (5) desistência rápida ao não conseguir realizar uma atividade.

Tabela 2 – Alunos reprovados vs estudantes aprovados em disciplinas de programação

Reprovados	Aprovados	Fatores
37%	53%	Uso de ônibus coletivo como meio de transporte
63%	47%	Possuir vínculo empregatício
63%	47%	Falta de prática exercícios de programação rotineiramente
81%	58%	Verificação da corretude da solução por tentativa e erro
44%	37%	Apresentam desconcentração durante as aulas
31%	11%	Desistência rápida ao não conseguir realizar atividade
38%	47%	Não frequentar biblioteca

Fonte: Adaptação de Costa (2013)

Baseado no que foi exposto, percebe-se que há necessidade de buscar formas alternativas de ensino-aprendizagem as quais estimulem o interesse dos estudantes pelos conteúdos da disciplina, e mantenham-os envolvidos durante todo o processo e contorne os principais problemas enfrentados pelos mesmos.

E uma proposta que corresponde às necessidades mencionadas acima e que vem ganhando muito destaque nos últimos anos é a utilização do conceito de gamificação, que é o uso de elementos de jogos para envolvimento de pessoas na realização de tarefas fora do contexto de *games*, com o objetivo de desenvolver a motivação, auxiliar na resolução de problemas e facilitar o aprendizado (KAPP, 2012).

Muitos autores defendem a aplicação de gamificação nas disciplinas de computação (RAPOSO; DANTAS, 2016), (NAGAI; IZEKI, 2016), (MONTEIRO; OLIVEIRA; MARTINS, 2015) e tais trabalhos concordam que o resultado é positivo no que diz respeito à melhora no desempenho dos estudantes onde essa técnica é aplicada.

Sendo assim, este trabalho visa aplicar a gamificação no ensino-aprendizagem de algoritmos em turmas de ciência da computação, através da construção de um sistema de avaliação automática de código, batizado de Posh Code - Online Judge, desenvolvido em plataforma web, utilizando as linguagens de programação Java[1] e Python[2], e provém ambiente para o estudante resolver problemas, submeter soluções, receber feedback do professor e comparar seu rendimento através de sistema de ranqueamento.

Esses sistemas de avaliação de código em geral levam em seu nome o termo "Online Judge", que em tradução livre seria "Juiz Online". Funciona a partir da disponibilização de problemas a serem solucionados, onde deve-se submeter código-fonte da solução escrito em alguma linguagem de programação, para que seja corrigido automaticamente pelo sistema (KURNIA; LIM; CHEANG, 2001).

Este trabalho tenciona contribuir com o aumento da interação dos estudantes

[1] Para mais informações acesse: <https://www.java.com/>
[2] Para mais informações acesse: <https://www.python.org/>

dentro da universidade, especificamente dentro de curso da área da computação, porém, sem se limitar a esse escopo, uma vez que qualquer pessoa da sociedade poderá utilizá-lo também. Através da estratégia de gamificação com o sistema de ranqueamento visa-se favorecer o aprendizado por reforço e consequentemente melhorar o rendimento do estudante. Pretende-se também disponibilizar o código fonte do sistema para que a comunidade possa fazer correções, evolução e estudo.

1.1 Objetivos

Utilizar técnicas de gamificação no ensino-aprendizagem de programação por meio de uma aplicação web de avaliação automática de código, e analisar se essas técnicas podem contribuir para o aumento da motivação na aprendizagem de lógica de programação.

1.1.1 Objetivos Específicos

a) desenvolver uma aplicação web que permita a utilização de gamificação no ensino-aprendizagem de algoritmos;

b) aplicar o sistema desenvolvido em disciplina de programação do curso de Ciência da Computação da UFMA (Universidade Federal do Maranhão);

c) verificar os benefícios que o sistema trouxe aos estudantes.

1.2 Juízes Online

Há muitos sistemas de avaliação automática de código, entre os mais conhecidos estão URI Online Judge [3], BOCA [4], Sphere Online Judge[5], UVA Online Judge[6], além de muitos outros. Trata-se de um sistema que compila ou interpreta o código e o executa, em seguida compara o resultado da execução com uma bateria de respostas corretas internas ao sistema, previamente cadastradas, e por fim mostra um resultado ao usuário, para informar se a solução testada está correta ou não. A questão cadastrada pode ainda ser configurada para ser limitada a certas condições, como limite de tempo, quantidade de memória disponível para uso, condições de segurança, entre outras.

A maioria desses sistemas são utilizados em maratonas de programação por terem um vasto banco de problemas e pela rápida e automática avaliação dos códigos. Dois exemplos desses sistemas, comumente chamados de *Online Judges* são: BOCA e URI Online Judge Academic.

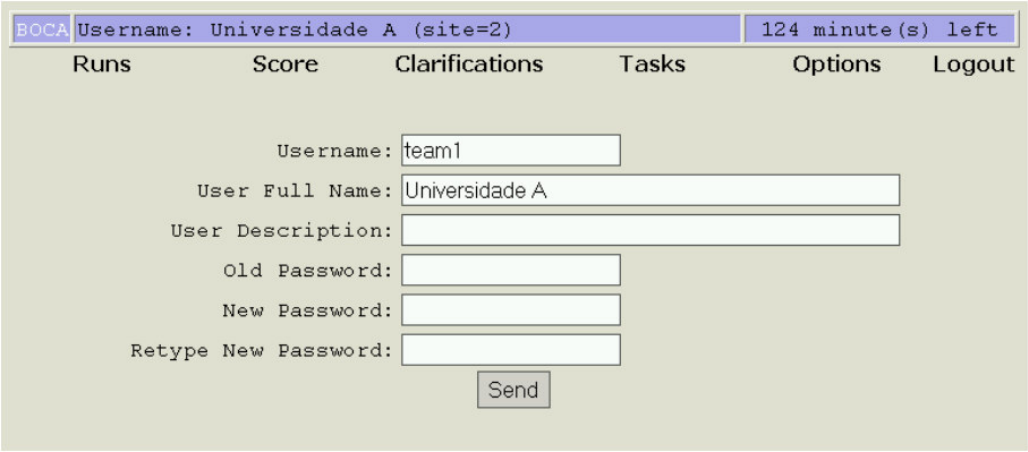
[3] Para mais informações acesse: <https://www.urionlinejudge.com.br/>
[4] Para mais informações acesse: <https://www.ime.usp.br/~cassio/boca/>
[5] Para mais informações acesse: <https://www.spoj.com/>
[6] Para mais informações acesse: <https://uva.onlinejudge.org/>

Nas subseções a seguir são apresentadas de forma simplificada as principais funcionalidades e telas dos sistemas BOCA e do módulo URI Online Academic, para que se tome conhecimento de seus funcionamentos e de suas propostas.

1.2.1 BOCA

O BOCA é um sistema de avaliação automática de código que tem como foco o apoio a competições de programação e foi desenvolvido para ser usado na Maratona de Programação da Sociedade Brasileira de Computação (SBC)[7]. É um sistema desenvolvido na linguagem de programação PHP e sua interface é dividida em quatro partes: Time, Juiz, Administrador e Staff (CAMPOS; FERREIRA, 2004).

Figura 1 – Interface do usuário Time

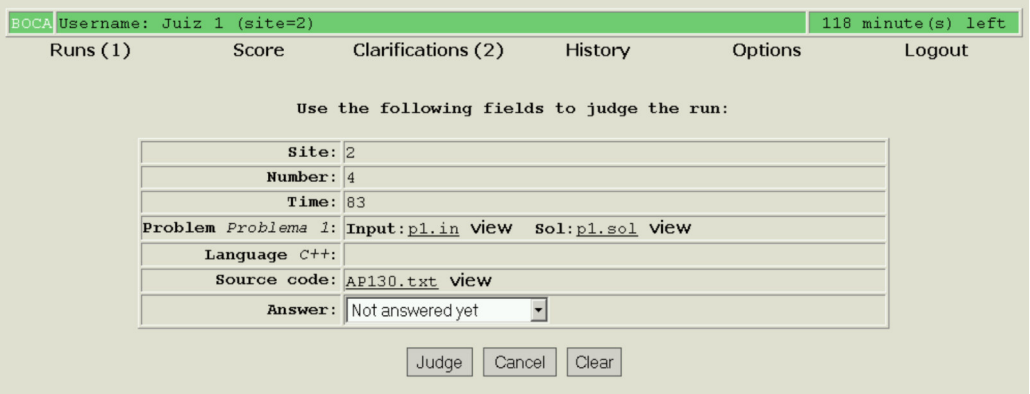


Fonte: Campos e Ferreira (2004)

No ambiente do usuário Time Figura 1 tem-se as funcionalidades: (1) submissão de código-fonte da solução do problema; (2) histórico de submissões; (3) criação de times; (4) espaço para enviar perguntas, afim de sanar dúvidas com relação às questões; e (5) lista de classificação ordenada por maior quantidade de questões resolvidas, em seguida ordenada por menor tempo para alcançar a solução.

[7] Para mais informações acesse: <http://sbc.org.br/>

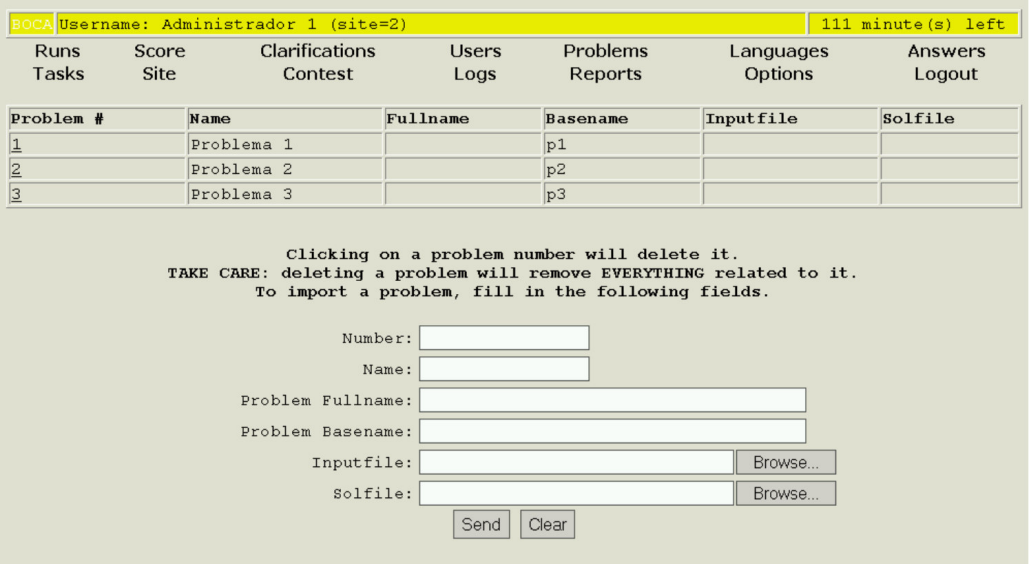
Figura 2 – Interface do usuário Juiz



Fonte: Campos e Ferreira (2004)

No ambiente do usuário Juiz Figura 2 tem-se as funcionalidades: (1) avaliar questões respondidas pelos estudantes; (2) acompanhamento do placar da competição; (3) área para responder perguntas dos estudantes e fazer perguntas para outros juízes;

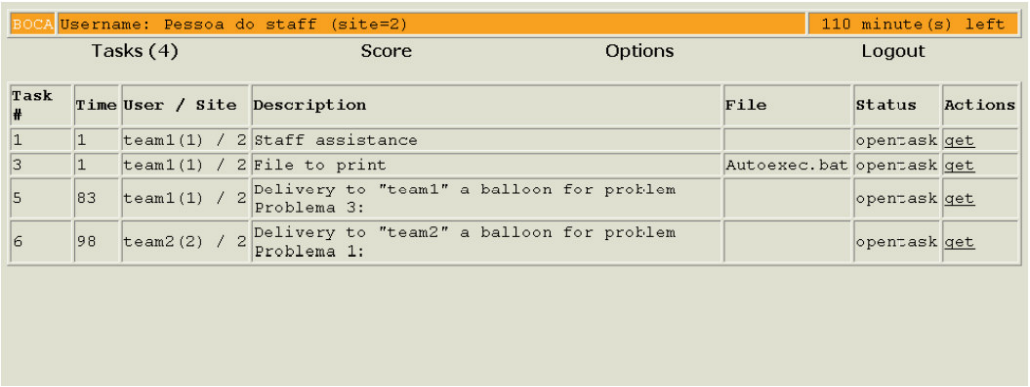
Figura 3 – Interface do usuário Administrador



Fonte: Campos e Ferreira (2004)

No ambiente do usuário Administrador Figura 3 tem-se as funcionalidades: (1) controle sobre a configuração da competição; (2) inclusão de problemas; (3) inclusão de usuários; e (4) acompanhamento completo da competição.

Figura 4 – Interface do usuário Staff



Fonte: Campos e Ferreira (2004)

Por fim, no ambiente do usuário Staff Figura 4 tem-se as funcionalidades: (1) controle de acesso e movimentação dos times no local da prova; e (2) recebimento de arquivos solicitados pelos times para impressão. Além disso, o usuário Staff tem ações fora do sistema, como entrega de impressão e de balões para os times em competição.

1.2.2 URI Online Judge Academic

Outro trabalho muito interessante é o URI Online Judge Academic: Integração e Consolidação da Ferramenta no Processo de Ensino/Aprendizagem. O URI Online Judge é um sistema de avaliação automática de código desenvolvido pela URI (Universidade Regional Integrada)[8], Campus de Erechim (SELIVON et al., 2015). Tem como principais características:

a) correção em tempo real;

b) utilização de juízes especiais;

c) fórum;

d) aceitação de soluções nas linguagens (C, C++, Java e Python);

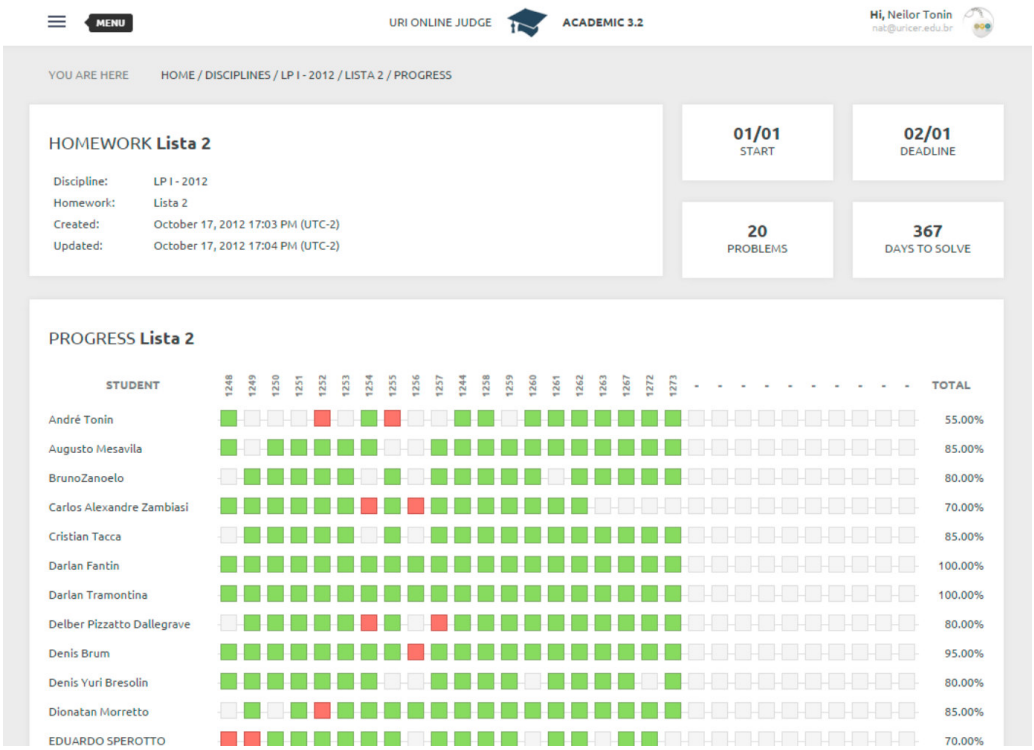
e) organização de problemas por níveis de dificuldade e categorias;

f) ranking por problema e por linguagem de programação.

