UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
CURSO DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

FÁBIO PACHECO ALVES

UM AMBIENTE VIRTUAL COM FEEDBACK PERSONALIZADO PARA APOIO A DISCIPLINAS DE PROGRAMAÇÃO

São Leopoldo
2014

FÁBIO PACHECO ALVES

UM AMBIENTE VIRTUAL COM FEEDBACK PERSONALIZADO PARA APOIO A DISCIPLINAS DE PROGRAMAÇÃO

Artigo apresentado como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, pela Universidade do Vale do Rio dos Sinos - UNISINOS

Orientadora: Profa. Dra. Patrícia A. Jaques Maillard

São Leopoldo

2014

Observação: Na UNISINOS, os trabalhos de conclusão de graduação em informática devem ser entregues no formato de artigo, com apenas as capas no formato de dissertação.

UM AMBIENTE VIRTUAL COM FEEDBACK PERSONALIZADO PARA APOIO A DISCIPLINAS DE PROGRAMAÇÃO

Fábio Pacheco Alves*

Patrícia A. Jaques Maillard**

Resumo: Disciplinas de programação são, para grande parte dos alunos, desafiadoras e muitas vezes frustrantes. Elas exigem raciocínio lógico e conhecimento da sintaxe da linguagem de programação a ser estudada, que podem demorar a serem adquiridos pelo estudante. Para o professor, a alta carga de trabalho pode prejudicar no acompanhamento dos alunos, onde estes esperam sempre contar com respostas imediatas para suas dúvidas. Embora existam alguns ambientes web para apoio a aprendizagem de programação, tais como JOnline, MOJO, entre outros, esses ambientes retornam *feedback* do tipo juiz online, notificando apenas se o programa está funcionando corretamente ou não, mas sem explicar a possível causa da execução incorreta. Esse tipo de *feedback* é insuficiente para alunos de programação novatos. Além disso, essas ferramentas provém poucas funcionalidades para alunos e professores comentarem sobre possíveis erros e correções no código. Sendo assim, como forma de apoiar as disciplinas de programação, tanto para alunos como para professores, este artigo descreve um ambiente virtual composto de recursos como *feedback* personalizado, facilidades para a interação entre aluno e professor e auxílio ao professor para acompanhamento dos seus alunos. Uma avaliação experimental do ambiente proposto realizada com um grupo de 24 alunos que empregou a ferramenta nas aulas em laboratório de uma disciplina de programação durante um semestre mostrou, através de um questionário com questões fechadas, que as funcionalidades providas pela ferramenta auxiliaram os estudantes a identificar mais facilmente problemas em seus programas e como corrigi-los, assim como melhorou a comunicação entre aluno e professor.

Palavras-chave: Ensino a distância. Juiz *Online*. Disciplinas de programação. Interação com o professor.

1 INTRODUÇÃO

O ensino de programação nos cursos de tecnologia da informação tem uma grande importância por ser um dos pilares para a vida profissional do aluno. Alunos com alguma experiência profissional em informática conseguem ter esta percepção, porém, isso não é tão claro para aqueles sem muita experiência (BORGES, 2000).

* Fábio Pacheco Alves, Graduando em Análise e Desenvolvimento de Sistemas Unisinos.
e-mail: arnistrong@gmail.com
** Profa. Dra. Patrícia A. Jaques Maillard, Professor Adjunto do PIPCA/UNISINOS.
e-mail: pjaques@unisinos.br

Para que o aprendizado de programação seja efetivo é fundamental que o aluno pratique os conteúdos vistos em aula. Sendo assim o professor elabora trabalhos com desafios graduais de programação e disponibiliza para seus alunos. Uma vez concluído estes exercícios pelos alunos, começa o maior trabalho do professor, que é a correção destes trabalhos e a sinalização dos problemas encontrados. Não sendo somente isto, no decorrer da execução dos exercícios é comum e até esperado que os alunos acionem o professor para esclarecer suas dúvidas e dificuldades encontradas. Devido ao grande volume de trabalho, o professor, geralmente, não consegue atender imediatamente o pedido de ajuda dos seus alunos. Por outro lado, os estudantes, quando não prontamente atendidos, acabam se desmotivando ou até mesmo ignorando o aprendizado desta matéria.

Para apoiar o ensino de programação e buscar minimizar estas dificuldades, professores buscam recursos e ferramentas computacionais que possam lhes auxiliar nessas tarefas. Uma das ferramentas empregadas são os Juízes *Online*. Os Juízes *Online* são sistemas utilizados em competições de programação, disponibilizados através de aplicações *web*, onde é possível submeter programas, em uma linguagem de programação escolhida, que são respostas a desafios. Este Juiz *Online* testa o programa do estudante, executando-o com entradas pré-cadastradas no sistema. Os dados de saída do programa executado são comparados com uma base de dados de saídas esperadas, onde cada saída esperada está relacionada com uma entrada cadastrada para teste. O Juiz pode então dar uma resposta de forma imediata, apontando se a resposta fornecida pelo estudante é válida ou não.

As plataformas de Juízes *Online* utilizadas em competições de programação são capazes de aplicar testes nos códigos submetidos a elas, porém não possuem o recurso de fornecer um retorno apontando o tipo de erro ocorrido, por que ele aconteceu e nem onde este erro ocorreu. Este recurso é inexistente devido à natureza da sua utilização. Em competições de programação não se pode apontar os problemas encontrados; o próprio competidor deve tentar descobrir os erros para submeter novamente.

Alguns trabalhos como JOnline (SANTOS; RIBEIRO, 2011), MOJO (CHAVES et al., 2013) e BOSS (JOY; GRIFFITHS; BOYATT, 2005) estão fazendo uso destes recursos para apoiar as disciplinas de programação, utilizando a automatização fornecida pelos Juízes *Online* para apresentar mensagens para o aluno de forma

instantânea. Os recursos de Juízes *Online* são também empregados para permitir que os alunos submetam seus trabalhos para os seus professores, e para que estes possam ter um controle sobre o que está sendo entregue, como é o caso do BOCA (CAMPOS; FERREIRA, 2004). Dentre estes trabalhos, não foi observado um aprofundamento no uso de mecanismos que facilitam a interação entre aluno e professor, limitando os trabalhos a simples comunicação via *chat* ou através de fórum. Além disso, com relação à correção automática, a grande maioria dos trabalhos utiliza apenas o simples retorno emitido pelo Juiz *Online*, o que não é suficiente para apoiar a aprendizagem de programação.

Visto estes pontos, este artigo descreve *feeper*, um ambiente virtual para apoiar o ensino/aprendizagem de programação no ensino superior. O trabalho desenvolvido fez uso de um mecanismo para melhorar a comunicação entre aluno e professor, permitindo sinalizar partes do código fonte escrito. Além disso, o *feedback* fornecido pelo Juiz *Online* foi personalizado para apresentar mensagens que indiquem ao aluno onde aconteceu o erro, por que ele aconteceu e como corrigi-lo e para permitir a execução de diferentes tipos de teste em cima do programa do aluno.

Como diferencial, além da correção automática, o trabalho proposto emite dicas sobre os erros gerados pelo programa do aluno, permitindo que este consiga evoluir na correção do seu programa. O aperfeiçoamento da interação entre aluno e professor também foi foco desse trabalho. O programa permite que o aluno possa adicionar marcações em seu programa que serão exibidas ao professor, facilitando o entendimento de ambas as partes em qual linha exata do código o aluno está tendo dúvidas e/ou dificuldades.

Foi realizada uma avaliação experimental do ambiente proposto com um grupo de 24 alunos que empregou a ferramenta nas aulas em laboratório de uma disciplina de programação durante um semestre. Ao final do experimento, que compreendeu o desenvolvimento de sete exercícios de programação no ambiente, foi aplicado um questionário para avaliar a experiência dos alunos e professor no uso do *feeper*. Este questionário consistia de 13 questões fechadas, cujas respostas seguiam a escala de *Likert*, e objetivou avaliar a usabilidade da interface do sistema bem como a impressão dos estudantes e professor sobre suas funcionalidades. Através deste questionário foi possível obter um resultado bastante positivo com relação o uso do *feeper* como uma ferramenta para apoiar os aluno e o professor. Para o professor ficou evidenciado a facilidade gerada na comunicação com os

alunos e para o acompanhamento destes, e para os alunos o retorno imediato da ferramenta ao avaliar seu código fonte, foram pontos que ganharam destaque na avaliação.

2 REFERENCIAL TEÓRICO

2.1 ENSINO DE PROGRAMAÇÃO

Programar é uma habilidade difícil de ser adquirida. Uma das formas mais efetivas de aprender é praticando. Essa estratégia é chamada de "*Learning by doing*" (ANZAI; SIMON, 1979). Para resolver algum problema, o estudante realiza sucessivas tentativas. A cada tentativa ele aprende com seus erros e elabora uma nova estratégia. Trazendo esta teoria para o ensino de programação, pode-se dizer que a prática de exercícios é um caminho para o aprendizado, onde ele fará tentativas sucessivas até que o exercício esteja resolvido.

O principal papel do professor é incentivar os estudantes a praticarem através do desenvolvimento de projetos de programação. Os alunos devem ser motivados para que eles se engajem apropriadamente. No estudo apresentado em (JENKINS, 2001), foi identificada que a principal motivação pela graduação é a aspiração: motivação centrada em algum objetivo futuro como carreira, trabalhos e dinheiro. Já para módulos de programação, geralmente, não há uma motivação particular. O módulo é visto como uma obrigação, uma parte mandatória para a graduação.

Há um grande número de reprovações e desistências nas disciplinas de programação, inclusive, relacionadas ao abandono do curso. Com a grande demanda por mão de obra de profissionais da área de computação, esta evasão dos cursos superiores se torna preocupante. De acordo com o site CODE.ORG (2013), em 2020 haverá 1,4 milhões de empregos na área da informática e apenas 400 mil graduados em cursos na área da informática. Menos de 2,4% dos estudantes universitários são graduados em ciências da computação e este número tem caído desde a última década.

2.2 JUÍZES ONLINE

Os Juízes *Online* são sistemas utilizados em competições de programação, disponibilizados através de aplicações *web*. Nas competições de programação é possível submeter programas, em uma linguagem de programação escolhida, que são respostas aos desafios estabelecidos. O Juiz *Online* testa se o programa passou no desafio, executando-o com entradas pré-cadastradas no sistema. Os dados de saída do programa testado são comparados com uma base de dados de saídas esperadas, onde cada saída esperada está relacionada a uma entrada cadastrada para teste. O Juiz pode assim apontar se o programa está gerando uma saída de dados válida.

As respostas geradas pelos Juízes *Online*, de forma geral, seguem a seguinte linha:

a) **Sucesso**;
b) **Erro de compilação**: não foi possível executar o programa porque o código apresenta erros de compilação;
c) **Erro em tempo de execução**: algum objeto gera um erro durante a execução do programa;
d) **Tempo limite excedido**: o programa demora a responder ultrapassando o tempo limite;
e) **Erro de apresentação**: o programa está apresentando as respostas no formato incorreto, impossibilitando a comparação do resultado com o gabarito;
f) **Resposta errada**: o programa gerou respostas no formato correto, porém as respostas não estão de acordo com o gabarito.

Estas respostas emitidas são genéricas, pois o principal objetivo do Juiz *Online* é o torneio de programação, e nestes torneios não se pode emitir outro tipo de resposta, pois acabaria ajudando os competidores. Além disso, o objetivo da competição é testar conhecimentos e o raciocínio lógico dos participantes. Alguns trabalhos tem empregado Juízes *Online*, tais como o BOCA (CAMPOS; FERREIRA, 2004), para apoiar disciplinas de programação. Neste caso, o Juiz *Online* pode apoiar disciplinas de programação recebendo códigos submetidos por alunos e emitindo respostas. Porém, ele não fornece uma resposta personalizada, apenas estas genéricas antes apresentadas, o que não ajuda o aluno a entender o seu erro.

