



**MINISTÉRIO DA EDUCAÇÃO
SECRETÁRIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO SERTÃO
PERNAMBUCANO**

GEIDSON BENÍCIO COELHO DE SOUZA

**SISTEMA DE CORREÇÃO AUTOMÁTICA DE CÓDIGOS-FONTE PARA AUXÍLIO
NA APRENDIZAGEM DE PROGRAMAÇÃO**

Petrolina

2014

GEIDSON BENÍCIO COELHO DE SOUZA

**SISTEMA DE CORREÇÃO AUTOMÁTICA DE CÓDIGOS-FONTE PARA AUXÍLIO
NA APRENDIZAGEM DE PROGRAMAÇÃO**

Trabalho de Conclusão de Curso submetido ao Instituto Federal de Educação, Ciência e Tecnologia Sertão Pernambucano – Campus Petrolina, como requisito parcial para obtenção do grau de Licenciando em Computação.

Orientadora: Jussara Moreira

Petrolina

2014

GEIDSON BENÍCIO COELHO DE SOUZA

**SISTEMA DE CORREÇÃO AUTOMÁTICA DE CÓDIGOS-FONTE PARA AUXÍLIO
NA APRENDIZAGEM DE PROGRAMAÇÃO**

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do grau de Licenciado em Computação pelo Instituto Federal de Educação, Ciência e Tecnologia Sertão Pernambucano, Campus Petrolina.

Aprovado em de de 2014

Banca Examinadora

Dedico este trabalho à Deus, pela sabedoria, paz e por ser o meu guia nesta caminhada e aos meus pais pelo cuidado e apoio em todos os momentos.

AGRADECIMENTOS

A Deus, Senhor da minha existência, pela vida, sabedoria, saúde, paz e pela oportunidade de desenvolver esse trabalho.

Aos meus pais, pela formação, apoio e carinho que me deram.

À minha noiva, pela compreensão e incentivo em todos os momentos desta caminhada.

À orientadora Jussara Moreira pelo apoio e auxílio durante o trabalho e que sempre incentivou no estudo de programação ao longo do curso.

Aos meus amigos, pelo companheirismo e por todos os momentos, sejam bons ou ruins, que passamos ao longo desta caminhada.

Agradeço aos integrantes da banca que se dispuseram a participar e prestigiar o meu trabalho.

RESUMO

As disciplinas de programação exigem um grande comprometimento do professor que vai desde a pesquisa de materiais, aplicação de conteúdos, acompanhamento individualizado na correção de exercícios, trabalhos práticos, dentre outras atividades. Devido a essa sobrecarga este profissional não consegue realizar um acompanhamento eficiente dos alunos, resultando aos estudantes, por vezes, à dispersão, desinteresse e até desistência de cursos que envolvem disciplinas de programação. Pretendendo instrumentalizar o professor para melhor condução de turmas de disciplinas de programação e a grande necessidade de exercícios e prática para a compreensão dos conteúdos aplicados, este trabalho apresenta um sistema *Web* integrado a um juiz *online*, para realizar correções instantâneas e automáticas dos programas desenvolvidos pelos alunos. Todas as operações são realizadas por meio de uma interface desenvolvida e integrada ao programa de correção. Para comprovar a funcionalidade do sistema, foram realizados testes unitários durante o desenvolvimento para validação das funcionalidades, por fim são apresentados os resultados da pesquisa e trabalhos futuros.

Palavras-Chave: ensino de programação, correção automática, informática na educação, computação.

ABSTRACT

The programming disciplines require a great teacher commitment ranging from materials research, application content, personalized support in correcting exercises, practical work and other activities. Due to this overhead this professional cannot realize an efficient monitoring of students, resulting to students, sometimes to the dispersion, disinterest and even cancellation of courses that involve programming disciplines. Intending to better equip the teacher conducting classes in programming disciplines and the great need for exercises and practice to understand the content applied, this paper presents an integrated web an online judge system, to perform instant and automatic corrections programs developed by students. All transactions are conducted through a developed and integrated into the correctional program interface. To prove the functionality of the system, unit tests were conducted during development to validate the functionality finally search results and future work are presented.

Keywords: *educational programming, automatic grading, computing in education, computing.*

LISTA DE ILUSTRAÇÕES

Figura 1 - Exemplo de problema utilizado em maratonas de programação	18
Figura 2 - Interface administrativa do sistema BOCA	20
Figura 3 - Interface principal do sistema JOnline.....	21
Figura 4 - Página inicial do site Project Euler	22
Figura 5 - Aplicativo do Project Euler	22
Figura 6 - Tela inicial do usuário no site URI Online Judge.....	23
Figura 7 - Tela de Problemas no site URI Online Judge	24
Figura 8 - Diagrama de Caso de Uso do Aluno.....	32
Figura 9 - Diagrama de Caso de Uso de Professor e Administrador.....	33
Figura 10 - Diagrama de atividades do processo de correção automática.....	34
Figura 11- Diagrama de Implementação representando a arquitetura física do sistema	35
Figura 12 - Tela de login do administrador	36
Figura 13 - Tela de login do aluno	36
Figura 14 - Tabela de problemas na tela administrativa.....	37
Figura 15 - Formulário de cadastro do problema	38
Figura 16 - Continuação do formulário de cadastro de problemas.....	38
Figura 17 – Visualização das últimas submissões.....	39
Figura 18 - Tela de problemas do usuário aluno	40
Figura 19 - Visualização de um problema.....	40
Figura 20 - Template padrão SBAdmin v2	44
Figura 21 - Representação do Banco de Dados	47
Figura 22 - Relação entre os componentes do sistema.....	48
Figura 23 - Comunicação entre os componentes do sistema.....	49

LISTA DE TABELAS

Tabela 1. Quadro comparativo dos sistemas corretores	24
--	----

SUMARIO

1	INTRODUÇÃO.....	12
2	FUNDAMENTAÇÃO TEÓRICA	17
2.1	Sistemas de Avaliação Automática de Código	17
2.1.1	Processos de Entrada.....	19
2.1.2	Processos de Saída	19
2.1.3	Ferramentas de correção automática existentes	19
2.2	Métodos de Análise	25
2.2.1	Validade do Resultado do Programa.....	25
2.2.2	Métodos de Avaliação da Qualidade do Programa	25
2.2.3	Métodos de Identificação de Plágio	26
2.3	Correção e <i>Feedback</i>	27
2.4	Execução Segura de Código	28
3	PROJETO DO SISTEMA	29
3.1	Visão Geral	29
3.1.1	Plataforma	29
3.1.2	Corretor Automático	30
3.2	Análise de Requisitos.....	30
3.3	Diagramas UML	31
3.3.1	Diagramas de Caso de Uso.....	32
3.3.2	Diagrama de Atividades	33
3.3.3	Diagrama de Implementação.....	35
3.4	Detalhamento do Sistema	35
3.4.1	Autenticação de usuários.....	35
3.4.2	Área Administrativa	36
3.4.3	Área do Aluno	39
4	IMPLEMENTAÇÃO	41
4.1	Linguagem PHP	41
4.1.1	PDO - PHP <i>Data Objects</i>	41
4.2	Servidor <i>Web</i>	42
4.3	Banco de Dados	43
4.4	<i>Front-End</i>	43
4.4.1	<i>SB Admin v2</i>	43

4.4.2	jQuery.....	44
4.4.3	Twitter Bootstrap.....	45
4.5	Execução de Programas: safeexec	46
4.6	Testes	46
5	ORGANIZAÇÃO DO SISTEMA.....	47
5.1	Modelagem do Banco de Dados	47
5.2	Relação entre elementos do sistema	48
6	RESULTADOS	50
6.1	Dificuldades encontradas	51
7	CONCLUSÃO.....	52
7.1	Trabalhos futuros	53
8	REFERÊNCIAS	56
APÊNDICES		

1 INTRODUÇÃO

O aprendizado de programação é essencial na formação de profissionais na área de computação, pois constituem a base do entendimento dos processos associados à construção de ferramentas computacionais. Essa importância é proveniente do modelo de ensino em que a programação é enfatizada nas disciplinas introdutórias nos cursos de graduação da área. Moreira e Favero (2009), afirmam que “além de capacitar o indivíduo a utilizar a lógica de programação na resolução de problemas, a programação é a base para muitos campos em que a computação é utilizada, fator relevante em disciplinas avançadas.”

Uma das etapas no aprendizado de programação é a construção de estruturas algorítmicas, como laços, desvios condicionais e operações matemáticas. Após a apresentação e explanação destas estruturas o aluno relaciona com elas por meio de exercícios passados pelo professor a fim de exercitar o que já foi discutido em sala. O meio mais utilizado para esta prática é a resolução de problemas pelos alunos.

O ambiente de estudos do aluno consiste em ferramentas para experimentar a linguagem de programação e verificar sua execução. Essas ferramentas normalmente contêm um editor de texto e um compilador. O compilador realiza análises sintáticas e semânticas no código fonte do programa, e verifica se o mesmo condiz com a estrutura da linguagem de programação utilizada. A indicação imediata de erros pelo compilador pode ajudar o aluno a gravar a sintaxe da linguagem de programação e se o código escrito condiz com a linguagem de programação escolhida, como a abertura e fechamento de blocos. Porém este método não é capaz de indicar ao aluno se ele está utilizando todas as estruturas exigidas pelo exercício, realizando cálculos corretamente, formatação da saída, etc.

Não há como o aluno saber de imediato se a solução construída está resolvendo o exercício corretamente a não ser que o professor corrija o que foi elaborado pelo aluno. Esta é uma tarefa que demanda muito tempo dos professores ou monitores das disciplinas de programação, principalmente quando estas são compostas por muitos alunos, sendo que cada um constrói uma lógica diferente. “Tradicionalmente, as correções sempre são realizadas manualmente. Para isso faz-se necessário que os alunos apresentem seus arquivos, seja eletronicamente, impresso ou manuscrito.” (KURNIA *et al.*, 2001, p. 300). Siebra *apud* Galasso e Moreira (2014, p. 24) cita que “é difícil para o professor compreender a lógica do

aluno e também é difícil mudar um raciocínio depois de construído e tentar construir outra forma de resolução para os problemas”.

Partindo desse princípio este projeto propõe o desenvolvimento de um sistema que facilite a disponibilidade de problemas pelo professor e o autoaprendizado de programação através do uso de um juiz *online*. O acompanhamento dos alunos se dará de maneira simples, através de um módulo integrado responsável em fornecer a interface e as funcionalidades necessárias para a gestão e acompanhamento dos problemas de programação.

Juízes *online* realizam o processo automático de correção de código-fonte. Eles recebem, compilam e executam os códigos-fonte enviados pelos usuários. “Durante a execução do programa, os juízes *online* utilizam dados formatados como a entrada do programa e, após o processamento, é feita a comparação dos resultados obtidos com os esperados”. (SANTOS; RIBEIRO, 2011, p.333 (a)).

Os juízes *online* já são amplamente utilizados para o apoio de competições de programação, onde são disponibilizados vários problemas a serem resolvidos e submetidos para correção. Os usuários também podem além de enviar soluções dos problemas para serem avaliadas, escolher a linguagem a ser utilizada na escrita do código, (SANTOS; RIBEIRO, 2011 (b)). Além disso, são disponibilizados fóruns para discussão, estatísticas dos problemas e ranking de usuários. Esses corretores automáticos podem ser encontrados na internet, a exemplo do SPOJ Brasil (<http://br.spoj.com/>), UVA (REVILLA *et. al*, 2008) e BOCA *Online Constest Administrator* (CAMPOS; FERREIRA, 2004).

Portanto, o uso de um corretor automático em disciplinas de programação poderá viabilizar o aumento da quantidade de exercícios oferecidos e fornecer ao aluno um *feedback*¹ rápido, potencializando a sua autoaprendizagem. O módulo de gerenciamento dará ao professor uma maneira inovadora de acompanhar a evolução do aprendizado do aluno e fornecerá um repositório de problemas que poderá ser utilizado nos laboratórios de programação.

A disciplina de Programação Estruturada, disponível no segundo semestre do curso de Licenciatura em Computação do Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano, Campus Petrolina, é o primeiro contato do aluno com a linguagem C e estruturação de algoritmos para resolução de problemas complexos. A linguagem C também é

¹ Dar resposta a um determinado evento.

utilizada na disciplina de Estrutura de Dados e a linguagem Java em Programação Orientada a Objetos.