O URI Online Judge Academic é um módulo do URI Online Judge que fora desenvolvido posteriormente para que fosse possível dar a oportunidade do professor acompanhar a evolução e o rendimento de seus estudantes. O novo ambiente é integrado ao portal do URI Online Judge e é dividido em dois ambientes distintos, um para o estudante e outro para o professor.

[8] Para mais informações acesse: <http://www.uri.br/>

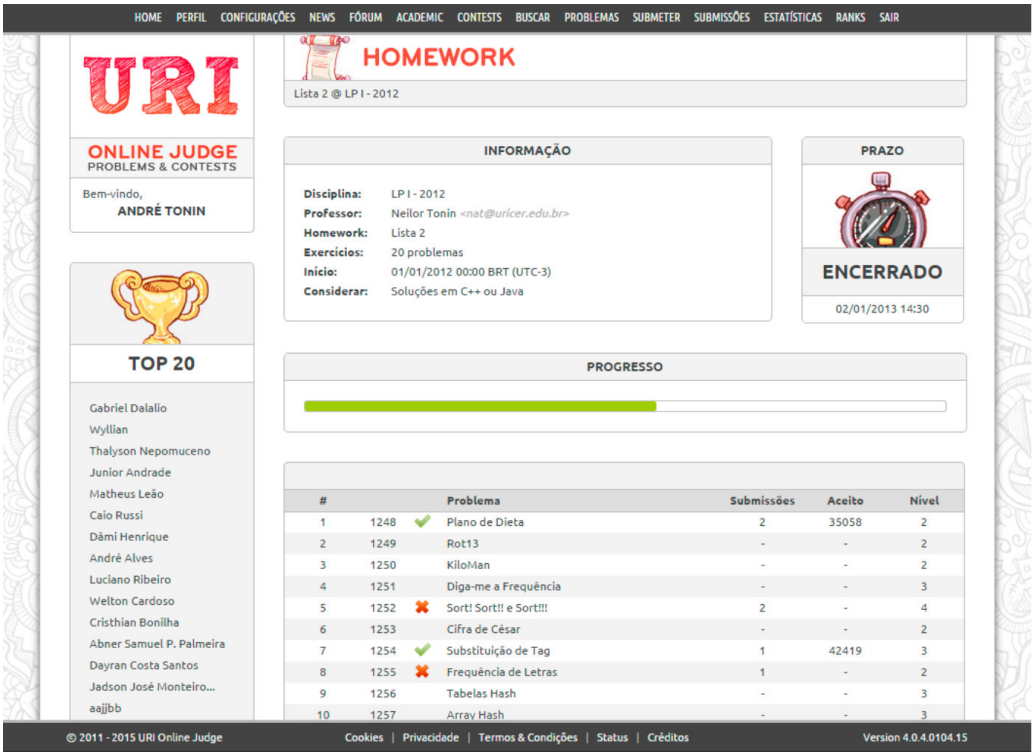
Figura 5 – Interface do usuário Professor



Fonte: Selivon et al. (2015)

No ambiente do professor é possível cadastrar turmas através de convites aos estudantes, cadastrar disciplinas e listas de tarefas associadas a cada disciplina, definir linguagem a serem utilizadas, definir prazo para a solução e submissão dos problemas, registrar instruções, e o principal, acompanhar o rendimento dos estudantes através do percentual e da marcação de problemas solucionados, conforme Figura 5.

Figura 6 – Interface do usuário Aluno



Fonte: Selivon et al. (2015)

O estudante, a partir do seu ambiente tem acesso às listas e turmas onde está inserido, e pode verificar o andamento de uma tarefa, visualizar informações dos problemas - como número de submissões, nível de dificuldade -, além de ter acesso às orientações do professor da disciplina e prazo para conclusão Figura 6.

É notório que existem muitos sistemas de avaliação automática de código, porém é importante destacar que a maioria tem foco em competição e em estimular a participação dos estudantes em maratonas de programação, ou mesmo, apenas ajudar o usuário a melhorar seu rendimento na construção de algoritmos.

Então cabe ressaltar que o foco deste trabalho é voltado ao meio acadêmico, com o objetivo de aumentar a interação estudante/professor através do sistema de feedback, e a interação estudante/ estudante através do sistema de ranqueamento, e principalmente aumentar a motivação e o interesse dos estudantes pelo conteúdo da disciplina de programação, a partir da aplicação da metodologia de gamificação.

1.3 Organização do trabalho

Os capítulos seguintes deste trabalho são estruturados da seguinte forma: No Capítulo 2 (Tecnologias Relacionadas) é apresentado um resumo das ferramentas e bibliotecas

de terceiros utilizadas para o desenvolvimento do sistema web. No Capítulo 3 (Proposta do Poshcode) é mostrado o funcionamento e as telas do sistema em produção, além dos diagramas, implementação e técnicas de desenvolvimento do Posh Code. No Capítulo 4 (Relato de Experiência) é apresentando o cenário onde a proposta foi aplicada, o resumo de utilização dos usuários, e a avaliação dos resultados de uso. Por fim, no Capítulo 5 (Conclusões) é apresentado o que se concluiu com o trabalho, bem como proposta para a evolução.

2 Tecnologias Relacionadas

O desenvolvimento de sistemas Web exige cada vez mais configurações em sua infraestrutura a fim de suprir a demanda de funcionalidades e segurança que um sistema exige para os padrões atuais, porém, há uma grande dificuldade em manter essas configurações, uma vez que qualquer tempo gasto na configuração é tempo gasto não escrevendo a lógica do sistema (WALLS, 2016).

Diante disso, muitas ferramentas são criadas para resolver essa questão de configuração, deixando o desenvolvedor focado nas regras de negócio do sistema. Uma delas é o eco-sistema Spring[1].

O Spring é um projeto *open source* (código livre) que tem como base os padrões de projeto Inversão de Controle (IoC) e a Injeção de Dependência (DI) na plataforma Java e como objetivo o auxílio na construção de aplicações robustas, flexíveis, de fácil manutenção e de baixo acoplamento.

> Injeção de dependências (*dependency injection* ou DI) é um padrão de desenvolvimento de software usado para manter o baixo acoplamento entre classes do sistema. As dependências de um objeto não são instanciadas programaticamente, mas sim injetadas de alguma forma (FARIA, 2015).

Foi desenvolvida por Rod Johson e descrita pela primeira vez em seu livro Expert One-on-one J2EE Development without EJB (JOHNSON; HOELLER, 2003) e em 2004 foi apresentada sua versão final através do livro Expert one-on-one J2EE design and development (JOHNSON, 2004). Foi criada como uma alternativa à complexidade do uso de EJBs.

2.1 Cliente *WEB*

As ferramentas utilizadas no desenvolvimento do sistema Posh Code são classificadas em duas partes: Cliente WEB e Serviço API. Dessa forma torna-se mais organizada a apresentação de cada tecnologia utilizada.

De forma concisa, o Cliente WEB é o site usado pelo usuário. É onde se gerencia toda a parte visual do site, o que inclui a disposição e configuração dos elementos que compõe a tela, como texto, imagens, animações, etc. O Cliente WEB também trata de parte dos dados consumidos pela aplicação, fazendo comunicação com o banco de dados, e ainda gerencia as configurações com relação à segurança do site.

[1] <https://spring.io/>

Nas subseções a seguir são apresentadas as principais ferramentas utilizadas na produção do sistema Posh Code.

2.1.1 Automatizador Spring Boot

Como foi dito anteriormente, o Spring foi apresentado como uma alternativa ao JEE (Java Enterprise Edition), mas se por um lado era simples em termos de código, era massivo em configuração, o que tornou o projeto pouco atraente no início, pois a configuração era toda feita utilizando XML (Extensible Markup Language). Mesmo depois, com a versão Spring 3.0, que permitiu configuração via código Java, o processo de configuração ainda existia e era muito trabalhoso.

Outro problema era a gestão das dependências, pois era preciso demandar tempo e esforço para verificar quais bibliotecas são compatíveis entre si e quais versões deveriam ser utilizadas nos projetos. Foi então que o Spring Boot foi lançado e resolveu tudo isso.

O Spring Boot nada mais é que um automatizador das configurações do projeto, e fornecedor de ferramentas para a construção e publicação da aplicação, com uma proposta em *micro service*. Basicamente o desenvolvedor só escolhe quais os módulos que deseja utilizar e incluí-os no `POM.XML`, podendo ser alterado em qualquer momento no ciclo de desenvolvimento.

Conforme Walls (2016) mostra em seu livro, a simples execução de um "*Hello Word*" em um projeto Spring demanda: a configuração da estrutura do projeto; a configuração do arquivo web.xml; a configuração do Spring que habilita o Spring MVC; uma classe do controlador que responderá a solicitações HTTP (Hypertext Transfer Protocol) com "Hello World"; e a configuração de um servidor de aplicação web. E como o autor destaca, o único item específico para desenvolver a funcionalidade é o controlador, o restante é genérico, portanto, a criação do Spring Boot tornou todo o processo mais simples, garantindo ao desenvolvedor aumento de produtividade, economia de tempo e a garantia da consistência da configuração do projeto.

Em resumo as principais vantagens de utilizar Spring Boot são: A proveniência de um jeito flexível de configurar XML, Java Beans e a transição com banco de dados; Configurações automáticas; Aplicações Spring via *annotations*; Simplicidade na gerencia de dependência; Suporte para SQL e NoSQL; e *container servlet* nativo.

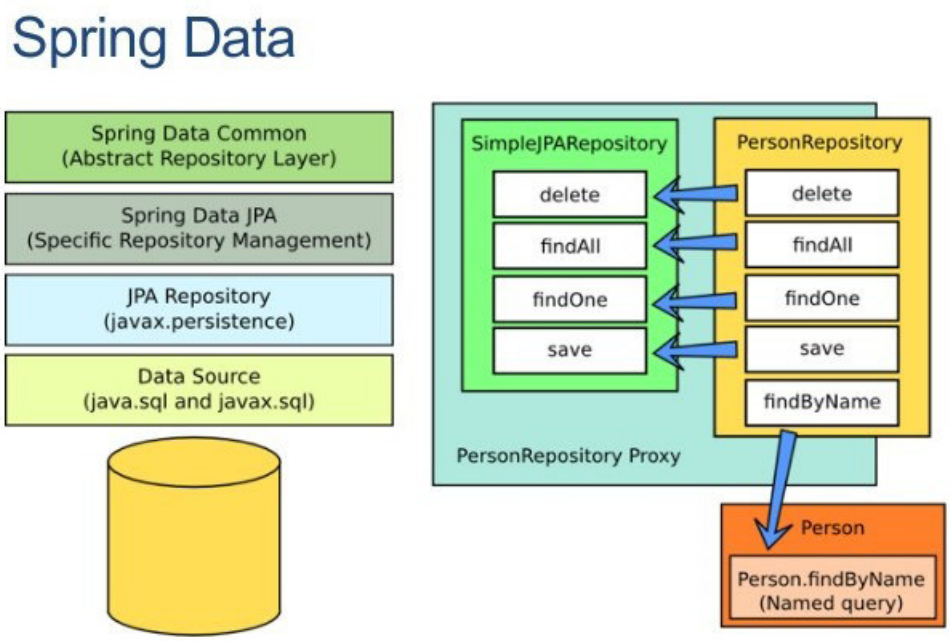
2.1.2 Spring Data

Podemos definir o Spring Data como um projeto Spring composto de vários subprojetos que visa abstrair e gerenciar toda a parte de dados de um sistema em produção. Seu foco é simplificar a utilização de bancos de dados, sejam eles relacionais, não-relacionais ou mesmo tecnologias novas como armazenamento de dados baseado em nuvem.

Sua criação surgiu da necessidade de tirar a responsabilidade da configuração de bancos de dados da mão do desenvolvedor, pois esse processo era trabalhoso e pouco flexível, já que o mesmo tinha que implementar a cada novo projeto - ou mesmo quando no mesmo projeto, por ventura, fosse mudado o SGBD - uma camada de acesso a dados usando APIs específicas, novamente escrevendo códigos repetitivos para alcançar os resultados (KAINULAINEN, 2012).

Entre seus vários subprojetos podemos destacar: Spring Data Commons, que provém interfaces e código compartilhados entre as várias implementações específicas do armazenamento de dados; Spring Data JDBC, que facilita a implementação de repositórios baseados em JDBC; Spring Data JPA, módulo que dá suporte aprimorado para camadas de acesso a dados baseadas em JPA; e o Spring Data REST, que analisa o modelo de domínio da aplicação e expõe recursos HTTP orientados por hipermídia para agregados contidos no modelo (SPRING, 2019).

Figura 7 – Representação da estrutura do Spring Data



Fonte: Silva (2016)

Na Figura 7 ilustra-se a estrutura por trás do Spring Data, que faz uso de um processo de simplificação da camada de comunicação com o banco de dados descrito na especificação conhecida como JPA (Java Persistence API) e adiciona mais duas camadas de abstrações afim de gerenciar uma simplificação das conhecidas Named Query (Spring Data JPA e Spring Data Common), esses dois projetos criam uma nova forma de se fazer queries semânticas em java, usando o poder das *reflection* para conseguir diminuir a complexidade de se definir um SQL e melhorando conceitos como a semântica.

No esquema de usabilidade do Springa Data (detalhe à direita), vemos que ele se baseia no padrão de projeto Repository para criar a camada de abstração que definirá as regras de manuseamento do banco, a Classe ilustrada como Person faz uso de todo o processo de simplificação para manter seu esboço simples e coeso.

2.1.3 Spring Security

O Spring Security é um framework com foco na segurança da aplicação. É uma estrutura poderosa e altamente personalizável de autenticação e autorização, que permite o uso de boas práticas de segurança de forma simples e direta.

Com poucas configurações é possível criar classes de usuário, proteger as requisições web, criar diferentes níveis de autorização de usuários, criar regras de autenticação globais aos recursos, criar regras de autorização aos diferentes níveis da aplicação, e ainda prevenir ataques comuns ao sistema.

2.1.4 Ferramentas Auxiliares

2.1.4.1 Gradle

Gradle é um sistema open source de automação de builds que se baseia no Apache Ant e Apache Maven, e utiliza a linguagem DSL que é baseada em Groovy[2], diferentemete da configuração via XML usado no Maven[3]. O Gradle fornece suporte poderoso para definir e organizar construções multiprojetos, bem como modelar dependências entre projetos.

2.1.4.2 Bootstrap

Bootstrap é um framework web open source para construção de front-end de aplicações web utilizando HTML, JavaScript e CSS , com o objetivo de torná-la elegante e responsiva, melhorando a experiência do usuário (EFRON; TIBSHIRANI, 1994).

2.2 Serviço API

Por motivos de facilitação no gerenciamento das execuções dos códigos-resposta dos estudantes faz-se necessário utilizar a linguagem de programação Python como alternativa ao Java para essa parte do sistema, através do consumo de um sistema RESTFull, pois o Python tem atuação direta ao Sistema Operacional.

[2] Para mais informações acesse: <https://groovy-lang.org/>
[3] Para mais informações acesse: <https://maven.apache.org/>

2.2.1 Django

Django é um *framework open source* para aplicações web utilizando Python, e tem como modelo de padrão de arquitetura o MVT (*model-template-view*). Foi lançado em 2005 e é mantido pela instituição sem fins lucrativos Django Software Foundation (DSF) (GSTI, 2016).

