HOLOS
ISSN 1807 - 1600

MOJO: UMA FERRAMENTA PARA INTEGRAR JUÍZES ONLINE AO MOODLE NO APOIO AO ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO

J. O. M. CHAVES[1*], A. F. CASTRO[1], R. W. LIMA[2], M. V. A. LIMA[2] e K. H. A. FERREIRA[2]
[1]Universidade Federal Rural do Semi-árido
[2]Universidade do Estado do Rio Grande do Norte
oswaldo.mesquita@gmail.com[*]

Artigo submetido em janeiro/2014 e aceito em setembro/2014

DOI: 10.15628/holos.2014.1904

RESUMO

Este artigo apresenta o MOJO, uma ferramenta que integra o Moodle e os Juízes Online. Os sistemas de Juízes Online são comumente utilizados em maratonas de programação. Eles são famosos pelo seu repositório de questões e avaliação automática de código-fonte. A ideia para esta integração surgiu da exigência de um maior envolvimento do professor de disciplinas de programação, e como uma forma de disponibilizar um maior número de questões para utilização na prática. Normalmente o professor dessas disciplinas fica sobrecarregado pelos processos de elaboração, avaliação e fornecimento de feedback das questões aos alunos. Esta ferramenta visa diminuir esta sobrecarga, por meio da automatização desses processos.

PALAVRAS-CHAVE: mojo, integração, moodle, juízes online, disciplinas de programação.

MOJO: A TOOL TO INTEGRATE THE ONLINE JUDGES IN MOODLE TO SUPPORT TEACHING AND LEARNING PROGRAMMING

ABSTRACT

This paper introduces MOJO, a tool that integrates Moodle and Online Judges. Online Judges are commonly used in programming marathons. They are famous for their repository of questions and automatic evaluation of source code. The idea of integrating these systems into Moodle came from the demand for more professor involvement in programming disciplines, and as a way to provide a greater number of issues for practice use. Usually the professor is overwhelmed by preparing, submitting, evaluating and providing the necessary feedback to the students of the issues. This tool aim to reduce this overhead, through automatization of these processes.

KEYWORDS: mojo, integration, moodle, online judges, programming disciplines.

1 INTRODUÇÃO

Disciplinas de programação são essenciais aos estudantes de computação, pois constituem a base para muitas áreas de aplicação da informática. O devido aprendizado dessas disciplinas torna o indivíduo apto a utilizar a lógica de programação na resolução de diversos problemas, fator importante e que será útil em disciplinas mais avançadas.

Por sua vez, as Tecnologias de Informação e Comunicação (TIC) têm contribuído de maneira significativa para a ampliação das estratégias de ensino e aprendizagem, proporcionando diferentes ferramentas e artefatos que apoiam o processo de aprendizagem (AMARAL *et al.*, 2011). Dessa forma, o sistema educacional tem caminhado rumo a novas possibilidades e sendo enriquecido com novas técnicas que visam um melhor desenvolvimento intelectual dos alunos.

A utilização dessas novas tecnologias tem sido de suma importância para auxiliar professores e alunos. Podem-se citar exemplos de sua utilização no auxílio a disciplinas de programação. Essas ferramentas, aplicadas a tais disciplinas, podem facilitar o aprendizado de programação, reduzindo os índices de reprovação e o mau desempenho em outras matérias que tenham programação como pré-requisito.

As dificuldades encontradas no aprendizado dessas disciplinas refletem em alguns dos problemas (índices de reprovação e mau desempenho) que podem provocar a evasão do curso. De acordo com o jornal Folha de São Paulo, que analisou dados do Ministério da Educação (MEC), existe uma alarmante taxa de 28%, em média, de evasão nos cursos de Ciência da Computação (TAKAHASHI, 2009) nas instituições de ensino superior brasileiras.

Sobre este problema, observa-se, principalmente, nos primeiros semestres dos cursos de graduação em computação, uma quantidade relevante de alunos que reprova, desiste ou obtém um baixo rendimento nas disciplinas que focam em programação. Isso é ocasionado, na maioria dos casos, devido à complexidade em desenvolver lógica de programação ou à sintaxe da linguagem de programação associada (MOTA *et al.*, 2009).

No sentido de melhorar esta situação, muitas ferramentas de TIC têm sido propostas para auxiliar o professor no ensino de programação. Um exemplo a ser citado é a utilização dos Ambientes Virtuais de Aprendizagem (AVA), categoria que tem como forte representante o Moodle (KUMAR *et al.*, 2011). Porém, mesmo com o advento dessas ferramentas, algumas barreiras ainda são encontradas pelo professor no ensino da disciplina, como, por exemplo, a dificuldade que se tem ao avaliar todas as atividades de uma turma extensa de alunos em pouco tempo.

No que diz respeito à avaliação de atividades de programação (atividades práticas de codificação), existem alguns sistemas como, por exemplo, os sistemas de Juízes *Online* (ZHIGANG *et al.*, 2012) que são utilizados em maratonas de programação, e conhecidos por seu amplo repositório de questões e pelo seu processo de avaliação automática de código-fonte. A avaliação feita por estes sistemas gera respostas como: certo, errado, erro de compilação e erro em tempo de execução.

Normalmente, quer seja na modalidade de Educação a Distância (EaD) ou na modalidade de educação presencial, as ferramentas de auxílio existentes fornecem um ambiente que permite

ao aluno criar seus algoritmos e codificá-los em alguma linguagem de programação. Porém, é de responsabilidade do professor a elaboração, submissão, avaliação das atividades e fornecimento de *feedback* a todos os seus alunos. Todo este processo, quando realizado para vários alunos em várias turmas, acaba causando desgaste (físico e mental) ao professor e uma sobrecarga de suas atividades.