Não é novidade que estas disciplinas apresentam altos índices de evasão e reprovação, sendo este um problema encontrado em várias universidades em todo o mundo, sendo os índices de reprovação acima de 55%, segundo (CAMPOS, 2010). Outras pesquisas afirmam que a lógica de programação é uma das disciplinas com maiores índices de reprovação em IES (CAMPOS, 2009) e (SILVA *et. al* 2009). Por esse motivo “este, é considerado um dos sete grandes desafios do ensino de computação”. (MCGETTRICK *apud* WESLEY; AMBROSIO, 2008)

Desta forma é grande a necessidade de exercícios e prática para a compreensão dos conteúdos, tendo como principais dificuldades encontradas no processo de aprendizagem:

- Os alunos são em sua maioria iniciantes em linguagem de programação avançada (diferentemente de outras disciplinas do início do curso de Licenciatura em Computação, a linguagem de programação não tem nenhuma base no ensino médio).
- O nível de abstração de linguagens de programação de médio e alto nível é muito significativo. “O nível de abstração está relacionado com a proximidade da linguagem natural dos computadores, quanto mais próxima, mais difícil de ser entendida”. (SCHILDT, 2006).
- A adaptação e o processo de aprendizagem são variados. Alguns alunos possuem maior facilidade para desenvolver a lógica computacional em relação aos demais. Este fato gera diferentes níveis de acompanhamento da turma;
- Necessidade de exercícios variados. O treinamento ajuda a desenvolver o processo de adaptação e fixar novas informações.
- A demora na correção causa a quebra da reflexão lógica, pois depois de algum tempo a sequência lógica é esquecida, tornando necessária a revisão de todos os passos desenvolvidos.
- Turmas com muitos alunos traz uma carga excessiva para o professor, prejudicando o acompanhamento individualizado.

Esses e outros problemas estão relacionados ao alto índice de reprovação das disciplinas de programação. Assim também estes problemas são difíceis de serem tratados, de forma que não se pode afirmar a existência de uma solução que resolva em sua totalidade.

Pensando em contribuir e amenizar os problemas mostrados, este trabalho traz como objetivo geral propor a implementação um corretor automático de códigos fonte para utilização em laboratórios de programação visando auxiliar o ensino e aprendizado de programação.

Para atingir o objetivo proposto neste trabalho foi:

- Levantado bibliografias dos trabalhos correlatos;
- Analisado os principais corretores automáticos de código;
- Identificado requisitos e especificado características do sistema quanto a sua estrutura e desenvolvimento;
- Projetado uma arquitetura para compilação de programas escritos nas linguagens C e C++, facilitando a implementação de outras linguagens no futuro.
- Desenvolvido parte do sistema de correção, utilizando padrões de desenvolvimento *web*.
- Avaliado os resultados obtidos e apontando pontos fortes e melhorias possíveis.

Este documento está estruturado em 7 capítulos. Este capítulo é uma introdução onde é apresentada uma visão geral sobre o trabalho e sua motivação. Posteriormente são apresentados o tema proposto e a justificativa bem como, os objetivos geral e específicos e o método a ser empregados para satisfazer os objetivos expostos nesse trabalho.

No capítulo 2 é apresentado a Fundamentação Teórica, um estudo sobre os sistemas de correção automatizada de códigos, principais ferramentas de correção, métodos de análise de códigos e soluções para o problema apresentado.

O Capítulo 3 apresenta uma visão geral do sistema, bem como seus requisitos, diagramas UML e um maior detalhamento do sistema.

O Capítulo 4 descreve tecnologias utilizadas no desenvolvimento e implementação do sistema.

O Capítulo 5 apresenta como o sistema está organizado, modelagem de banco de dados e relação entre seus componentes principais.

No Capítulo 6 são apresentados os resultados e as dificuldades encontradas na execução projeto.

O último capítulo se encontra a conclusão e as propostas de trabalhos futuros.

O texto contém também um apêndice que complementa as informações apresentadas no trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Sistemas de Avaliação Automática de Código

O índice elevado de reprovação, desistência e métodos de avaliação nas disciplinas de programação tem sido constantemente foco de pesquisa em todas as partes do mundo (DELGADO *et al.* 2004);(KOLIVER *et al.* 2004).

Para (RODRIGUES JR, 2004), as pesquisas mostram que a melhoria no processo de ensino e qualidade oferecida para os alunos, passa-se na alteração tanto na parte didática como na metodologia apresentada.

Essas mudanças envolvem a utilização de ferramentas que colaborem no processo de aprendizagem do aluno, como também no trabalho do professor, procurando contribuir minimizando dificuldades envolvidas nestes processos. Neste cenário de estudo e análise aparecem as ferramentas de correção automática de códigos.

Os corretores automáticos têm como objetivo julgar a consistência de algoritmos através de vários casos de teste. Esses sistemas recebem o código-fonte enviado pelo usuário e posteriormente compilam e executam esse código. “Durante a execução do programa, estes utilizam dados formatados como a entrada do programa, então processam esses dados e realizam a comparação dos resultados obtidos com os resultados esperados”. (CHAVES *et. al.*, 2013).

Na maioria dos casos, “[...] na avaliação automática de problemas de programação são levadas em consideração duas métricas: 1 - O resultado do programa; 2 - A complexidade da solução” (MOREIRA; FAVERO, 2009). Além desses pontos, outras ferramentas implementam outras funcionalidades como execução segura de código e facilidade na formulação dos problemas.

As principais competições de programação utilizam corretores automáticos para automatizar o processo de correção dos códigos submetidos. A principal competição de programação é realizada anualmente pelo *International Collegiate Programming Contest* (ACM)², sendo as fases brasileiras realizadas pela Sociedade Brasileira de Computação

² <http://icpc.baylor.edu/>

(SBC)³, evento que reúne participantes de várias universidades do Brasil. Outras competições de programação são realizadas internamente por universidade com objetivo de incentivar e treinar as equipes para as competições oficiais. Exemplo disso foram as competições realizadas em 2011 e 2013 no IF SERTÃO - PE. Essas competições têm como regra manter o mesmo padrão de apresentação dos problemas utilizado nas competições oficiais. O exemplo a seguir retirado da seletiva realizado na Nova Zelândia em 2010 e utilizado na 2ª Maratona de Programação do IF SERTÃO – PE em 2013 serve para demonstrar como é estruturado um problema, suas entradas padrão, restrições e saídas padrão.

Figura 1 - Exemplo de problema utilizado em maratonas de programação

Problema J: Juz by letters

Nome do arquivo fonte: letras.[c|java]
Fonte original: Selective New Zealand

An exercise done in elementary school is to supplement missing letters in a sentence. With this the student learns the letters that form a word and read the sentence with more attention. Thus, teachers are requesting an automatic generator of these incomplete sentences, which is always missing two letters.

Will be provided two letters you want to delete a particular phrase. In place of these letters, your program should print the "_" (*underscore*).

Input

The input consists of several test cases, ending with a line containing only # #.

Each test case starts with two lowercase letters in the same line, separated by a space. On the next line we have an integer n ($1 \leq n \leq 100$), which indicates the number of lines that will be built on the phrases that will be replaced. In these n rows each containing a maximum number of 1 to 255 characters.

Output

The output of each case have the case number (see examples below), and sentences with letters replaced. Although the lyrics are provided lowercase, replace them also when they are uppercase.

Leave a blank line after each test case.

Exemplo de entrada	Exemplo de saída
<pre>a e 2 Atarefado foi como estive Esse exemplo eh claro o h 3 Hoje talvez chova Que bom que hoje faz frio Ontem fez calor # #</pre>	<pre>Caso 1 _t_r_f_do foi como _stiv_ _ss_ _x_mplo _h cl_ro Caso 2 __je talvez c__va Que b_m que __je faz fri_ _ntem fez cal_r</pre>

Fonte: (MARATONA, 2013⁴)

³ Sociedade Brasileira de Computação - <http://maratona.ime.usp.br/>

⁴ MARATONA DE PROGRAMAÇÃO. Disponível em: <<http://maratonaisertao.orgfree.com/2013/index.php>>

A Figura 1 demonstra o modelo padrão de problema que é utilizado nas maratonas de programação. Esse modelo também é utilizado pelos corretores analisados. O modelo é composto por um título para o problema e orientações sobre o nome do arquivo fonte a ser submetido. Seguindo traz no corpo a descrição do problema e regras para as entradas e saídas do problema. Por fim traz exemplos de entrada e saída que podem ser utilizados como teste na elaboração dos algoritmos. Este modelo é bem estruturado e possui todas as informações importantes para análise e resolução de um problema. Assim este é o modelo a ser seguido e implementado no sistema.

2.1.1 Processos de Entrada

O processo de entrada define quais valores as variáveis do problema podem assumir. A elaboração do arquivo de entrada deve ser muito cuidadosa, pois este deve garantir a consistência da solução e não pode conter divergências. No exemplo anterior há a limitação de 100 linhas para cada caso de teste e cada linha contendo até 255 caracteres. O fim dos testes é determinado pelos símbolos # #. Este padrão de entradas é utilizado por todos os juízes *online* analisados.

2.1.2 Processos de Saída

A saída do programa é o que de fato permite a correção do problema. As saídas geradas na compilação do programa são comparadas as saídas do problema a ser resolvido. O processo de saída possui limitações e deve seguir a padronização e regras de formatação determinadas no corpo do problema.

2.1.3 Ferramentas de correção automática existentes

2.1.3.1 BOCA

O BOCA (BOCA *Online Contest Administrator*) é um sistema de código aberto desenvolvido na USP⁵ por Cassio Polpo de Campos, para apoiar as competições de programação, dentre elas a Maratona de Programação da Sociedade Brasileira de

⁵ Universidade de São Paulo.

Computação. Desenvolvido em PHP⁶ e Postgresql⁷, o sistema é utilizado também no apoio a disciplinas onde são realizadas submissão e correção de trabalhos durante as aulas. (CAMPOS, 2004).

Figura 2 - Interface administrativa do sistema BOCA

BOCA Username: Administrator (site=1) 4h 43 min(s) left

Runs Tasks Score Site Clarifications Contest Users Logs Problems Reports Languages Backups Answers Options Export Logout

Use the following fields to judge the run:

Site:	1
Number:	4
Time:	11
Problem:	problembebHcS_d.zip
Language:	Python
Team's code:	YES.py view
Answer:	YES [judge=vinicius (1)]
Answer 1:	Not answered yet
Answer 2:	Not answered yet

Judge Open run for rejudging Cancel editing Delete Clear

Autojudging: Renew

Autojudging answer:	(YES)Files match exactly
Autojudged by:	local from 03:14 to 03:14
Standard output:	stdout view
Standard error:	stderr view

Fonte: (GITHUB, 2014)

A Figura 2 mostra a interface utilizada pelo administrador para correções dos problemas enviados durante as competições de programação. Nela é possível visualizar o evento, id do problema, código do time que enviou, arquivo com código-fonte entre outras opções. É uma interface simples que apresenta diversas funcionalidades.

2.1.3.2 JONLINE

O JOnline é uma proposta de juiz *online* didático para o ensino de programação, desenvolvido por Joanna C.S. Santos da UFS⁸. O sistema visa estender as funcionalidades dos juízes *online* adicionando características educativas que auxiliem os discentes no aprendizado de programação, tendo como principais características programação colaborativa e implementação de fluxo de vídeo (SANTOS; RIBEIRO, 2011).

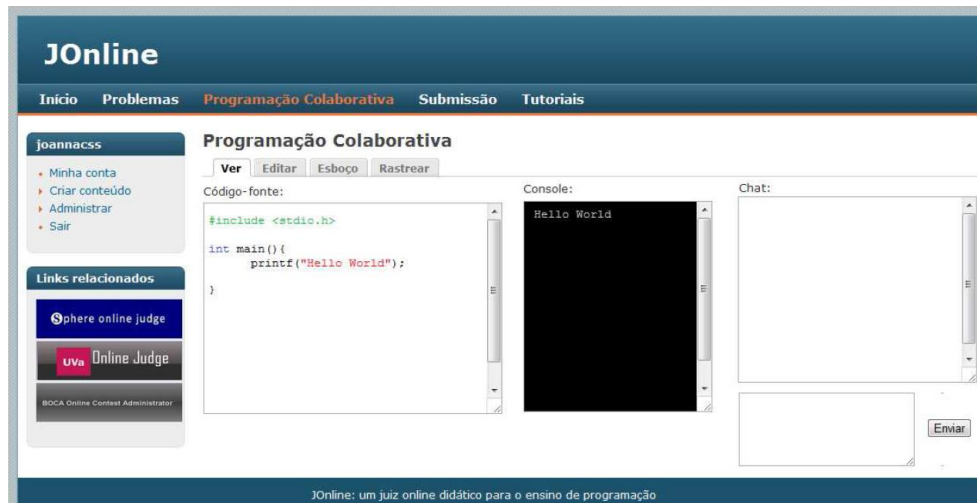
A Figura 3 apresenta a interface do sistema JOnline, sendo esta agradável e amigável. A imagem mostra a área de programação colaborativa, onde é possível enxergar a área reservada para escrita de código-fonte, um console e chat.