O *framework* Django facilita a construção de sites, de simples a complexos, com banco de dados orientado a objetos, destacando-se em reutilização e a conexão de componentes. Permite um desenvolvimento rápido evitando repetição e ainda utiliza modelos criados através do ORM (*Object Relational Mapping*) para criar, ler e excluir dados através de interfaces (GSTI, 2016).

Entre suas principais características pode-se destacar:

a) gerenciamento da autenticação do usuário, administração de conteúdo, mapas do site, feeds RSS e muitas outras tarefas;

b) ajuda os desenvolvedores a evitar muitos erros comuns de segurança, como injeção de SQL, script entre sites, falsificação de solicitações entre sites e *clickjacking*.

c) mapeamento objeto relacional (*Object Relational Mapping* - ORM) que define a modelagem de dados via código Pyhton, tornando facultativo a utilização de SQL;

d) interface para administração dos modelos criados através do ORM;

e) criação automática de formulários através dos modelos de dados;

f) possibilidade de criação de URL amigável;

g) linguagem de *template* onde é possível separar design, conteúdo e código;

h) um sistema de cache integrado ao *memcached* ou em outros frameworks de cache.

2.2.1.1 Docker

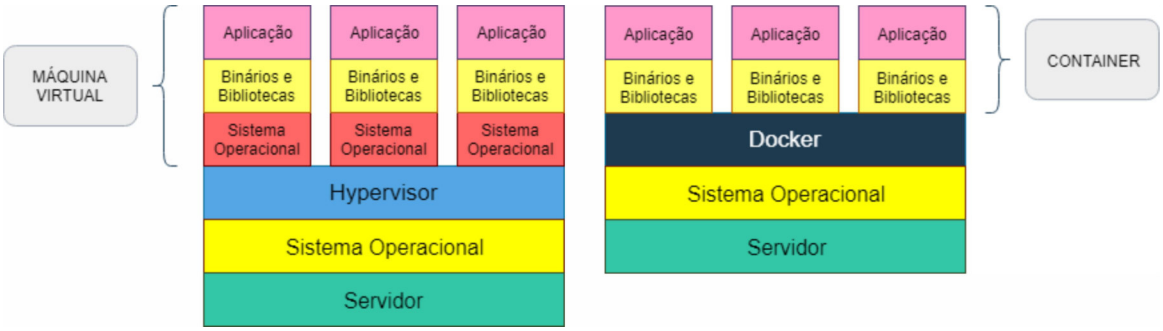
O Docker é uma ferramenta *open source* escrita em Go[4] para criação de containers linux, que possibilita o empacotamento de uma aplicação com todas as dependências necessárias para o seu funcionamento (ROLLA, 2018).

Um *container* é um conjunto de processos isolados do restante do sistema que compartilha o *kernel* da máquina física em vez de simular todos os componentes como a virtualização tradicional faz, dessa forma é permitido uma virtualização em nível de

[4] Para mais informações acesse: <https://golang.org/>

sistema operacional que mantém a aplicação isolada utilizando menos recursos. Na Figura 8 podemos notar essa diferença.

Figura 8 – Comparação da representação do paradigma de servidor entre Máquinas Virtuais e Container



Fonte: Imagem adaptada de Rolla (2018)

O Docker fornece uma API para interação com o Docker daemon, chamado Docker Engine API, bem como SDKs para Python. Os SDKs permitem que você crie e dimensione aplicativos e soluções do Docker de maneira rápida e fácil.

A API do Docker Engine é uma API RESTful acessada por um cliente HTTP, como wget ou curl, ou a biblioteca HTTP, que faz parte da maioria das linguagens de programação modernas.

3 Proposta do PoshCode

Neste capítulo é abordado o conceito de gamificação e a aplicação de suas técnicas a fim de buscar melhoria no rendimento dos estudantes da área da computação por meio do desenvolvimento de um sistema online de análise automática de código, intitulado Posh Code. Além disso, é também descrita a arquitetura, modelagem, implementação e as telas desse sistema.

3.1 Gamificação

Segundo Kapp (2012), gamificação é o uso de mecânicas, estéticas e pensamentos dos *games* para engajar pessoas, motivar a ação, promover a aprendizagem e resolver problemas. O termo Gamificação (do inglês, *gamification*, surgiu em 2002 cunhado pelo programador e inventor britânico, Nick Pelling. No entanto, o termo só se tornou popular em 2010, a partir de uma apresentação de TED[1] realizada por Jane McGonigal, famosa game designer norte-americana e autora do livro "A realidade em jogo: por que os games nos tornam melhores e como eles podem mudar o mundo", considerada uma das obras primas a respeito da gamificação (TANAKA et al., 2013).

Por meio da Tabela 3 abaixo é possível diferenciar *game* e gamificação, para que se possa entender melhor o papel de cada um.

Tabela 3 – Diferença entre *game* e gamificação

Game	Gamificação
Os games têm regras e objetivos definidos.	Pode ser apenas uma coleção de tarefas com pontuação e algum tipo de recompensa.
Existe a possibilidade de perder.	Perder pode ou não ser uma possibilidade, uma vez que o objetivo é motivar as pessoas a entrar em ação e fazer algo.
Às vezes, apenas o ato de jogar o game já é intrinsecamente gratificante.	Ser intrinsecamente gratificante é opcional.
Os games geralmente são caros e difíceis de desenvolver.	A gamificação é geralmente mais fácil e mais barata de se implementar.
O conteúdo é geralmente transformado para caber na história e nas cenas do game.	Normalmente recursos com aparência de game são adicionados, sem realizar muitas alterações no conteúdo.

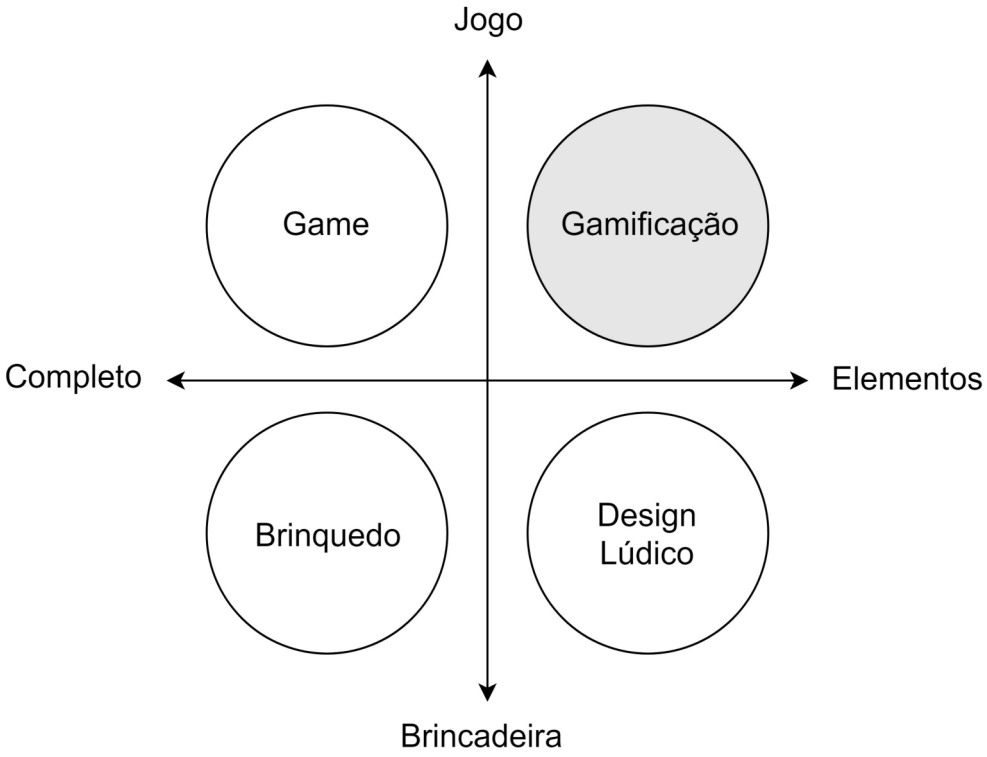
Fonte: Jose (2015)

Deterding et al. (2011) propõe um esquema que insere a gamificação entre dois eixos Figura 9. O horizontal traz a ideia de um jogo (no caso, *game*) completo até as suas

[1] Para mais informações acesse: <https://www.ted.com/>

partes (elementos) e o vertical vai da brincadeira (livre e descontraída) para o jogo (mais formal). Sendo assim, a gamificação pressupõe o uso de elementos dos *games*, sem que o resultado final seja um *game* completo, e também se diferencia do design lúdico na medida em que este pressupõe apenas um aspecto de maior liberdade, de forma lúdica, quanto ao contexto em que está inserido (FARDO, 2013).

Figura 9 – Contextualização da gamificação



Fonte: Deterding et al. (2011)

Com relação aos tipos, a gamificação pode ser dividida em 3 tipos comuns: interna, externa e para a mudança de comportamento. Na gamificação interna o foco está na situação interna de uma empresa com o objetivo de aumentar o engajamento dos funcionários em busca de melhores resultados. Já na gamificação externa o olhar está voltado para os clientes, com o objetivo de fidelizar os que já existem e atrair novos para melhorar as vendas. Por fim, na gamificação para a mudança de comportamento há uma visão mais ampla com a finalidade de mudar os hábitos das pessoas, proporcionando novas ferramentas para que elas atinjam os objetivos ou as metas propostas. Assim pode ser utilizada em outras áreas e não apenas na comercial (WERBACH; HUNTER, 2012).

3.1.1 A gamificação na educação

Atualmente, os jovens vivem em um ambiente digital com diversas tecnologias disponíveis e ampla variedade de opções de entretenimento e acesso ao conhecimento.

Essa realidade da nova geração não condiz com o modelo tradicional de ensino existente, onde o professor possui papel centralizador e o estudante permanece passivo recebendo as informações. Isso faz com que o estudante se sinta desconectado do seu ambiente de aprendizado, perca o interesse nas informações repassadas e se sinta desmotivado para estudar (BRITO, 2017).

Dentro desse contexto educacional é importante que as práticas destinadas ao processo de aprendizagem sejam constantemente ajustadas à realidade dos indivíduos e com foco no acompanhamento das transformações tecnológicas da sociedade. Sendo assim é necessário estimular a aprendizagem através de meios multi e transdisciplinares com o objetivo de elevar os níveis motivacionais e de engajamento dos indivíduos e assim proporcionar experiências mais efetivas e relevantes ao sujeito (BUSARELLO, 2016).

Nessa perspectiva identificam-se os jogos como mídias capazes de motivar os indivíduos, se apresentando como alternativas eficientes no processo de geração de conhecimento. Por meio dos jogos é possível desenvolver alguns princípios de aprendizagem como: identidade, interação, produção, riscos e consolidação (GEE, 2009).

Como uma das possibilidades de aplicação dos jogos digitais para a educação temos a gamificação de atividades educacionais. A mesma utiliza elementos comumente encontrados na maioria dos games como ferramenta de ensino, com o intuito de gerar níveis semelhantes de envolvimento e dedicação daqueles que os games normalmente conseguem gerar. A gamificação também se dispõe a transpor os métodos de ensino e aprendizagem presentes nos games para a educação formal (FARDO, 2013).

Dessa forma, a gamificação surge como uma forma de conectar a escola ao universo dos jovens com foco na aprendizagem por meio de práticas como sistema de ranqueamento e fornecimento de recompensas. No entanto, ao invés de enfatizar efeitos tradicionais como notas, utilizam-se elementos alinhados com a mecânica dos jogos para promover experiencias que envolvem emocionalmente e cognitivamente os estudantes (SILVA et al., 2014).

Para entender melhor o processo de gamificação, deve-se entender sobre os elementos dos *games*, pois os mesmos dão origem à atividade gamificada (MARTINS et al., 2014). A Tabela 4 sintetiza as principais características dos games:

Deste modo, a gamificação dentro do processo de aprendizagem, traz para o indivíduo uma nova forma de obter competências e habilidades, de uma maneira mais suave e prazerosa. Isto porque os *games*, por meio dos elementos que vimos, geram experiências com estímulos emocionais que geram impacto nesse processo de aprendizagem (FERNANDES; RIBEIRO, 2018).

Tabela 4 – Elementos de games

Elementos	Descrição
Narrativa	História que promove a imersão do jogador no jogo.
Níveis	Divisão do jogo em partes, geralmente com dificuldades incrementais; também chamadas de fases.
Desafio/Missões	Objetivos que o jogador deve alcançar.
Regras	Restrições ou limitações impostas pelo jogo.
Feedback	Resposta a uma ação do jogador, que possibilita imediatamente uma confirmação ou reavaliação das escolhas e táticas.
Competição	Relacionamento entre jogadores ou times, que promove a busca por ser o melhor. Se bem estimulada, pode promover inúmeras aprendizagens. Pode-se também competir consigo mesmo numa busca por superação.
Engajamento ("ciclo mágico")	O que motiva o jogador a jogar.
Recompensa	Benefício adquirido após alguma ação ou conclusão de uma missão.
Pontuação/Progressão	Forma quantificável do status do jogo.

Fonte: Fardo (2013)

3.1.2 A gamificação do Posh Code - Online Judge

O objetivo do sistema Posh Code é melhorar o rendimento dos estudantes de cursos da área de computação nas disciplinas de programação. Conforme mostrado nas subseções anteriores, utilizar os elementos da gamificação tornam a atividade mais atrativa. Portanto, faz-se uso de alguns elementos da gamificação no Posh Code.

O ranqueamento é o principal elemento da gamificação utilizado no sistema Posh Code. E uma vez que o Posh Code possibilita a criação rankings de muitas maneiras, mesmo quando o estudante não tiver uma boa colocação em rankings gerais, sua colocação em rankings compostos por seus amigos certamente será melhor, por exemplo.

O ranqueamento também promove desafios e competição, dois outros elementos da gamificação, uma vez que o estudante se sente desafiado a conseguir sempre uma boa colocação ou na tentativa de superar seus resultados e de seus amigos.

O sistema Posh Code conta ainda com um sistema feedbacks, onde o professor tem o papel de auxiliar o estudante nas respostas das questões, dando dicas, sugerindo melhorias no código, etc. Além disso, a resposta que o sistema dá após análise da solução submetida pelo estudante também caracteriza um feedback, pois, o sistema retorna o próprio retorno do compilador, então, em caso de resposta errada, o fato do estudante receber aquele retorno do compilador, o ajuda a entender melhor o erro do seu código.

3.2 Visão do usuário do Posh Code - Online Judge

O Posh Code contempla duas visões, uma para o estudante, que é o usuário que consulta problemas e submete código-soluções, afim de receber uma resposta após correção do sistema, bem como uma classificação nos rankings. A outra visão é a do professor, que é o usuário que cadastra problemas, dá feedbacks aos estudantes, cria e gerencia turmas.

Na visão do estudante é possível ver as questões disponíveis para solução e visualizar seu conteúdo através do próprio navegador web, tendo a opção de download em formato PDF. Para realizar uma submissão o estudante deve selecionar uma questão e enviar código em um único arquivo entre os formatos .c, .cpp, .java ou .py (Python 3).

O estudante pode acompanhar o resultado das submissões na página principal no histórico de submissões, onde é mostrado a resposta da execução do código, o tempo que levou-se para terminar sua execução, e também o retorno do compilador ou interpretador, que serve para ajudar o estudante a entender melhor o que deu errado em caso de erro. Ainda na visão do estudante é possível ver a sua classificação nos rankings, mensagens de feedback repassadas pelo professor, e também a possibilidade de consultar questões através da busca no banco de questões.

Na visão do professor permite-se cadastrar questões, onde é preciso informar título da questão, categoria, tempo para processamento de código, enviar arquivo PDF com o conteúdo da questão, arquivo de entrada e arquivo de saída com baterias de testes.

Ao professor é dado a possibilidade de cadastrar as instituições, cursos e turmas, para que o usuário estudante utilize essas informações na edição de seu perfil. Ainda na visão do professor é mostrado as questões cadastradas pelo mesmo e área que provém feedback aos estudantes.