Nesta situação o emprego de respostas personalizadas, que emitam um detalhamento do erro encontrado, com informações pertinentes ao problema, podem ajudar o aluno a identificar em que ponto ele está errando, e assim aprender.

3 TRABALHOS RELACIONADOS

JOnline é um Juiz *Online* didático para o ensino de programação (SANTOS; RIBEIRO, 2011). A proposta deste trabalho foi a de utilizar uma plataforma já existente de Juiz *Online* denominada BOCA (CAMPOS; FERREIRA, 2004) e agregar a ela novas funcionalidades, entre as quais:

a) A apresentação de dicas em língua portuguesa para corrigir erros de compilação de código-fonte submetido;
b) Apresentação de casos de teste que geram resultados errados;
c) Organização de problemas por assunto e grau de dificuldade e votação do nível de dificuldade dos problemas pelos usuários, e;
d) Programação colaborativa.

A apresentação dos resultados em língua portuguesa é feita quando um erro ocorre, seja um erro de compilação, de lógica ou de execução. Para cada tipo de erro, são cadastradas mensagens diferentes. Ainda é possível realizar a programação colaborativa pela ferramenta, um recurso que permite aos alunos desenvolver a solução para um exercício de forma simultânea. Segundo os autores, a ferramenta ainda está sendo desenvolvida, utilizando a linguagem de programação *Java*, porém, por enquanto apenas suporta a submissão de códigos em *C* e *C++*.

Outro trabalho interessante proposto é o MOJO, uma ferramenta para auxiliar o professor em disciplinas de programação (CHAVES et al., 2013). No MOJO o foco é o professor. Ele visa diminuir consideravelmente a sobrecarga de trabalho do docente na correção e apresentação dos resultados das atividades desenvolvidas pelos alunos. Como o foco é no professor, o MOJO recebe a resposta do Juiz *Online* e apresenta para o professor, para que este então dê um *feedback* para os seus alunos.

Lançado em 2005, o BOSS é um sistema de submissão e avaliação de código fonte que suporta a avaliação de trabalhos por meio da coleta de submissões, executando testes automatizados para correção e qualidade, verificando plágios, e fornecendo uma interface para obtenção de *feedbacks* (JOY; GRIFFITHS; BOYATT,

2005). O BOSS realiza dois tipos de teste automatizado. O primeiro define entradas e saídas e executa o código inserindo as entradas e comparando as saídas, clássico teste caixa preta realizado pelos Juízes *Online*. Já o segundo tipo é aplicado somente para a linguagem *Java*, pois utiliza o *JUnit* (BECK; GAMMA, 2010), um framework com suporte à criação de testes automatizados para *Java*. Neste caso as entradas e saídas são objetos *Java*, listas contendo os dados que serão repassados para os métodos, e permitindo que o próprio *JUnit* execute e aponte o resultado da execução. O JBOSS também disponibiliza métricas básicas de software, tais como número de comentários, percentual de métodos declarados como abstratos, entre outros, assim como permite a criação de novas métricas. Além disso, essa ferramenta permite aos professores executarem os testes automatizados, detectar plágios, visualizar os códigos submetidos, e fornecer um *feedback* para os alunos. Este sistema executa tanto no *Linux* como no *Windows*.

Outro sistema estudado, denominado BOCA, é um Juiz *Online* desenvolvido para suportar competições de programação e que também pode ser utilizado para apoiar disciplinas de programação, através da submissão e correção automatizada de trabalhos desenvolvidos pelos alunos (CAMPOS; FERREIRA, 2004). Com o foco em competições de programação, o BOCA permite, além de criar as competições, que as equipes façam um aquecimento antes da competição, resolvendo alguns exercícios para testar as funcionalidades, permitindo que a equipe se adapte à ferramenta. É ainda possível enviar dúvidas aos juízes de prova onde eles podem responder através de uma interface específica. Dentro desta interface exclusiva, os juízes podem avaliar os códigos submetidos e, além de obter a resposta do autojulgamento realizado pelo Juiz *Online*, reavaliar a solução proposta e escolher a melhor resposta para devolver aos times. O BOCA permite a submissão de códigos em C, *C++* e *Java* e executa somente no sistema operacional *Linux*.

O *Moodle* (MOODLE, 2013) permite um acoplamento de funcionalidades personalizadas e para aproveitar toda a base fornecida pelo *Moodle*, a ferramenta OnlineJudge foi desenvolvida para gerenciar a submissão de códigos fontes internamente pelo *Moodle*. Nativamente, a aplicação é capaz de compilar as linguagens *C* e *C++*, porém possui um recurso que pode ser ativado, permitindo executar mais de 40 diferentes tipos de linguagens. Este recurso é disponibilizado por uma ferramenta externa denominada *Ideone*, um *WebService* comercial desenvolvido e mantido pela *Sphere*. Porém, por ser uma ferramenta comercial, o

restrito pacote gratuito é bastante limitado, permitindo que poucos códigos sejam compilados diariamente e respeitando um intervalo de tempo entre cada compilação.

Outra plataforma para a submissão e correção de exercícios é o URI Online Judge (URI ERECHIM, 2011). Ele possui além do recurso da submissão de códigos, fórum de discussão para cada exercício, permitindo que todos os usuários possam discutir e tirar dúvidas sobre a resolução do exercício. Também possui um recurso interessante para apoiar o desenvolvimento que é a geração de dados de entrada de forma randômica. Através de um painel é possível visualizar um *ranking* geral dos usuários, onde os resultados são exibidos tanto através de uma listagem, como também através de gráficos avançados, o que permite a comparação do desempenho com os outros usuários.

3.1 TABELA DE COMPARAÇÃO DE CARACTERÍSTICAS

Nesta seção é apresentada uma tabela para comparação entre as características e funcionalidades das ferramentas acima citadas com o trabalho desenvolvido. Nas linhas da tabela estão listadas todas as ferramentas a serem comparadas e nas colunas estão representadas as principais funcionalidades presentes nas ferramentas estudadas.

Abaixo um detalhamento das colunas da tabela de comparação:

a) **Sistema Operacional**: serve para ilustrar que a maioria das ferramentas são voltadas para o *Linux* tendo estas comandos exclusivos para o Sistema Operacional *Linux*. Apenas para a ferramenta URI Online Judge não foi possível identificar quais sistemas operacionais são suportados;

b) **Compilação em C, *C++* ou *Java***: as principais linguagens de programação na área acadêmica ilustra a abrangência das ferramentas. Apenas para a ferramenta MOJO não foi possível identificar quais as linguagens que ele é capaz de compilar;

c) **Integração com o *Moodle***: apresenta as ferramentas que foram desenvolvidas para funcionarem integradas ao *Moodle*;

d) ***Feedback* automático para o usuário**: esta funcionalidade está presente nos Juízes *Online*, onde ao submeter um código para avaliação o usuário tem um retorno imediato de forma automatizada;

e) **Retorno da compilação detalhado**: esta comparação apresenta as ferramentas que possuem um tratamento no retorno do Juiz *Online*, com uma resposta personalizada mais detalhada;

f) **Apoia disciplinas de programação**: esta característica aponta as ferramentas que possuem recursos capazes de suportar disciplinas de programação, tais como dispor de recursos para a entrega de exercícios e obtenção de notas através da ferramenta;

g) **Permite interação com o professor**: indica que a ferramenta possui recursos para que seja possível estabelecer alguma forma de interação com o professor a fim de tirar dúvidas, seja através de mensagens diretas enviadas ao professor, ou através do uso de fóruns de discussão onde há a intervenção do professor ou até mesmo através de *chats*;

h) **Organiza competições de programação**: esta característica aponta as ferramentas que possuem recursos capazes de organizar e suportar competições de programação, disponibilizando acessos para as equipes e para os avaliadores durante uma competição.

Tabela 1 - Comparativo entre os trabalhos relacionados

	Sistema Operacional	Compilação em C, *C++* ou *Java*	Integração com o *Moodle*	*Feedback* automático para usuário	Retorno da compilação detalhado	Apoia disciplinas de programação	Permite interação com o professor	Organiza competições de programação
JOnline	Linux	✓	✗	✓	✓	✓	✗	✗
MOJO	Linux	--	✓	✗	✗	✓	✓	✗
BOSS	Windows, Linux	✓	✗	✓	✗	✓	✓	✗
BOCA	Linux	✓	✗	✓	✗	✓	✗	✓
OnlineJudge	Linux	✓	✓	✓	✗	✓	✓	✗
URI Online Judge	--	✓	✗	✓	✗	✗	✗	✓
Trabalho Proposto	Windows, Linux	✓	✗	✓	✓	✓	✓	✗

O trabalho desenvolvido abrange a maioria das características acima apresentadas, ficando sem cobertura apenas para a integração com o *Moodle* e para a organização de competições de programação. Mesmo o *Moodle* sendo uma importante ferramenta para o ensino a distância, este trabalho não implementou integração ao *Moodle* em um primeiro momento, pois algumas das funcionalidades

aqui relacionadas exigiram a utilização de uma estrutura arquitetural não disponível no *Moodle*. Com relação a organização de competições, o foco do trabalho é o apoio ao aluno e ao professor, dando a estes mais recursos para trabalharem e aprenderem. A competição está em outro contexto, ao qual este trabalho não se encaixa.

Como diferencial neste trabalho, para a interação entre alunos e professor, foi desenvolvida uma funcionalidade que permite registrar dúvidas e comentários diretamente nas linhas do código fonte do aluno, permitindo que ambos possam visualizar e discutir sobre o código fonte. Outro diferencial neste trabalho é a forma como os exercícios são cadastrados pelo professor, que permite atingir um alto nível de detalhamento na correção das respostas submetidas pelos alunos, através do cadastro de classes de teste e também de massa de dados para testes de entrada e saída de informação.

4 FEEPER: O SISTEMA DESENVOLVIDO

O sistema desenvolvido recebeu o nome de *feeper*, onde é um acrônimo entre as palavras "***fee****dback* **per**sonalizado", recurso este, tido como principal característica do sistema.

O *feeper* consiste em uma ferramenta *web* para apoiar a aprendizagem de programação em classe, assim como em extraclasse. Nesta ferramenta são disponibilizados exercícios práticos que são organizados de forma a desafiar o aluno de forma gradual na resolução e também acompanhar a sua evolução desde o início do uso desta ferramenta. Da mesma forma, vários recursos foram disponibilizados para auxiliar o professor no acompanhamento e avaliação dos aprendizes.

Para auxiliar o professor, o seu módulo conta com funcionalidades para acompanhar as tentativas dos alunos de solucionar os exercícios, permitindo que o professor interaja com o aluno, registrando comentários nos códigos dos alunos de forma simples e direta. Todas estas mensagens enviadas e recebidas estão disponíveis dentro da caixa de mensagens, desenvolvida como forma de centralizar todas as conversas entre professor e alunos. O professor pode também gerenciar a sua turma, para adicionar novos exercícios e liberá-los gradativamente conforme necessidade. Estes exercícios são mantidos pelo professor, podendo este inserir novos, modificar e até excluir exercícios. Através do cadastro dos exercícios o

professor poderá montar toda a inteligência para a correção do exercício, fornecendo mensagens que irão ajudar seus alunos quando estes encontrarem problemas para resolver o exercício.