⁶ Linguagem de programação criada em 1994 por Rasmus Lerdorf. (DALL'OGGIO, 2012). <http://php.net>

⁷ É um sistema de banco de dados objeto relacional desenvolvido como projeto de código aberto. <http://www.postgresql.org/>

⁸ Universidade Federal do Sergipe. <http://www.ufs.br/>

Figura 3 - Interface principal do sistema JOnline



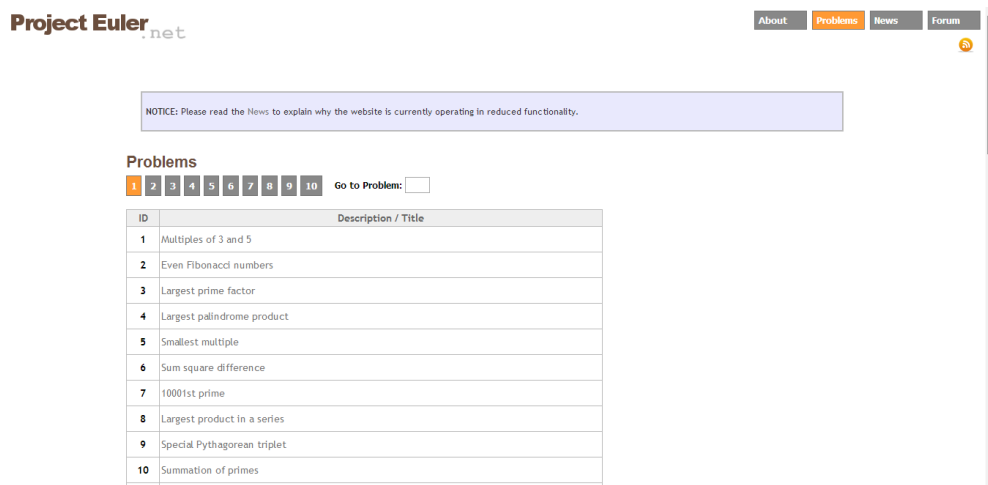
Fonte: (SANTOS; RIBEIRO, 2011).

2.1.3.3 PROJETO EULER

É um site dedicado a uma série de problemas computacionais a serem resolvidos por programas de computador. Desde a sua criação em 2001 por Colin Hughes, o projeto ganhou notoriedade por todo o mundo. Inclui mais de 450 problemas, dos mais diferentes graus de dificuldade. Os participantes podem acompanhar seu progresso através de 16 níveis de desempenho baseado no número de problemas resolvidos. Em 2014 o site ultrapassou o número de 360 mil usuários que resolveram pelo menos um problema. Possui versão *Web* no idioma português e um aplicativo para Android⁹.

⁹ Sistema operacional do Google para dispositivos móveis baseado no Linux.

Figura 4 - Página inicial do site Project Euler

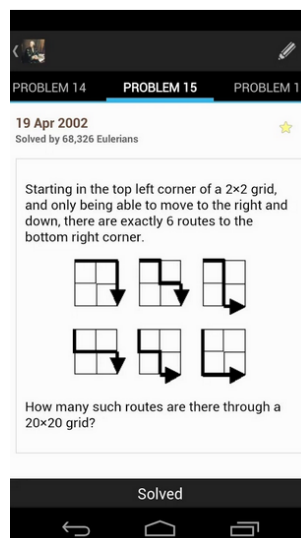


Fonte: (PROJETO EULLER, 2014)

A interface do site oficial do Projeto Euler é simples e bastante eficiente, como visto na Figura 4. Os problemas são listados de acordo com a quantidade de soluções corretas recebidas. Esse tipo de abordagem traz uma melhor interação do usuário com o serviço, tendo este facilitado às questões mais fáceis.

A interface do aplicativo desenvolvido pelo Projeto Euler disponível para plataforma Android, pode ser observada na Figura 5. A interface é básica e traz as principais funcionalidades disponíveis no site oficial.

Figura 5 - Aplicativo do Project Euler

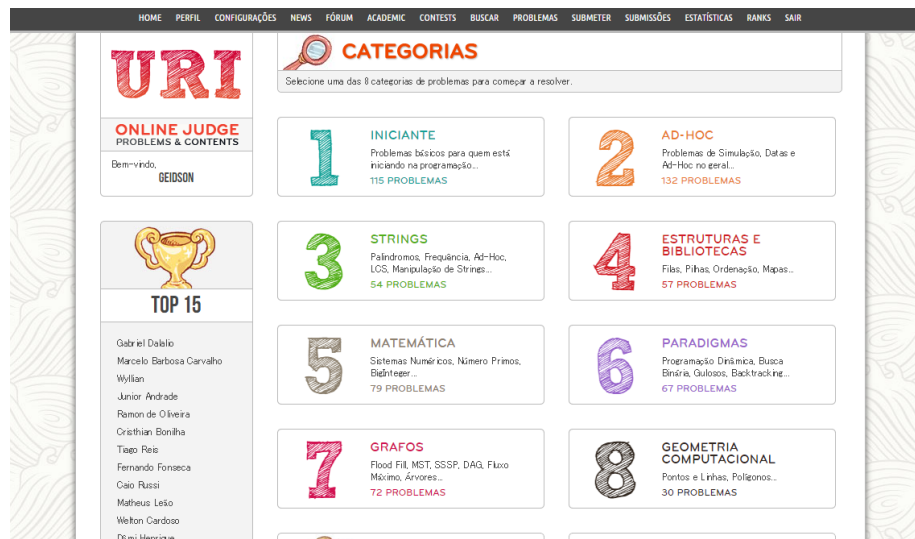


Fonte: (GOOGLE PLAY, 2014)

2.1.3.4 URI ONLINE CONTEST

O URI *Online Judge* é um portal desenvolvido pela coordenação de Computação da Universidade Regional Integrada Erechim em conjunto com o Departamento de Ciência da Computação do Instituto de Matemática e Estatística da USP, com o objetivo de estimular a prática de programação. O portal agrega os recursos dos melhores portais de programação do mundo e, além disso, oferece recursos não encontrados em nenhum outro. Possui interface amigável, disponibiliza fóruns de discussão, ranking de classificação, além de tutoriais e materiais extras (BEZ; TONIN, 2014). O Portal já recebeu mais de 500 mil arquivos de problemas e atualmente conta com cerca de 15 mil usuários, professores de mais de 300 universidades do mundo.

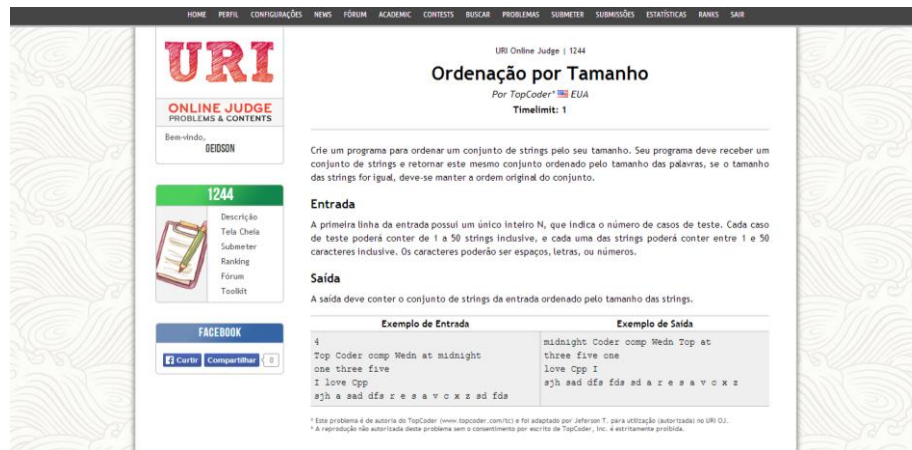
Figura 6 - Tela inicial do usuário no site URI Online Judge



Fonte: (URI, 2014).

Na Figura 6 é possível visualizar que o URI organiza os problemas por categoria. Na Figura 7 observa-se um exemplo de um problema de ordenação. Como pode ser visto o problema segue a estrutura apresentada na Figura 1.

Figura 7 - Tela de Problemas no site URI Online Judge



Fonte: (URI, 2014).

Na Tabela 1, é feito comparativo entre os sistemas corretores existentes no que se refere à interface, interação, acesso e funcionalidades.

Tabela 1. Quadro comparativo dos sistemas corretores

Funcionalidades	BOCA	JOnline	Project Euler	URI
Interface Amigável	-	X	X	X
Interação remota	X	X	X	X
Acesso via Web	-	-	X	X
Vários tipos de questões	-	-	X	X
Questões apresentam índices de dificuldades	-	-	-	X
Relatório de desempenho do aluno	-	X	-	X
Acesso às correções dos alunos	X	X	-	-

Fonte: Elaborada pelo autor

Os juízes *online* estudados apresentam características distintas, sendo que uns apresentam mais recursos que outros ou são utilizados para fins diferentes. O BOCA é um ambiente voltado para competições de programação, desta maneira apresenta poucas funcionalidades didáticas. O JOnline explora funcionalidades didáticas aliadas à correção

automática de códigos, mas é um sistema que ainda não foi disponibilizado na em uma plataforma *Web*. Desta forma fica difícil a sua avaliação. O Projeto Euler e o URI *Online Contest* são duas ferramentas completas, que exploram as principais funcionalidades dos corretores automáticos existentes, sendo que o segundo apresenta recursos inovadores voltados para a necessidade do aluno. Os dois sistemas são disponibilizados na internet, de forma que a falta de conexão torna o serviço indisponível.

É válido lembrar que a conexão à internet disponibilizada no IF Sertão – PE sempre apresentou problemas, e este foi um fator que dispensou o uso dos juízes analisados, por ser a *internet* um item essencial para o uso destes serviços. Neste sentido, foi proposto então o desenvolvimento de um corretor que apresente as principais funcionalidades dos sistemas estudados e que fique disponível tanto na *internet* como também na rede interna do instituto, apoiando as atividades de laboratório nas disciplinas de programação. Assim também, será possível em pouco tempo ter um grande repositório de questões disponível para ser utilizado nas aulas de programação.

2.2 Métodos de Análise

Os métodos de análise de programação podem ser divididos em métodos de validação do resultado, validação da qualidade, identificação de plágio, execução segura do código e facilitação na criação de questões.

Vale ressaltar que esta funcionalidade é proposta de melhorias futuras para o sistema, conforme o capítulo de Trabalhos Futuros.

2.2.1 Validade do Resultado do Programa

Considera-se um dos métodos mais básicos. O método consiste em comparar se a saída esperada é igual à saída gerada pelo compilador. Essa abordagem é a primeira medida de correção possível.

2.2.2 Métodos de Avaliação da Qualidade do Programa

(MOREIRA e FAVERO, 2009), recomenda que o aluno seja avaliado também com base na qualidade da solução.

Este método “mede a qualidade do algoritmo através da análise de estrutura do código”, (SAIKKONEN *et al.*, *apud* PONTES; MIRANDA, 2013) ou utilizando métricas de engenharia de software, avaliando a complexidade de um código-fonte.

Segundo (ALMEIDA, 2005), algumas dessas métricas são:

- Número de linhas do programa;
- Quantidade de uso de funções;
- Número de palavras reservadas;
- Número de declarações; e
- Complexidade de McCabe¹⁰;

Juízes *Online* utilizados em competições de programação implementam essa funcionalidade. O código submetido passa por uma série de testes para que este seja compilado em um tempo de execução pré-determinado pelo problema.

Naudé *apud* Pelz (2007) destaca que o valor pedagógico em medir eficiência do tempo de execução é incerto. Na maioria das correções é improvável que isso seja relevante, particularmente no ensino introdutório de programação.

2.2.3 Métodos de Identificação de Plágio

A identificação de plágio é uma preocupação não raramente encontrado em laboratórios de programação, onde ocorre cópia total ou parcial de soluções entre colegas, frequentemente com mudança de nomes de variáveis ou inserção de comentários, para dificultar a identificação.

A detecção é feita através da comparação da estrutura de dois programas. Inicialmente, o programa analisa cada trabalho e armazena informações sobre certos tipos de declarações. “As declarações mais relevantes para estrutura são marcadas com um caractere único. Cada trabalho é então representado por uma *string* de caracteres” (MUSSINI, 2008, p.7).

¹⁰ Conhecida como Complexidade ciclomática ou complexidade condicional, é uma métrica de software usada para indicar a complexidade de um programa de computador. http://pt.wikipedia.org/wiki/Complexidade_ciclomática.

Embora a detecção desse tipo de conduta seja um aspecto importante em uma turma de programação, para França e Soares (2013), a experiência trazida com a utilização de ferramentas de análise de similaridade revelou que os resultados da comparação entre códigos podem indicar outros aspetos que são de natureza irreprensível. Dependendo do contexto, esta semelhança pode indicar trabalho colaborativo, referências encontradas em livros ou exemplos fornecidos pelo professor.

2.3 Correção e *Feedback*

Quando executado o programa pode apresentar vários tipos de respostas, com intuito de proporcionar um adequado *feedback* da autocorreção, sendo os principais:

- **Correto** – Quando a submissão está totalmente correta. Neste caso o programa retornou a saída esperada para o problema em um tempo de execução determinado.
- **Erro de Compilação** - Esse tipo de erro ocorre quando alguma sintaxe é declarada de forma incorreta, quando isso acontece o arquivo executável não é gerado, impossibilitando a correção. É incomum acontecer esse tipo de erro, pois o aluno executa seu teste antes da submissão.
- **Erro de Saída** – Acontece quando o programa submetido não gera as mesmas saídas determinadas no problema.
- **Erro de Execução** – Quando o programa não executou ou teve um término inesperado. Acesso ilegal à memória ou acesso ao índice inexistente de um vetor são exemplos que causam erros de execução.
- **Tempo Limite Excedido** – Quando o programa não executa em um tempo determinado pelo problema. Existem duas possibilidades de acontecer esse tipo de erro, a primeira quando uma estrutura de repetição apresenta loop infinito. Outra maneira é quando um determinado código utiliza uma grande sequência de passos para solucionar um problema, tornando-se um algoritmo lento.