3.3 Arquitetura do Posh Code - Online Judge

Como metodologia de desenvolvimento do sistema Posh Code fora utilizado o método Kanban com auxílio da ferramenta Trello. O Kanban é um método de gestão que agrega os conceitos de otimização de produção de código, e segundo Boeg (2010) suas principais vantagens são:

a) começar com o que está se fazendo agora;

b) buscar mudanças incrementais e evolucionárias;

c) respeitar o processo atual, com seus papéis, responsabilidades e cargos;

d) possibilidade de visualizar tudo que está acontecendo em determinado momento;

e) limitar a capacidade do trabalho em progresso (WIP - Work in Progress), limitando a quantidade de trabalho permitido para cada etapa;

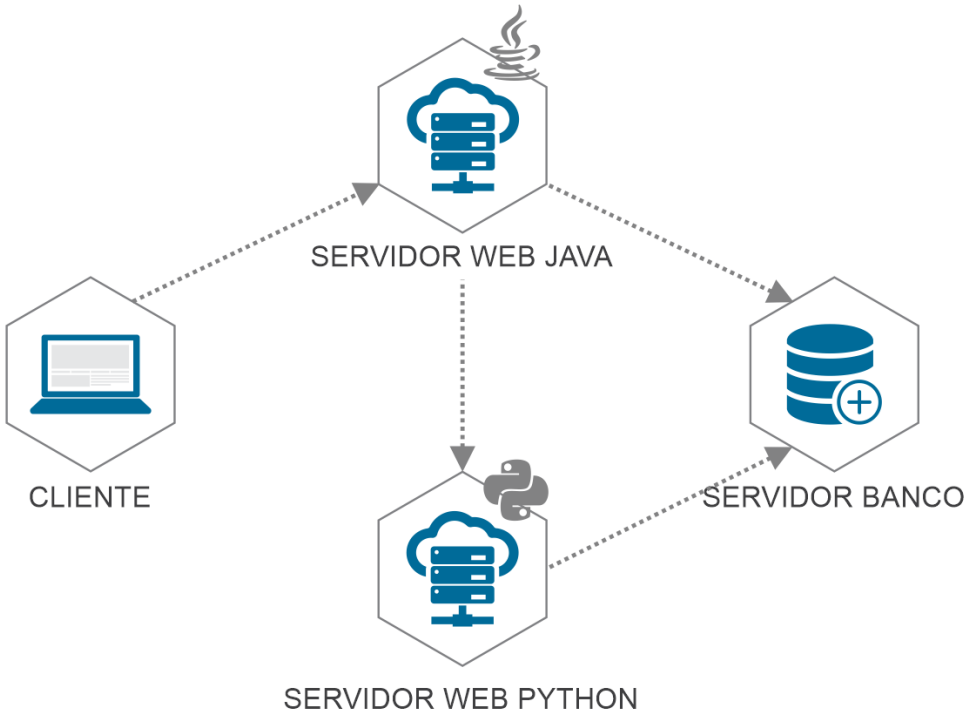
f) deixar explícita as políticas a serem acompanhadas;

g) medir e gerenciar o fluxo para que a tomada de decisões sejam bem fundamentadas, bem como poder analisar suas consequências.

O sistema em questão adota o modelo de Três Camadas, que consiste em separar o código em uma camada de apresentação, uma camada de negócios e uma camada de dados, fazendo com que cada área seja responsável somente pelos itens que correspondem à ela. Dessa forma, o sistema torna-se mais flexível, permitindo que cada parte possa ser modificada livremente.

A arquitetura proposta para o desenvolvimento do sistema do Posh Code utiliza o Spring Boot como framework web para aplicações Java, aplicando o padrão de arquitetura de software MVC (Model-View-Controller) que separa a aplicação em três camadas: modelo, tela e controlador. E também a utilização do framework web para aplicações desenvolvidas em Python, o DJango. Dividido em duas principais infraestruturas responsáveis em manter o funcionamento do sistema: o cliente e o servidor remoto. A arquitetura do sistema é ilustrada pelo fluxograma na Figura 10.

Figura 10 – Arquitetura do Sistema



Fonte: Produzido pelo autor

Cada módulo nessa arquitetura tem uma papel próprio e bem definido. O Cliente é encarregado de prover interface gráfica para interação do usuário e é pela qual envia e recebe dados. O Servidor Web Java é responsável pelo gerenciamento de recebimento de dados do cliente, e se comunica com o Servidor Web Python quando o dado for relacionado

a execução de código-fonte submetido pelo usuário, além de enviar e receber dados do servidor de banco de dados. O Servidor Web Python, por sua vez, processa os códigos-fonte e envia o resultado ao servidor de banco de dado. O banco de dados por fim é quem gerencia e persiste todos os dados da aplicação.

A vantagem de usar essa arquitetura é a possibilidade do cliente realizar tarefas paralelamente ao processamento de requisições do servidor. Desse forma o cliente não fica preso à realização de uma tarefa, pois cada parte tem autonomia de funcionamento. Outra vantagem é a possibilidade de atualizar, reparar ou mesmo substituir o servidor sem afetar os clientes.

3.4 Modelagem do Posh Code - Online Judge

A modelagem é uma das atividades principais que resultam em uma boa implementação de softwares. Através dela é possível construir modelos para comunicar a estrutura e o comportamento desejado do sistema, visualizar e controlar a arquitetura e compreender melhor o sistema a ser elaborado (DEVMEDIA, 2011).

Neste projeto optou-se por utilizar o modelo UML (Unified Modeling Language)[2] para que fosse garantido uma melhor estruturação do software, assim como a padronização no processo de modelagem a fim de deixar mais claro todo o seu funcionamento.

A seguir são apresentados os diagramas de Casos de Uso e Atividades na anotação padrão UML, com o propósito de demonstrar as principais funcionalidades do sistema Posh Code. A escolha de somente dois diagramas para a modelagem do sistema, deu-se por conta da metodologia de desenvolvimento escolhida e mencionada na seção anterior. Por se tratar de uma metologia ágil, esses diagramas são suficientes para auxiliar na implementação do sistema.

3.4.1 Diagrama Casos de Uso

De acordo com Guedes (2018), o diagrama de casos de uso representa uma ideia geral de como o sistema irá se comportar, a partir de uma linguagem simples e de fácil compreensão, e serve como base de consulta durante todo o processo de modelagem como a construção dos demais diagramas. O Casos de uso identifica os atores. Ator é tudo aquilo que pode interagir de alguma forma com o sistema, como por exemplo, usuários, outros sistemas ou até mesmo um hardware.

Nesta sub-seção é apresentado o diagrama geral de Casos de Uso do Posh Code, com o objetivo de esclarecer as principais funcionalidades e de que forma os atores interagem com o sistema.

[2] Para mais informações acesse: <https://www.uml.org/>

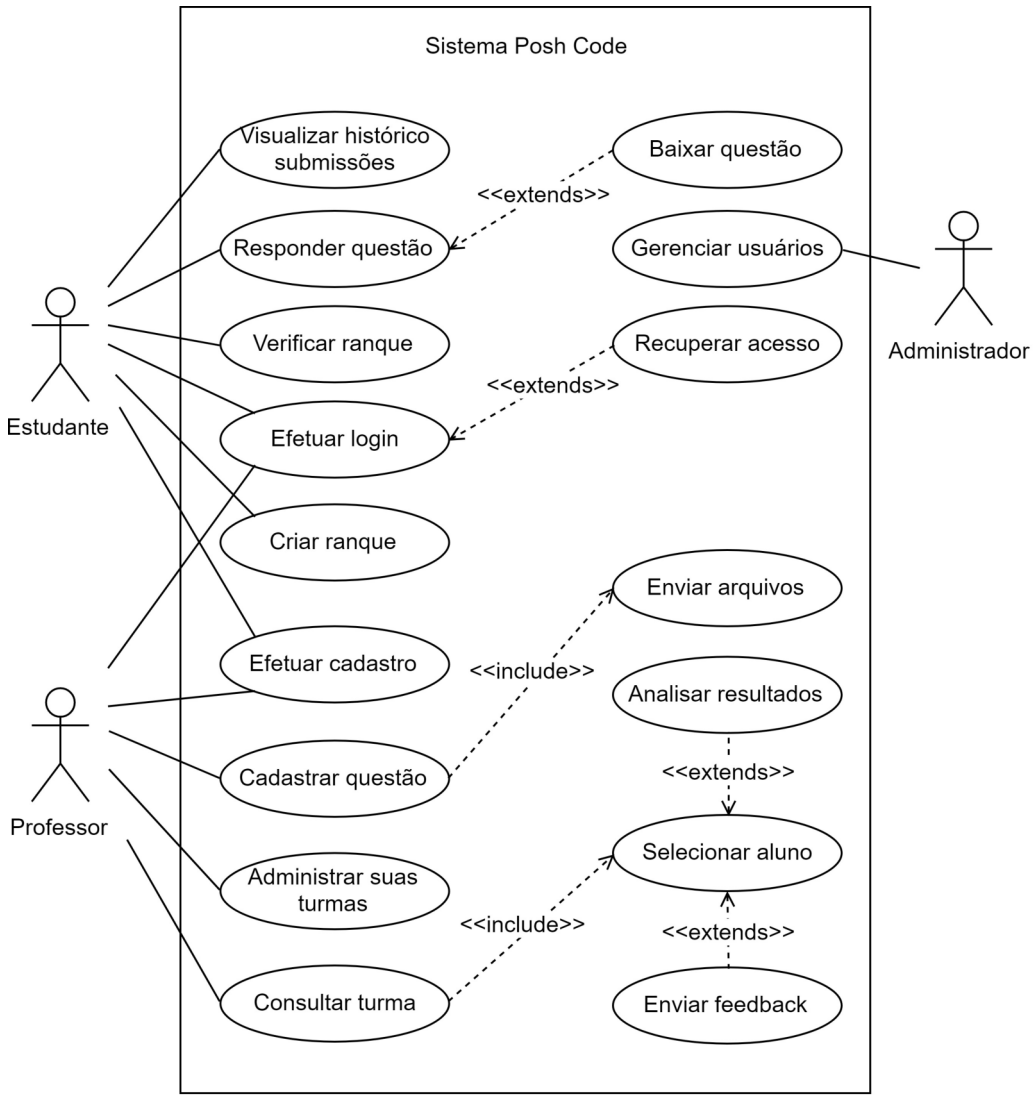
No diagrama de Casos de uso do Sistema Posh Code Figura 11 observamos três atores: estudante, professor e administrador, e todos interagem com o sistema a fim de realizar uma ação. É importante destacar que a descrição a seguir do diagrama de Casos de uso é feita de forma simplificada, deixando o detalhamento de cada função para as descrições dos diagramas de atividades na sub-seção seguinte.

Como pode-se observar na Figura 11, o ator Estudante age sobre seis casos de uso, o que significa que há seis principais funcionalidades que atendem esse ator, entre elas cabe-se destacar: (1) responder questão; (2) visualizar histórico de submissões; (3) verificar ranque e (4) criar ranques.

Por outro lado,o ator Professor atua sobre cinco principais casos de uso, sendo que compartilha as funcionalidades de “Efetuar login” e “Efetuar cadastro” com o ator Estudante, uma vez que essas funcionalidades funcionam iguais para ambos. Cabe ainda frisar as principais funcionalidades permitidas desse ator, que são: (1) cadastrar questão; (2) administrar suas turmas e (3) consultar turma. Este último caso de uso permite ao professor pesquisar estudantes, analisar seus resultados e enviar feedback.

Por fim, o diagrama de Casos de Uso: Posh Code representa o ator Administrador, que tem como papel o de gerenciar usuários. De forma lacônica, o Administrador é quem age quando uma ação do Estudante ou Professor foge do seu escopo de atuação, como por exemplo, quando o Professor precisa alterar uma informação não permitida a ele pelo sistema.

Figura 11 – Casos de uso: Sistema Posh Code



Fonte: Produzido pelo autor

3.4.2 Diagramas de Atividades

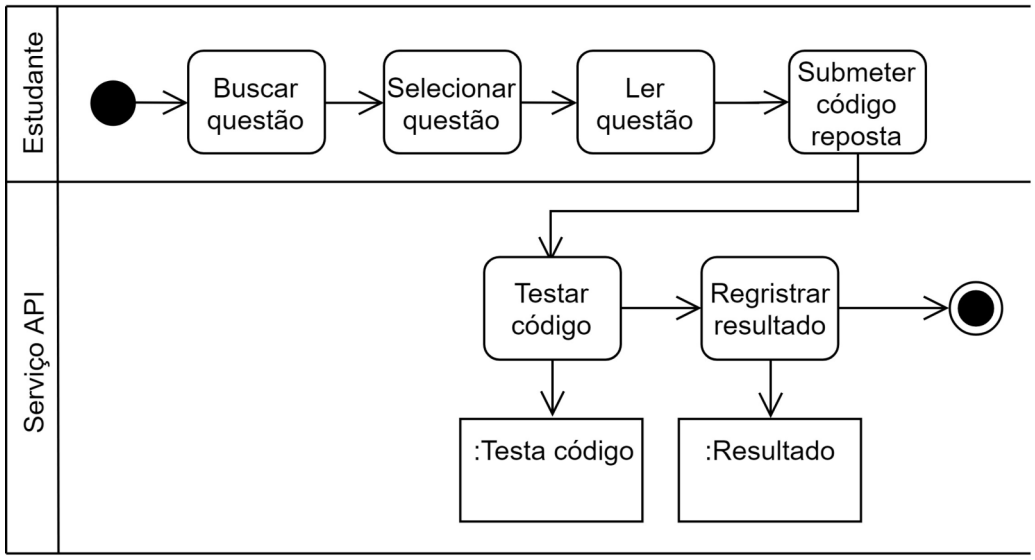
O diagrama de Atividades enfatiza a demonstração da sequência e condições para gerenciar um comportamento de um método, algoritmo ou um processo completo. Representa os métodos correspondentes às operações sobre classes. Cada atividade contem ações, mas não necessariamente sempre às representa dentro da atividade, como quando faz referência a uma atividade já modelada em outro momento (GUEDES, 2018).

A partir dos Casos de uso do Sistema Posh Code Figura 11 na sub-seção anterior foram retirados alguns casos de uso a fim de descrever com mais detalhe o seu funcionamento, através da utilização dos diagramas de atividades, sendo eles os seguintes: (a) responder questão; (b) cadastrar questão; (c) enviar feedback; (d) efetuar login e (e) efetuar cadastro.

O diagrama de atividade Responder questão Figura 12 é representado na forma

de Partição de Atividade, pois, faz-se necessário evidenciar que o fluxo do processo em determinado momento muda de ator. A atividade começa quando o ator Estudante deseja responder uma questão. Para isso, ele precisa primeiramente usar a busca, selecionar a questão escolhida, então o sistema apresenta o conteúdo da questão, e por fim submeter sua resposta. Em seguida, o processo muda seu fluxo, deixando a função de testar o código e registrar o resultado ao serviço API.