Em um painel de resultados, o professor consegue visualizar o andamento dos exercícios, onde através de uma grade que exibe todos os alunos e todos os exercícios da turma é possível visualizar o status do exercício para cada aluno, se o estudante já conseguiu resolver o exercício ou se exercício está com algum outro problema. Ao clicar nesta sinalização, a ferramenta apresenta um histórico de respostas submetidas pelo aluno, onde todos os detalhes são exibidos, como a data da submissão, os códigos fontes utilizados e o *feedback* que o sistema apresentou para o aluno.

4.1 ARQUITETURA DO SISTEMA

O *feeper* foi desenvolvido como uma aplicação para a *internet*, para permitir que os alunos e os professores usufruam desta onde quer que estejam. A aplicação foi dividida em três módulos de uso e um módulo automatizado: o primeiro módulo para uso dos alunos, o segundo módulo para uso dos professores e o terceiro módulo para a administração geral da aplicação. O último módulo é composto de rotinas automatizadas que darão suporte para a aplicação como, por exemplo, realizar a avaliação automática das respostas submetidas pelos alunos. Para tal funcionalidade foi utilizado um componente Juiz *Online* no formato de *Servlets*, que está sempre aguardando por demandas a serem validadas.

A Figura 1 ilustra a arquitetura do sistema, exibindo todos os módulos envolvidos e o componente Juiz *Online*, ambos suportados por um banco de dados.

Figura 1 - Diagrama da Arquitetura do Sistema

Fonte: Elaborado pelos autores.

Toda esta estrutura foi hospedada em nuvem, para permitir o fácil escalonamento de memória e processamento. A demanda de servidor é alta durante o período de aula, onde todos os alunos estão conectados ao mesmo momento. Após a aula, a capacidade do servidor pode ser reduzida, mantendo assim uma economia de recursos financeiros.

4.2 FUNCIONALIDADES DO SISTEMA

Abaixo são descritas as funcionalidades que estão disponíveis na ferramenta para uso dos usuários e também o funcionamento do Juiz *Online*, executada de forma automatizada. As funcionalidades são apresentadas nos módulos correspondentes.

4.2.1 Funcionalidades do Módulo do Aluno

4.2.1.1 Enviar/Atualizar um Código Fonte

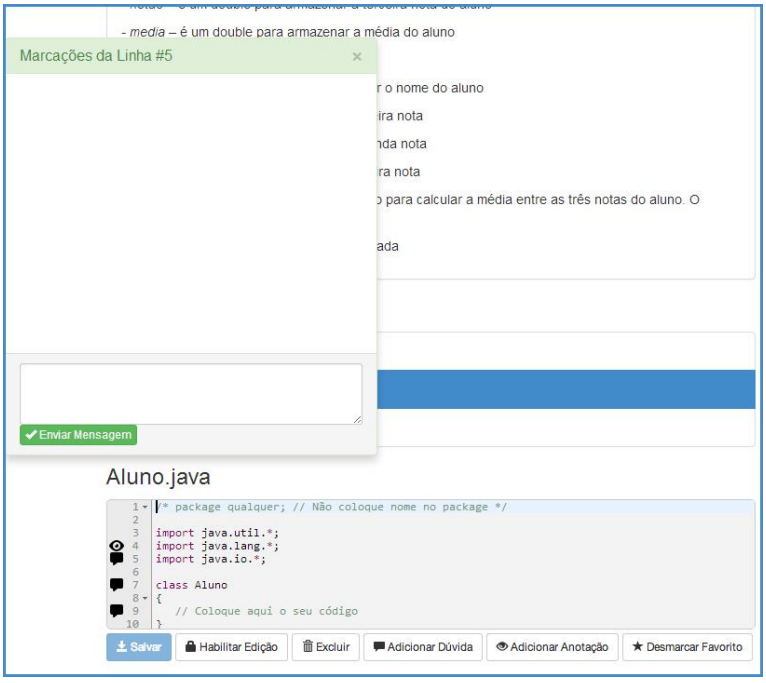
A resposta ao exercício possui uma etapa onde o aluno realiza o *upload* do arquivo do seu código, ou ele digita o código fonte em um editor de codificação para a *web*. É permitido ao aluno adicionar múltiplas classes *Java* para solucionar o exercício. Uma vez o código inserido na ferramenta, ele pode registrar anotações

para cada linha do código fonte e também enviar dúvidas para seu professor. A qualquer momento é possível realizar uma revisão dos códigos cadastrados, e, durante esta revisão, o aluno pode enviar perguntas para o professor.

4.2.1.2 Enviar Dúvidas para o Professor

Este recurso de marcar as linhas permite que o aluno aponte a linha em que ele está tendo dificuldades, possibilitando estreitar a comunicação com o professor. O professor, ao receber a dúvida do aluno, consegue visualizar exatamente os trechos de código aos quais o aluno está se referindo, evitando assim a necessidade de fazer *prints* de telas ou enviar arquivos por *e-mail* para o professor, apenas para exemplificar o problema real. O aluno registra as dúvidas sempre vinculando a dúvida a uma linha do seu código fonte; é possível registrar dúvidas para todas as linhas do arquivo. No momento em que o professor for auxiliar o aluno, ele visualiza o código fonte do aluno e também todas as linhas marcadas, facilitando a localização das dúvidas. Após a marcação das linhas, o aluno registra uma pergunta para o professor. Após enviar uma pergunta para o professor, o aluno pode ver toda a conversa clicando na linha em que as marcações foram originadas.

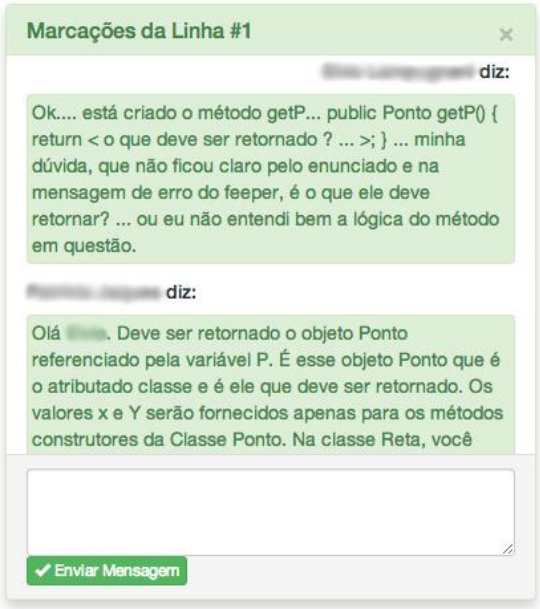
Figura 2 - Tela para envio de dúvidas ao professor



Fonte: Print-screen da tela do *feeper*.

A Figura 2 ilustra a tela para a marcação da linha e o registro da dúvida enviada ao professor. A Figura 3 ilustra a conversa entre aluno e professor, registrada através desta funcionalidade.

Figura 3 – Conversa com o professor



Fonte: Print-screen da tela do *feeper*.

4.2.1.3 Adicionar Anotação no Código Fonte

As anotações no código fonte servem para auxiliar o aluno na sua revisão, e permite que ele registre comentários para revisões futuras ou até mesmo para explicar algum trecho de código.

4.2.1.4 Submeter para Validação um Código Fonte

Depois de feitas as revisões do código fonte, e quando o aluno desejar, é possível realizar a submissão do seu código fonte para validação. Um processo automatizado é iniciado para validação do código fonte através do Juiz *Online*. Após esta validação, o aluno receberá um retorno, que indicará a existência ou não de erro. Esta etapa de validação executa um teste por vez, então caso o programa falhe na primeira execução, o Juiz *Online* para e responde com a mensagem de erro

referente ao teste executado. Dessa forma, caso haja erro, o aluno receberá uma mensagem personalizada do Juiz *Online* e também o erro que foi gerado durante a validação. Do contrário, uma mensagem de sucesso é apresentada para o estudante. A qualquer momento é possível que o aluno volte e corrija seu código, podendo excluir ou modifica-lo rapidamente utilizando o editor de códigos para a *web*.

Cada vez que ocorrer uma submissão para validação do código fonte, é criada uma versão deste código fonte, para que ocorra uma rastreabilidade das alterações aplicadas. Elas ficam visíveis para o professor, para que este acompanhe a evolução do aprendiz e também visualize a quantidade de vezes que o aluno tentou resolver o exercício proposto.

4.2.1.5 Realizar Download dos Códigos Fonte

O aluno pode realizar o *download* dos seus códigos cadastrados para a resolução dos exercícios. A funcionalidade é apresentada dentro de cada exercício, onde um pacote contendo todas as classes é gerado e disponibilizado para *download*.

4.2.1.6 Marcar Código Fonte como Favorito

O aluno pode marcar um código como favorito, colocando este em um repositório que terá como objetivo o reaproveitamento de código. Através da listagem de códigos favoritos, o aluno pode resgatar um código que ele já desenvolveu e utilizar novamente na resolução de outro exercício.

4.2.1.7 Visualizar Lista de Exercícios

A lista de exercícios, mantida pelo professor, apresenta para os alunos os exercícios que estão disponíveis para serem respondidos. O professor pode liberar os exercícios conforme a necessidade. Nesta lista são apresentados para o aluno apenas os exercícios da sua turma, exibindo junto ao título do exercício a data da última resposta submetida pelo aluno.

4.2.1.8 Visualizar Lista de Colegas

A lista de colegas exibe para o aluno quem é o seu professor e todos os seus colegas de turma, além de si mesmo. Todas as pessoas são apresentadas com suas fotos cadastradas e seus respectivos nomes.

4.2.1.9 Visualizar suas Turmas

Através do cabeçalho da ferramenta, o aluno tem acesso rápido as suas turmas, facilitando a navegação entre os conteúdos de cada.

4.2.1.10 Atualizar Perfil

O aluno pode acessar as configurações do seu perfil para trocar o seu nome de exibição, sua senha e também realizar o *upload* de uma foto de exibição para o seu perfil.

4.2.1.11 Visualizar Resultados

Através da tela de resultados, o aluno consulta a sua evolução na resolução dos exercícios propostos. Esta visualização apresenta apenas os dados autenticados do aluno na ferramenta. É possível, através da tela de resultados, que o aluno consulte um histórico de todas as suas submissões por exercício. Ao visualizar as versões, o aluno ainda pode realizar o *download* destes arquivos, e até mesmo registrar dúvidas e enviar para o professor, da mesma forma como citado anteriormente.

A Figura 4 ilustra a tela de resultados, exibindo um detalhamento de um dos exercícios já submetidos.

4.2.1.12 Visualizar Caixa de Mensagens

A caixa de mensagens centraliza todas as mensagens enviadas para o professor, e também as mensagens que o professor enviou para o aluno. Quando há novas mensagens estas são contabilizadas e exibidas em forma de contador ao lado

do menu “Mensagens”. Acessando a tela, as novas mensagens são destacadas, facilitando assim a localização destas.

Figura 4 – Tela de resultados

Fonte: Print-screen da tela do *feeper*.