Com base neste cenário atual da tecnologia educacional, no que diz respeito ao ensino de programação, é possível integrar dois ou mais ambientes distintos de maneira complementar para um propósito comum. Desta forma, fazendo surgir um novo sistema que contém as características dos sistemas originais, e embora esta integração seja um processo complexo, este ainda é o cenário mais comumente encontrado.

Mesmo os Ambientes Virtuais de Aprendizagem atuais, que apresentam um conjunto de ferramentas de propósito geral e podem ser empregados em vários cursos, raras vezes são concebidos levando em consideração a possibilidade de extensão ou de integração com outras plataformas. Uma exceção a este modelo é o Moodle que possui documentação específica para a agregação de novas funcionalidades.

No que diz respeito às disciplinas de programação, mesmo com o uso das ferramentas existentes nestes ambientes virtuais, como, disponibilização de notas de aulas, exercícios e fóruns; percebe-se que estas ferramentas não são o bastante para solucionar a dificuldade do professor de acompanhar e dar o *feedback* necessário ao aprendizado de todos os seus alunos.

Como forma de contribuir para melhorar as condições de ensino e aprendizagem em disciplinas de programação, e se aproveitando da possibilidade de extensão oferecida pelo Moodle, este artigo apresenta o Módulo de Integração com os Juízes *Online* (MOJO), uma ferramenta que integra os sistemas de Juízes *Online* ao Moodle. Esta integração permitirá ao professor gerenciar os recursos didáticos necessários (disponibilização de notas, material de estudos, atividades), além de automatizar o processo de elaboração, submissão e avaliação de atividades, propiciando um melhor acompanhamento e *feedback* aos alunos.

2 AMBIENTES ENVOLVIDOS E TRABALHOS RELACIONADOS

2.1 MOODLE

Moodle é um acrônimo para *Modular Object-Oriented Dynamic Learning Environment*. Foi desenvolvido pelo australiano Martin Dougiamas em 1999 e possui tradução para diversos idiomas, incluindo o português. É uma plataforma *open source* (código aberto) de *e-learning* (aprendizagem eletrônica) que é utilizada em todo o mundo, em 193 países, com mais de 400.000 usuários registrados (KUMAR *et al.*, 2011).

O Moodle faz parte do grupo de Ambientes Virtuais de Aprendizagem. Esses ambientes permitem, por meio da Internet, a produção de conteúdos e canais variados de comunicação, bem como o gerenciamento de dados e controle total de informações. Além disso, possui uma comunidade formada por professores, pesquisadores, e, principalmente, programadores que mantém um Portal na *Web* funcionando como uma central de informações, discussões e colaborações.

No contexto educacional, a ferramenta oferece a professores e alunos um ambiente capaz de reunir a maioria das informações e eventos relevantes, associados a uma disciplina de determinado curso. Uma importante observação é que esta é a plataforma oficial do Ministério da Educação (MEC) para as escolas públicas brasileiras (MARTINS e GIRAFFA, 2008), e que se mostrou, também, uma importante ferramenta auxiliar no ensino superior, podendo ser utilizada tanto na modalidade de ensino à distância como na modalidade de ensino presencial.

A plataforma conta com uma diversidade de recursos educacionais, permitindo larga flexibilidade para configuração e utilização. Por meio do seu desenvolvimento extremamente modular é possível, ainda, a inclusão de novos recursos que melhor se adaptem às necessidades de quem utiliza o ambiente. Porém, na prática, para agregar um recurso externo ao Moodle como, por exemplo, um novo módulo, é preciso ter conhecimento de sua estrutura assim como saber utilizar os recursos internos por meio de suas bibliotecas de desenvolvimento.

O modelo de desenvolvimento do Moodle prevê a sua extensão podendo, assim, serem desenvolvidas e disponibilizadas novas funcionalidades por meio de *plugins* para que outros usuários possam usufruir dos novos recursos. Uma das formas de extensão e criação de novas funcionalidades é a implementação de módulos. Esta característica se mostra relevante para o desenvolvimento da integração realizada neste trabalho.

Todo esse potencial oferecido pela plataforma para a criação e extensão de funcionalidades justifica sua integração com outras ferramentas como, no caso deste trabalho, os sistemas de Juízes *Online*.

2.2 JUÍZES ONLINE

A maioria dos programas de natureza algorítmica necessita apenas obter como entrada um padrão de dados devidamente formatado e, a partir desses dados, realizar o devido processamento. Após o processamento, os resultados são apresentados de maneira formatada em uma saída padronizada. Dessa maneira, é possível que a avaliação de programas seja feita automaticamente utilizando uma ferramenta que gere os dados de entrada e outra que obtenha e verifique os resultados obtidos (dados de saída).

Os Juízes *Online* são sistemas que compilam, executam e testam códigos-fonte com base em dados padronizados para julgar se estão corretos (ZHIGANG *et al.*, 2012). Basicamente, este processo de avaliação automática ocorre da seguinte maneira: o juiz recebe o código-fonte, durante a execução do código, o juiz utiliza dados formatados como a entrada do programa, processa esses dados e realiza a comparação dos resultados obtidos com os resultados esperados, dando uma resposta apropriada com base nessas comparações (certo, errado, erro de compilação ou de execução). Este método é utilizado em muitos concursos de programação, tais como *ACM Collegiate Programming Contest International* (ACM-ICPC, 2014) e *TopCoder* (INC., 2014).

Estes sistemas podem ser facilmente encontrados na Internet e como exemplos podem ser citados o *Timus Online Judge* (TIMUS ONLINE JUDGE TEAM, 2014), o *UVA Online Judge* (UNIVERSIDAD DE VALLADOLID, 2014) e o SPOJ Brasil (SPHERE RESEARCH LABS, 2014), este último utilizado no desenvolvimento da ferramenta. Nestes sistemas são disponibilizados vários problemas a serem submetidos para resolução e são tão atraentes que muitos alunos começam a praticar suas habilidades de programação neles (ZHIGANG *et al.*, 2012).