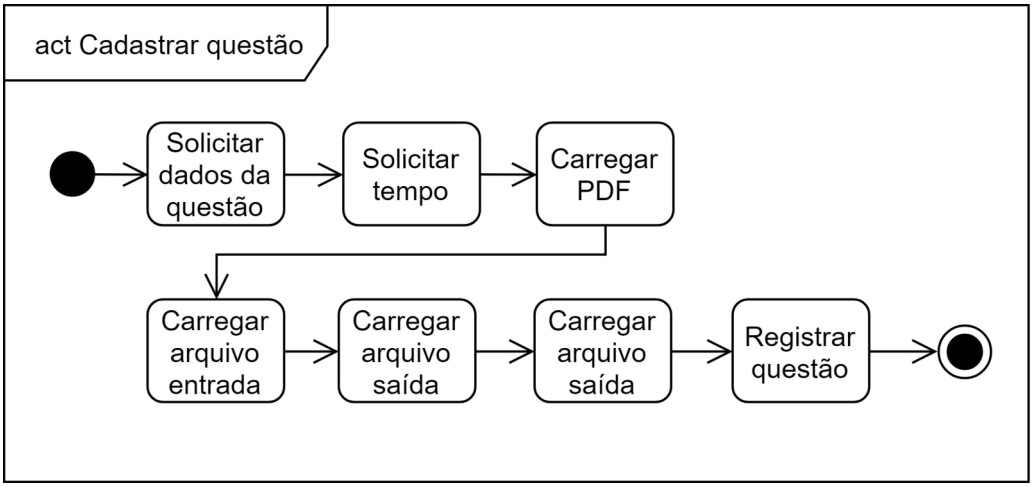
Figura 12 – Diagrama de Atividade: Responder questão



Fonte: Produzido pelo autor

Como já foi dito anteriormente, ao Professor fica o cargo de cadastrar questões, e quando for realizar essa tarefa, precisa repassar ao sistema os dados da questão, seguido do tempo máximo permitido para execução do código-resposta do estudante para aquela questão. Em seguida deve enviar o PDF com o conteúdo da questão, depois enviar dois arquivos de texto. O primeiro contendo várias entradas válidas e o segundo arquivo, as saídas equivalentes a essas entradas, para que o sistema use-os como bateria de teste para validação. Por fim, com tudo informado e carregado, a questão é registrada. O diagrama de atividade Cadastrar questão Figura 15 é quem representa o passo a passo dessas ações.

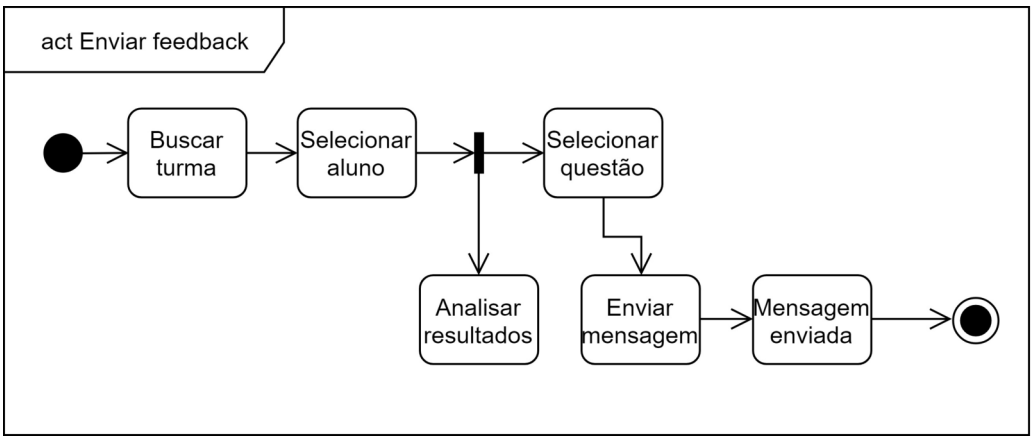
Figura 13 – Diagrama de Atividade: Cadastrar questão



Fonte: Produzido pelo autor

Outra atividade muito importante é a de dar feedback ao estudante. Para realizar essa tarefa, conforme podemos analisar no diagrama Enviar feedback Figura 14, o Professor inicialmente busca uma turma, em seguida seleciona um estudante dessa turma, nesse momento, o nó de bifurcação/união demonstra que ele tem a opção de analisar de forma geral os resultados desse estudante, conforme é representado no nó de ação Analisar resultados, ou selecionar uma submissão do estudante, e por fim mandar uma mensagem a cerca dessa questão selecionada.

Figura 14 – Diagrama de Atividade: Enviar feedback

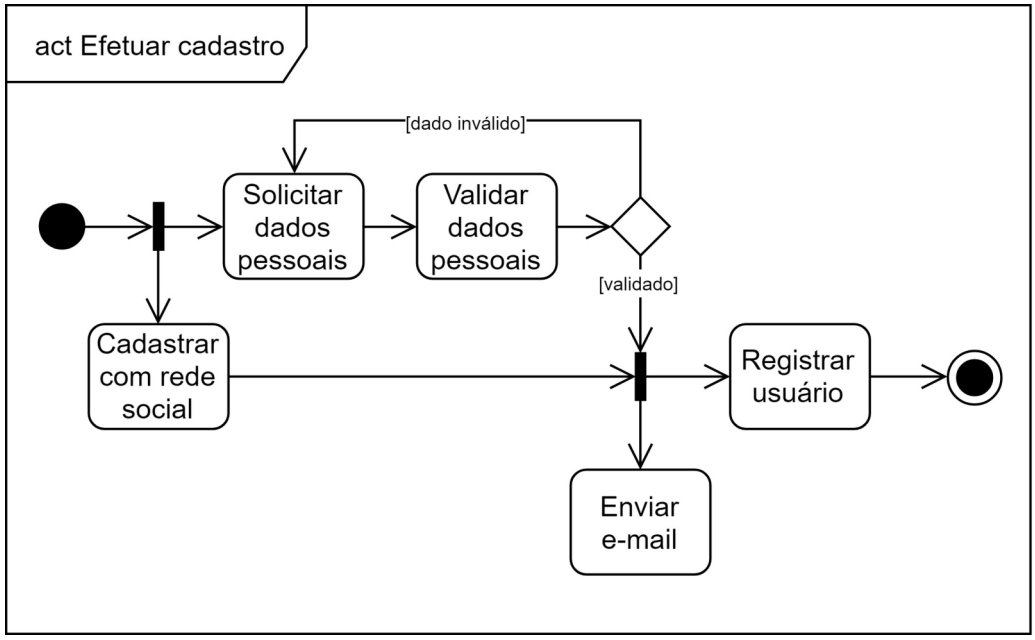


Fonte: Produzido pelo autor

O diagrama de atividade Efetuar cadastro é atuado tanto pelo estudante quanto pelo professor. O nó de bifurcação/união no diagrama representa a opcionalidade de se cadastrar do modo tradicional ou através dos dados de sua rede social. Pela via tradicional, ao usuário é solicitado os dados e-mail e senha, em seguida, conforme é demonstrado no nó de ação Validar dados pessoas, é analisado se o usuário colocou um e-mail no padrão

de e-mails e se sua senha compreende a obrigação de requisitos para senhas. Continuando há um nó de decisão que representa as duas opções pós-validação, caso um dos dados não contemple o exigido, então a ação apresenta mensagem de erro e volta para o nó de ação Solicitar dados pessoais. Caso o dado seja validado, então é enviado um e-mail de confirmação para esse usuário e em seguida o mesmo é registrado no sistema. Pela via Cadastrar com rede social, o sistema se comunica com a rede social escolhida a ação é direcionada ao segundo nó de bifurcação/união, e os passos seguintes já foram mencionados.

Figura 15 – Diagrama de Atividade: Efetuar Cadastro

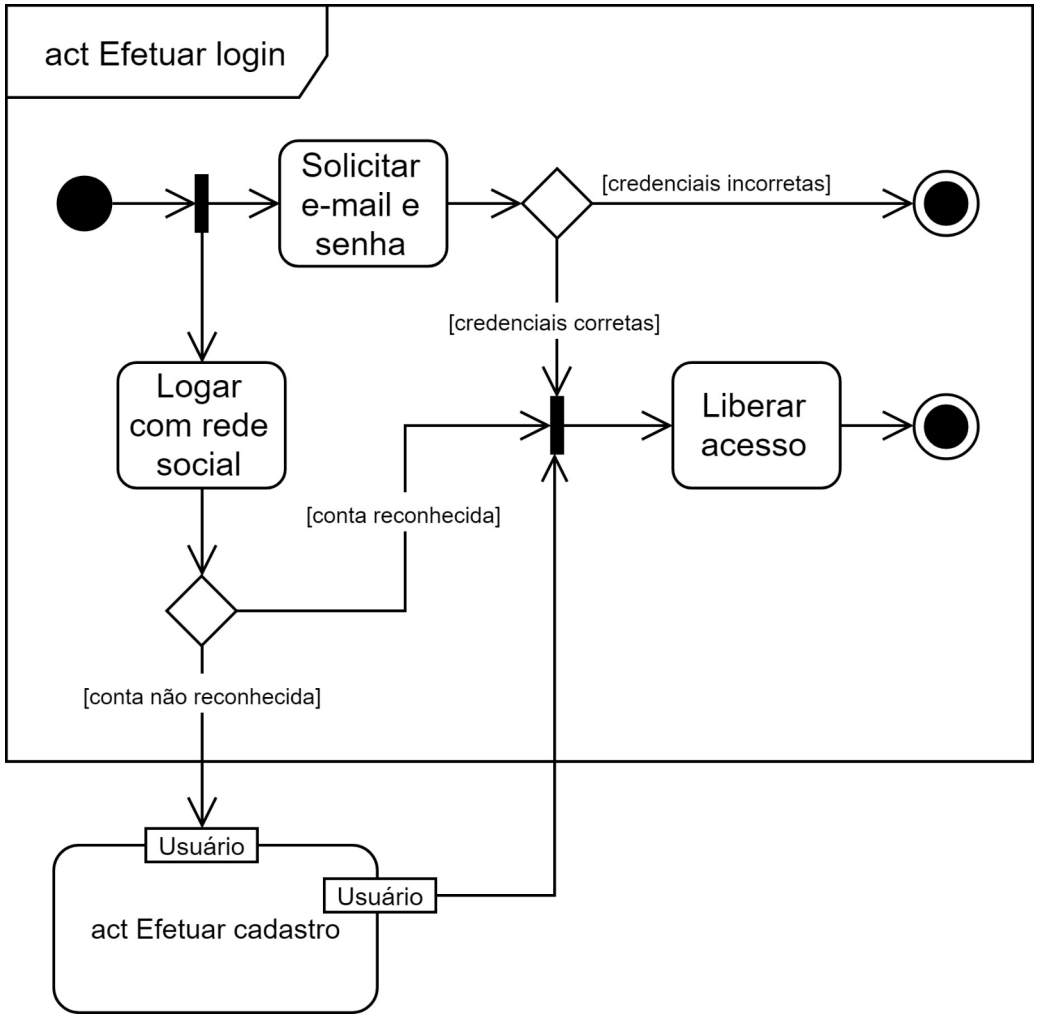


Fonte: Produzido pelo autor

Por último, apresenta-se o diagrama de atividade: Efetuar login. Novamente é uma opção disponível aos dois principais atores do sistema, o estudante e o professor. O diagrama começa com um nó de bifurcação/união representando a opção do cliente logar com sua rede social ou colocando e-mail e senha. Se o usuário escolher a primeira opção, o sistema verificará se aquela conta da rede social está associada ao sistema, demonstrado conforme nó de decisão, caso contrário, o sistema libera o acesso. Porém, quando a conta da rede social não tiver vínculo com o sistema, então a ação é redirecionada para o diagrama de atividade Efetuar cadastro, já descrito no parágrafo anterior.

O nó de ação Solicitar e-mail e senha representa o caso em que o usuário deseja fazer login informando seu e-mail e sua senha. Então após informação o nó de decisão representa a validação desses dados, caso um dos dados esteja incorreto, a atividade acaba, e caso, esteja correto, então o acesso ao sistema é liberado.

Figura 16 – Diagrama de Atividade: Efetuar Login



Fonte: Produzido pelo autor

Como pode ser observado, não foi abordado os diagramas das atividades Verificar *ranking* e Verificar histórico, mencionados no diagrama de casos de uso Figura 11. A explicação para isso é que ambas as ações são triviais, pois, essas informações já aparecem no próprio perfil do usuário, cabendo ao mesmo somente clicar no link de ranking ou histórico para ter acesso a uma tabela com essas informações.

3.5 Seção do Usuário

Esta seção descreve de forma resumida como fora utilizado as tecnologias e de que forma elas se relacionam na implementação do sistema Posh Code. Ainda neste seção é apresentado o resultado da implementação do sistema através da apresentação das telas do sistema.

3.5.1 Implementação

No Capítulo 2 deste trabalho são apresentadas tecnologias utilizadas para o desenvolvimento do sistema Posh Code. Nos próximos parágrafos desta subseção são apresentadas de forma simplificada de que maneira essas ferramentas foram utilizadas e como elas se comunicam.

A base do desenvolvimento do sistema Posh Code foi o framework Spring Boot. A partir das abstrações feitas pelo Spring Boot, no que diz respeito à segurança do site, com Spring Security, a comunicação com o banco de dados, com o Spring Data, foi possível desenvolver as lógicas para o funcionamento de todo o cliente web.

O front-end foi desenvolvido com a utilização do framework Bootstrap. O que permitiu tornar o site responsivo e com aparência mais agradável, garantido uma boa disposição dos elementos do site em qualquer resolução de tela.

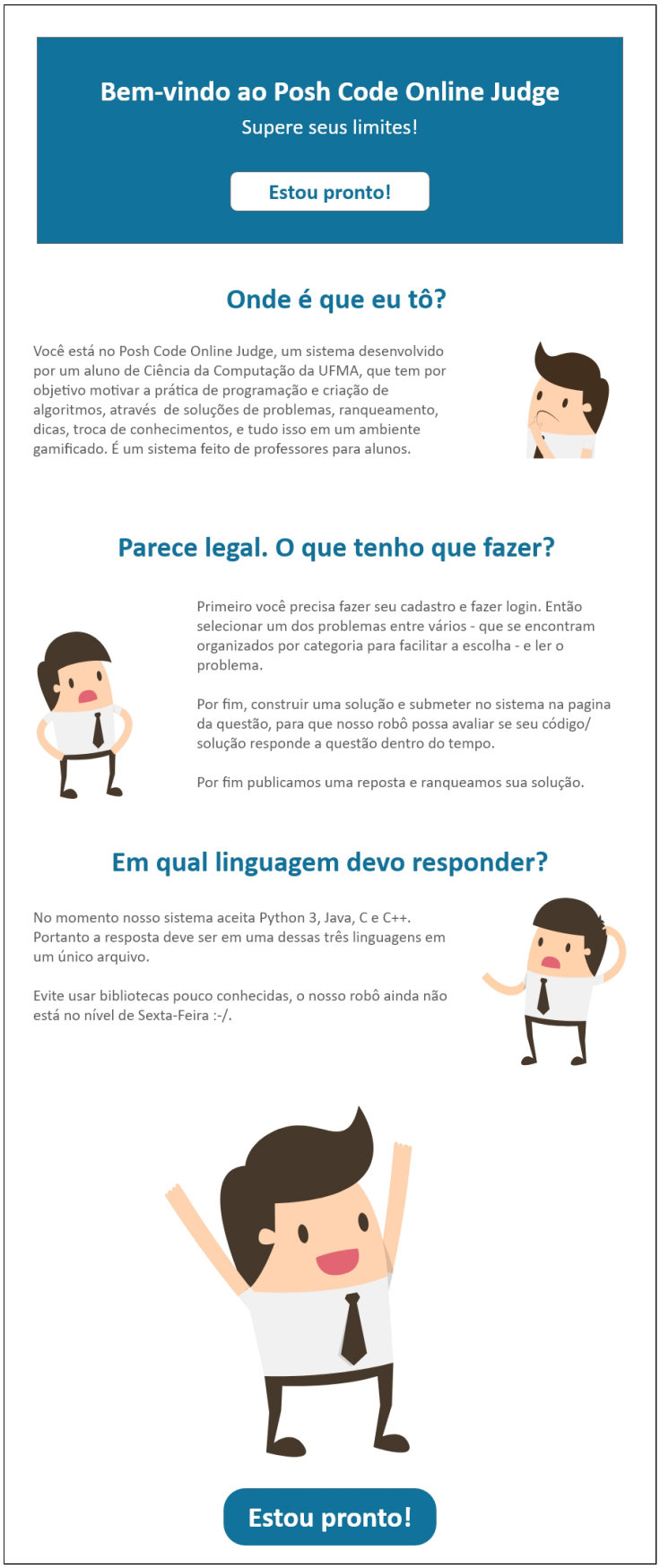
Por fim, cabe destacar a utilização do Python, com a ajuda do framework Django para a construção do Serviço API, que é requisitado sempre que é preciso testar o código-solução do estudante. Para isso, o client web chama à API, passando o código-solução e os arquivos de entrada e saída da questão para serem usados na avaliação, e então é criado uma cópia de um container, utilizando o PyDocker, com toda a configuração necessária para executar e testar o resultado do código. Em seguida a própria API manda o resultado da avaliação da questão ao banco de dados, e então o client web faz a leitura desse resultado e apresenta ao usário.