4.2.2 Funcionalidades do Módulo do Professor

4.2.2.1 Cadastrar Exercícios

A ferramenta disponibiliza um repositório de exercícios, acessíveis a todos os professores, permitindo através de um formulário a inserção de novos exercícios. Este formulário é composto por dois blocos, o primeiro servirá para registrar a parte

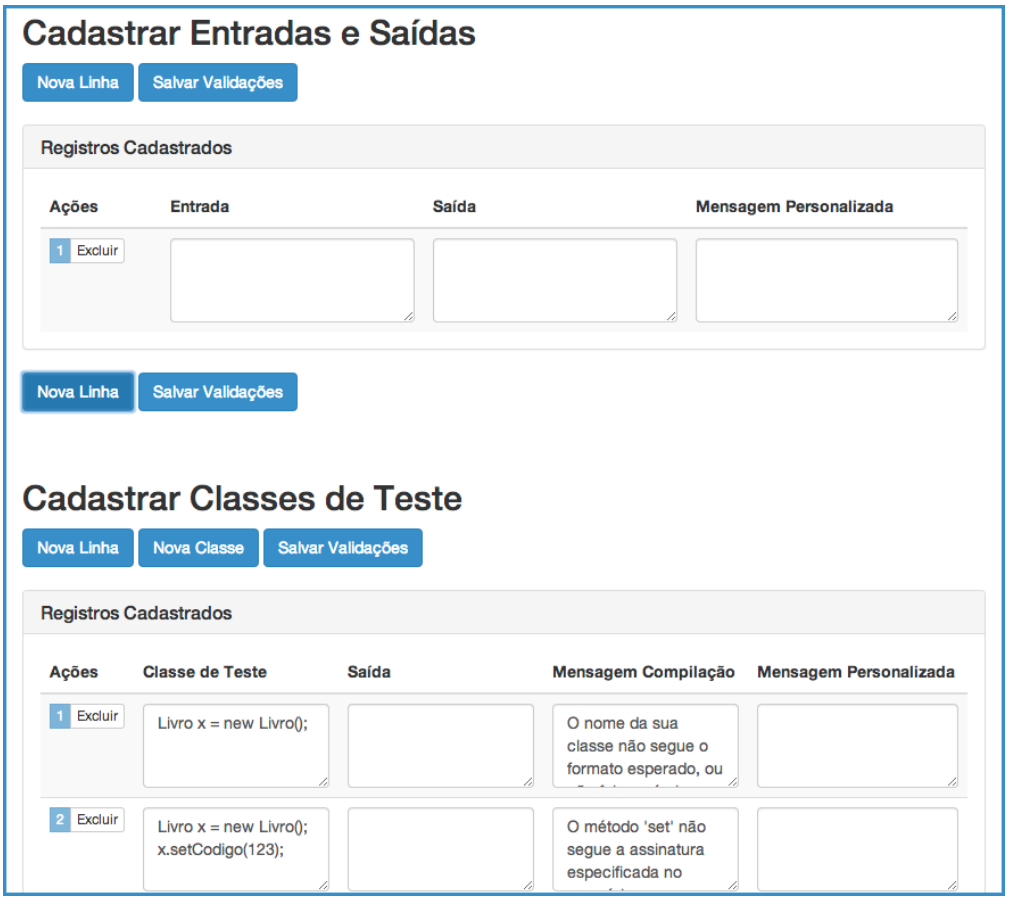
descritiva do exercício, podendo ser feita através de um editor de texto *html*. O segundo bloco será utilizado para cadastrar os testes a serem executados pelo Juiz *Online* para determinar que a resposta do aluno está satisfatória. É possível cadastrar dois tipos de testes. No primeiro modelo são cadastrados dados de entrada, saídas esperadas e as mensagens personalizadas para serem exibidas aos alunos em caso de insucesso na tentativa de resolver o exercício. No segundo modelo, o professor pode cadastrar classes de teste, as saídas esperadas destas classes, a mensagem personalizada para caso ocorra algum erro de compilação e as mensagens personalizadas, ambas serão exibidas para o aluno da mesma forma do primeiro modelo. O uso das mensagens personalizadas ajuda o aluno a resolver possíveis problemas encontrados, dando a liberdade ao professor de estabelecer o nível de detalhamento que este dará ao exercício. Exemplificando seu funcionamento, para um exercício que realize algum cálculo com números, ao tentar inserir um valor inesperado, o programa deverá estar apto a validar tal valor, porém, caso essa validação não seja possível, o professor, já prevendo este tipo de situação, cadastra uma mensagem instrutiva para o aluno, orientando este a corrigir o seu programa, indicando qual validação deverá ser aplicada para tratar estes valores inválidos.

A Figura 5 ilustra a tela para cadastro do enunciado do exercício. A Figura 6 ilustra a continuação da tela para cadastro dos testes para o exercício.

Figura 5 – Tela de cadastro do enunciado do exercício

Fonte: Print-screen da tela do *feeper*.

Figura 6 – Tela de cadastro dos testes do exercício



Fonte: Print-screen da tela do *feeper*.

4.2.2.2 Editar Turma

O professor tem acesso total para poder editar os dados da sua turma. Ele pode vincular alunos na sua turma, desde que estes estejam previamente cadastrados na ferramenta, remover alunos da turma e enviar o *e-mail* de convite para acessar a ferramenta, onde este envio de convite gera uma senha aleatória e temporária para que o aluno possa acessar a ferramenta. Ainda, na tela de cadastro da turma, o professor pode selecionar os exercícios que serão disponibilizados para seus alunos. Ao selecionar um exercício, este não é exibido de imediato para os alunos; é necessário que o professor realize a liberação para visualização do exercício. Uma vez vinculado, o exercício pode ser removido ou ter a sua exibição desabilitada.

4.2.2.3 Visualizar Resultados

Através da tela de resultados, o professor consulta a evolução de seus alunos na resolução dos exercícios propostos. É possível, através da tela de resultados, que o professor consulte o histórico de todas as submissões de todos os alunos por exercício, apresentando data e hora da submissão da resposta, todas as mensagens de erro, de sucesso emitidas pelo Juiz *Online*, e também todos os arquivos utilizados pelo aluno na submissão. O professor pode, ao visualizar o histórico de submissões, enviar observações para o aluno em qualquer um dos arquivos submetidos pelo aluno.

A Figura 7 ilustra a tela de resultados na visão do professor, exibindo todos os seus alunos e o resultado obtido por estes para cada exercício. Em seguida a Figura 8 ilustra a tela de resultados com o detalhamento de execução de determinado exercício para um estudante em específico. E por fim a Figura 9 ilustra o envio de mensagens para o aluno ao acessar o seu código fonte, através da listagem de respostas.

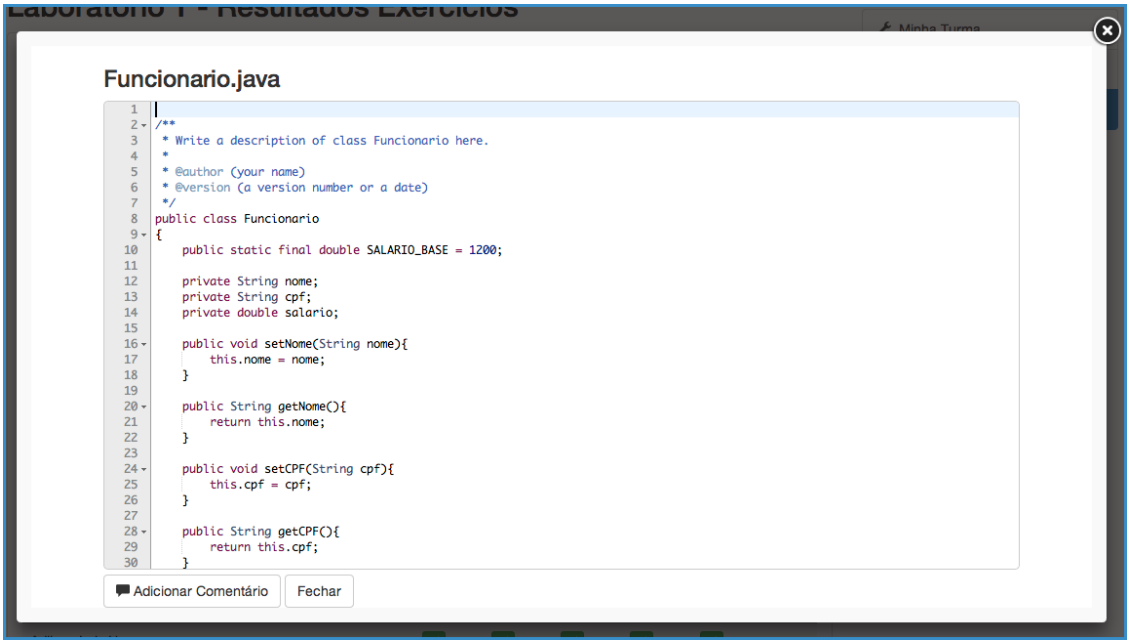
Figura 7 – Tela de resultados na visão do professor

Fonte: Print-screen da tela do *feeper*.

Figura 8 – Tela de resultados com detalhamento de execução

Fonte: Print-screen da tela do *feeper*.

Figura 9 – Tela de resultados com exibição do código fonte do aluno



Fonte: Print-screen da tela do *feeper*.

4.2.2.4 Visualizar Caixa de Mensagens

Outro recurso fundamental para a ferramenta é o acompanhamento das perguntas, onde através de uma caixa de mensagens o professor tem acesso rápido às novas perguntas e também a todas as conversas e interações que fez com os seus alunos. A cada nova pergunta que surja para o professor, este será avisado por *e-mail* e também aparece um aviso em forma de contador de mensagens quando acessar o sistema. Ao listar as mensagens, as novas são destacadas, facilitando assim a sua localização dentre as demais.

4.2.2.5 Visualizar suas Turmas

Através do cabeçalho da ferramenta o professor tem acesso rápido as suas turmas, facilitando a navegação entre os conteúdos de cada.

4.2.2.6 Atualizar Perfil

O professor pode acessar as configurações do seu perfil para trocar o seu nome de exibição, sua senha e também realizar o *upload* de uma foto de exibição para o seu perfil.

4.2.3 Funcionalidades do Módulo do Administrador

4.2.3.1 Cadastrar Pessoas

Para que o sistema possa ter seu uso iniciado é imprescindível à existência de pessoas cadastradas. O gerenciamento de pessoas está ativo apenas para o módulo do administrador, permitindo que o administrador adicione novos professores, alunos e até mesmo outros administradores.

4.2.3.2 Cadastrar Turmas

O administrador pode criar turmas e simultaneamente relacionar o professor responsável pela turma. Ainda no cadastro de turmas, é possível vincular ou

desvincular alunos da turma. No momento em que os alunos são vinculados, eles ainda não receberão o *e-mail* com o convite de acesso; para que isto ocorra há um botão para disparar os convites. Os convites serão enviados de forma individual, informando ao aluno seus dados para acessar a ferramenta, como usuário e senha. O usuário é o próprio *e-mail* do aluno e a senha é gerada automaticamente pelo sistema, caso o aluno ainda não possua senha. Caso o aluno já possua senha de acesso, apenas é enviado o convite para acesso ao sistema, sem apresentar a senha. O administrador também pode relacionar os exercícios para a turma da mesma forma que o professor.

4.2.3.3 Cadastrar Exercícios

Assim como para os professores, a funcionalidade de castrar exercícios também fica visível para o administrador, permitindo que este insira novos exercícios, exclua ou edite exercícios existentes.

4.2.3.4 Visualizar os Logs

Fica disponível somente para o administrador o acesso aos *logs* do sistema, onde são apresentadas todas as ações dos usuários dentro da ferramenta. É possível realizar uma consulta pelo conteúdo descritivo do *log*, pela data do *log* através de um campo data inicial e um campo data final, e um campo de seleção tipo do *log*. Através desta tela, o administrador consegue acompanhar o uso da ferramenta.

4.2.4 Juiz Online

Após o código ter sido desenvolvido e submetido pelo aluno, este código é avaliado pelo Juiz *Online*. Ele é capaz de processar o código submetido, aplicando testes previamente cadastrados no exercício. O Juiz *Online* aplica dois tipos de testes para validar o código fonte, o teste com inserção de dados e comparação dos resultados, e o teste de execução de classes de teste.