2.3 TRABALHOS RELACIONADOS

Entre os ambientes que fornecem apoio à submissão e avaliação automática de atividades de programação destaca-se a PROGTEST (SOUZA *et al.*, 2012). A PROGTEST é um ambiente *Web* automatizado que apoia a submissão e avaliação de atividades práticas de programação, e se baseia em atividades de teste de *software*. Atualmente, dá suporte a apenas duas linguagens de programação (Java e C) e utiliza um programa referência (programa oráculo) que deve ser fornecido pelo professor para avaliação das atividades dos alunos, além de utilizar diferentes ferramentas para testes.

Uma iniciativa que também utiliza Juízes *Online* é a de Santos e Ribeiro, que propõe o *JOnline* (SANTOS e RIBEIRO, 2011) um sistema que possui um Juiz *Online* próprio no qual é possível acessar problemas para serem resolvidos e submeter códigos-fonte para a visualização do resultado. Além da programação colaborativa, o sistema ainda adiciona funcionalidades didáticas ao juiz auxiliando o aluno no processo de aprendizagem.

Embora esses trabalhos tragam importantes estudos para auxiliar no ensino de programação, eles não estão integrados a um ambiente que forneça outros recursos educacionais como, por exemplo, suporte a gestão de conteúdo e ferramenta de discussão. Estes são importantes recursos, principalmente no que diz respeito às disciplinas ministradas à distância, recursos que um AVA como, por exemplo, o Moodle pode fornecer. E em algumas ferramentas específicas, além do professor ter que criar programas referências para auxiliar na correção das atividades, tem-se ainda a limitação de se trabalhar restrito a poucas linguagens de programação.

Em um contexto semelhante ao abordado nesta pesquisa, algumas iniciativas foram realizadas no sentido de integrar recursos de apoio a disciplinas de programação ao ambiente Moodle, como é o caso da iniciativa de Sirotheau *et al*. (2011), do BOCA-LAB (FRANÇA, 2012) e do *Onlinejudge* para Moodle (ONLINEJUDGE, 2014). Em Sirotheau *et al*. (2011), com o objetivo de contribuir para uma melhor compreensão do estudante no aprendizado de programação, a ferramenta *JavaTool* (MOTA *et al.*, 2009) foi integrada ao Moodle juntamente com o avaliador automático de Moreira e Favero (2009), propiciando uma maneira de visualizar e simular programas no Moodle e permitindo a combinação de técnicas para avaliação da complexidade do código. Desta forma, colaborando para uma melhor avaliação e *feedback* das atividades.

O BOCA-LAB foi desenvolvido no Departamento de Engenharia de Teleinformática (DETI) da Universidade Federal do Ceará (UFC) e surgiu da extensão de um sistema utilizado em maratonas de programação – o BOCA (CAMPOS e FERREIRA, 2004). O BOCA-LAB foi integrado ao Moodle por meio de *Web Services*. A ferramenta é capaz de compilar e executar programas escritos em diversas linguagens de programação. Os programas submetidos são então avaliados quanto a erros de compilação e execução em um processo automático.

O *Onlinejudge* para Moodle é composto por *plugins* e também foi desenvolvido para gerenciar a submissão e avaliação de códigos-fontes no Moodle. Ele pode ser integrado a outras duas aplicações: a *Sandbox* (SANDBOX, 2014) e a *Ideone* (SPHERE RESEARCH LABS, 2014). A *Sandbox* se restringe a submissões que executem em C/C++ no ambiente Linux. Já a *Ideone* permite escrever códigos-fonte em aproximadamente 40 linguagens de programação diferentes, sendo executados diretamente a partir do navegador. Entretanto, a *Ideone* é uma aplicação comercial e permite submissão de apenas 2000 códigos-fonte por mês em uma conta gratuita (ZHIGANG *et al.*, 2012). Sem a integração o *Onlinejudge* para Moodle suporta apenas as linguagens C e C++.

Outra importante iniciativa é o *JAssess* (YUSOF *et al.*, 2012), que propõe um sistema intermediário desenvolvido para fornecer, no Moodle, uma maneira prática de gerenciar a apresentação de exercícios de programação, em Java, desenvolvidos pelos alunos. Um ponto fraco encontrado no *JAssess* é o fato da ferramenta se limitar a exercícios que utilizem somente Java como linguagem de programação, enquanto outras linguagens como, por exemplo, C e C++, não são atendidas pela ferramenta.

Todos os trabalhos citados contêm importantes contribuições para o ensino de programação. Porém, mesmo com a integração ao Moodle, o professor ainda sofre com a sobrecarga de atividades no que diz respeito ao processo de elaboração, submissão, avaliação e *feedback* das atividades para uma turma extensa de alunos.

Este trabalho, em complemento aos demais trabalhos, apresenta um ambiente que, integrando os Juízes *Online* ao Moodle, fornece: 1 – Auxílio necessário ao professor no que diz respeito ao processo de elaboração, submissão e avaliação de atividades de programação; 2 – Maior agilidade nas tarefas do professor; 3 – Acompanhamento de resultados através de uma mesma interface disponível no ambiente virtual; 4 – *Feedback* mais rápido ao aluno; 5 – Suporte a uma variedade de linguagens de programação e; 6 – Reuso das atividades.

2.4 MÓDULO DE INTEGRAÇÃO COM OS JUÍZES ONLINE (MOJO)

O Moodle oferece a professores e alunos um ambiente que reúne as principais informações e eventos relativos a uma disciplina (em determinado curso). Com base nisto, sua utilização em disciplinas voltadas para o ensino de programação, seja na modalidade presencial ou à distância, pode trazer importantes benefícios no que diz respeito à organização da mesma.