3.5.2 Telas do sistema

Ao acessar o endereço do sistema Posh Code, em www.poshcode.com.br, a primeira tela que o usuário tem contato é a de lançamento Figura 17. Trata-se de uma tela de apresentação do Posh Code, onde, através de um sistema de perguntas e respostas é apresentado um pouco da história do Posh Code e a sua finalidade. Em seguida indica-se o que deve ser feito para poder utilizar o sistema. Por fim, é informado quais as linguagens de programação aceitáveis, além de dá outros avisos menores. Todos os textos dessa tela são acompanhados da ilustração de uma pessoa, na tentativa de chamar a atenção do usuário e mantê-lo interessado na leitura.

Como pode ser observado, essa tela possui um botão com o nome “Estou pronto!”, e ao clicá-lo, o usuário é direcionado à tela de login. Cabe ressaltar que praticamente todas as páginas têm no seu cabeçalho os menus: questões; ranks; fórum; contato e login/sair, e por ser de fácil entendimento, a explicação dos mesmos é omitida neste texto.

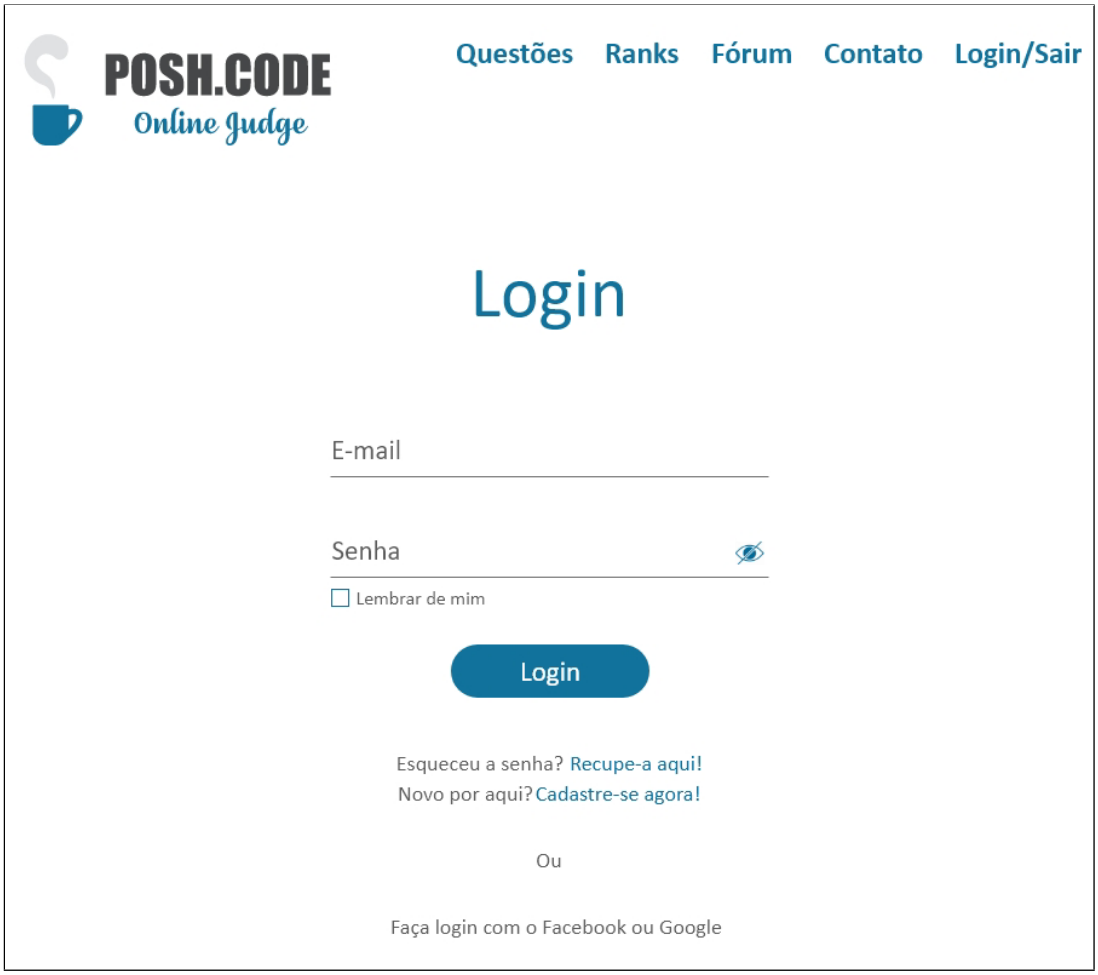
Figura 17 – Tela de lançamento



Fonte: Produzido pelo autor

Na tela de login Figura 18, é mostrado os campos para preenchimento de e-mail e senha para se ter acesso ao sistema. Ao clicar no símbolo de olho barrado, a senha digitada passa a ser visível. O usuário pode salvar suas credenciais de acesso no navegador marcando a opção "Lembrar de mim". Outra opção é fazer login com uma conta Facebook ou conta Google clicando no link na parte inferior da tela. É possível ainda, recuperar as credenciais de acesso em "Recupere-a aqui!" ou ainda fazer um cadastro no site caso seja um usuário novo, em "Cadastre-se agora!". Ao clicar no botão de login o sistema verificará se as informações de acesso são válidas e caso seja, o usuário é direcionado para a página do perfil do usuário.

Figura 18 – Tela de login



Fonte: Produzido pelo autor

Na tela de cadastro Figura 19, o usuário só precisa informar um e-mail e uma senha, e confirmar ambos digitando-os mais uma vez. Em seguida o usuário deve clicar no botão "Cadastre-se" para efetuar o cadastro. O sistema então validará as informações e encaminhará para a tela de login. Nesse primeiro cadastro, o sistema só exige e-mail e senha para que o usuário se sinta mais a vontade e só depois atualize seu cadastro. Para

usuários-professor é preciso posteriormente ao cadastro mandar mensagem ao administrador informando que sua conta é do tipo professor.

Figura 19 – Tela de cadastro

Fonte: Produzido pelo autor

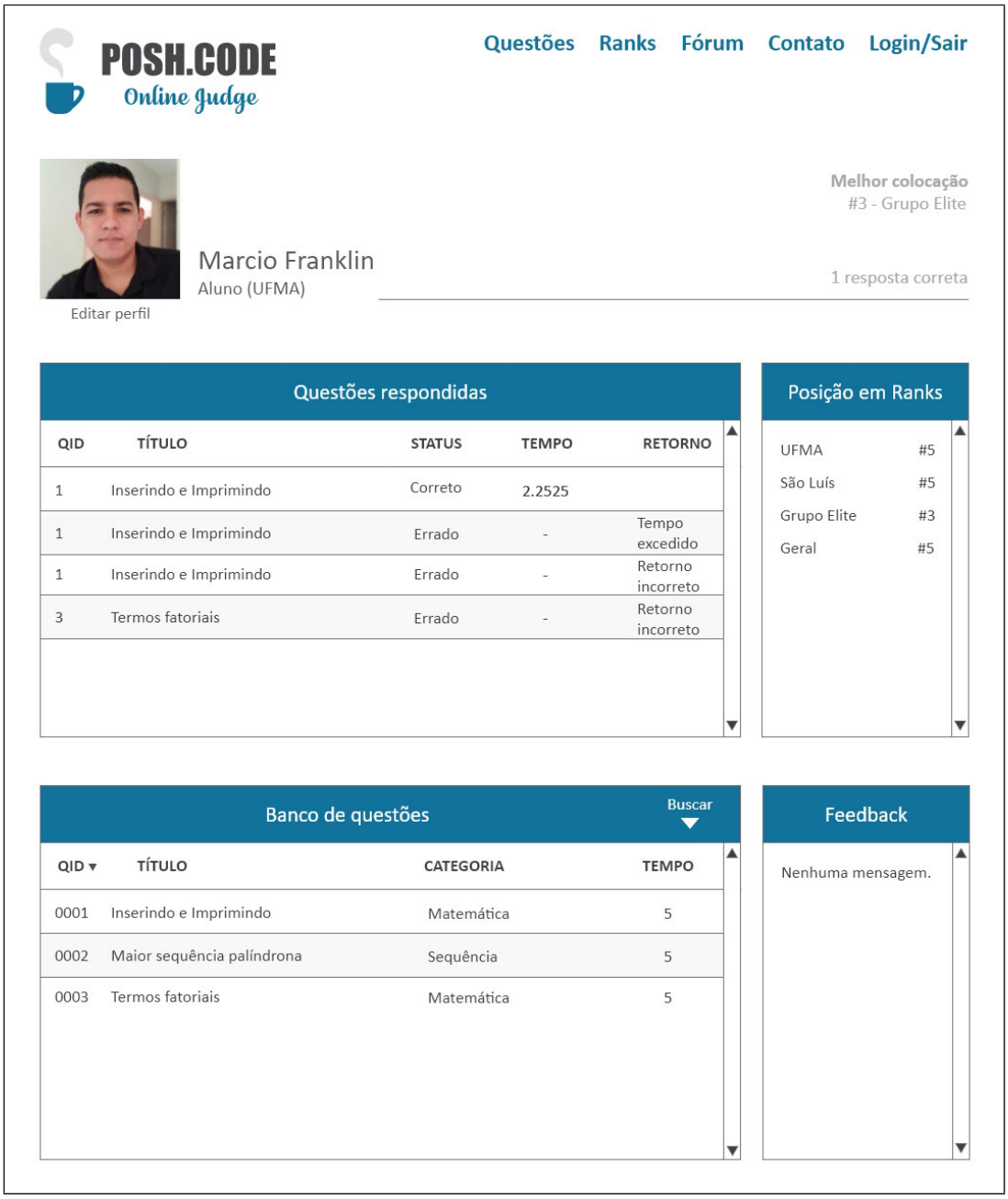
Como foi dito, no momento do cadastro do usuário, o mesmo só precisa informar e-mail e senha válidos. Porém, para que o usuário possa usufruir de todas as funcionalidades do sistema, o mesmo deve atualizar seu cadastro na tela Atualização do cadastro Figura 20. Nessa tela, o usuário deve informar um apelido (*nick*), seu nome e seu sobrenome, o curso que está fazendo ou que fez, uma turma, caso faça parte de alguma - podendo escolher mais de uma -, a instituição onde estuda e a cidade onde reside. O usuário tem ainda a opção de enviar uma foto para o seu perfil. Para confirmar a alteração o usuário deve clicar no botão “Salvar”, então o mesmo é direcionado para a tela do perfil do usuário.

Figura 20 – Tela de atualização de cadastro

Fonte: Produzido pelo autor

A tela do perfil do estudante Figura 21 como o próprio nome sugere, apresenta o perfil do estudante, onde é possível ver um histórico das suas questões respondidas, a sua posição nos rankings que participa, a lista de questão em um resumo do banco de questões ordenado por questões mais recente, e ainda um chat com os feedbacks recebido dos professores.

Figura 21 – Tela do perfil do estudante



Fonte: Produzido pelo autor

Parecida com a tela do perfil do estudante, a tela do perfil do professor Figura 22 mostra um botão para cadastro de questões, uma tabela com a lista de questões cadastradas, um botão para editar seu perfil e ainda um chat com as mensagens de feedbacks enviadas.

Figura 22 – Tela do perfil do professor

Fonte: Produzido pelo autor

Para adicionar novas questões, a partir do perfil de usuário, o professor deve clicar no botão Cadastrar questão, então será direcionado para a tela cadastro de questão Figura 23 onde primeiramente deve informar o título da questão, em seguida há a opção de escrever uma observação sobre a questão. Continuando o professor deve definir a categoria da questão e qual o tempo máximo para um código-resposta em execução solucionar a questão. Por fim deve clicar no botão “PDF” para adicionar o arquivo com o conteúdo da questão em formato PDF, clicar no botão “Entradas” para adicionar arquivo de texto com entradas usadas para validação da questão, e clicar no botão “Saídas” para adicionar arquivo de texto com as respectivas saídas do arquivo de entrada enviado anteriormente. Após preencher todos os campos obrigatórios e carregar os arquivos, o professor deve clicar no botão “Enviar” para que a questão seja cadastrada no sistema.

Figura 23 – Tela de cadastro de questão

Fonte: Produzido pelo autor

Para responder uma questão, o usuário estudante deve primeiramente selecioná-la no banco de questão, e após selecionada, o estudante é direcionado para a tela de submissão de questão Figura 24. Nessa tela é reproduzido o PDF com o conteúdo da questão.

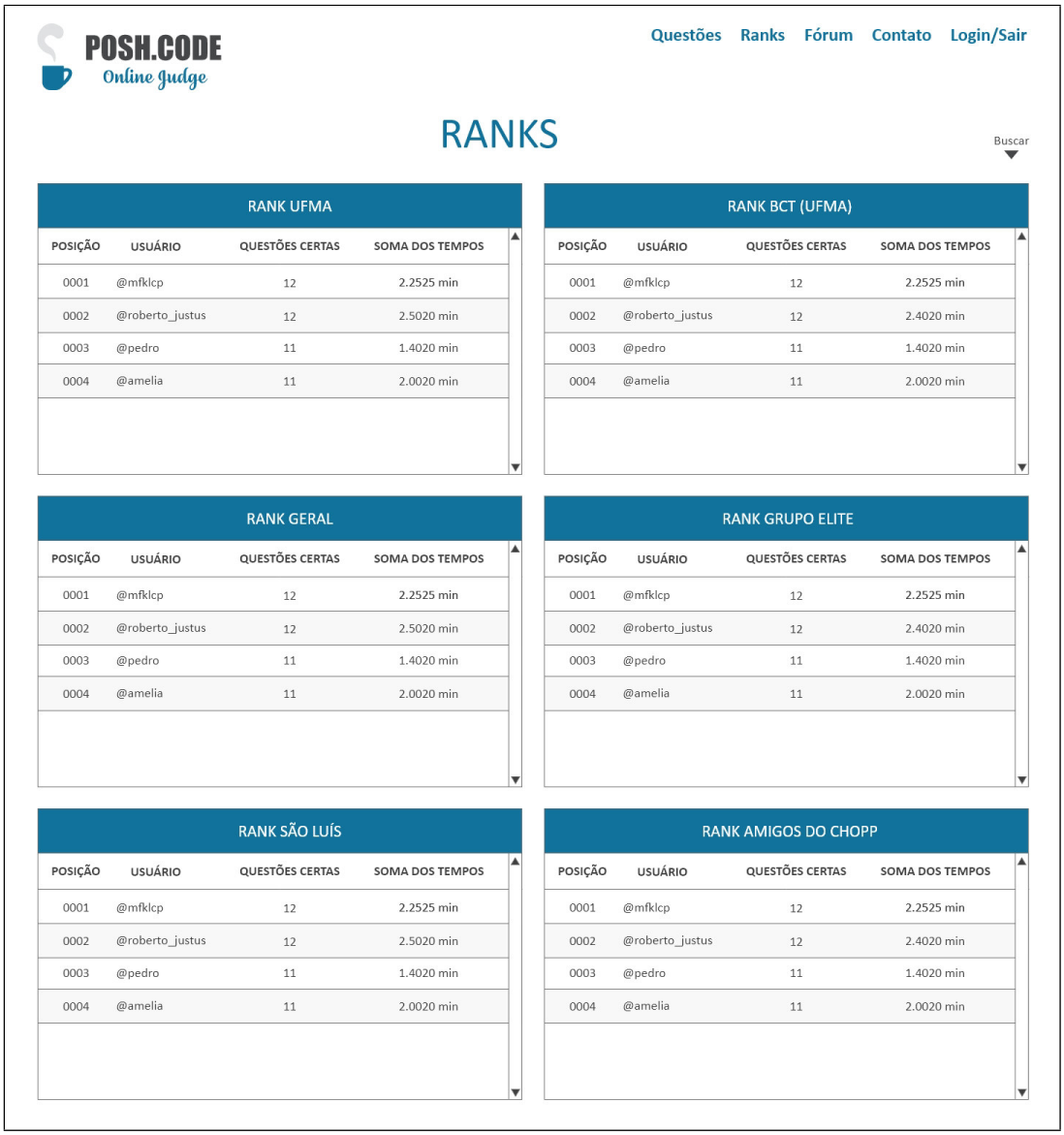
Após código-solução pronto, o estudante deve clicar no botão "Selecionar arquivo" para carregar ao sistema a sua solução, que precisa ser um único arquivo em um dos formatos: Python 3, Java, C e C++. Respeitado as condições, o estudante deve clicar no botão "Submeter Solução", nesse momento o sistema compila o código, executa-o, e passa como entrada do executável o arquivo-entrada previamente cadastrado, por fim, compara a saída do código do estudante com o arquivo-saída cadastrado previamente, então se as saídas forem exatamente iguais, o sistema apresenta o resultado *true*, significando que a resposta está correta. Mas caso as saídas sejam diferentes, caracteriza-se resposta incorreta, e o usuário recebe a mensagem *false*, junto do retorno do compilador.