No primeiro teste, o código fonte é compilado e executado, e uma série de dados de entrada é enviada para o programa do aluno, para que este responda com

outra série de dados de saída. Estes dados são comparados com as saídas esperadas (registradas em um banco de dados). Portanto, para que um exercício seja aprovado por esta validação, a resposta do aluno deve ser construída seguindo o padrão descrito no exercício. Este padrão é aplicado para os dados de entrada e os dados de saída. Por exemplo, para um exercício que identifique se um número é par ou ímpar, pode ser estabelecido que o programa receba um número qualquer e a sua saída deve ser 0 para números pares e 1 para números ímpares. Sendo assim, quando o Juiz *Online* executar este programa, ele saberá o que inserir de informação (entrada) e também saberá como o resultado do programa (saída) será apresentado. Para que se tenha uma certeza de que o programa de fato está aplicando uma lógica ou um cálculo, o cadastro do exercício deve possuir uma lista considerável de dados de teste para estas execuções, com valores bem distintos, a fim de identificar qualquer falha cometida na programação. Desta forma, o Juiz *Online* é capaz de determinar se o programa foi construído corretamente.

No segundo teste, o código fonte é compilado, porém agora ele será executado utilizando classes de teste cadastradas no exercício. O objetivo deste teste é criar múltiplas classes que irão instanciar os códigos desenvolvidos pelo aluno para testar de diversas formas possíveis estes códigos, utilizando todos os recursos disponíveis no *Java*. Por exemplo, é possível criar uma classe de teste que inicialize uma classe do aluno, atribua valores para os atributos desta classe e execute os métodos desta classe, dando assim um maior controle para o professor, na hora de planejar como pretende que o Juiz *Online* corrija o exercício.

Todos estes dados de teste e classes de teste, são mantidos pelos professores e montados de forma a identificar os principais erros cometidos pelos alunos. Por exemplo, em um exercício que calcule a diferença em dias entre duas datas, um dos testes que o professor pode cadastrar para ser aplicado é registrar uma data inválida, para se certificar que o aluno aplicou as restrições e validações proposta pelo exercício. Para que o Juiz *Online* julgue corretamente é imprescindível que o cadastro dos exercícios, e suas respectivas validações de teste, sejam feitos de forma correta.

A tecnologia do Juiz *Online* foi reaproveitada de um trabalho existente denominado CodeJudge (GURIA, 2013), o qual foi selecionado devido a fácil utilização e a fácil customização. Ele foi desenvolvido para funcionar no sistema operacional *Linux*, e interpretar as linguagens de programação *Java*, *C* e *C++*. Em

cima do CodeJudge foram aplicadas as personalizações anteriormente citadas, porém neste primeiro momento, apenas foi habilitado no *feeper* o uso da linguagem *Java*, deixando em aberto a possibilidade, se necessário, de expandir as linguagens as quais ele é capaz de interpretar.

4.3 ASPECTOS TÉCNICOS

A aplicação foi desenvolvida utilizando o modelo de arquitetura de software *Model-Controller-View* (*MVC)* e para sua concepção os seguintes recursos técnicos foram empregados:

a) Plataforma *Java J2EE* (ORACLE, 2013);
b) *JDK* 1.6 (ORACLE, 2013);
c) *Spring Web MVC Framework* (SPRING, 2013);
d) Biblioteca de estilos *Bootstrap* (BOOTSTRAP, 2013);
e) Framework *javascript jQuery* e componentes derivados deste (JQUERY, 2013);
f) Gerenciador de acesso a dados *Hibernate* (HIBERNATE, 2013);
g) Servidor de Aplicação *Glassfish* 3.1 (GLASSFISH, 2013);
h) Servidor de Banco de Dados *MySQL* 5.5 (MYSQL, 2013);
i) Hospedagem da aplicação na nuvem provida pela *Jelastic* (JELASTIC, 2014).

5 AVALIAÇÃO

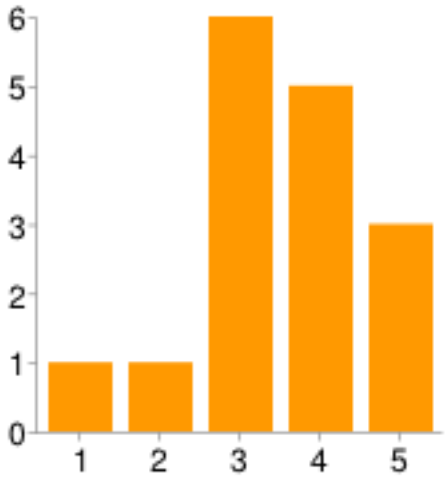
Uma avaliação experimental foi realizada em uma turma com 24 alunos de uma disciplina de programação em laboratório do primeiro semestre dos cursos de tecnologia da informação de uma universidade da grande Porto Alegre. O *feeper* foi utilizado pelos alunos e pelo professor durante um semestre de aula, mais especificamente dois meses. Após este período, para avaliar a experiência no uso da ferramenta, foi aplicado um questionário diferente para alunos e professor. As perguntas foram montadas utilizando a escala de *Likert*, onde os alunos deveriam escolher um valor entre 1 a 5. Os dados foram analisados com a medida por moda (*Mo*) ou a resposta mais frequente, assim como a média (μ). 17 alunos responderam esse questionário final.

O questionário do aluno foi dividido em partes, onde a primeira parte serviu para identificar o aluno, onde este informou seu nome, idade, a quantos semestres está estudando na universidade, o seu nível de conhecimento em programação antes de cursar as cadeiras de programação e o seu nível de conhecimento atual. Estas perguntas relacionadas ao nível de conhecimento serviram para mapear a evolução que o aluno teve no decorrer do seu estudo, onde a pesquisa mostrou que o conhecimento inicial passou de nenhum ou muito baixo para um conhecimento intermediário. Por ser uma disciplina de primeiro semestre de curso, já era esperado que o conhecimento do aluno não atingisse o nível avançado.

Na segunda parte da pesquisa foram aplicadas perguntas relacionadas com a usabilidade da ferramenta e também a importância deste quesito em ferramentas desta categoria. O resultado da pesquisa mostrou que os alunos tiveram muita facilidade para aprender a utilizar o *feeper* (*Mo*=5; μ=4,3) e que o *layout* da ferramenta facilitou muito o entendimento das funcionalidades (*Mo*=5; μ=4,4). Os alunos também indicaram a grande importância que um *layout* bonito e agradável tem em ferramentas que objetivam o aprendizado (*Mo*=5; μ=4,4). De uma forma geral, a interatividade do *feeper* foi avaliada entre intermediária e boa (*Mo*=4; μ=3,6).

A última parte da pesquisa focou nas funcionalidades disponibilizadas pelo *feeper* e também na utilidade do uso de ferramentas que apoiam o aluno nas cadeiras de programação. A grande maioria dos alunos, de acordo com a pesquisa, acham que o uso de ferramentas para apoia-los nas cadeiras de programação tem muita utilidade (*Mo*=5; μ=4,5). A utilidade do *feeper* no aprendizado foi avaliada como intermediária, tendendo para boa (*Mo*=3; μ=3,5). A Figura 10 ilustra o gráfico em colunas para as respostas dadas pelos estudantes para a pergunta “O feeper foi útil para o seu aprendizado de programação?”. O eixo horizontal representa as possíveis respostas do aluno (1 a 5, segundo escala de *Likert*) e o eixo vertical o número de estudantes que escolheu aquela resposta.

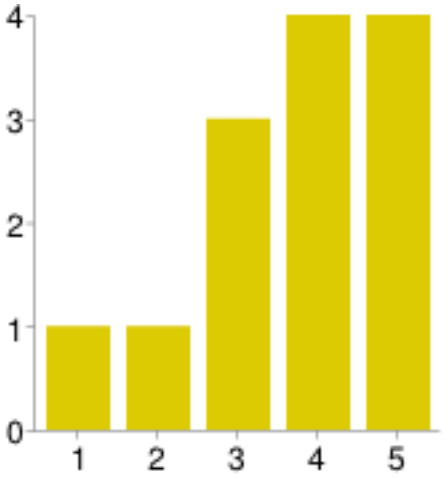
Figura 10 – Gráfico de respostas da pergunta



Fonte: Google Forms

A funcionalidade para envio (e recebimento) de mensagens ao professor foi muito bem avaliada, ficando empatada entre muito útil e útil (*Mo*=5; μ=3,7). A Figura 11 ilustra o gráfico em colunas para as respostas dadas pelos estudantes para a pergunta “A funcionalidade do *feeper* de enviar (e receber) mensagens ao professor foi útil na sua comunicação com o professor?”. O eixo horizontal representa as possíveis respostas do aluno (1 a 5, segundo escala de *Likert*) e o eixo vertical o número de estudantes que escolheu aquela resposta.

Figura 11 – Gráfico de respostas da pergunta



Fonte: Google Forms

A grande maioria dos alunos julgou que o retorno fornecido pelo *feeper* após corrigir seus códigos fonte foi útil (*Mo*=4; μ=3,6), porém as mensagens que validam as respostas objetivando ajudar a solucionar o erro teve dois picos na avaliação, no primeiro sendo avaliadas como úteis e no segundo pico avaliadas como um nível abaixo de intermediário (*Mo*=4; μ=3,4). Ao final do questionário foi disponibilizado um campo para que os alunos informassem sugestões de funcionalidades que julguem necessárias, onde surgiram ideias como apresentação de dicas e exemplos para solucionar os erros e a tradução de mensagens de erro geradas ao compilar o código fonte.

O questionário do professor também foi dividido em partes, porém diferente do questionário do aluno, neste não foi incluída a parte de identificação do professor. Como diferencial o questionário do professor tem, para cada pergunta na escala de *Likert*, um campo de texto para que o professor justifique sua resposta, permitindo agregar mais informação qualitativa no seu retorno.

A primeira parte do questionário apresentou as mesmas perguntas referentes a usabilidade da ferramenta e também a importância da usabilidade neste categoria de ferramenta. De acordo com a pesquisa, o professor julgou ser fácil o aprendizado para usar a ferramenta, justificando que a ferramenta é bastante intuitiva, mas que estaria faltando apenas um tutorial para explicar o funcionamento do cadastro dos testes para os exercícios de programação, uma vez que o cadastro deve ser muito bem feito preenchido para gerar um retorno positivo ao aluno. O professor também julgou como muito importante que ferramentas desta categoria tenham o *layout* bonito e agradável. De forma geral a interatividade e o layout aplicado no *feeper* foram muito satisfatório.

A segunda parte do questionário foi referente as funcionalidades disponibilizadas para o professor e também na utilidade do uso de ferramentas que apoiam o aluno nas cadeiras de programação. O professor julgou que o uso de ferramentas para apoiarem os aprendizes é muito útil pois podem fornecer um *feedback* mais individualizado e também mais rápido a eles. Com relação ao cadastro de exercícios, a funcionalidade foi avaliada como flexível, faltando uma ajuda para o professor, pois o cadastro de testes é bastante exaustivo e propício a erros humanos. O apoio fornecido pelo *feeper* aos alunos foi muito útil pois a ferramenta forneceu o retorno imediato aos alunos quando estes enviavam seus programas para validação. Outra funcionalidade que recebeu nota máxima foi a

funcionalidade para envio de mensagens aos alunos a qual foi muito elogiada pelo professor por ser extremamente útil, funcional e de fácil uso. A exibição do resultado do exercício foi útil, mas, como depende diretamente do cadastro das validações feitas pelo professor, está suscetível a falhas. O tempo de resposta da ferramenta foi bem avaliado, com ressalva ao Juiz *Online*, que demorava um pouco a mais para corrigir o exercício do aluno. De acordo com a pesquisa o professor ficou muito motivado para usar a ferramenta para apoiar as suas aulas de programação, justificando que ela facilitou muito seu trabalho na comunicação com os alunos para sinalizar que parte do código continha erro ou que precisava ser melhorado. Outro ponto ressaltado na justificativa do professor é que a ferramenta facilitou o acompanhamento das soluções dos alunos, permitindo que ele visualizasse todas as versões das respostas de forma extremamente rápida agilizando a correção do exercício.