Por meio desta plataforma, é permitido ao professor disponibilizar, em um espaço reservado à turma, todo o material didático digital relativo, além de propor atividades e recebimento de trabalhos no próprio ambiente virtual. Além disso, com a possibilidade de fazer lançamentos de notas, é possível aos alunos o acompanhamento de seu rendimento em cada atividade. A plataforma também facilita a interação entre o professor e o aluno por intermédio de ferramentas agregadas como fóruns e chats (FRANÇA, 2012).

Entretanto, no que diz respeito ao processo de compilação e avaliação automática de códigos-fonte, o Moodle, por padrão, não oferece nenhum recurso. Porém, é possível, para este propósito, instalar alguns *plugins* como, por exemplo, o *Onlinejudge* para Moodle, apresentado como um dos trabalhos relacionados neste artigo, mas, conforme dito, ele possui suas limitações.

Diante desta perspectiva, e visando diminuir a sobrecarga de tarefas do professor, foi feita uma pesquisa com alguns professores que lecionam disciplinas de programação na Universidade do Estado do Rio Grande do Norte (UERN), Campus Mossoró. Com isto foi possível idealizar a integração entre os Juízes *Online* ao Moodle, alvo deste trabalho. Pelo fato de não existir qualquer ferramenta que faça esta integração, entre os Juízes *Online* existentes e o Moodle, fez-se necessária a concepção de uma nova ferramenta.

2.5 INTEGRAÇÃO ENTRE AMBIENTES

O MOJO é a ferramenta, propriamente dita (encapsulada em um módulo instalado no Moodle), responsável pela integração dos ambientes. Este módulo é o responsável pela

comunicação e interação que irá ocorrer entre o Moodle e os Juízes *Online* envolvidos nas operações. Na arquitetura geral de integração entre os ambientes, apresentada na Figura 1, o Moodle fornece a interface e o conjunto de funcionalidades necessárias à gestão e ao acompanhamento das atividades de programação, e o MOJO fornece a interação com os Juízes *Online*.

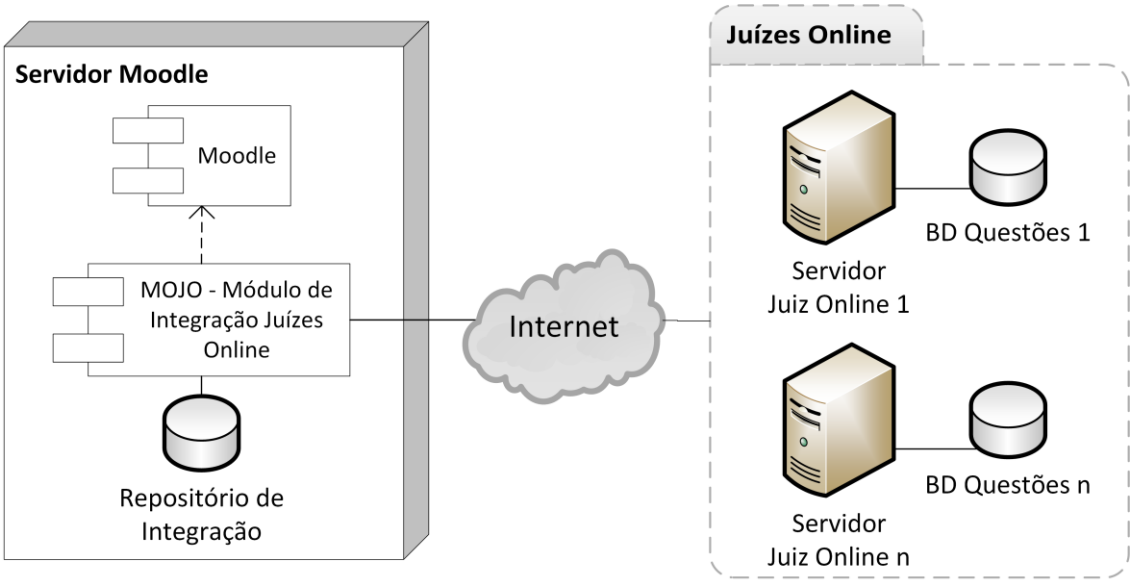

Figura 1 – Arquitetura geral de integração entre os ambientes.

2.6 ARQUITETURA INTERNA DO MOJO

Com a finalidade de garantir a integração entre os dois ambientes envolvidos (Moodle e Juízes *Online*), a arquitetura do MOJO é composta por dois módulos que se comunicam entre si: o Módulo Principal (MOP) e o Módulo de Carga e Atualização (MOCA), além de um Repositório de Integração, todos apresentados na Figura 2 e detalhados a seguir.