Com essa classificação dos possíveis resultados da correção professores e alunos conseguem entender melhor em que ponto o programa não está executando, oferecendo compreensão necessária para a devida correção do código.

2.4 Execução Segura de Código

Em ambientes computacionais é preciso tomar cuidado com o que o usuário pode fazer em um sistema. Em corretores automáticos de código, deve-se levar em consideração que os códigos mal intencionados podem ser enviados por alunos para benefício próprio ou apenas para desestabilizar o serviço.

Desta forma o processo de correção precisa acontecer em um ambiente controlado, conhecido como sandbox¹¹. Assim, os programas só podem realizar operações que estejam dentro do controle desse ambiente.

Em (SAIKKONEN *apud* PONTES; MIRANDA, 2013), foi implementado uma sandbox que executa os códigos dos alunos, na linguagem Scheme¹², em um ambiente interpretador Scheme metacircular especialmente desenvolvido para esse propósito. São funcionalidades desse interpretador:

- Desabilitar acesso de leitura e escrita em arquivo;
- Verificar existência de loops infinitos no código;
- Medir se o algoritmo funciona em tempo linear.

Já em (CAMPOS; FERREIRA, 2004), é empregada outra abordagem em sistemas como o BOCA, em que todos os programas do aluno são executado utilizando um programa chamado SafeExec.

O SafeExec, é desenvolvido em C, executa código em um ambiente seguro como um usuário desprivilegiado e com diversos tipos de controle, tais como:

- Número máximo de *threads*¹³ que podem ser abertas pelo programa;
- O tamanho máximo de memória que pode ser utilizada;
- O tempo máximo de execução, entre outros;

¹¹ Sistema de máquina virtual focado na segurança. Todos os registros e danos causados na execução dentro dele são imediatamente apagados ao fim da execução.

¹² Linguagem de programação multiparadigma, característica pela sua linguagem simples e com poucas regras.

¹³ É uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrentemente.

3 PROJETO DO SISTEMA

Este capítulo traz de forma geral a descrição do sistema. Assim, é apresentado as suas funcionalidades. Para isso foi feito estudo dos requisitos, utilizando a linguagem UML¹⁴ para descrever diagramas de casos de uso, diagramas de atividades e detalhamento das telas do usuário.

3.1 Visão Geral

O sistema possui duas interfaces de usuário, uma para professor e administrador e outra para o aluno. Ambas serão descritas em detalhes posteriormente.

De forma geral as interfaces iniciais do administrador e professor são parecidas, apresentando um *dashboard*¹⁵ com informações relevantes do uso do sistema, uma área pra consulta e cadastro de problemas ou área de conhecimento, área para consulta de submissões e outra área para acesso às estatísticas do sistema. O administrador possui total acesso, podendo assim incluir e excluir usuários, esta função esta não é permitida para professores. Na área de cadastro de problemas, o professor ou administrador deve escolher uma área de conhecimento, permitindo assim a classificação das questões. Em seguida são informados os dados, os arquivos de entrada e saída e o nível da questão do problema adicionado. Os problemas cadastrados podem ser consultados, podendo ser filtrados a partir de critérios estabelecidos.

A interface do aluno possui também um *dashboard* com estatísticas da utilização do sistema. É possível também visualizar os problemas cadastrados classificados por área do conhecimento e nível de dificuldade. Em outra área é possível consultar as submissões realizadas. Para responder um problema o aluno envia o arquivo-fonte corresponde a questão, recebendo assim um *feedback* do resultado da correção automática realizada pelo corretor.

3.1.1 Plataforma

Pensando em facilitar o uso do sistema em plataformas diferentes, como também possibilitar o acesso ao sistema por máquinas de baixo desempenho, adotou-se desenvolver

¹⁴ *Unified Modelling Language* - linguagem ou notação de diagramas para especificar, visualizar e documentar modelos de *software* orientadas à objetos.

¹⁵ Tela, composta de uma ou mais camadas, sob a forma de um painel.

para plataforma *Web*. Assim é possível viabilizar o uso da ferramenta em laboratórios de computação que não possuem muitos recursos.

3.1.2 Corretor Automático

O corretor automático toma como base a abordagem utilizada pelo sistema BOCA. Essa abordagem funciona como um teste de caixa preta, onde o professor ao criar o problema fornece as entradas e saídas, ambas para serem testadas e comparadas. Durante a correção, o sistema compila o código do aluno e em seguida compara a saída gerada pela compilação com a saída cadastrada pelo professor.

3.2 Análise de Requisitos

A análise de requisitos procura obter os requisitos do sistema como um todo estabelecendo um conjunto de objetivos gerais que o sistema deve cumprir.

Os requisitos identificados podem ser divididos em dois tipos: Funcionais e Não Funcionais.

Os requisitos funcionais descrevem o que o sistema deve fazer, isto é, as funções necessárias para atender aos objetivos do sistema. Partindo de estudo observatório definem-se os requisitos funcionais do corretor automático sendo:

- Cadastrar e manter usuários;
- Criar e manter questões e submissões;
- Enviar e realizar a correção de problemas automaticamente;
- Apresentar *feedback* para os alunos e professores;
- Armazenar códigos-fonte, sendo este com identificar único.
- Gerar estatísticas do uso do sistema;

Os requisitos não funcionais dizem respeito às características que sistema deve possuir e que estão relacionadas às suas funcionalidades, como performance, portabilidade,

segurança, usabilidade, entre outros. Partindo do princípio que o sistema irá funcionar em plataforma web, são definidos como requisitos não funcionais:

- Disponibilidade 24 horas por dia, 7 dias por semana;
- Acesso via navegador e multiplataforma;
- Controle de acesso dos usuários;
- Interface agradável;
- Utilização de ferramentas livres no desenvolvimento e implantação;

Procurando atender esses requisitos foram escolhidas algumas tecnologias:

- Implantação do sistema em um servidor, permitindo a disponibilidade diária do sistema.
- Uso da linguagem dinâmica PHP de fácil acessibilidade pelos principais sistemas operacionais.
- Uso de autenticação, registrando em banco de dados as informações dos usuários, permitindo assim que cada um tenha acesso a sua área.

3.3 Diagramas UML

O UML (*Unified Modelling Language*) surgiu em 1997 e é uma linguagem ou notação de diagramas para especificar, visualizar e documentar modelos de *software* orientadas a objetos. O UML, hoje mantida pela OMG¹⁶ (*Object Management Group*) apresenta um conjunto muito significativo de diagramas que representam as diferentes partes ou ponto de vista de um sistema, sendo que estes agregam uma fácil visualização e possuem uma grande aceitação no mercado. Por este motivo esta linguagem foi escolhida para descrever a ferramenta proposta neste trabalho. (UML, 2014).

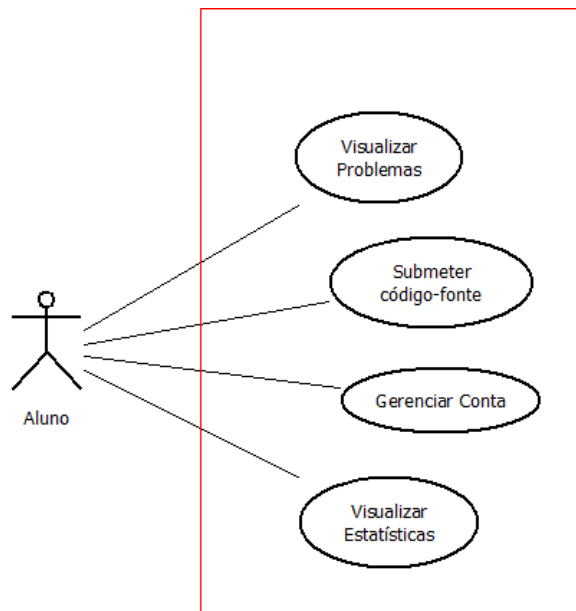
¹⁶ <http://www.omg.org>

3.3.1 Diagramas de Caso de Uso

O diagrama de casos de uso descreve um cenário que mostra as funcionalidades de um sistema a partir do ponto de vista do usuário. Este diagrama apresenta atores (tipos de usuários), casos de uso e os relacionamentos entre esses elementos. O sistema apresenta três tipos de atores.

A Figura 8 representa os casos de uso para o ator aluno: visualizar problemas, submeter código-fonte, gerenciar conta e visualizar estatísticas. Traz uma visão diferente para o entendimento dos requisitos deste ator.

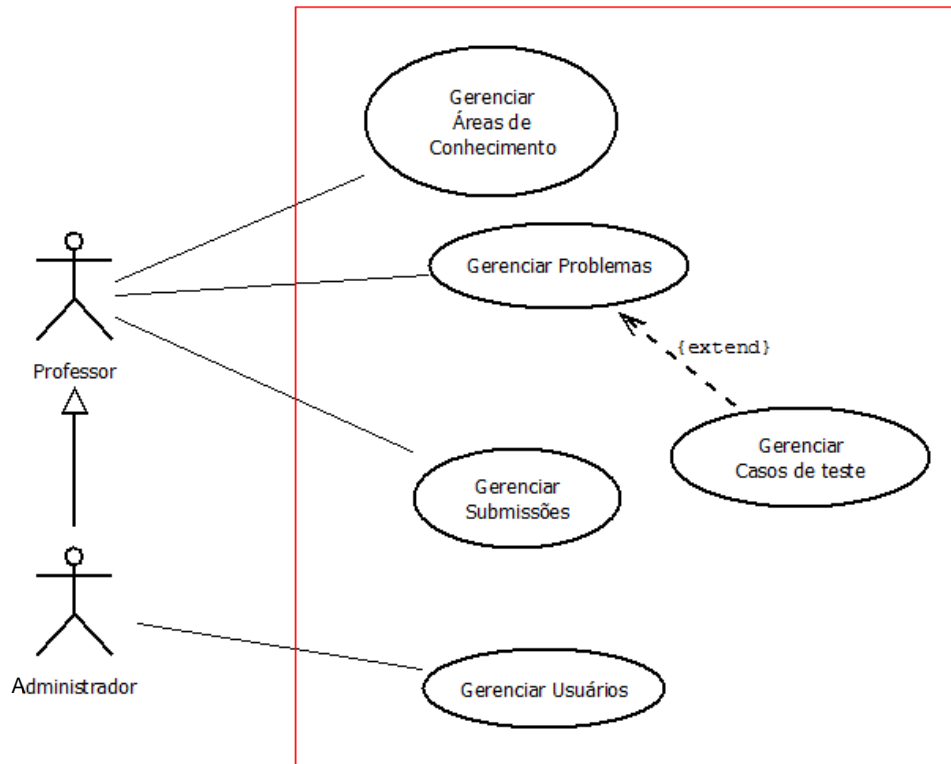
Figura 8 - Diagrama de Caso de Uso do Aluno



Fonte: Elaborada pelo autor

A Figura 9 representa os casos de uso para o professor e administrador: gerenciar áreas de conhecimento, gerenciar problemas, gerenciar casos de teste, gerenciar submissões e gerenciar usuários.

Figura 9 - Diagrama de Caso de Uso de Professor e Administrador



Fonte: Elaborada pelo autor

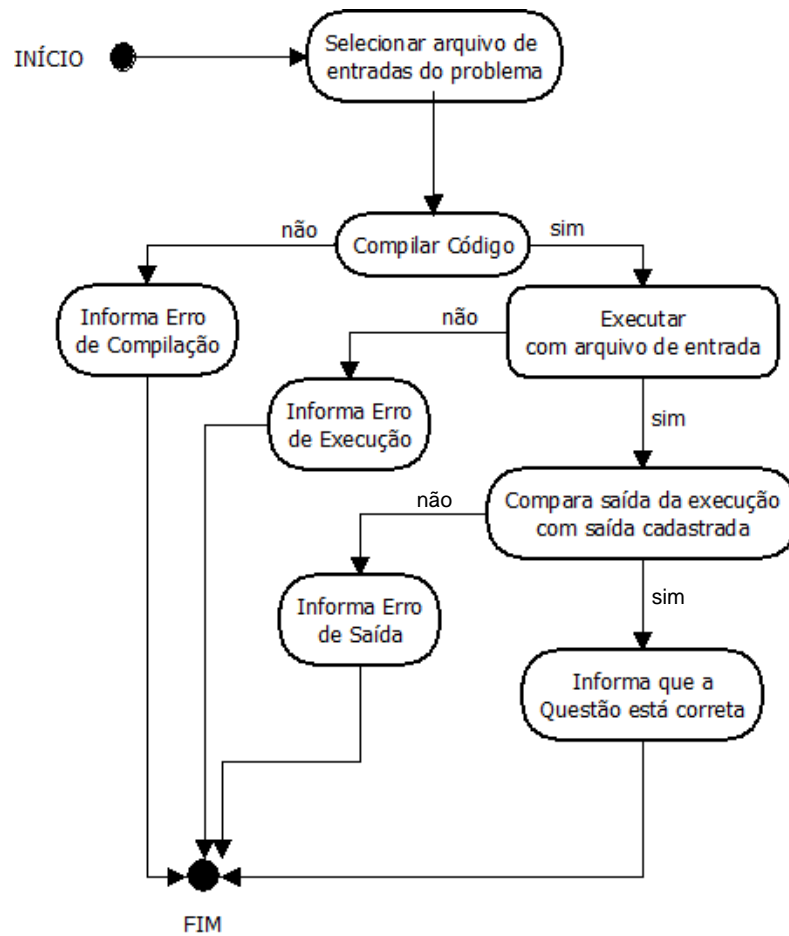
3.3.2 Diagrama de Atividades

O diagrama de atividades descreve o fluxo de atividades envolvidas em um único processo. As atividades são sempre dependentes umas das outras através de suas transições, seguindo uma sequência lógica de ações. (STADZISZ, 2002).