O questionário teve ainda uma pergunta listando todas as funcionalidades disponíveis para o professor, permitindo que este selecionasse múltiplas funcionalidades que mais lhe agradaram. Foram selecionadas as funcionalidades de envio de mensagens aos alunos, a visualização dos resultados dos exercícios e a recepção de dúvidas dos alunos permitindo a visualização do código fonte no mesmo momento.

Da mesma forma que no questionário disponibilizado para os alunos, o professor também pode sugerir outras funcionalidades importantes para melhorar o uso do *feeper*, tais como, uma funcionalidade para geração automática de testes, sistema de dicas em português para os alunos baseado nos principais erros de compilação dos alunos, inserção de técnicas de gamificação para aumentar a motivação dos alunos, e por fim uma evolução no editor de código fonte para permitir *debug* tanto para o professor quanto para o aluno.

6 CONCLUSÃO

Este artigo apresentou uma ferramenta *online* para ser utilizada como apoio, para alunos e professores, em disciplinas de programação, oferecendo funcionalidades para tal. Dentre todas as funcionalidades, ganham destaque o recurso de *feedback* personalizado, onde através de um cadastro de exercícios o professor pode montar cenários de teste com mensagens personalizadas para

quando o teste falhar. Mensagem esta que será apresentada de forma imediata ao aluno, assim que este submeter sua resposta. Outro recurso muito importante apresentado neste trabalho é a facilidade criada para enviar mensagens ao professor e também para que o professor envie mensagens para seus alunos, a fim de comentar sobre partes específicas dos códigos dos programas dos seus alunos. Quando o estudante manda mensagens para o professor, esta mensagem é vinculada a um trecho de código fonte, o que permite destacar a linha a qual está ocorrendo sua dúvida. Para o professor a facilidade de acompanhar a sua turma através de um painel de resultados dos exercícios, permite que este visualize através de uma grade como os seus alunos estão evoluindo na resolução dos exercícios, permitindo inclusive que o professor visualize um histórico de todas as respostas do aluno, com a data da resposta, os códigos utilizados e o *feedback* emitido pelo Juiz *Online*.

Os resultados deste trabalho foram obtidos através de uma avaliação da experiência de uso da ferramenta, onde um questionário com questões fechadas foi aplicado para os alunos e para o professor que usaram a ferramenta durante um semestre de uma disciplina de laboratório de programação de primeiro semestre de cursos em informática, resultados estes que, serviram para identificar se os objetivos foram ou não atingidos. A pesquisa indicou que as principais funcionalidades propostas foram úteis para os avaliados, e que de fato agregaram para a sua experiência de uso. Com a avaliação também foi possível identificar pontos a serem melhorados, como por exemplo, automatizar a criação de cenários de teste, para diminuir o trabalho do professor, evitando também eventuais falhas humanas no momento do cadastro destas informações, que são cruciais para uma boa avaliação dos exercícios. Outras sugestões dadas pelo professor e pelos alunos poderão ser adicionadas em trabalhos futuros.

Com o resultado desta avaliação também é possível perceber que há uma necessidade de ferramentas que apoiem os alunos nas disciplinas de programação, pois como apresentado pela pesquisa, a grande maioria dos alunos ou nunca teve contato com uma linguagem de programação ou o seu conhecimento é muito baixo. Sendo assim a pesquisa mostrou os pontos que ainda devem ser aprimorados para que a ferramenta possa evoluir e não ser mais apenas um experimento, e sim um ambiente efetivo para apoiar estudantes em disciplinas de programação.

6.1 TRABALHOS FUTUROS

Durante o desenvolvimento do *feeper* algumas funcionalidades foram identificadas como possíveis melhorias que possam ser aplicadas para aperfeiçoar o uso da ferramenta.

Uma das principais melhorias identificadas, é o uso de gamificação como forma de motivação para os alunos. Este tema será pesquisado e trabalhado em uma tese de mestrado que dará sequencia a este trabalho.

Outra funcionalidade que será bastante útil é a detecção de plágio, onde através de um componente será possível comparar códigos fonte históricos e obter um retorno que indique o percentual de chance de um código ser plágio.

O professor atualmente apenas visualiza o resultado dos exercícios. Dessa forma, outra melhoria interessante é a de permitir atribuir notas aos exercícios, para que o professor possa corrigir uma prova e já obter a nota do aluno.

A Virtual Environment with Custom Feedback
To Support Programming Disciplines

Abstract: For most students, programming courses are challenging and often frustrating, because they demand logical and language syntax knowledge, which can take time to be acquired by the student. For teacher, the high workload can hamper the students monitoring, where they always expect to have immediate answers to their questions. Although there are some web environments to support programming learning, like JOnline, MOJO, among others, these environments return a kind of feedback used in Online Judges, notifying only if the program is working or not, but without explaining a possible cause about a wrong execution. This kind of feedback is insufficient for novice students. Moreover, these tools have few features for students and teachers comment about possible mistakes and corrections inside the code. Thus, as a way of supporting programming courses, both for students and teachers, this paper describes a virtual environment composed of resources, such as custom feedback, facilities for interaction between student and teacher and to help teachers to monitor students. An experimental evaluation conducted with a group of 24 students who used the tool in a programming classroom over a semester showed, through a questionnaire with closed questions, that the functionalities provided by the

tool helped students to more easily identify problems in their programs and how to fix them, as well as improved communication between students and teacher.

Keywords: eLearning. Online Judge. Programming Disciplines. Interaction with the Teacher

REFERÊNCIAS

AMBRÓSIO, Ana Paula L. et al. **Programação de Computadores**: Compreender as dificuldades de aprendizagem dos alunos. Revista Galego-Portuguesa de Psicoloxía e Educación, v. 19, p. 185-197, 2011.

ANZAI, Yuichiro; SIMON, Herbert A. **The Theory of Learning by Doing**. Carnegie-Mellon University, Pittsburgh, 1979.

BECK, Kent; GAMMA, Erich. **JUnit Cookbook**. Disponível em: <http://junit.sourceforge.net/doc/cookbook/cookbook.htm>. Acesso em: 15 nov. 2013.

BOOTSTRAP. **Getting started** Disponível em: <http://getbootstrap.com/>. Acesso em: 10 nov. 2013.

BORGES, Marcos A. **Avaliação de uma metodologia alternativa para a aprendizagem de programação**. In: VIII Workshop de Educação em Computação. Campinas: Faculdade Campo Limpo Paulista, 2000. Disponível em: <http://www.niee.ufrgs.br/eventos/SBC/2000/pdf/wei/relatos/selecionados/wei006.pdf >. Acesso em: 13 nov. 2013.

CAMPOS, Cassio P De; FERREIRA, Carlos E. **BOCA**: um sistema de apoio a competições de programação. In: Workshop de Educação em Computação. São Paulo: Pontifíıcia Universidade Católica, 2004.

CHAVES, José O. et al. **MOJO**: uma ferramenta para auxiliar o professor em disciplinas de programação. In: X Congresso Brasileiro de Ensino Superior a Distância. Belém: UNIREDE, 2013.

CODE.ORG. **What's wrong with this picture?** Disponível em: <http://code.org/stats>. Acesso em: 10 nov. 2013.

ERECHIM, URI. **URI Online Judge**. Disponível em: <http://www.urionlinejudge.com.br>. Acesso em: 22 set. 2013.

GLASSFISH. **Glassfish Server** Disponível em: <https://glassfish.java.net>. Acesso em: 10 nov. 2013.

GURIA, Sankha N. **CodeJudge** Disponível em: <http://sankhs.com/codejudge/>. Acesso em: 21 nov. 2013.

HIBERNATE. **Hibernate ORM** Disponível em: <http://hibernate.org/orm>. Acesso em: 10 nov. 2013.

JELASTIC. **Platform-as-Infrastructure Provider** Disponível em: <https://www.jelastic.com>. Acesso em: 20 mar. 2014.

JENKINS, Tony. **The motivation of students of programming**. ACM SIGCSE Bulletin, v. 33, n. 3, p. 53-56, 2001. Disponível em: <http://portal.acm.org/citation.cfm?doid=507758.377472>. Acesso em: 28 nov. 2013.

JOY, Mike; GRIFFITHS, Nathan; BOYATT, Russell. **The BOSS Online Submission and Assessment System**. ACM, v. 5, n. 3, p. 1-28, 2005.

JQUERY. **jQuery.** Disponível em: <http://jquery.com/>. Acesso em: 10 nov. 2013.

MOODLE. **Moodle**: Modular Object-Oriented Dynamic Learning Environment. Disponível em: <http://moodle.org>. Acesso em: 15 nov. 2013.

MYSQL. **MySQL – The world's most popular open source database** Disponível em: <https://www.mysql.com>. Acesso em: 10 nov. 2013.

ORACLE. **Java Plataform, Enterprise Edition.** Disponível em: <http://www.oracle.com/technetwork/java/javaee/overview/index.html>. Acesso em: 10 nov. 2013.

SANTOS, Joanna C S; RIBEIRO, Admilson R L. **JOnline**: proposta preliminar de um juiz online didático para o ensino de programação. In: XXII Simpósio Brasileiro de Informática na Educação. São Cristóvão: Universidade Federal de Sergipe, 2011, p. 964-967.

SPRING. **Spring Framework** Disponível em: <http://projects.spring.io/spring-framework>. Acesso em: 10 nov. 2013.

WOOLF, Beverly P. **Building Intelligent Interactive Tutors**: Student-centered strategies for revolutionizing e-learning. 1. ed. Amherst: Morgan Kaufmann, 2009.

GLOSSÁRIO

BOOTSTRAP: Biblioteca para o desenvolvimento de telas responsivas para aplicações *web*.

C: Linguagem de programação.

C++: Linguagem de programação.

CHAT: Recurso *online* para a troca de mensagens em tempo real entre indivíduos.

CLASSE: Arquivos de código fonte das linguagens de programação.

COMPILAÇÃO: Processo automatizado de análise do código fonte escrito e transformação em linguagem intermediária reconhecida pelo computador.

DEBUG: Ação de executar o código fonte pausadamente linha a linha para acompanhar o que está acontecendo.

DOWNLOAD: Ação de transferir arquivos de um local remoto para um computador.

FEEDBACK: Termo inglês que tem o significado de um parecer, um comentário a respeito de determinado assunto.

FEEPER: Nome dado ao sistema desenvolvido neste trabalho, é um acrônimo entre as palavras *feedback* personalizado.

GLASSFISH: Servidor que realiza a armazenagem de uma aplicação *web*, é responsável por disponibilizar esta aplicação acessível a todos, gerenciando requisições dos usuários.

HIBERNATE: Biblioteca de gerenciamento de acesso a dados.

HTML: Linguagem de marcação utilizada para *web*.

IDEONE: Serviço ofertado pela empresa *Sphere*, é um compilador *web* para diversas linguagens de programação.

INSTANCIAR: Na programação orientada a objetos, instanciar significa criar um novo objeto em memória para ser utilizado.

J2EE: Plataforma para desenvolvimento e construção de aplicações *web*.

JAVA: Linguagem de programação.

JAVASCRIPT: Linguagem de programação.

JDK: Ambiente de desenvolvimento de software utilizado para desenvolver aplicações em *Java*.