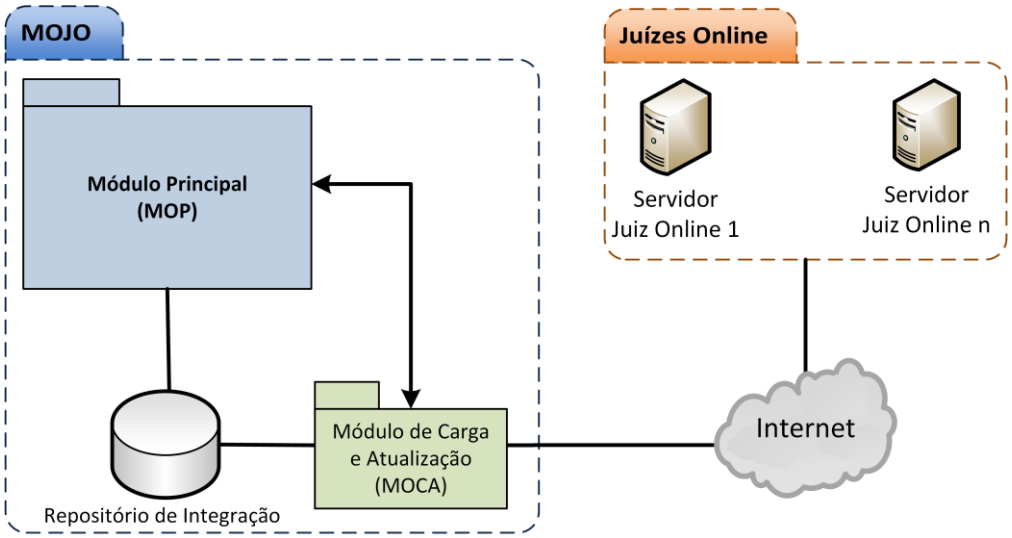

Figura 2 – Arquitetura interna do MOJO.

- **Módulo Principal (MOP)** – é o principal módulo da ferramenta, é ele quem gerencia, controla e fornece todas as funcionalidades necessárias ao bom funcionamento do MOJO, tais como a disponibilização das questões e dos resultados no Moodle. Conforme dito anteriormente, o

MOP interage diretamente com o Módulo de Carga e Atualização, desta forma gerenciando suas ações.

- **Módulo de Carga e Atualização (MOCA)** – é o responsável, mediante delegação do MOP, pela carga de questões no repositório local (Repositório de Integração) da ferramenta e por sua constante atualização. Em um primeiro momento, o MOJO precisa de uma carga inicial em seu repositório. Esta carga é feita pelo MOCA que interage, por meio de requisições *Web*, com o repositório de questões dos Juízes *Online*, obtendo as questões para seu repositório local. Além disso, o MOCA também oferece o monitoramento do repositório de questões dos juízes a fim de descobrir se novas questões foram adicionadas; em caso afirmativo, é realizada a atualização no Repositório de Integração do MOJO, mediante nova requisição *Web*.

- **Repositório de Integração** – consiste de uma base de dados exclusiva criada para o MOJO, pois foi identificada a necessidade de tabelas específicas para suas funcionalidades. Estas tabelas interagem diretamente com as tabelas do Moodle, propiciando a real integração alvo desta pesquisa. É neste repositório onde dados como, por exemplo, as questões dos juízes ficarão armazenadas, assim como os resultados das mesmas.

2.7 PRINCIPAIS FUNCIONALIDADES

Após estudo das funcionalidades necessárias relativas às operações do MOJO e suas limitações (implementações específicas para cada um dos juízes envolvidos) para o contexto deste trabalho, foi feito um levantamento com o seguinte conjunto de operações a serem implementadas:

- **Importação de questões** – funcionalidade diretamente relacionada ao Módulo de Carga e Atualização. É a operação responsável por realizar a importação das questões do repositório de questões de cada um dos Juízes *Online*. Ainda sinaliza quando a importação foi bem sucedida ou não. As informações obtidas na importação de cada questão são: 1 – Código da questão; 2 – Juiz responsável pela questão; 3 – Título da questão; 4 – Descrição da questão e; 5 – Linguagem de programação permitida.

- **Atualização de questões** – outra funcionalidade diretamente relacionada ao Módulo de Carga e Atualização. Esta operação é responsável por verificar se houve a adição de novas questões no repositório de questões dos juízes. Havendo novas questões disponíveis é feita a atualização no repositório do MOJO. A busca por novas questões leva em consideração a quantidade de questões que havia anteriormente no repositório, além de considerar também o código da questão, para que não exista o risco de duplicação de questões. A operação ainda sinaliza se houve atualização, e se foi bem sucedida ou não.

- **Submissão de questões para resolução** – por meio desta operação o professor poderá analisar e selecionar, em uma lista de questões, a questão que melhor se aplica a seus alunos. O professor então submete a questão para resolução pelos alunos. A questão ficará visível para resolução pelos alunos pelo tempo estabelecido pelo professor.

- **Submissão de código-fonte** – por meio desta operação são enviados os dados do código-fonte para o processo de avaliação automática nos Juízes *Online*. Uma vez submetido os dados do código-fonte, o MOJO armazena essas informações em sua base de dados. O processamento

desses códigos é realizado nos servidores dos juízes, e após esse processo, o *feedback* retornado é armazenado na base de dados para posteriormente ser utilizado para visualização. A forma que o processamento do código acontece em cada um dos juízes é de responsabilidade dos mesmos. O MOJO apenas envia o código para avaliação e recebe a resposta (*feedback*) do processo.

- **Visualização de resultados** – Esta operação fornece a opção de recuperar o resultado da avaliação automática realizada pelos Juízes *Online*. O resultado é recuperado utilizando o código da questão, obtendo o *feedback* da base de dados do MOJO e retornando essas informações para visualização. O professor terá a opção de visualizar os resultados por aluno (onde é possível visualizar as questões e resultados de um aluno específico) e por questão (onde é possível visualizar os resultados dos alunos que submeteram código para determinada questão). O professor também pode consultar os códigos-fonte enviados. Por outro lado, o aluno terá apenas a opção de visualizar seus resultados por questão e de consultar o seu código submetido.

2.8 LINGUAGENS DE PROGRAMAÇÃO DISPONÍVEIS

Para esta primeira versão da ferramenta, optou-se por disponibilizar para utilização, no desenvolvimento dos códigos-fonte, 5 (cinco) linguagens de programação, que de acordo com TIOBE (TIOBE SOFTWARE BV, 2014), fazem parte do grupo das 10 linguagens de programação mais populares atualmente (com base em dados de janeiro de 2014). São estas: C, C++, C#, Java e PHP. Para um melhor entendimento, a Figura 3 ilustra o ranking do TIOBE com as linguagens citadas.