O sistema proposto fundamenta-se em um processo central denominado correção automática de código. Este processo é o responsável em substituir o processo humano de correção, o que lhe faz ser o mais complexo e a parte principal do sistema.

Para melhor entendimento, a Figura 10 apresenta o diagrama de atividades deste processo.

Figura 10 - Diagrama de atividades do processo de correção automática



Fonte: Diagrama criado pelo autor

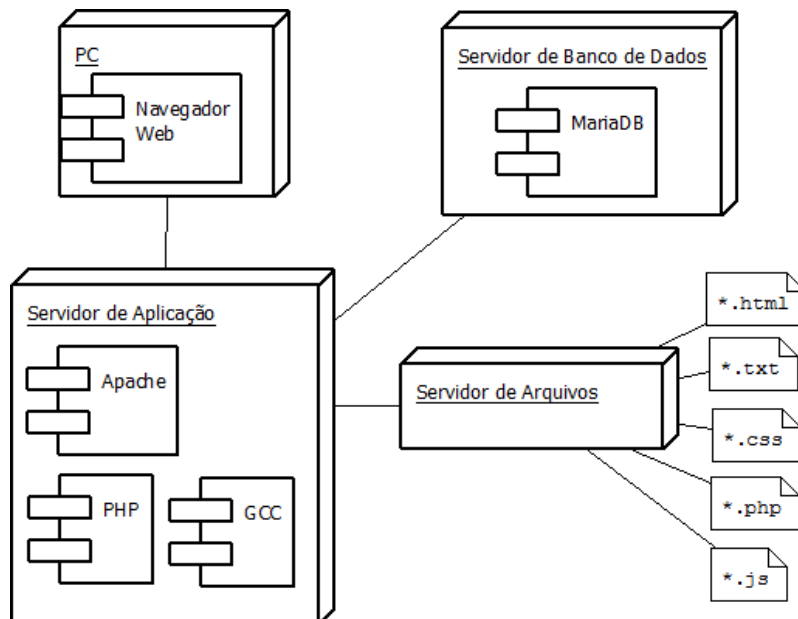
É possível verificar na Figura 10, que há uma sequência lógica para realizar o processo de correção. Na primeira etapa é selecionado o arquivo de entradas cadastradas para o problema. Na segunda etapa é verificada a tentativa de compilar o código submetido. Este teste gera dependência para os demais passos, pois caso esta etapa não seja realizada, as demais ficam impossibilitadas de serem realizadas.

Na terceira etapa é verificado se o código consegue fazer todas as operações atribuídas, se há exceções que abortam a execução e assim o erro é informado ao sistema. Em caso de sucesso na terceira etapa, enfim a quarta etapa é realizada, onde são comparadas as saídas geradas na execução do código com as saídas cadastradas para o problema no sistema. Sendo estas iguais o sistema informa o acerto, pois teve sucesso em todos os passos do processo, mas caso exista alguma diferença o erro é informado ao sistema.

3.3.3 Diagrama de Implementação

O diagrama de implementação representam a arquitetura física do sistema, relacionados a hardware e software. A Figura 11, traz o diagrama de implementação do sistema de correção automática. Mostra a comunicação dos diversos itens do sistema.

Figura 11- Diagrama de Implementação representando a arquitetura física do sistema



Fonte: Diagrama elaborado pelo autor

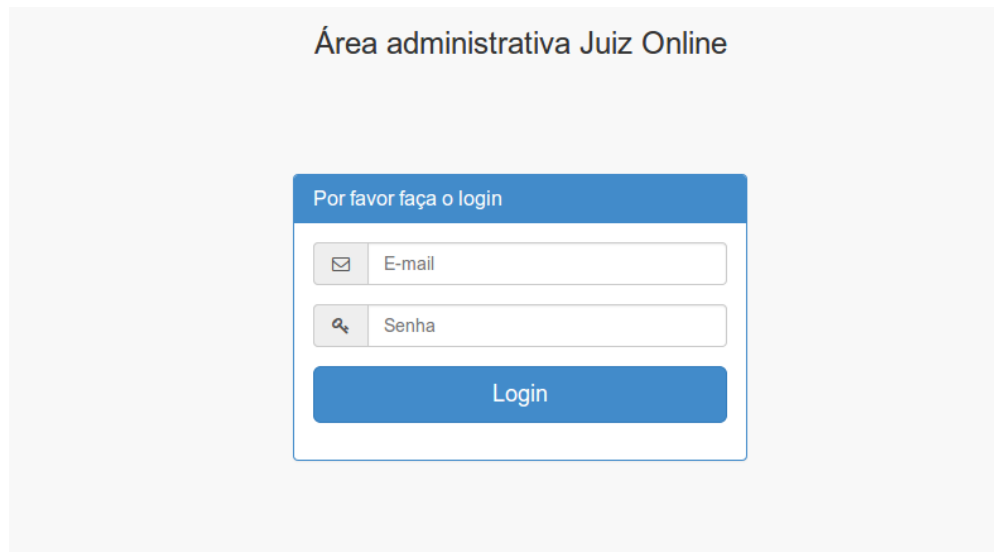
3.4 Detalhamento do Sistema

Neste tópico são detalhados os diversos pontos do sistema proposto. Algumas telas desenvolvidas são mostradas a seguir.

3.4.1 Autenticação de usuários

A autenticação traz como objetivo proporcionar mais segurança ao sistema. Para autenticação do aluno e professor é necessário informar e-mail e senha conforme mostram as Figuras 12 e 13.

Figura 12 - Tela de login do administrador



Área administrativa Juiz Online

Por favor faça o login

E-mail

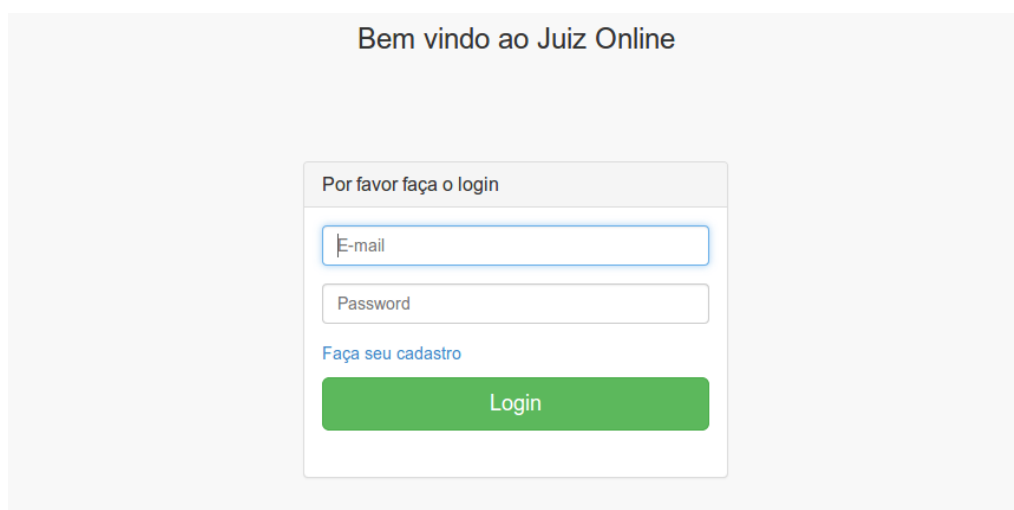
Senha

Login

Fonte: Elaborada pelo autor

A interface de login do professor e administrador é a mesma, sendo este redirecionado para sua categoria de acordo com as permissões atribuídas ao seu usuário.

Figura 13 - Tela de login do aluno



Bem vindo ao Juiz Online

Por favor faça o login

E-mail

Password

[Faça seu cadastro](#)

Login

Fonte: Elaborada pelo autor

3.4.2 Área Administrativa

Os usuários administradores e professores são redirecionados para o mesmo ambiente. A única funcionalidade que não está presente para o professor é o gerenciamento de usuários. As funcionalidades Problemas e Submissões estão disponíveis para os dois usuários.

A Figura 14 mostra a tela de gerenciamento de problemas. Esta área pode ser acessada através do menu lateral. Lá são apresentados os problemas cadastrados e também fornece as funcionalidades de inserir novos problemas, filtrar as informações, ocultar ou ativar problemas para o usuário aluno e visualizar um problema específico.

Figura 14 - Tabela de problemas na tela administrativa

The screenshot displays the administrative interface for managing problems. The main table, 'Tabela de Problemas', lists the following data:

ID	Título	Autor	Área	Data	Status	Ações
3	Problemas das Moscas	Maria	Estrutura Condicional	12/08/2014	On	[Search] [Edit] [Eye]
5	Idade em Dias	Administrador	Iniciante	12/08/2014	On	[Search] [Edit] [Eye]
7	Teste2	Administrador	Estrutura Condicional	29/08/2014	Off	[Search] [Edit] [Eye]
8	Teste3	Administrador	Estrutura Condicional	29/08/2014	Off	[Search] [Edit] [Eye]

Below the table, there are two summary tables:

PROBLEMAS POR CATEGORIA

Área	Problemas	Resolvidos

ESTATÍSTICAS

#	Problema	Tentativas	Acertos	Erros

Fonte: Elaborada pelo autor

Para inserir um novo problema é necessário clicar no botão Novo Problema. Na elaboração do problema é necessário informar os itens a seguir:

- Área de conhecimento;
- Título do problema;
- Descrição do problema;
- Exemplos de entrada e saída;
- Anexar o arquivo com entradas e saídas;
- Tempo limite para execução da questão;
- Nível de conhecimento;

Outros dados como data de cadastro, autor do problema são inseridos automaticamente pelo sistema no banco de dados.

Pode ser visto nas Figuras 15 e 16 o formulário disponibilizado para elaborar os problemas. Pensando em facilitar a formatação do enunciado é disponibilizado nesta área um editor de texto desenvolvido em Java Script. As questões cadastradas podem ser utilizadas por diferentes usuários.

Figura 15 - Formulário de cadastro do problema

Cadastrar Problema

Fonte: Elaborada pelo autor

Figura 16 - Continuação do formulário de cadastro de problemas

Fonte: Elaborada pelo autor

No menu lateral da área administrativa é possível encontrar também o *link* para a página de Submissões. Nesta área são listadas as últimas submissões para os problemas realizados pelos alunos, ordenadas pela data e hora da submissão.

A tabela visualizada na Figura 17 mostra como estão organizadas as informações da submissão: Autor da submissão, título do problema, data de envio, linguagem utilizada no arquivo fonte, tempo de execução e resultado da correção.

Figura 17 – Visualização das últimas submissões

The screenshot shows the 'Submissões' page in the Juiz Online IF system. The page header includes 'Juiz Online IF' and 'Bem-Vindo Administrador'. The left sidebar contains navigation links: Painel, Problemas, Soluções, Usuários, Alunos, and Estatísticas. The main content area is titled 'Submissões' and contains a table of submissions and a summary table below it.

Tabela de Submissões						
ID	Autor	Problema	Data	Linguagem	TempoExec	Status
5	Geldson Benício Coelho de Souza	Idade em Dias	28/08/2014 03:23:01	C	0.1200000	Erro de Execução
1	Geldson Benício Coelho de Souza	Problemas das Moscas	28/08/2014 02:58:15	C++	0.0000000	Correto
2	João de Souza Silva	Teste	28/08/2014 02:57:15	C	0.2100000	Erro de Compilação

ESTATÍSTICAS DE SUBMISSÕES						
Total	Corretas	Incorretas	Erro de Compilação	Erro de Saída	Erro de Execução	Tempo Limite

Fonte: Elaborada pelo autor

3.4.3 Área do Aluno

A interface do aluno é baseada na mesma interface do usuário administrativo. O usuário aluno pode ver os problemas cadastrados pelos professores e enviar soluções para os problemas disponíveis.

Na área de problemas são listados os problemas que estão ativos no sistema, disponibilizados em uma tabela com as colunas título, área de conhecimento, nível de conhecimento, números de tentativas de resolução para o problema, porcentagem de acertos e a opção de visualizar um problema específico, como é mostrado na Figura 18. Também é disponibilizado um filtro e a funcionalidade de ordenar as questões por qualquer coluna.