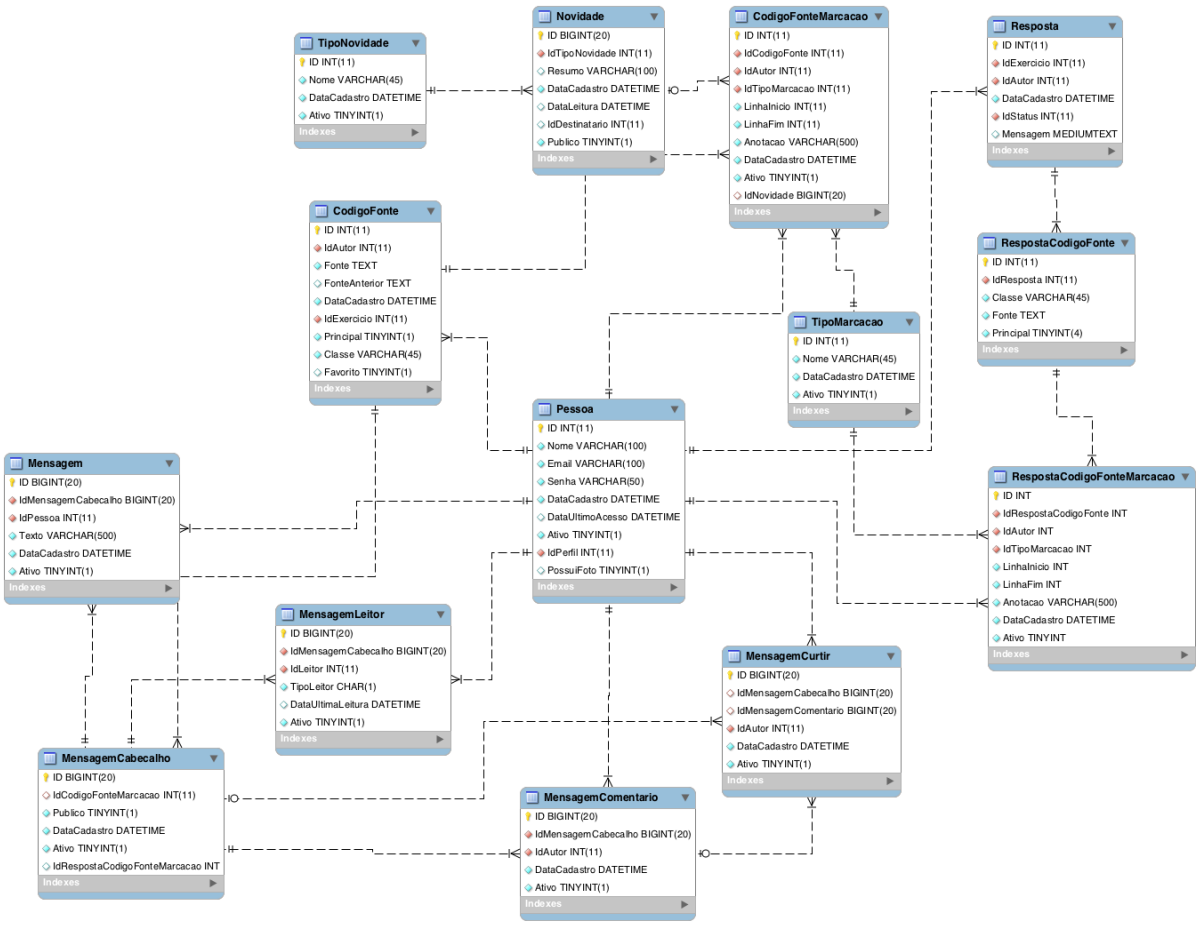
JELASTIC: Serviço de hospedagem de aplicações na nuvem.

JQUERY: Biblioteca de programação para a linguagem de programação *Javascript*.

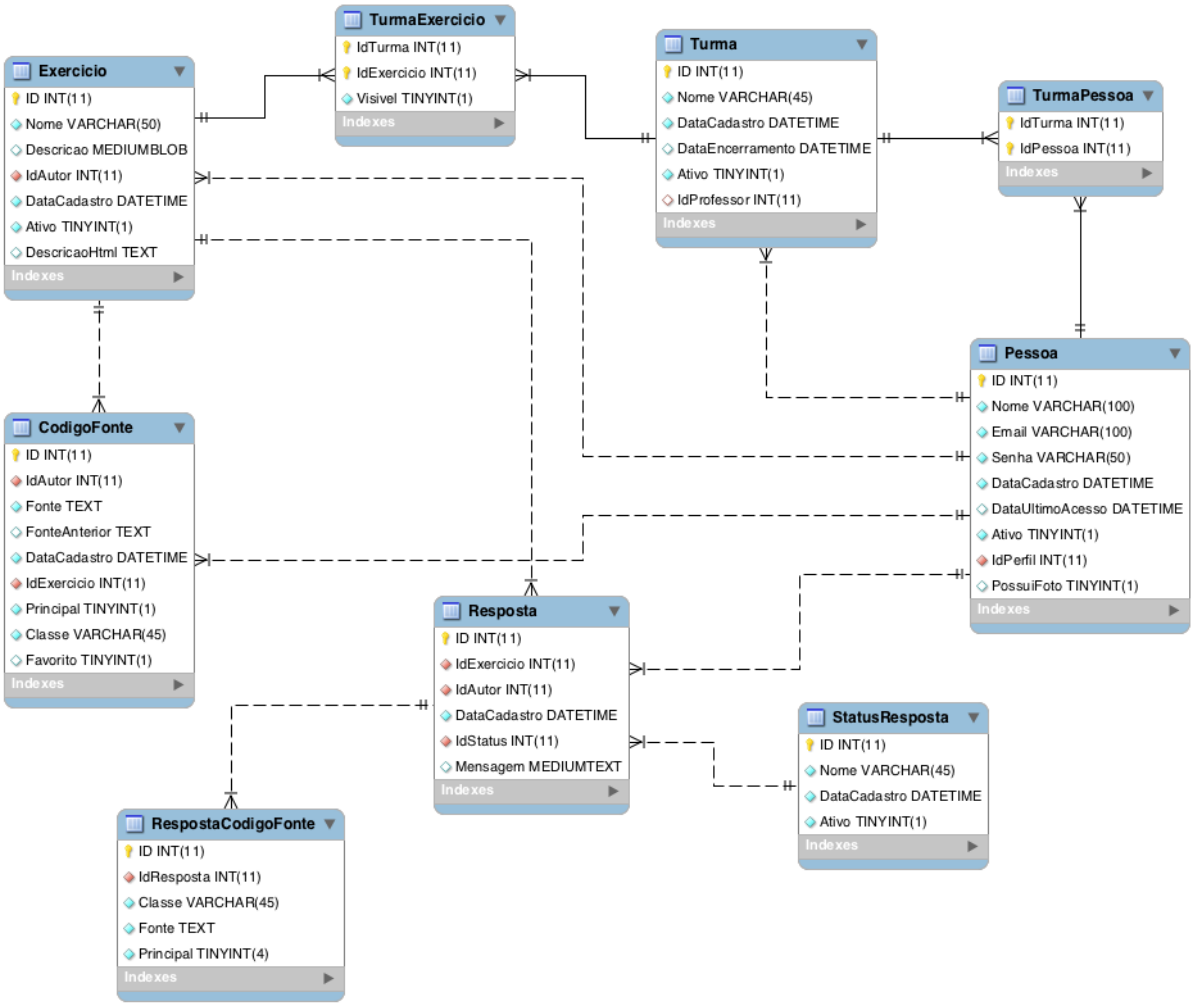
JUNIT: Biblioteca para criação de testes para a linguagem de programação *Java*.

LEARNING BY DOING: Termo em inglês que tem o significado de aprender fazendo.
LINUX: Sistema operacional de software livre.
LOGS: Registro de informações históricas de ações executadas na aplicação.
MOODLE: Plataforma *web* para ensino a distância.
MVC: Padrão de arquitetura de software.
MYSQL: Banco de dados relacional gratuito.
PRINTS: Ou PRINTSCREEN, é a ação de capturar/tirar uma foto da tela do computador através de uma funcionalidade nativa do sistema operacional.
RANKING: Termo em inglês que tem o significado de posição.
SERVLETS: Classe *Java* utilizada para estender funcionalidades de um servidor.
SPHERE: Empresa responsável pelo produto *Ideone* citado acima.
SPRING WEB MVC FRAMEWORK: Biblioteca para utilização do padrão de arquitetura *MVC* no *Java*.
UPLOAD: Ação de transferir arquivos de um computador para um local remoto.
WEBSERVICE: Solução utilizada para a integração de sistemas e na comunicação entre aplicações distintas.
WINDOWS: Sistema operacional.

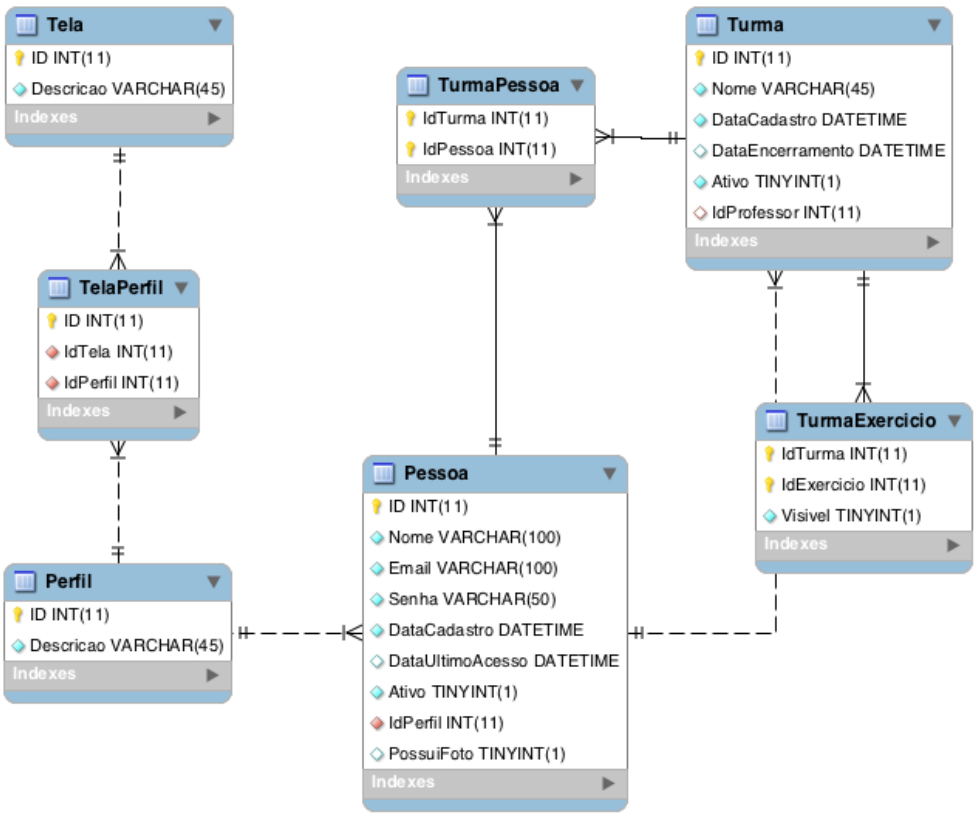
APÊNDICE A – Modelo ER para o envio de mensagens