Ainda seria possível disponibilizar outras linguagens, mas por questão de praticidade resolveu-se utilizar estas, que normalmente são as mais utilizadas no ensino de disciplinas de programação.

Jan 2014	Jan 2013	Change	Programming Language	Ratings	Change
1	1		C	17.871%	+0.02%
2	2		Java	16.499%	-0.92%
3	3		Objective-C	11.098%	+0.82%
4	4		C++	7.548%	-1.59%
5	5		C#	5.855%	-0.34%
6	6		PHP	4.627%	-0.92%
7	7		(Visual) Basic	2.989%	-1.76%
8	8		Python	2.400%	-1.77%
9	10	^	JavaScript	1.569%	-0.41%
10	22	^^	Transact-SQL	1.559%	+0.98%
11	12	^	Visual Basic .NET	1.558%	+0.52%
12	11	v	Ruby	1.082%	-0.69%
13	9	vv	Perl	0.917%	-1.35%
14	14		Pascal	0.780%	-0.15%
15	17	^	MATLAB	0.776%	+0.14%
16	45	^^	F#	0.720%	+0.53%
17	21	^^	PL/SQL	0.634%	+0.05%
18	35	^^	D	0.627%	+0.33%
19	13	vv	Lisp	0.604%	-0.35%
20	15	vv	Delphi/Object Pascal	0.595%	-0.32%

Figura 3 – Ranking das linguagens de programação mais populares.

2.9 TECNOLOGIAS UTILIZADAS NO DESENVOLVIMENTO DA FERRAMENTA

Devido ao fato de nenhuma API (*Application Programming Interface*) ou serviço *Web* (*Web Service*) ser disponibilizado por qualquer um dos juízes utilizados nesta versão da ferramenta, foi necessário fazer implementações bem específicas para cada juiz individualmente. Esta etapa, em especial, foi a que necessitou de mais tempo e atenção no decorrer do desenvolvimento da ferramenta.

Para estas implementações, e visando facilitar a integração com a plataforma Moodle, foi utilizada a linguagem de programação PHP (THE PHP GROUP, 2014), mesma linguagem utilizada no desenvolvimento da plataforma Moodle.

Em conjunto com a linguagem PHP, também foi utilizada *JavaScript* (W3SCHOOLS, 2014), uma linguagem de programação *client-side*, que possibilita que os *scripts* sejam executados do lado do cliente e interajam com o usuário sem a necessidade de passar pelo servidor.

HOLOS
ISSN 1807 - 1600

Ainda sobre o uso da linguagem PHP, foi utilizada a biblioteca *client URL* ou simplesmente *cURL* (THE PHP GROUP, 2014), uma biblioteca bem específica para casos em que se precisa trabalhar obtendo partes específicas da estrutura HTML (W3Schools, 2014) de uma página *Web*. Esta biblioteca permite a conexão e comunicação com diferentes tipos de servidores com a utilização de diferentes tipos de protocolos.

Em relação à base de dados adotada, optou-se por utilizar como Sistema Gerenciador de Banco de Dados (SGBD) o *PostgreSQL* (THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2014), por sua característica *open source* (*software* livre). E também pela facilidade em sua utilização e suporte, pois o mesmo SGBD já vem sendo utilizado no laboratório onde a pesquisa foi desenvolvida.

2.10 NECESSIDADES DA INTEGRAÇÃO

A característica dos Juízes *Online* utilizados de não fornecerem qualquer API ou funcionalidades via *Web Service* dificultou, mas não impediu o desenvolvimento da ferramenta. Com isso, algumas necessidades relevantes à integração do ambiente foram identificadas, conforme explicado a seguir.

O MOJO é uma ferramenta que se propõe a obter questões, de programação, de terceiros (Juízes *Online*) para resolução pelo aluno. Para a obtenção dessas questões, e demais informações necessárias ao MOJO, fez-se o uso da *cURL*, uma biblioteca bem específica para casos em que se precisam obter informações de uma página HTML.

A partir dos pedaços de informações "brutos" obtidos com a utilização da *cURL*, foi feito o devido tratamento, utilizando funções disponibilizadas pela própria linguagem PHP, até a obtenção da informação "lapidada". Para um melhor entendimento, a Figura 4 exibe um trecho de código utilizando a biblioteca *cURL* para obtenção de informações de uma página.

```
$ch = curl_init();
$timeout = 0;
curl_setopt($ch, CURLOPT_URL, $paginaHTML);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, $timeout);
$url = curl_exec ($ch);
curl_close($ch);
```

Figura 4 – Trecho de código exemplificando a utilização da biblioteca *cURL*.

Conforme a Figura 4, a variável **$paginaHTML** (terceira linha, contando de cima para baixo) recebe o endereço URL da página que se deseja obter a informação. Essa informação é colocada dentro de outra variável a **$url** (segunda linha, contando de baixo para cima). Uma vez em posse dessa informação é possível manipulá-la até a obtenção do trecho de informação necessário.

Percebeu-se, ainda, a necessidade de criação de um conjunto de novas tabelas para possibilitar a integração do MOJO ao Moodle, dando origem ao Repositório de Integração da ferramenta. Estas novas tabelas, que interagem diretamente com as tabelas do Moodle, tem entre as principais finalidades, armazenar as questões dos juízes e os resultados das mesmas, além de armazenar informações individualizadas dos alunos, permitindo o correto direcionamento do *feedback* dado pela ferramenta.