Figura 18 - Tela de problemas do usuário aluno

The screenshot shows the 'Problemas' page in the Juiz Online IF system. The page header includes 'Juiz Online IF' and 'Bem-Vindo Geidson'. The left sidebar contains navigation options: 'Painel', 'Problemas', 'Soluções', 'Estatísticas', 'Flot Charts', and 'Morris.js Charts'. The main content area is titled 'Problemas' and features a 'Tabela de Problemas' section. This section includes a dropdown for 'resultados por página' (set to 10) and a 'Filtrar:' input field. The table below has the following data:

ID	Título	Área	Nível	Tentativas	% acertos	
3	Problemas das Moscas	Estrutura Condicional	2	1	100%	Visualizar
5	Idade em Dias	Iniciante	1	1		Visualizar
7	Teste2	Estrutura Condicional	1	2	50%	Visualizar
8	Teste3	Estrutura Condicional	1	0		Visualizar

Below the table, it says 'Mostrando 1 a 4 de 4 resultados'. There are two summary tables: 'PROBLEMAS POR CATEGORIA' and 'ESTATÍSTICAS'.

Área	Problemas	Resolvidos

#	Problema	Tentativas	Acertos	Erros

Fonte: Elaborada pelo autor

Ao selecionar um problema este é aberto como mostrado na Figura 19. Esta área traz o título, autor, enunciado da questão, especificações para entrada e saída, exemplos de entrada e saída e um formulário onde o aluno poderá anexar o seu código-fonte.

Figura 19 - Visualização de um problema

The screenshot shows the 'Idade em Dias' problem page. The page header includes 'Problemas' and 'Soluções'. The main content area is titled 'Idade em Dias' and includes the following information:

Idade em Dias
 Autor: Administrador
 Problema retirado do site URI Online Contest.

Lela um valor inteiro correspondente à idade de uma pessoa em dias e informe-a em anos, meses e dias.
 Obs: apenas para facilitar o cálculo, considere todo ano com 365 dias em todo mês com 30 dias. Nos casos de teste nunca haverá uma situação que permite 12 meses e alguns dias, como 360, 363 ou 364. Este é apenas um exercício com objetivo de testar raciocínio matemático simples.

Entrada
 O arquivo de entrada contém um valor inteiro.

Saída
 Imprima a saída conforme exemplo fornecido

Exemplo Entradas
 400

Exemplo Saídas
 1 ano(s)
 1 mes(es)
 5 dia(s)

Submeter Solução
 Escolher arquivo Nenhum arquivo selecionado
 Submeter Solução

Fonte: Elaborada pelo autor

Outras funcionalidades encontram em desenvolvimento e são citados no capítulo de trabalhos futuros.

4 IMPLEMENTAÇÃO

Neste capítulo são apresentadas as tecnologias utilizadas para o desenvolvimento, apresentação e execução do sistema.

4.1 Linguagem PHP

O sistema de correção automática foi desenvolvido em PHP, uma linguagem de *script open source* muito utilizada para o desenvolvimento de aplicações *Web* dentro do HTML¹⁷. A escolha do PHP vem do fato dele uma sintaxe extremamente simples, oferecendo muitos recursos para a programação, como suporte a vários servidores *Web* e banco de dados, resolução problemas básicos de performance, modularidade e seções, como também suporte à orientação à objetos. (DALL’OGLIO, 2012).

4.1.1 PDO - PHP *Data Objects*

O PHP evoluiu a partir de colaboradores distribuídos geograficamente ao redor do mundo, movido pelas resoluções de problemas individuais e razões diversas. A colaboração fez o PHP crescer rapidamente, por outro lado geraram uma fragmentação das extensões de acesso a bases de dados, cada qual com sua implementação, não havendo real consistência entre as interfaces. (DALL’OGLIO, 2012).

O PHP *Data Objects* - PDO surgiu da necessidade de unificar o acesso a diferentes bancos de dados, com objetivo de prover uma API limpa e consistente, unificando a maioria das características presentes nas extensões de acesso a banco de dados.

“O PDO não é uma biblioteca completa para abstração do acesso à base de dados, uma vez que não faz a leitura nem a tradução das instruções SQL, adaptando-as aos mais diversos ‘*drives*’ de banco de dados existentes” (DALL’OGLIO, 2012). Assim, unifica as chamada de métodos de forma simples, delegando-os a suas extensões correspondentes.

Pensando nessa facilidade de acesso ao banco de dados através da API PDO, o sistema proposto faz o uso dessa biblioteca para conexão segura do banco de dados MariaDB¹⁸. No

¹⁷ *HiperText Markup Language* – Linguagem de Marcação de Hipertexto. Utilizada para produzir página web.

¹⁸ <https://mariadb.org>

Linux, sistema utilizado para desenvolvimento do sistema, é preciso habilitar, no arquivo **php.ini**, os drivers de acesso ao banco de dados da nossa aplicação.

Passos realizados no sistema Linux:

- Descomentar a linha **extension=mysql.so** no **php.ini**, pois o banco de dados MariaDB utiliza os mesmos drivers de conexão do MySQL.
- Configurar *String* de conexão: **new PDO('mysql:host=localhost;port=3306;dbname=juizonline', 'user', 'senha');**

O código referente à implementação da API PDO está em anexo.

4.2 Servidor Web

O servidor web é o responsável por processar e disponibilizar as páginas e todos os demais recursos que o cliente queira acessar. Este processa solicitações HTTP (*Hyper-Text-Transfer-Protocol*), o protocolo padrão da *web*.

Para implementação do sistema foi escolhido o servidor web Apache Server¹⁹ que é uma ferramenta que executa tanto HTTP, como outros protocolos, tais como HTTPS (HTTP + SSL – *Secure Socket Layer*), FTP (*File Transfer Protocol*), entre outros. É capaz de executar código em PHP, Perl, Shell Script, ASP, etc. Outras características são o suporte à criptografia MD5²⁰, personalização de *logs*, suporte a tipos mime²¹, suporte à autorização de acesso, facilidade de adicionar ou remover recursos sem a necessidade de recompilar o programa, entre outros. (APACHE, 2014).

O Apache Server é disponibilizado na forma de software livre, podendo ser utilizado gratuitamente. Graças a essas características ele continua sendo melhorado ao passar dos anos, o que lhe tornou o servidor mais utilizado no mundo.

¹⁹ <http://www.apache.org/>

²⁰ *Message-Digest algorithm 5* - é um algoritmo de *hash* de 128 bits unidirecional.

²¹ *Multipurpose Internet Mail Extensions*, uma norma da internet para o formato das mensagens de correio eletrônico.

4.3 Banco de Dados

Como o sistema utiliza a biblioteca PDO, todo acesso ao banco de dados é realizado pelos *drivers* da API, tornando o sistema independente de qualquer banco de dados. Assim para trocar o banco de dados, basta somente alterar a configuração na *string* de conexão do PDO para o driver do banco escolhido.

Como banco de dados para ser usado na etapa de desenvolvimento, foi selecionado o MariaDB, um servidor de banco de dados relacional que oferece as funcionalidades e substituição para o MySQL²². Além das funcionalidades básicas do MySQL, o Maria DB oferece um rico conjunto de recursos, incluindo mecanismos de armazenamento alternativo, otimização, *patches*, além de uma vasta documentação. Outra vantagem dessa ferramenta é ser de código aberto, assim existe uma grande comunidade de desenvolvedores sempre trabalhando para melhorá-la.

4.4 Front-End

Esta seção apresenta as ferramentas utilizadas para desenvolvimento da parte visual do site, a forma como o conteúdo é apresentado na tela e aplicação do design para exibição das informações. Pensando em um rápido desenvolvimento sem comprometer a performance e compatibilidade entre navegadores foram escolhidas ferramentas bastante conhecidas do mercado.

4.4.1 SB Admin v2

É um *template open source* para websites baseado no *framework* Bootstrap²³, com suporte para HTML5 e compatível com vários navegadores, incluído navegadores mobile.

O SB Admin v.2²⁴ inclui uma base de CSS, *plug-ins* jQuery, fontes, ícones dentre outras estruturas prontas para acelerar o desenvolvimento *front-end*, com característica na UI Twitter Bootstrap. No sistema foi aproveitado várias estruturas prontas fornecidas pelo template. A Figura 20 representa o painel inicial da interface SB Admin, com diversas funcionalidades pré-configuradas que podem ser reutilizadas para diversos tipos de projetos.

²² <http://www.mysql.com>

²³ *Framework front-end* para desenvolvimento ágil de projetos web. <http://getbootstrap.com/>

²⁴ <http://startbootstrap.com/template-overviews/sb-admin-2/>

Figura 20 - Template padrão SBAdmin v2



Fonte: www.startbootstrap.com

4.4.2 jQuery

jQuery é uma biblioteca JavaScript criada em 2005 por John Resign²⁵, disponibilizada como software livre e aberto. Essa biblioteca destina-se a adicionar interatividade e dinamismo às páginas web, proporcionando ao desenvolvedor as funcionalidades necessárias para criação de scripts que visem a incrementar, de forma progressiva e não obstrutiva, a usabilidade, a acessibilidade e o design, enriquecendo a experiência do usuário.

Para Silva (2010) deve-se utilizar o jQuery para:

- Adicionar efeitos visuais e animações;
- Acessar e manipular o DOM²⁶;
- Buscar informações no servidor sem necessidade de recarregar a página;
- Prover interatividade;
- Simplificar tarefas específicas de JavaScript;

²⁵ <http://ejohn.org/>

²⁶ Document Object Model. - multi-plataforma que representa como as marcações em HTML, XHTML e XML são organizadas e lidas pelo navegador.

- Modificar apresentação e estilização;

O template do sistema SB Admin v.2 utiliza o jQuery para localizar elementos componentes da estrutura HTML das páginas, como também para anular as inconsistências de renderização entre navegadores.

4.4.3 Twitter Bootstrap

O Twitter Bootstrap é um *framework front-end* de código aberto desenvolvido pela equipe do Twitter. Tem como objetivo tornar mais fácil o desenvolvimento de interfaces para páginas web, disponibilizando padrões para os principais elementos HTML, além de elementos personalizados com classes CSS padrões (BOOTSTRAP, 2014).

Compatível com HTML5 e CSS3, o *framework* possibilita a criação de layouts responsivos e o uso de *grids*, permitindo que o conteúdo se comporte de maneira diferente para cada resolução.

Como qualquer outra ferramenta apresenta vantagens e desvantagens.

Vantagens:

- Documentação detalhada;
- Otimizado para layouts responsivos;
- Possui componentes suficientes para o desenvolvimento de qualquer site ou sistema web com interface simples;
- Funciona em todos os navegadores atuais;

Desvantagem:

- O código da aplicação terá que seguir os “padrões de desenvolvimento Twitter Bootstrap”;

A estrutura é simples e seu pacote contém arquivos CSS, *JavaScript* e *Fonts*. O *template* escolhido para o sistema é desenvolvido totalmente no padrão Twitter Bootstrap, explorando toda a sua estrutura e componentes.

4.5 Execução de Programas: safeexec

O safeexec é um programa *sandbox* escrito na linguagem C (um ambiente controlado e restringido) para executar de forma segura e manter o controle de programas do usuário. Essa *sandbox* garante que os programas executados pelo corretor automático não comprometam o resto do sistema, seja em questão de segurança ou performance.

A versão escolhida para o sistema é baseada na versão presente no sistema BOCA, com a funcionalidade de *chroot*²⁷, permitindo controlar o ambiente em que serão executados os códigos dos alunos.

4.6 Testes

Os testes apresentados à seguir foram realizados durante o desenvolvimento das funcionalidades apresentadas neste trabalho.

O teste de configuração foi realizado a fim de verificar o funcionamento do sistema no hardware escolhido. Durante os testes foram realizadas diversas solicitações HTTP e de banco de dados, sendo que o sistema apresentou consistência em uma máquina com 2gb de memória, processador Intel Dual Core.

Foi realizado teste de segurança com objetivo de verificar se os dados são acessados de maneira segura somente pelo autor das ações. Durante esse teste foi utilizado em momentos distintos os usuários aluno, professor e administrador para realizar operações no sistema e verificar o acesso a determinadas áreas com ou sem permissão. Ao final foi verificado que cada usuário tem acesso a sua área específica, sem autorização para acessar outras áreas.

Outro teste realizado foi o teste funcional. Este teste valida os requisitos funcionais e os casos de uso do sistema. Durante os testes foram analisados se as funcionalidades atendem aos requisitos determinados para o sistema. Ao final foi possível observar que o requisito Visualizar estatísticas do ator Aluno e Gerenciar Áreas de Conhecimento do ator Professor não foram atendidos pelo sistema. Os demais requisitos foram implementados.

²⁷ Operação que muda o diretório root do processo corrente e de seus processos filhos em sistemas operacionais Linux.