Figura 24 – Tela de submissão de solução

Fonte: Produzido pelo autor

Por fim, tem-se a tela de rankings Figura 25, que é carregada quando um usuário clica no link superior Ranks. Essa tela consiste de várias tabelas com os rankings que o usuário participa. E tem o objetivo de permitir que o estudante analise o seu desempenho e possa se inspirar a melhorar suas colocações ou seus tempos.

Figura 25 – Tela de rankings



Fonte: Produzido pelo autor

4 Relato de Experiência

Este capítulo apresenta as estatísticas de utilização do sistema Posh Code - Online Judge e a discussão dos resultados obtidos, através de questionários de avaliação, a partir da aplicação da técnicas da gamificação.

4.1 Cenário

O cenário escolhido para teste inicial do sistema foi a disciplina de Projeto e Análise de Algoritmos do Programa de Pós Graduação da área de computação da Universidade Federal do Maranhão. A disciplina é obrigatória e portanto essencial para a formação dos estudantes, dispondo de uma carga horária de 60 horas, com turma de 24 estudantes e a execução ocorreu durante o período 2019.1.

A escolha desse cenário se deu ao fato da experiência da turma com sistemas online de análise automática de código, uma vez que durante toda a graduação é comum professores utilizarem esse tipo de sistema para corrigir os códigos dos alunos.

Outro motivo relevante a essa escolha foi a circunstância do orientador deste projeto ser o professor dessa disciplina, o que garantiu uma melhor comunicação com a turma.

No sistema foi criado acesso do tipo estudante para todos os 24 alunos da turma e 1 acesso do tipo professor. Porém, 9 estudantes não fizeram uso do sistema, sendo assim, 15 estudantes e 1 professor foram os que contribuíram com a avaliação do sistema através de sua utilização durante o período.

É sabido que o cenário escolhido não é o ideal para a aplicação da proposta, uma vez que os estudantes dessa turma de mestrado têm experiência com programação e já não sofrem tanto com a desmotivação comum nos primeiros períodos de curso da área de computação, como foi abordado no Capítulo 1.

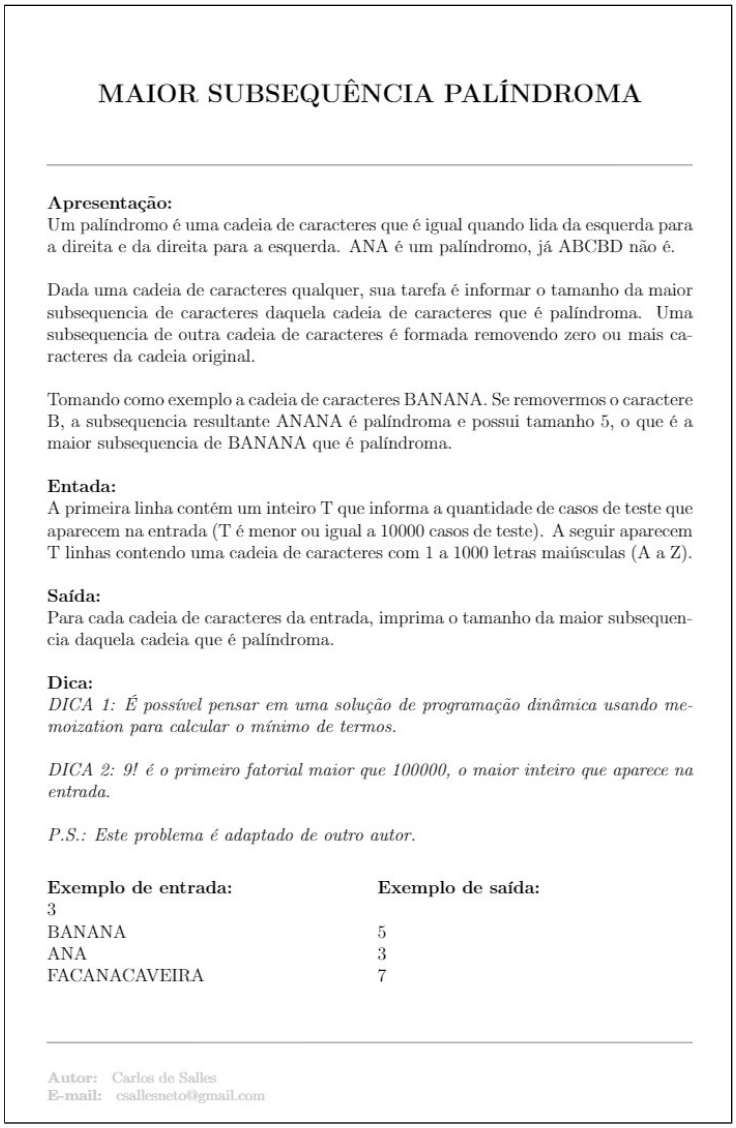
4.2 Resumo da utilização

Foi desenvolvido um *template* utilizando o sistema de tipografia LaTeX que é uma extensão da tecnologia TeX e está disponível através do link: <https://www.overleaf.com/project/5c805e0210f5971613760ab6> para padronizar a formatação do conteúdo das questões, conforme exemplo na Figura 26.

A estrutura da questão deve conter um título em caixa alta e negrito. Um corpo divido em quatro partes obrigatórias: (1) apresentação; (2) entrada; (3) saída; (4) exemplo

de entrada e de saída. E um parte opcional: (1) dica. Além disso, a página deve conter em seu rodapé o autor da questão e seu e-mail para contato, sendo este último opcional.

Figura 26 – Exemplo do padrão da formatação das questões



MAIOR SUBSEQUÊNCIA PALÍNDROMA

Apresentação:
Um palíndromo é uma cadeia de caracteres que é igual quando lida da esquerda para a direita e da direita para a esquerda. ANA é um palíndromo, já ABCBD não é.

Dada uma cadeia de caracteres qualquer, sua tarefa é informar o tamanho da maior subsequencia de caracteres daquela cadeia de caracteres que é palíndroma. Uma subsequencia de outra cadeia de caracteres é formada removendo zero ou mais caracteres da cadeia original.

Tomando como exemplo a cadeia de caracteres BANANA. Se removermos o caractere B, a subsequencia resultante ANANA é palíndroma e possui tamanho 5, o que é a maior subsequencia de BANANA que é palíndroma.

Entada:
A primeira linha contém um inteiro T que informa a quantidade de casos de teste que aparecem na entrada (T é menor ou igual a 10000 casos de teste). A seguir aparecem T linhas contendo uma cadeia de caracteres com 1 a 1000 letras maiúsculas (A a Z).

Saída:
Para cada cadeia de caracteres da entrada, imprima o tamanho da maior subsequencia daquela cadeia que é palíndroma.

Dica:
DICA 1: É possível pensar em uma solução de programação dinâmica usando memoization para calcular o mínimo de termos.

DICA 2: 9! é o primeiro fatorial maior que 100000, o maior inteiro que aparece na entrada.

P.S.: Este problema é adaptado de outro autor.

Exemplo de entrada:	**Exemplo de saída:**
3	
BANANA	5
ANA	3
FACANACAVEIRA	7

Autor: Carlos de Salles
E-mail: csallesneto@gmail.com

Fonte: Produzido pelo autor

Para a realização dos primeiros testes o professor cadastrou três questões, com os títulos: (1) maior subsequência palíndroma; (2) inserindo e imprimindo; e (3) termos fatoriais. Essas questões foram as mesmas utilizadas como atividade avaliativa na disciplina de Projeto e Análise de Algoritmos, portanto, os alunos precisavam respondê-la independentemente do projeto, mas com ajuda do sistema, puderam testar a correção de seus códigos-soluções de forma automática. A seguir são apresentados os resultados dessa utilização.

Conforme Tabela 5 pode-se observar que a questão “Maior subsequência palíndroma” foi a que teve mais submissões de código-solução e o maior índice de acerto. A questão

"Inserindo e imprimindo" teve menos submissões que a primeira questão, e índice de acerto um pouco menor. Por último, a questão termos fatoriais, teve a menor quantidade de submissões entre as três questões e também a maior quantidade de erros.

Tabela 5 – Submissões de código-solução

Questões	Tentativas	Acertos	%	Erros	%
Maior subsequência palíndroma	36	20	55%	16	45%
Inserindo e imprimindo	21	8	38%	13	62%
Termos fatoriais	16	5	31%	11	69%
Total	73	33	45%	40	55%

Fonte: Produzido pelo autor

Com relação à quantidade de questões resolvidas, a Tabela 6 mostra que quatro estudantes conseguiram resolver todas as questões. Quatro estudantes resolveram até duas questões. Seis estudantes conseguiram responder apenas uma das três questões, apesar de terem tentado por até três vezes solucionar as outras duas questões. E um estudante que tentou responder as três questões, entretanto sem resultado positivo.

Tabela 6 – Quantidade de questões respondidas

	Questões respondidas
Responderam as 3	4
Responderam até 2	4
Responderam apenas 1	6

Fonte: Produzido pelo autor

É importante salientar que foi observado que algumas avaliações de código-solução feitas pelos estudantes localmente e manualmente, acusavam de resposta correta mas o sistema Posh Code acusava de resposta errada. Na maioria das vezes isso ocorria porque a IDE utilizada pelo usuário corrigia erros comuns no código do usuário e o sistema Posh Code não. Outras vezes, o usuário enviava arquivo com formato não compatível com o sistema ou versão da linguagem de programação diferente da aceitável.

4.3 Avaliação

Para avaliar a qualidade da proposta implementada foi desenvolvido um questionário baseado na Escala Likert, que trata-se de uma escala de autorelato[1] usada para medir opiniões e determinar o nível de concordância ou discordância dos respondentes com relação a um produto (AGUIAR; CORREIA; CAMPOS, 2011).

[1] Escalas onde o voluntário avalia algo por si próprio, sem influência externa, através de respostas dadas num questionário (AGUIAR; CORREIA; CAMPOS, 2011).

O questionário foi criado utilizando a ferramenta Formulário do pacote Google Docs[2] e continha questões com as opções de respostas dispostas em uma ordem descendente onde a primeira opção indicava o maior grau de concordância e a última opção o menor grau, sempre com uma opção média para quando a resposta fosse neutra.

Foi então criado o formulário com dez questões, sendo três a respeito da experiência de uso, cinco sobre os resultados que a ferramenta pode proporcionar e mais duas com caráter complementar à pesquisa. Os respondentes foram aqueles que testaram o sistema, ou seja, os alunos da disciplina de Projeto e análise de algoritmos do Programa de Pós Graduação na área de computação da UFMA. No total 13 estudantes responderam o questionário.

As questões elaboradas para os questionário foram as seguintes:

a) o Posh Code, no geral, apresenta uma boa experiência de uso;

b) a aparência visual do Posh Code é agradável;

c) o sistema Posh Code funcionou de maneira estável durante a sua utilização;

d) o Posh Code motivou a responder as questões avaliativas;

e) o Posh Code pode ser utilizado nas disciplinas de programação como forma de motivação;

f) o sistema de ranqueamento proporciona um ambiente de disputa saudável, que como consequência melhora o rendimento do estudante;

g) receber feedback do professor ajuda na solução das questões;

h) o Posh Code cumpre com a sua finalidade;

i) as respostas das avaliações de código feitas pelo Posh Code são perfeitamente corretas;

j) continuarei usando o Posh Code.

E as opções de respostas para cada questão foram: (1) concordo totalmente (CT); (2) concordo parcialmente (CP); (3) indiferente (IN); (4) discordo parcialmente (DP); e (5) discordo totalmente (DT). Assim, cada opção, a partir do eixo neutro, sobe ou desce de grau de qualidade com uso dos marcadores verbais: totalmente e parcialmente. Organizar dessa maneira permite que o respondente mostre mais especificamente o quanto ele concorda ou discorda da questão.

Para análise dos resultados da pesquisa de opinião, os dados foram dispostos em 3 tabelas e a partir delas produziu-se gráficos de barra e serão discutidos a seguir.

[2] Para mais informações acesse: <https://docs.google.com/>

A Tabela 7 apresenta o resultado da resposta para cada uma das três primeiras questões. E a Figura 27 apresenta o gráfico desse mesmo resultado em forma de porcentagem.

Com relação à primeira questão, a partir do que foi apresentado no gráfico, podemos inferir que a experiência de uso do sistema foi em geral boa, uma vez que todos concordaram parcialmente ou totalmente com essa questão. Cabe observar que provavelmente os 54% que concordam apenas parcialmente com isso tiveram um ou outro problema com o sistema, o que era esperado de um sistema em sua primeira versão.

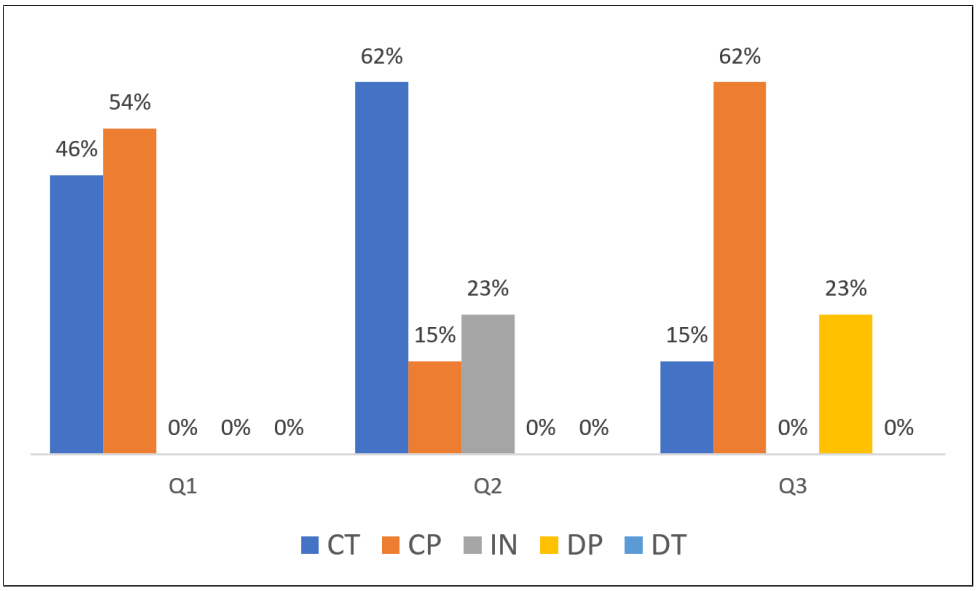
A maioria respondeu que a aparência visual do Posh Code é agradável, apesar de 23% ter uma opinião indiferente à ela. Já com relação à estabilidade do sistema, infere-se que 23% tiveram problema de instabilidade com sistema, outro resultado já esperado, pois o sistema em alguns momentos ficou offline, uma vez por queda do servidor, outras vezes por atualização do sistema, porém, esses problemas foram sempre rapidamente resolvidos.