2.11 MOJO EM AÇÃO

Com o Repositório de Integração carregado com as questões dos juízes, o MOJO vai propiciar uma forma de colaboração, ainda que indireta, entre o Moodle e os Juízes *Online*. Neste processo, cada envolvido (professor, estudante e Juiz *Online*) possui papel importante. Para um melhor entendimento, a Figura 5 ilustra o fluxo deste processo com o MOJO.

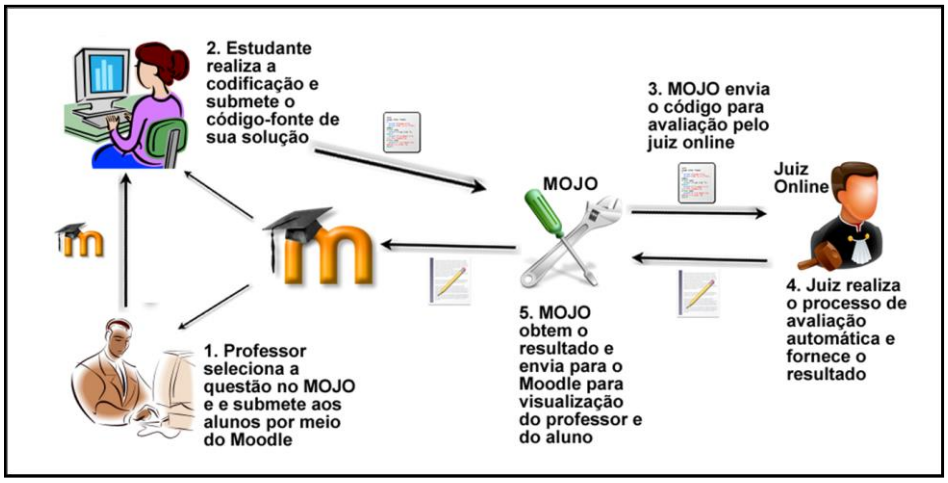

Figura 5 – Fluxo do processo com o MOJO.

Conforme a figura 5, podemos identificar as seguintes etapas no processo:

- **Etapa 1** – o professor, no Moodle, define a questão e a submete para resolução.
- **Etapa 2** – o aluno visualiza, desenvolve e submete uma solução para a questão.
- **Etapa 3** – o MOJO entra em contato com o juiz responsável pela questão e envia o código-fonte para avaliação.
- **Etapa 4** – o Juiz *Online* realiza os devidos processos de avaliação automática para a solução proposta e devolve o resultado.
- **Etapa 5** – o MOJO obtém o resultado da avaliação e o disponibiliza para visualização pelo professor e pelo aluno.

Com a possibilidade de visualizar os resultados, o professor poderá fazer o devido acompanhamento de seus alunos. Vale ressaltar que o professor terá acesso aos códigos submetidos pelos alunos, e as atividades ficam armazenadas para reutilização posteriormente.

3 RESULTADOS

Para coleta dos primeiros resultados, foi realizada uma conversa informal com 7 (sete) professores de programação, onde foi apresentado o MOJO e seus principais objetivos. Por meio dessa conversa, foram obtidos os seguintes resultados mostrados na Tabela 1.

Tabela 1 – Resultados iniciais.

PORCENTAGEM	SIGNIFICADO
71,4%	Professores que concordaram que a ferramenta pode reduzir o tempo gasto no processo de elaboração, submissão e avaliação.
28,6%	Professores que sentiram falta da possibilidade de elaborarem suas próprias atividades e terem as mesmas avaliadas automaticamente.
42,8%	Professores que fizeram observações sobre a legibilidade do código.

28,6%	Professores que fizeram observações sobre a clareza nos resultados retornados pela ferramenta (resposta dos juízes).
85,7%	Professores que concordaram que a ferramenta pode ajudar a fornecer um *feedback* mais rápido ao aluno.
100%	Professores que concordaram que com a diminuição na sobrecarga do professor, será possível um melhor acompanhamento dos alunos.

Pode-se perceber, com base nos resultados obtidos, que o *feedback* dado pela ferramenta deve ser melhorado, assim como deve-se atentar a possíveis problemas na legibilidade do código. Observa-se, ainda, que alguns professores preferem elaborar suas próprias atividades, e que a avaliação automática destas também pode ser bem útil. Estas observações serão tratadas em trabalhos futuros.

Ainda foram realizados alguns testes em laboratório quanto à velocidade na obtenção dos resultados. Nestes testes, o tempo de resposta era de apenas alguns segundos ou até menos (para problemas mais básicos), mas deve-se frisar que foram testes realizados em laboratórios com um número reduzido de usuários.

4 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

O MOJO está em sua fase final de testes no laboratório em que é desenvolvido. Com o fim dos testes laboratoriais, pretende-se, para uma melhor avaliação do desempenho da ferramenta junto a professores e alunos, disponibilizar o MOJO em uma turma regular de programação, com o objetivo de verificar a aplicação prática da ferramenta e possíveis ajustes a serem realizados. Além de disponibilizar uma versão estável da ferramenta na comunidade do Moodle na *Web*, para que o fruto deste trabalho possa alcançar o seu público-alvo (professores e alunos de programação).

Como trabalhos futuros, pretende-se integrar ao MOJO um Laboratório Virtual de Programação, como, por exemplo, o VPL (VPL, 2014). Este laboratório virtual irá auxiliar o aluno na edição e avaliação de seus códigos, além de permitir ao professor elaborar suas próprias atividades e avaliá-las automaticamente. Com este laboratório virtual também se pretende tratar os problemas de legibilidade e do *feedback* retornado na avaliação da atividade, ambos apontados nos resultados deste trabalho. A inclusão de outros Juízes *Online* para ampliar o repositório do MOJO também está prevista.