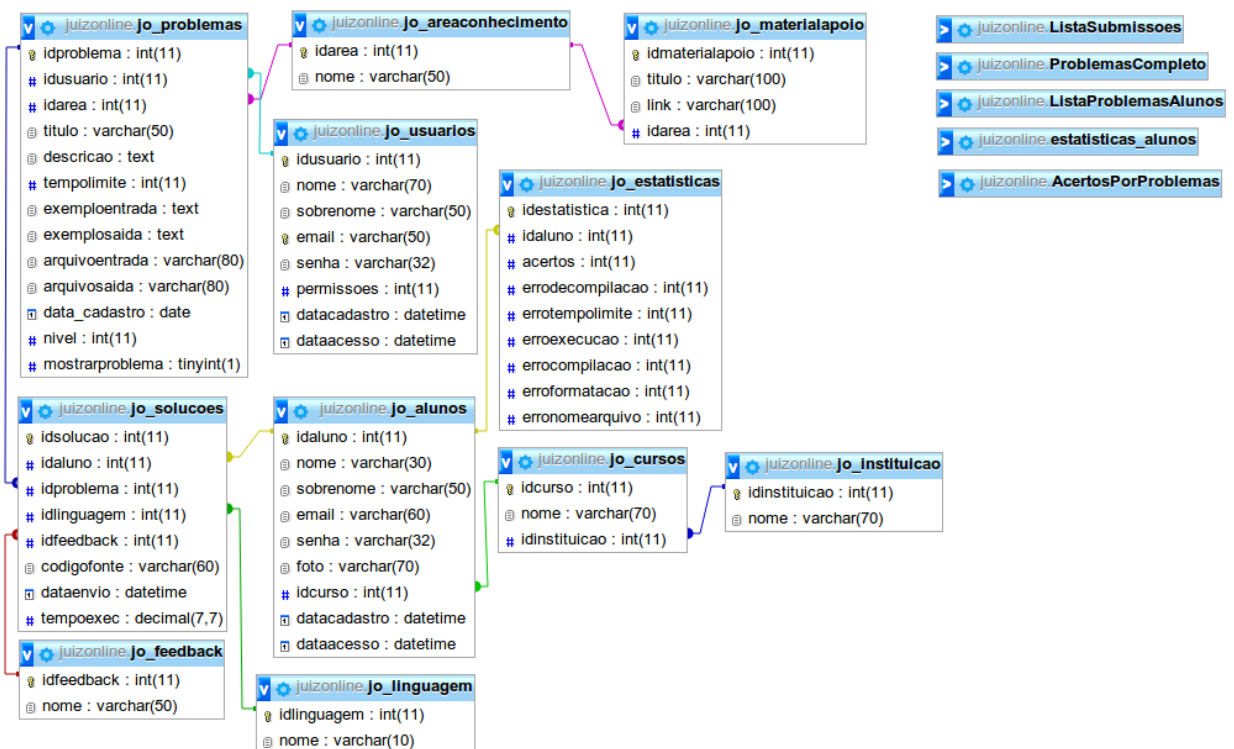
5 ORGANIZAÇÃO DO SISTEMA

Neste capítulo apresenta a organização do banco de dados e a relação entre componentes do sistema.

5.1 Modelagem do Banco de Dados

Buscando um melhor entendimento do banco de dados, foi utilizado o modelo entidade relacionamento para representa-lo.

Figura 21 - Representação do Banco de Dados



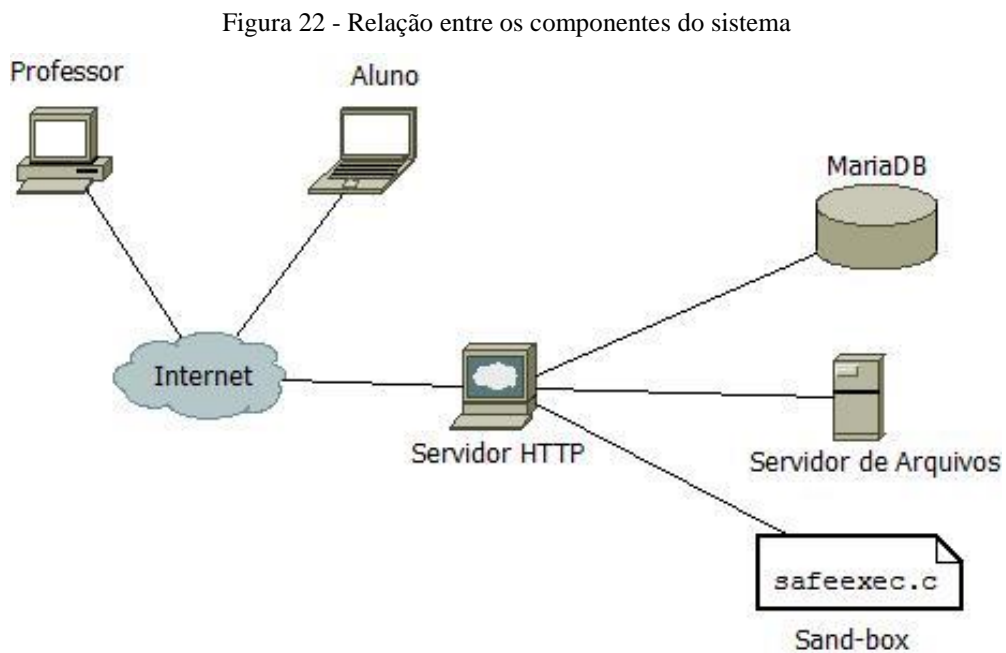
Fonte: Elaborada pelo autor

Na Figura 21 visualizamos 11 tabelas e o relacionamento entre elas. Há também *views* que ajudam a selecionar dados para o sistema. Foram também criados procedimentos que executam tarefas de inserção e seleção de dados no próprio banco de dados, deixando para o servidor HTTP²⁸ outras tarefas mais importantes. É válido lembrar que é preciso desenvolver outros procedimentos e também gatilhos para melhor performance do banco de dados.

²⁸ Hypertext Transfer Protocol – Protocolo de Transferência de Hipertexto.

5.2 Relação entre elementos do sistema

A implementação do sistema de correção automática é realizado em uma máquina na função de servidor web, servidor de aplicação e servidor de banco de dados. O esquema é representado na Figura 22, que descreve a forma como os componentes do sistema se relacionam.



Fonte: Elaborada pelo autor

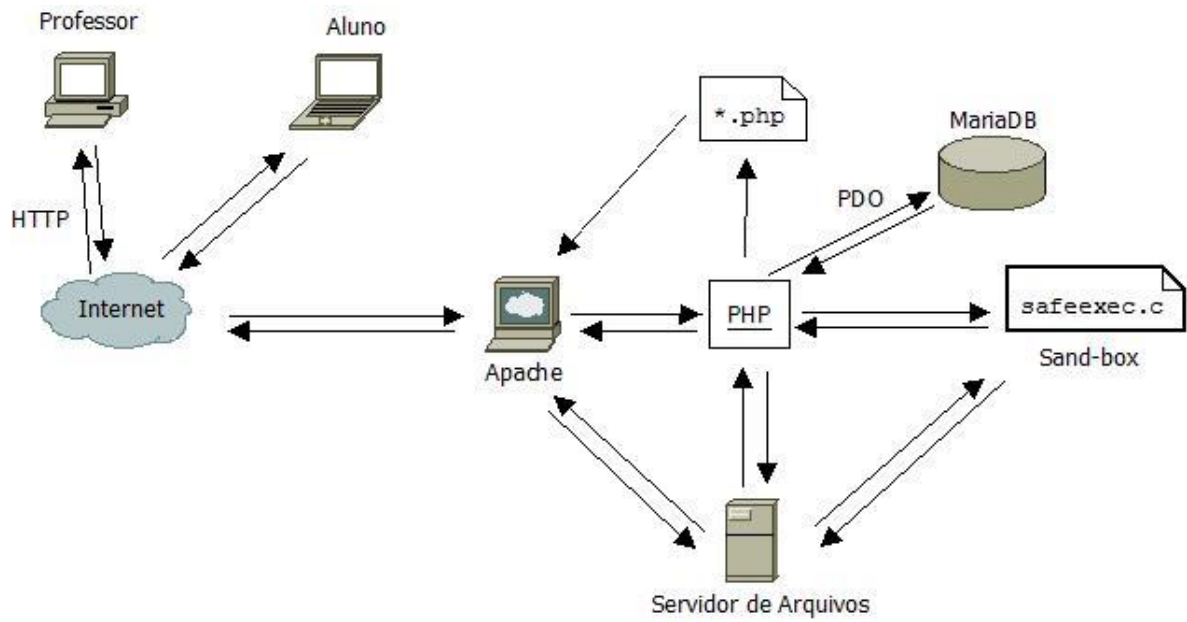
No servidor rodam os seguintes serviços:

- Servidor HTTP (Apache 2.2);
- PHP 5.3;
- Banco de dados (Maria DB);
- *Sandbox* (safeexec.c);

As estações de Usuário Professor e Aluno podem acessar o servidor através da rede interna ou internet.

A Figura 23 exemplifica a representação da comunicação realizada entre os componentes.

Figura 23 - Comunicação entre os componentes do sistema



Fonte: Elaborada pelo autor

6 RESULTADOS

O sistema está em fase de desenvolvimento, mas já apresenta funcionalidades relevantes que permitem a sua avaliação.

O sistema proposto neste trabalho apresenta funcionalidades similares com as ferramentas de correção automática utilizadas por diversas universidades do mundo.

A pesquisa permitiu conhecer o funcionamento dos principais juízes *online* existentes, o que permitiu projetar o sistema escolhendo as funcionalidades que atendessem os requisitos observados para o ensino de programação do IF Sertão – PE. Também esse estudo permitiu escolher as melhores ferramentas e práticas de desenvolvimento, que contribuíssem para o bom resultado do projeto.

No módulo de gerenciamento de questões os professores podem gerenciar os problemas cadastrados de forma simplificada, como também acompanhar o desempenho dos alunos na tentativa de solucionar as questões.

A *sandbox* safeexec atendeu as necessidades do projeto constituindo um ambiente seguro de execução de código. Trouxe recursos que garantem que um programa não danifique o servidor, nem consuma excessivos recursos computacionais. O corretor automático se apresentou eficaz nos testes realizados com problemas de nível iniciante. Este mostrou os *feedbacks* corretos para as avaliações realizadas com os códigos-fonte de teste.

O corretor apresenta potencial para auxiliar os alunos, embora os testes só tenham sido realizados durante o desenvolvimento, mas as expectativas são grandes para que ele sirva como facilitador para o professor, poupando-o em trabalhos de correção e forneça um ambiente de aprendizado para os alunos.

Ainda há muito trabalho a ser feito para que o sistema alcance todos os resultados esperados, pois apenas uma pequena parte das funcionalidades foram implementadas durante esse TCC, porém foi possível visualizar novos objetivos e preparar o ambiente para futuras melhorias e novas funcionalidades.

6.1 Dificuldades encontradas

O desenvolvimento do trabalho foi prejudicado pela greve que paralisou as atividades do instituto por três meses. Assim o calendário precisou ser alterado tornando curto o tempo de desenvolvimento de funcionalidades do projeto e para realização de testes com alunos das turmas de programação.

Foram encontradas algumas dificuldades na etapa de especificação, devido a pouca experiência com modelagem UML.

Outro problema foi a transcrição do código SQL das views, procedimentos e gatilhos do sistema para o padrão SQL do MariaDB, já que este foi desenvolvido durante a disciplina de banco de dados II no padrão Firebird.

Já a maior fonte de problemas foi realmente a implementação do sistema de correção, já que esse demandou muitas configurações e testes, aumentando consideravelmente o tempo de desenvolvimento.

7 CONCLUSÃO

Este trabalho vem debater a relevância de uma ferramenta que auxilie a criação e a correção de problemas de programação de forma rápida. A ideia geral foi propor um sistema de autocorreção de código, fazendo o uso de software livre, boas práticas de desenvolvimento e tecnologias de referências no mercado de tecnologia, visando tornar o aprendizado mais dinâmico e atrativo, amenizando problemas como a demora na correção de exercícios de programação e o gerenciamento de exercícios e correções.

Primeiramente o objetivo foi identificar as características da ferramenta, onde foram levantados os requisitos básicos usados pelos principais sistemas de correção automatizada. Após esse estágio foram escolhidas e estudadas algumas bibliotecas para uso no processo de desenvolvimento, optando-se sempre por ferramentas de software livre. Assim foi possível obter várias experiências reunindo diversas tecnologias muito utilizadas pelo mercado.

Seguindo, observou-se que a utilização da ferramenta de correção automática de código em laboratórios de programação pode favorecer o trabalho do professor reduzindo a sua carga de trabalho e trazendo uma melhor experiência prática para o aluno. A metodologia utilizada pelo sistema e a forma como os problemas são apresentados, permite que o aluno pratique o aprendizado de programação ao mesmo tempo em que se prepara para as principais competições de programação.

Desta forma, pode-se afirmar que o sistema derivado desta pesquisa irá representar um avanço no auxílio ao processo de ensino e aprendizagem em disciplinas de programação. Aplicar e corrigir problemas em um prazo curto, menos trabalho repetitivo, respostas sobre a compilação e execução dos problemas, métricas de desempenho, repositório de questões online serão grandes benefícios alcançados pela ferramenta, tornando o exercício e a prática mais ágil e dinâmica.