Tabela 7 – Experiência de uso

Questão	CT	CP	IN	DP	DT
O Posh Code, no geral, apresenta uma boa experiência de uso	6	7	0	0	0
A aparência visual do Posh Code é agradável	8	2	3	0	0
O sistema Posh Code funcionou de maneira estável durante a sua utilização	2	8	0	3	0

Fonte: Produzido pelo autor

Figura 27 – Experiência de uso



Fonte: Produzido pelo autor

No tocante ao sentimento de aumento de motivação para solucionar as questões avaliativas aplicadas pelo professor conforme Tabela 8 e Figura 28, 62% dos respondentes

concordam que o Posh Code lhes favoreceu essa melhora, 31% concordam parcialmente e 8% não têm certeza se isso aconteceu. No que se refere à opinião a cerca do Posh Code ser utilizado nas disciplinas de programação como forma de motivação, a grande maioria tem convicção que sim, que pode ser usado.

Com relação às técnicas de gamificação utilizadas no Posh Code, um pouco mais da metade dos respondentes afirmam que o sistema de ranqueamento pode influenciar na melhora do rendimento do estudante, e o restante fica divido entre os que parcialmente acham isso e os que não sabem dizer. Porém quando a questão é com relação à efetividade do sistema de feedback, a resposta é unânime, todos concordam completamente com isso.

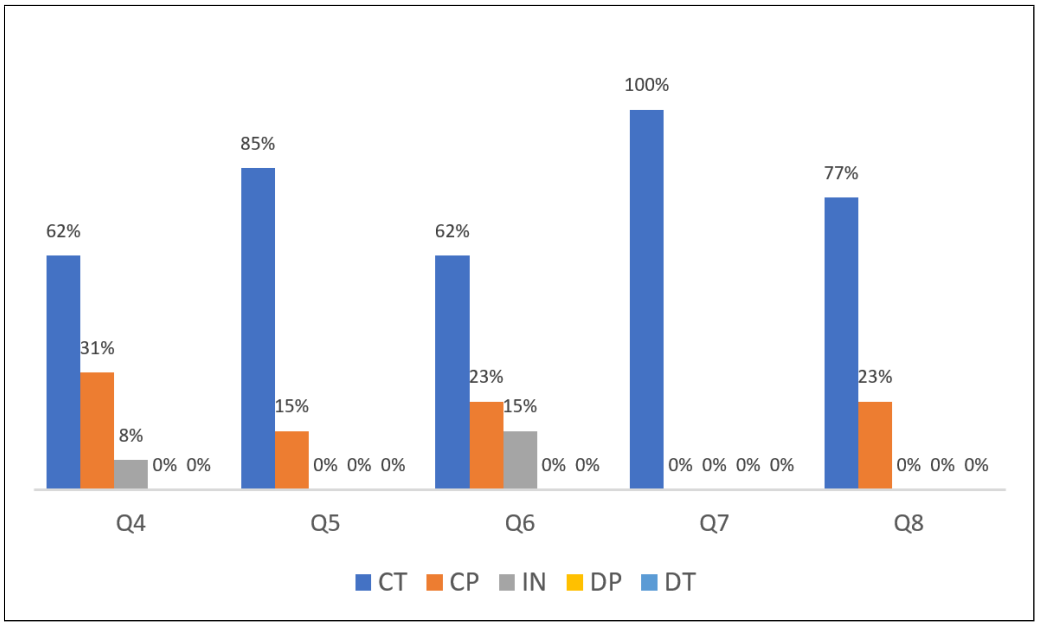
A questão 8 visa descobrir a opinião do estudante com relação ao cumprimento da proposta estabelecida pelo Posh Code, onde 77% concordam totalmente e 23% concordam parcialmente, o que pode significar que 1/3 dos respondentes consideram que o sistema pode melhorar para se tornar mais eficaz.

Tabela 8 – Resultados que o Posh Code pode proporcionar

Questão	CT	CP	IN	DP	DT
O Posh Code lhe motivou a responder as questões avaliativas?	8	4	1	0	0
Posh Code pode ser utilizado nas disciplinas de programação como forma de motivação?;	11	2	0	0	0
O sistema de ranqueamento proporciona um ambiente de disputa saudável, que como consequência melhora o rendimento do estudante?	8	3	2	0	0
Receber feedback do professor ajuda na solução das questões?	13	0	0	0	0
Você concorda que o Posh Code cumpre com a sua finalidade?	10	3	0	0	0

Fonte: Produzido pelo autor

Figura 28 – Resultados que o Posh Code pode proporcionar



Fonte: Produzido pelo autor

As duas últimas questões, apresentadas pela Tabela 9 e Figura 29, buscam levantar dados complementares, no sentido de saber mais da opinião do respondente a respeito do sistema Posh Code. A primeira delas, afirma que o resultado pós avaliação de código-resposta do Posh Code é totalmente correto, e 54% respondem que confiam na avaliação feita pelo sistema, o que é um número expressivo quando é levado em consideração que muitos estudantes tiveram problemas com isso, conforme é comentado no sexto parágrafo da seção 4.2 deste trabalho.

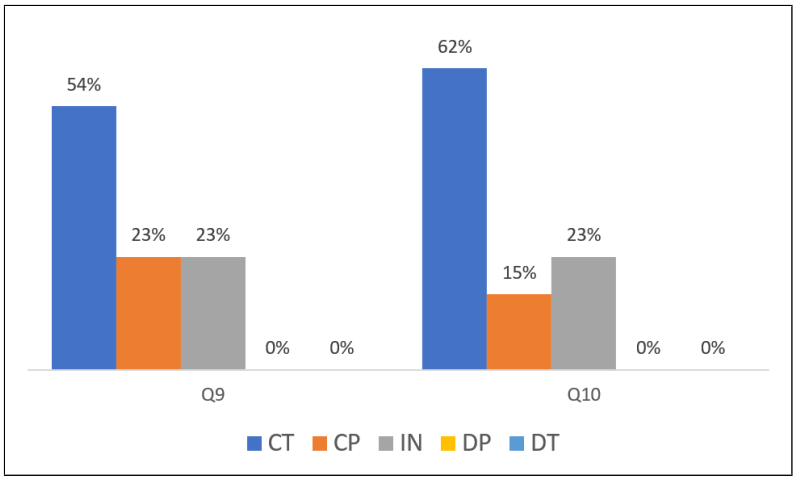
Por fim, a última questão tem o objetivo de saber se o respondente pretende continuar usando o Posh Code como forma de avaliação automática de código. A maioria responde com convicção que sim, o que pode significar que o Posh Code atende os requisitos de um sistema online judge e que a experiência obtida pelo respondente durante todo o tempo de uso foi boa.

Tabela 9 – Questões complementares

Questão	CT	CP	IN	DP	DT
Concorda que as respostas das avaliações de código feitas pelo Posh Code são perfeitamente corretas?	7	3	3	0	0
Você vai continuar usando o Posh Code?	8	2	3	0	0

Fonte: Produzido pelo autor

Figura 29 – Questões complementares



Fonte: Produzido pelo autor

5 Conclusões

Através de dados levantados sobre o índice de abandono de estudantes em cursos de nível superior da área de computação no Brasil, observou-se que mais da metade dos novos ingressantes acabam por abandonar o curso. A partir disso, este trabalho veio propor a utilização da gamificação aplicada ao ensino-aprendizado de programação através de um sistema online de análise automática de código, com objetivo de motivar os alunos e consequentemente melhorar esse índice.

Foi desenvolvido então o sistema web de análise automática de código para dar apoio aos estudantes no estudo de programação, e recebera o nome de Posh Code - Online Judge. O mesmo provém o acesso a dois tipos principais de usuários: professor e estudante. Onde o professor tem o papel de gerenciar suas turmas, cadastrar questões e dar feedback ao estudante. E o estudante é quem responde as questões cadastradas pelo professor.

A partir da implementação desse sistema foi possível testar na prática se os elementos da gamificação utilizados neste trabalho influenciaram no rendimento do estudante durante o período de utilização. E para constatar isso, foi aplicado questionário de caráter qualitativo após o uso sistema pelos estudantes, onde responderam questões opinativas a cerca de sua experiência de uso e sobre a possível motivação o que o mesmo poderia ter-lhe proporcionado.

Segundo o resultado da pesquisa de satisfação, em geral, os estudantes concordaram que as técnicas da gamificação serviram de motivadoras para o estudo-aprendizado de programação durante o tempo em que a utilizaram. Afirmaram ainda que o sistema cumpriu com o seu objetivo e que pode ser utilizado em disciplinas de estudo de programação.

Com essas observações, podemos afirmar que o Posh Code - Oline Judge cumpre com a sua proposta, uma vez que os estudantes que participaram da utilização e em seguida da avaliação do sistema através do questionário, o avaliaram de forma positiva. Vale lembrar que ao sistema ainda cabe melhorias e evolução, pois outros elementos relevantes da gamificação, como por exemplo, sistema de recompensa, sistema de pontuação, organização por níveis, etc, poderia ter grande poder influenciador no engajamento do estudante com o sistema e consequentemente no estudo e aprendizado de programação. Por fim, seria interessante melhorias também no sentido de aumentar a segurança do sistema, e também a inclusão de editor do conteúdo das questões, sistema de avaliação de plágio, entre outras.

Referências

AGUIAR, B.; CORREIA, W.; CAMPOS, F. Uso da escala likert na análise de jogos. *Anais do X Simpósio Brasileiro de Games e Entretenimento Digital, 07-09 de novembro de 2011 Salvador*, 2011. 53

BOEG, J. Kanban em 10 passos. *Tradução de Leonardo Campos, Marcelo Costa, Lúcio Camilo, Rafael Buzon, Paulo Rebelo, Eric Fer, Ivo La Puma, Leonardo Galvão, Thiago Vespa, Manoel Pimentel e Daniel Wildt. C4Media*, 2010. 32

BRITO, A. L. d. S. *Level up: uma proposta de processo gamificado para a educação.* Dissertação (Mestrado) — Brasil, 2017. 30

BUSARELLO, R. I. *Gamification: princípios e estratégias.* [S.l.]: Pimenta Cultural, 2016. 30

CAMPOS, C. P. de; FERREIRA, C. E. Boca: um sistema de apoio a competições de programação. 2004. 16, 17, 18

COSTA, T. H. Análise dos problemas enfrentados por alunos de programação. 2013. 13, 14

DETERDING, S. et al. Gamification: Toward a definition. In: VANCOUVER BC, CANADA. *CHI 2011 gamification workshop proceedings.* [S.l.], 2011. v. 12. 28, 29

DEVMEDIA. *Modelagem de software com UML.* 2011. <https://www.devmedia.com.br/modelagem-de-software-com-uml/20140>. Acesso em: 04-06-2019. 34

EFRON, B.; TIBSHIRANI, R. J. *An introduction to the bootstrap.* [S.l.]: CRC press, 1994. 25

FARDO, M. L. A gamificação aplicada em ambientes de aprendizagem. *RENOTE*, v. 11, n. 1, 2013. 29, 30, 31

FARIA, T. Java ee 7 com jsf, primefaces e cdi. *SI sn*, 2015. 22

FERNANDES, C. W. R.; RIBEIRO, E. L. P. Games, gamificação e o cenário educacional brasileiro. *CIET: EnPED*, 2018. 30

GEE, J. P. Bons video games e boa aprendizagem. *Perspectiva*, v. 27, n. 1, p. 167–178, 2009. 30

GSTI, P. *O que é Django.* 2016. <https://www.portalgsti.com.br/django/sobre/>. Acesso em: 24-06-2019. 26

GUEDES, G. T. *UML 2-Uma abordagem prática.* [S.l.]: Novatec Editora, 2018. 34, 36

INEP, I. N. D. E. E. P. E. A. T. *Sinopse Estatística da Educação Superior 2016, Brasília.* 2017. <http://inep.gov.br/sinopses-estatisticas-da-educacao-superior>. Acesso em: 19-06-2019. 13

JOHNSON, R. *Expert one-on-one J2EE design and development*. [S.l.]: John Wiley & Sons, 2004. 22

JOHNSON, R.; HOELLER, J. *Expert One-on-one J2EE Development without EJB*. [S.l.]: Pequim: Imprensa da Indústria Mecânica, 2003. 22

JOSE, A. B. Gamificação e ead: Recursos gamificados como aporte para uma educação inclusiva com foco no aluno. Revista do ISAT, 2015. 28

KAINULAINEN, P. *Spring Data Standard Guide*. [S.l.]: Packt Publishing Ltd, 2012. 24

KAPP, K. M. *The gamification of learning and instruction*. [S.l.]: Wiley San Francisco, 2012. 14, 28

KURNIA, A.; LIM, A.; CHEANG, B. Online judge. *Computers & Education*, Elsevier, v. 36, n. 4, p. 299–315, 2001. 14

MARTINS, T. et al. A gamificação de conteúdos escolares: uma experiência a partir da diversidade cultural brasileira. *X Seminário de Jogos Eletrônicos, Educação e Comunicação*, 2014. 30

MONTEIRO, W. M.; OLIVEIRA, T. M. de; MARTINS, D. J. S. Gamificação na educação a distância: Possibilidades para o ensino de programação. *Revista Tecnologias na Educação*, 2015. 14

NAGAI, W.; IZEKI, C. As estratégias de gamificação da disciplina de projeto e análise de algoritmos segundo o modelo dinâmico de aprendizado baseado em jogos. In: *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*. [S.l.: s.n.], 2016. v. 5, n. 1, p. 1159. 14

RAPKIEWICZ, C. E. et al. Estratégias pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais. *RENOTE: revista novas tecnologias na educação [recurso eletrônico]. Porto Alegre, RS*, 2007. 13

RAPOSO, E. H. S.; DANTAS, V. O desafio da serpente-usando gamification para motivar alunos em uma disciplina introdutória de programação. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2016. v. 27, n. 1, p. 577. 14

RODRIGUES, M. C. "como ensinar programação?". ULBRA. Canoas, RS, Brasil, 2002. 13

ROLLA, C. *DevMedia - Hello World com Docker*. 2018. <https://www.devmedia.com.br/hello-world-com-docker/40174>. Acesso em: 16-04-2019. 26, 27

SELIVON, M. et al. Uri online judge academic: Integraç ao e consolidaç ao da ferramenta no processo de ensino/aprendizagem. 2015. 18, 19, 20

SILVA, A. R. L. da et al. *Gamificação na educação*. [S.l.]: Pimenta Cultural, 2014. 30

SILVA, R. da. *JavaOne LATAM 2016 - RESTful Services Simplificado com Spring Data REST*. 2016. <https://pt.slideshare.net/rcandidosilva/javaone-latam-2016-restful-services-simplificado-com-spring-data-rest>. Acessado: 2019-06-16. 24

SPRING. *Spring Data*. 2019. <https://spring.io/projects/spring-data>. Acesso em: 16-06-2019. 24

TANAKA, S. et al. Gamification, inc.: como reinventar empresas a partir de jogos. mjv Press, 2013. 28

WALLS, C. *Spring Boot in action*. [S.l.]: Manning Publications, 2016. 22, 23

WERBACH, K.; HUNTER, D. *For the win: How game thinking can revolutionize your business*. [S.l.]: Wharton Digital Press, 2012. 29