5 REFERÊNCIAS BIBLIOGRÁFICAS

1. AMARAL E. M. H; ÁVILA, B; ZEDNIK, H; TAROUCO, L. Laboratório Virtual de Aprendizagem: Uma Proposta Taxonômica. RENOTE: Revista Novas Tecnologias na Educação, v. 9 n. 2, 2011.
2. TAKAHASHI, F. Matemática e ciências da computação têm alta taxa de abandono. Folha de São Paulo, São Paulo, 06 abr. 2009. Disponível em: http://www1.folha.uol.com.br/folha/educacao/ult305u546576.shtml. Acesso em: 06 de jan. de 2014.
3. MOTA, M. P; PEREIRA, L. W. K; FAVERO, E. L. JavaTool: Uma Ferramenta Para Ensino de Programação. In: Simpósio Brasileiro de Informática na Educação (SBIE 2009), 20, 2009, Florianópolis.

4. KUMAR S; GANKOTIYA, A. K; DUTTA, K. A Comparative Study of Moodle with other e-Learning Systems. In: International Conference on Eletronics Computer Technology (ICECT 2011), 3, 2011, Kanyakumari, p. 414-418.
5. ZHIGANG, S; XIAOHONG, S; NING, Z; YANYU, C. Moodle Plugins for Highly Efficient Programmin Courses. In: Moodle Research Conference, 1, 2012, Heraklion, p. 157-163.
6. MARTINS, C; GIRAFFA, L. M. M. Capacit@ndo: uma proposta de formação docente utilizando o Moodle. RENOTE: Revista Novas Tecnologias na Educação, v. 6, n. 1, 2008.
7. ACM-ICPC. The ACM-ICPC International Collegiate Programming Contest. Disponível em: http://icpc.baylor.edu/. Acesso em: 07 de jan. de 2014.
8. INC., T. TopCoder. Disponível em: http://www.topcoder.com/. Acesso em: 06 de jan. de 2014.
9. TIMUS ONLINE JUDGE TEAM. Timus Online Judge. Disponível em: http://acm.timus.ru/. Acesso em: 07 de jan. de 2014.
10. UNIVERSIDAD DE VALLADOLID. UVA Online Judge. Disponível em: <http://uva.onlinejudge .org/>. Acesso em: 06 de jan. de 2014.
11. SPHERE RESEARCH LABS. SPOJ Brasil. Disponível em http://br.spoj.pl/. Acesso em: 06 de jan. de 2014.
12. SOUZA, D. M; MALDONADO, J. C; BARBOSA, E. F. Aspectos de Desenvolvimento e Evolução de um Ambiente de Apoio ao Ensino de Programação e Teste de Software. In: Simpósio Brasileiro de Informática na Educação (SBIE 2012), 23, 2012, Rio de Janeiro.
13. SANTOS, J. C. S; RIBEIRO, A. R. L. JOnline: proposta preliminar de um juiz online didático para o ensino de programação. In: Simpósio Brasileiro de Informática na Educação (SBIE 2011), 22, 2011, Aracaju.
14. SIROTHEAU, S; BRITO, S. R; SILVA, A. S; ELIASQUEVICI, M. K; FAVERO, E. L; TAVARES, O. L. Aprendizagem de iniciantes em algoritmos e programação: foco nas competências de autoavaliação. In: Simpósio Brasileiro de Informática na Educação (SBIE 2011), 22, 2011, Aracaju.
15. FRANÇA, A. B. Sistema de apoio a atividades de laboratório de programação com suporte ao balanceamento de carga e controle de plágio. 145 f. Dissertação (Mestrado em Engenharia de Teleinformática) – Programa de Pós-Graduação em Engenharia de Teleinformática, Universidade Federal do Ceará, Fortaleza, 2012.
16. ONLINEJUDGE. Onlinejudge. Disponível em: https://github.com/hit-moodle/onlinejudge/. Acesso em: 07 de jan. de 2014.
17. MOREIRA, M. P; FAVERO, E. L. Um Ambiente Para Ensino de Programação com Feedback Automático de Exercícios. In: Workshop sobre Educação em Computação (WEI 2009), 17, 2009, Belém.
18. CAMPOS, C. P; FERREIRA, C. E. BOCA: Um sistema de apoio para competições de programação. In: Workshop sobre Educação em Computação (WEI 2004), 12, 2004, Salvador.
19. SANDBOX. Sandbox. Disponível em: https://github.com/openjudge/sandbox/. Acesso em: 07de jan. de 2014.
20. SPHERE RESEARCH LABS. IDE ONE. Disponível em http://ideone.com/. Acesso em: 07 de jan. de 2014.

21. YUSOF, N; ZIN, N. A. M; ADNAN, N. S. Java programming assessment tool for assignment module in moodle e-learning system. Procedia - Social and Behavioral Sciences, v. 56, 2012, p. 767-773.
22. TIOBE SOFTWARE BV. TIOBE Index for January 2014. Disponível em: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html. Acesso em: 06 de jan. de 2014.
23. THE PHP GROUP. PHP: Hypertext Preprocessor. Disponível em: http://www.php.net/. Acesso em: 08 de jan. de 2014.
24. W3SCHOOLS. JavaScript Tutorial. Disponível em: http://www.w3schools.com/js/default.asp. Acesso em: 08 de jan. de 2014.
25. THE PHP GROUP. Biblioteca Cliente URL. Disponível em: http://php.net/manual/pt_BR/book.curl.php. Acesso em: 08 de jan. de 2014.
26. W3SCHOOLS. HTML Tutorial. Disponível em: http://www.w3schools.com/html/default.asp. Acesso em: 08 de jan. de 2014.
27. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. PostgreSQL. Disponível em: http://www.postgresql.org/. Acesso em: 08 de jan. de 2014.
28. VPL. Virtual Programming Lab. Disponível em: http://vpl.dis.ulpgc.es. Acesso em: 08 de jan. de 2014.