Outra contribuição está em aproximar o aluno com a disciplina de programação. O processo de correção e *feedback* abre a oportunidade de tentativas de erro, redirecionando o foco do aluno para a lógica da programação propriamente dita. O gosto do estudante pela programação também o interesse pelos estudos, contribuindo para reduzir os índices de abandono do curso.

Portanto, conclui-se que essa ferramenta será um recurso didático facilitador no processo de ensino e aprendizagem de programação.

7.1 Trabalhos futuros

Aqui estão listadas as possíveis melhorias para o sistema em fase de desenvolvimento.

Melhorar a exibição e estatísticas

Mostrar estatísticas em forma de gráfico e histórico uso do sistema pelo usuário poderá tornar o ambiente mais atrativo, além de facilitar a análise dos dados.

Sistema de Pontuação

Implementar um sistema de pontuação em forma de ranking. A competição estimula a interação do usuário com o sistema. Os pontos serão gerados a partir de cálculos entre nível de dificuldade e área de conhecimento dos problemas solucionados.

Testes e avaliação do sistema

Este é um trabalho que deverá ser realizado com uma turma de alunos e professores a fim de testar as funcionalidades, visando validar os requisitos, verificar a performance do corretor e avaliar se este atende as necessidades observadas nos laboratórios de programação. Ao final gerar dados estatísticos do que foi obtido.

Suporte a outras linguagens de programação

O suporte a outras linguagens é uma funcionalidade essencial para o sistema. Com isto será possível que o aprendizado possa ocorrer sob as diferentes linguagens, tendo o aluno a opção de escolher a linguagem que lhe seja mais familiar.

Fórum de discussão

A possibilidade de implementar um fórum de discussão, amplia a possibilidade de aquisição e interação com o conhecimento. O fórum fornece o debate, a troca de opiniões, construção de pensamentos, cooperação e interação entre os usuários visando entendimento entre todas as partes. Este recurso será essencial para o sistema, pois irá agregar valor ao conhecimento e servir de apoio as mais diversas situações que ocorrerão no uso do corretor automático.

Gerenciamento de Turmas

Uma continuidade interessante para esse projeto é permitir a divisão de alunos por turmas. Essa funcionalidade irá permitir que o professor tenha o controle das ações em sala de aula e também irá facilitar o acompanhamento de progresso dos seus alunos. Assim ele pode ajudar os que estão com maiores dificuldades ao mesmo tempo em que orienta alunos a explorarem problemas mais complexos.

Análise de Plágio

Integrar ao sistema um analisador de similaridade entre códigos-fonte. Para isso é preciso realizar estudo de trabalhos relacionados visando escolher o melhor analisador. Essa funcionalidade permitirá ao professor analisar comparações das soluções apresentadas pelos alunos e verificar se alguma solução foi beneficiada por uso de plágio. O professor será responsável em ativar essa funcionalidade em um problema qualquer.

Sistema de Dicas

Realizar um estudo mais aprofundado, focando o desenvolvimento de uma metodologia para disponibilizar dicas dentro do sistema. Essas dicas podem estar relacionadas ao tipo de erro ou sobre o desempenho do aluno.

Material de Apoio

Esta é uma funcionalidade que trará para o aluno um ambiente com conteúdos relevantes disponibilizados pelos professores para servir de suporte no desenvolvimento das atividades e no estudo das técnicas de programação. Esse recurso será apresentado respeitando as categorias cadastradas para os problemas, visando uma maior organização do material de apoio dentro do sistema.

Ao final deste projeto todo o código-fonte do sistema será repassado para a Fábrica de Software do IF Sertão – PE, permitindo assim que novos desenvolvedores possam agregar essas funcionalidades para que outras ideias sejam implementadas no decorrer do uso do sistema, a partir de um desenvolvimento colaborativo.

Que no futuro esta ferramenta possa contribuir para o desenvolvimento do raciocínio lógico dos estudantes de programação e possa estimular a participação de alunos de maratonas

de programação, criando uma cultura de aprendizado prático de programação, facilitando o trabalho dos profissionais de educação.

8 REFERÊNCIAS

ALMEIDA, V.C. Proposta de Implementação de Métricas de Complexidade em um Avaliador Automatizado de Programas – VDSP. Monografia de diplomação do curso de Sistemas de Informação da PUC-MG. Arcos, 2005.

APACHE. The Apache Software Foundation. Disponível em: <<http://www.apache.org>>. Acesso em: Out. 2014.

BEZ, J. L.; TONIN, N. A. URI Online Judge e a Internacionalização da Universidade. Vivências: Revista Eletrônica de Extensão da URI, vol. 10, N. 18: p. 237-249, 2014.

BOOTSTRAP. Disponível em: <<http://getbootstrap.com>> Acesso em: Set. 2014

CAMPOS, C. P.; FERREIRA, C. E. BOCA: um sistema de apoio para competições de programação. Workshop de Educação em Computação, Anais do Congresso da SBC, Salvador-BA, 2004.

CAMPOS, R. L. B. L. Metodologia ERM2C: Para melhoria do processo de ensino-aprendizagem de lógica de programação. In: Congresso da Sociedade Brasileira de Computação: Workshop de Informática na Educação, 30, 2010. Anais do Congresso da SBC, Belo Horizonte – MG, 2010.

CAMPOS, R. L. B. L. Lógica de programação: Há como melhorar o aprendizado fugindo dos padrões estabelecidos nos livros didáticos e adotados pela maioria dos docentes? XVII Congresso Iberoamericano de Educacion Superior em Computacion (CLEI-2009-CIESC), 22-25/Set/2009. Brazil, RS, Pelotas.

CHAVES, J. O. *et al.*. Mojo: Uma ferramenta para auxiliar o professor em disciplinas de programação. In: Congresso Brasileiro de Ensino Superior à Distância, 10, 2013. Belém/PA, 2013.

DALL’OGLIO, P. PHP-GTK: Criando aplicações gráficas com PHP. 3. ed. São Paulo: Novatec Editora, 2012.

DELGADO, C. *et al.*. Uma Abordagem Pedagógica para a Iniciação ao Estudo de Algoritmos. In: Workshop de Educação em Computação, 12, 2004. Anais do WEI. Salvador, BA, Brasil.

FRANÇA, A. B.; SOARES, J. M. Sistema de apoio a atividades de laboratório de programação via Moodle com suporte ao balanceamento de carga. Anais do XXII SBIE – XVII WIE, Aracaju, 2011. p. 710-719.

GALASSO, R. H.; MOREIRA, B. G. Integração do ambiente BOCA com o ambiente Moodle para avaliação automática de algoritmos. In: *Computer on the Beach 2014*, Florianópolis/SC. Anais do Computer on the Beach, 2014, p. 22-31.

GITHUB. BOCA Online Contest Administrador. Disponível em: <https://github.com/viniciusmarangoni/Boca_Python>. Acesso em: 29 ago. 2014.

GOOGLE PLAY. Projeto Euler. Disponível em: <<https://play.google.com/store/apps/details?id=in.nishitp.euler>>. Acesso em: 09 abr. 2014.

KOLIVER, C.; DORNELES, R. V., CASA, M. E. Das (muitas) dúvidas e (poucas) certezas do ensino de algoritmos”. XII Workshop de Educação em Computação (WEI'2004). Salvador, BA, Brasil.

KURNIA, A; LIM, A.; CHEANG, B. Online Judge. *Computer & Education*, v. 36, n. 4, p. 299-315, 2001.

MOREIRA, M. P; FAVERO, E. L. Um Ambiente Para Ensino de Programação com Feedback Automático de Exercícios. In: Workshop Sobre Educação em Computação, 17., 2009, Bento Gonçalves. Anais do Congresso da SBC. Bento Gonçalves, 2009.

MUSSINI, J. A. Novas arquiteturas para detecção de plágio baseadas em redes P2P. Dissertação de Mestrado em Informática. Pontifícia Universidade Católica do Paraná. Curitiba, PR, 2008.

PELZ, F. D. Correção automática de algoritmos no ensino introdutório de programação. Trabalho de Conclusão do Curso de Ciência da Computação, Universidade do Vale do Itajaí, SC, 2011.

PROJETO EULLER. Disponível em: <<https://projecteuler.net/problems>>. Acesso em: 09 abr. 2014.

PONTES, F.A. MIRANDA, Z. C. AMAO – Desenvolvimento de um ambiente online de auxílio à correção e resolução de avaliações de programação. Projeto de Graduação para título de Bacharel em Sistemas de Informação. UNIRIO, Rio de Janeiro, 2013.

REVILLA, M. A.; MANZOOR, S.; LIU, R. Competitive Learning in Informatics: The UVa Online Judge Experience. *Olympiads in Informatics*, 2008, vol. 2, 131-148. Institute of Mathematics and Informatics, Vilnius.

RODRIGUES JR, M. C. “Experiências Positivas para o Ensino de Algoritmos”, II Workshop de Educação em Computação e Informática Bahia-Sergipe. Disponível em: <<http://www.uefs.br/erbase2004/documentos/weibase/Weibase2004Artigo001.pdf>>. Acesso em: Jul. 2014.

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), 22, 2011, Aracaju – SE. Anais do XXII SBIE – XVII WIE, 2011, p. 964-967.

SANTOS, J. C. S.; RIBEIRO, A. R. L. Uma proposta de um juiz *online* didático para o ensino de programação. In: Encontro Nacional de Informática e Educação, 2, 2011, Cascavel/PR. Anais UNIOESTE Campus Cascavel: 2011. p. 332-341.

SCHILDT, H. **C Completo e Total**. 4ª. ed. São Paulo: MaKron Book do Brasil, 2006.

SILVA, I. F. A.; SILVA. I. M. M.; SANTOS. M. S. Análise de problemas e soluções aplicadas ao ensino de disciplinas introdutórias de programação. IX Jornada de Ensino, Pesquisa e Extensão - UFRPE, Recife, 2009. Disponível em: <<http://www.eventosufrpe.com.br/jepex2009/cd/resumos/R1479-1.pdf>>. Acesso em: 27 ago. 2014.

SILVA, M. S. jQuery: a biblioteca do programador JavaScript / Maurício Samy Silva. 2. ed. rer. e ampl. São Paulo: Novatec Editora, 2010.

SPOJ BRASIL. “*Sphere Online Judge*”. Disponível em: <<http://br.spoj.com/>>. Acesso em: 19 mai. 2014.

STADZISZ, P. C. Projeto de Software usando a UML, Versão 2002. Centro Federal de Educação Tecnológica do Paraná. Departamento Acadêmico de Informática. Paraná, 2002.

UML. The *Unified Modeling Language*. Disponível em: <<http://www.uml.org>> Acesso em: Set. 2014.

UNIVERSIDAD DE VALLADOLID, “UVA Online Judge”. Disponível em: <<http://uva.onlinejudge.org/>> Acesso em: 19 mai. 2014.

URI ONLINE JUDGE. Disponível em: <<https://www.urionlinejudge.com.br/judge/pt>>. Acesso em: 17 jun. 2014.

WESLEY, H.; GONDIM, A. P.; AMBROSIO, A. P. Esboços de fluxogramas no ensino de algoritmos. In: Workshop sobre Educação em Computação – WEI. Anais do XXVIII Congresso da Sociedade Brasileira de Computação. Belém do Pará, PA, Brasil, 2008.

APÊNDICES

CONFIGURAÇÃO DO SAFEEXEC.C

Setar permissões para as pastas "/upload".
\$ chmod 777 /problema/upload

Para a correção dos códigos funcionar de forma correta e segura é preciso copiar o arquivo "safeexec" para "/usr/bin/safeexec" e definir suas permissões.

Este procedimento é feito usando os seguintes comandos, como root:
\$ cd /var/htdocs/www/doc/tools/
\$ cp safeexec /usr/bin
\$ cd /usr/bin
\$ chown root.root safeexec
\$ chmod 4555 safeexec

INSTRUÇÕES PARA CONFIGURAÇÃO DO PDODataBase.php

Includes/PDODatabase.class.php

Este arquivo é essencial para o funcionamento do sistema.

```
class PDODataBase {  
  
    public $conn = null;  
  
    function conecta() {  
        try {  
            $this->conn = new  
PDO('mysql:host=$host;port=3306;dbname=$db', '$login', '$pass');  
        }  
        catch(PDOException $e) {  
            print "Erro de Conexão!: ".$e->getMessage()."\n";  
        }  
    }  
  
    function desconecta() {  
        try {  
            $this->conn = null;  
        }  
        catch (PDOException $e) {  
            print "Erro! Por Favor contacte o suporte.".$e->getMessage()."\n";  
        }  
    }  
}
```

Ele possui dois métodos: conecta() e desconecta();

No método é conecta() é preciso definir parâmetros que deveram ser modificados, são eles:

\$host = Endereço do seu banco de dados
\$login = Login do banco de dados
\$pass = Senha do banco de dados
\$db = Nome do banco de dados