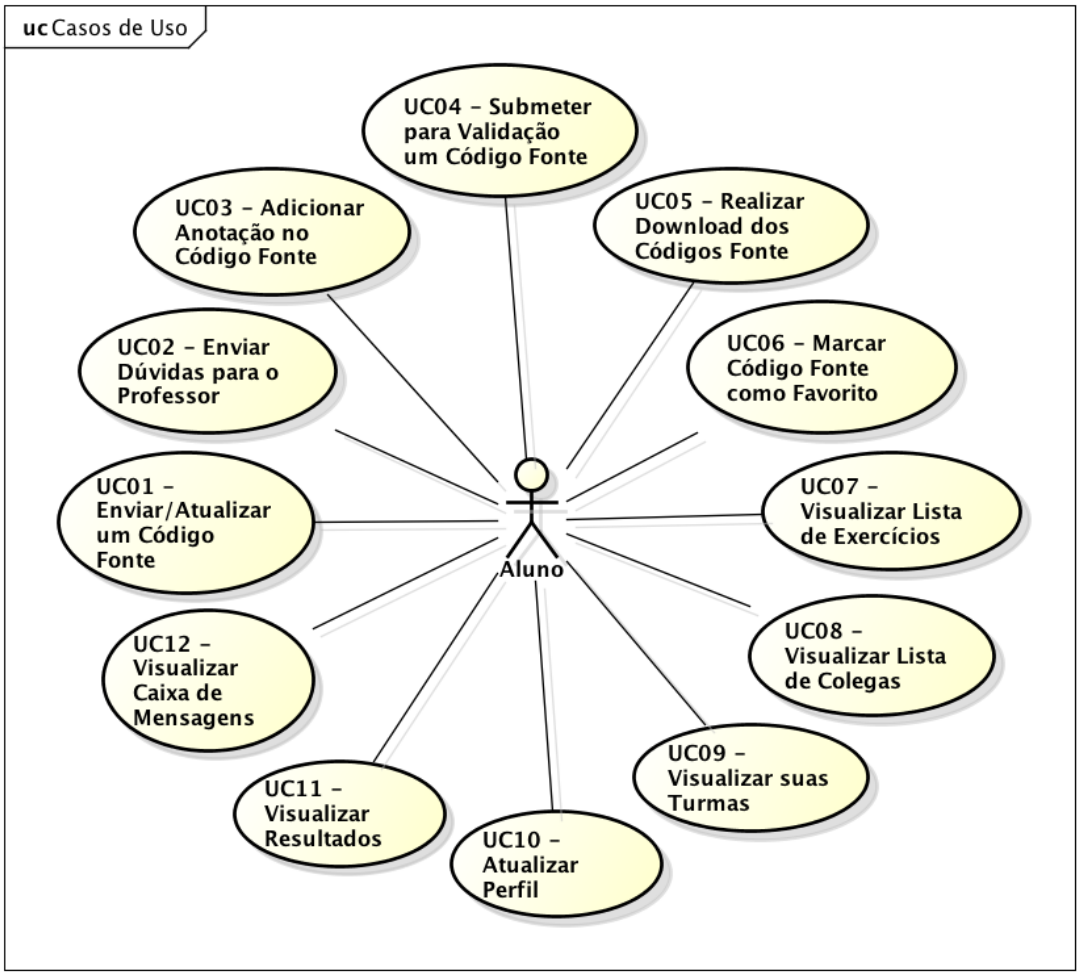
APÊNDICE B – Modelo ER para responder exercícios



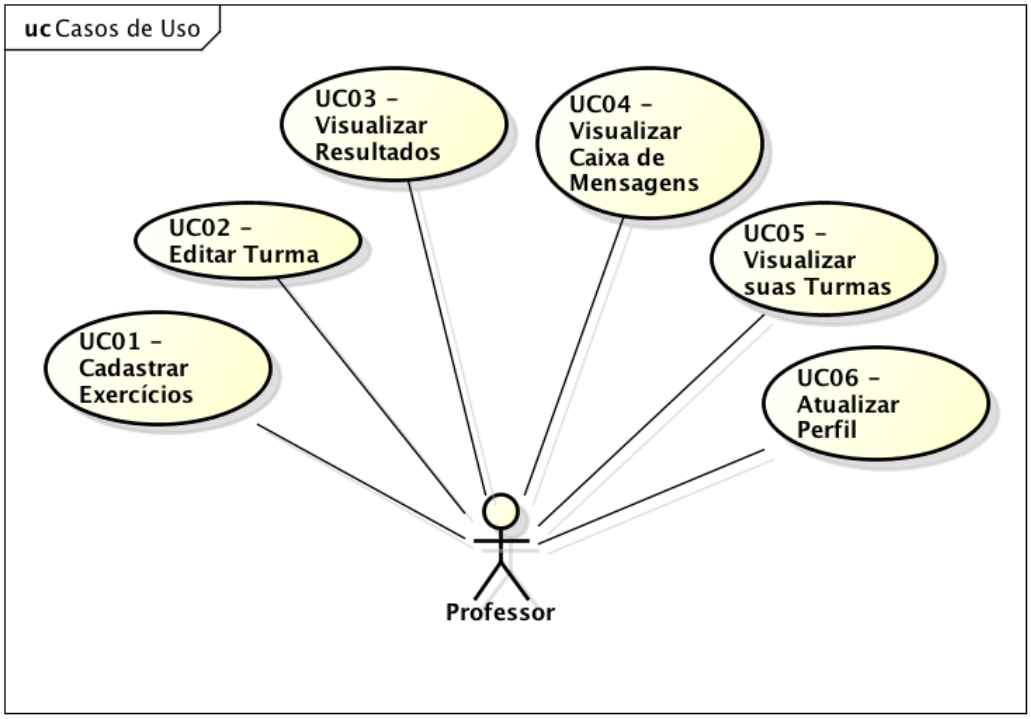
APÊNDICE C – Modelo ER para o controle de acessos



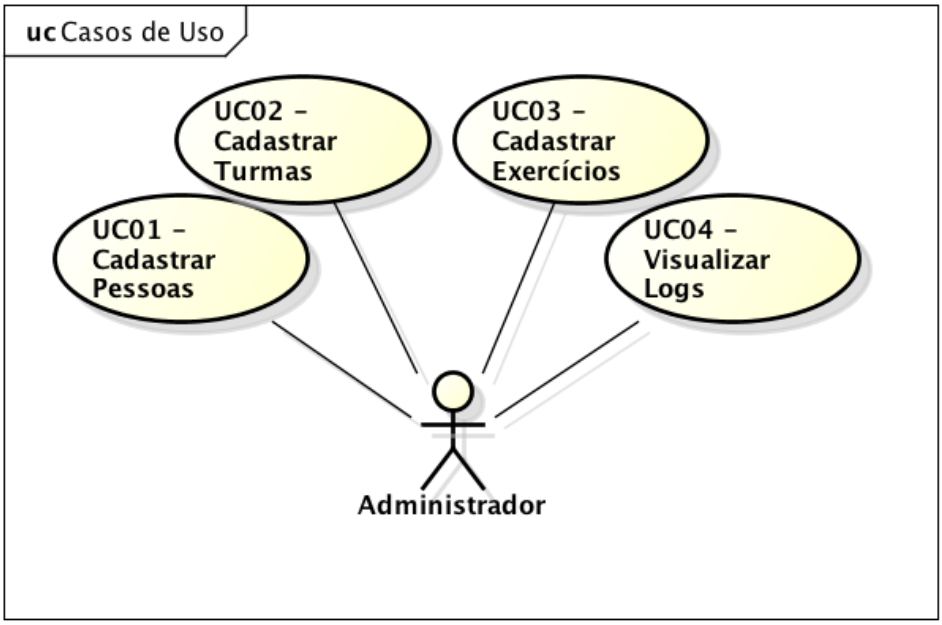
APÊNDICE D – Diagrama de casos de uso do módulo do aluno



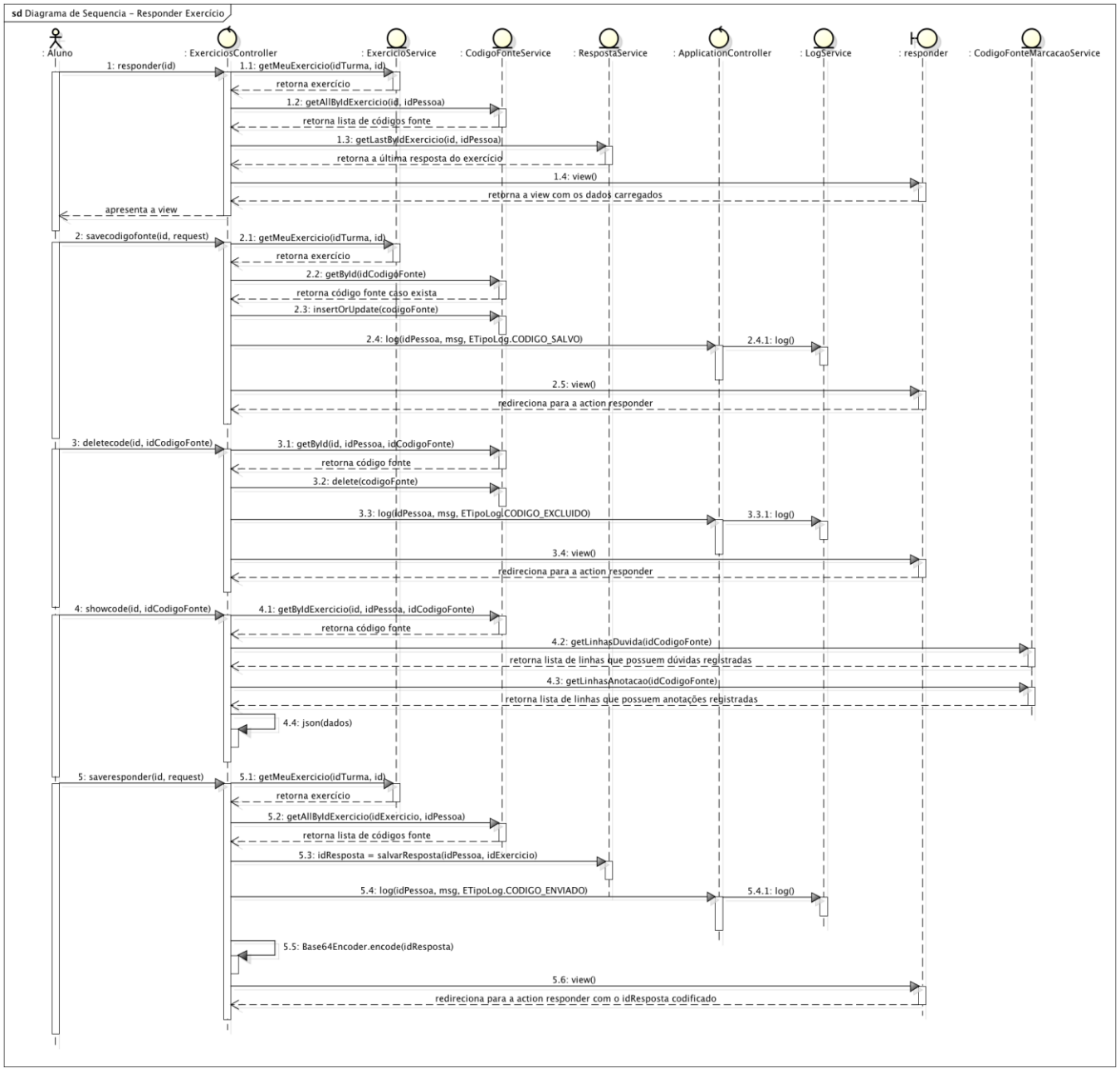
APÊNDICE E – Diagrama de casos de uso do módulo do professor



APÊNDICE F – Diagrama de casos de uso do módulo do administrador



APÊNDICE G – Diagrama de sequência da funcionalidade “Responder Exercício”



APÊNDICE H – Diagrama de sequência da funcionalidade “Juiz *Online*”