CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS TIMÓTEO

Otávio Cesário Oliveira

AMBIENTE COLABORATIVO PARA ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO DE COMPUTADORES E TREINAMENTO PARA COMPETIÇÕES

Timóteo

2017

Otávio Cesário Oliveira

AMBIENTE COLABORATIVO PARA ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO DE COMPUTADORES E TREINAMENTO PARA COMPETIÇÕES

Monografia apresentada à Coordenação de Engenharia de Computação do Campus Timóteo do Centro Federal de Educação Tecnológica de Minas Gerais para obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Aléssio Miranda Júnior

Timóteo

2017

Otávio Cesário Oliveira

AMBIENTE COLABORATIVO PARA ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO DE COMPUTADORES E TREINAMENTO PARA COMPETIÇÕES

Monografia apresentada à Coordenação de Engenharia de Computação do Campus Timóteo do Centro Federal de Educação Tecnológica de Minas Gerais para obtenção do grau de Bacharel em Engenharia de Computação.

Trabalho aprovado. Timóteo, 05 de dezembro de 2017:

Aléssio Miranda Júnior
(Orientador)

Douglas Nunes de Oliveira
(Banca Examinadora)

Odilon Correa da Silva
(Banca Examinadora)

Timóteo
2017

Agradecimentos

Agradeço a Deus, aos meus ancestrais, à minha família, aos meus pais, professores, amigos e colegas.

“[...] há certas e grandes razões de sabedoria ou congruência,
desconhecidas dos mortais mas assentes na ordem geral,
cujo fim é a maior perfeição do universo, e observadas por Deus”.
Gottfried Wilhelm Leibniz

Resumo

O ensino de lógica computacional em cursos de vários níveis de ensino, tem sido um desafio para os professores desta área, estes experienciam a realidade de um alto número de evasão nos cursos e a falta de motivação dos alunos. A evolução de ferramentas de educação à distância especializados tem sido um fator bem positivo para se adaptar à algumas necessidades desta nova geração de alunos. Dentre as ferramentas utilizadas no ensino de programação de computadores, o uso de ambientes automatizados de correção e auxílio especializado ainda está no início. Esses sistemas são fomentados principalmente pelo aumento da qualidade e número de ambientes de sistemas de avaliação online com repositório de problemas de competições de programação, que por sua vez podem ser adaptados ao ensino e aprendizagem dos estudantes. Entendendo que estes ambientes disponíveis ainda trabalham com um foco em treinamento para competições, apresentamos uma proposta modular de ambiente colaborativo personalizável e adaptável para professores e alunos, com o objetivo de melhorar os problemas e gerir a qualidade dos alunos com foco no aprendizado e acompanhamento. Desenvolveu-se neste trabalho uma API de integração com o avaliador automático de soluções, o Domjudge. Sugeriu-se também as primeiras taxonomias para problemas. Como resultado, tem-se uma implementação capaz de receber cadastros de problemas, submissões de solução e acompanhamento de resultados.

Palavras-chave: Sistemas colaborativos. E-Learning. Sistemas de avaliação online. Arquiteturas da Web.

Abstract

The teaching of computational logic on courses of many range of difficult, have been a challenge for professors of this area, they experience a reality with a high number of evasion in these courses and a lack of motivation of these students. The evolution of tools specialized in education on distance, have been a positive factor, adapting to needs of this new generation of students. One of this tools to teach computer programming, are automated correction environment, which are still developing, being fomented mainly by the increasing of quality and number of automated correction environment with competition repository problems, which in turn, may be adapted to the teaching and learning of students. Understanding that these environments are still working with focus in training to competitions, we present a modular proposal of a customized and adaptable collaborative environment to teachers and students, with the goal to increase the quality of problems and students with the objective in the learning and supervision. We developed in this work a API of integration with the automated solution evaluator, Domjudge. Was suggest too, the first taxonomies to problems. As a result, we have a implementation able to receive register of problems, submission of solutions and supervising of solutions results.

Keywords: Collaborative Learning, E-Learning, Automated Judge Systems, Web Architecture.

Lista de ilustrações

Lista de abreviaturas e siglas

TIC	Tecnologia da Informação e Comunicação
API	Interface de Programação de Aplicação
RPT	Repositórios de Problemas para Treinamento
CETECOP	Collaborative Environment for Teaching Computer Programing

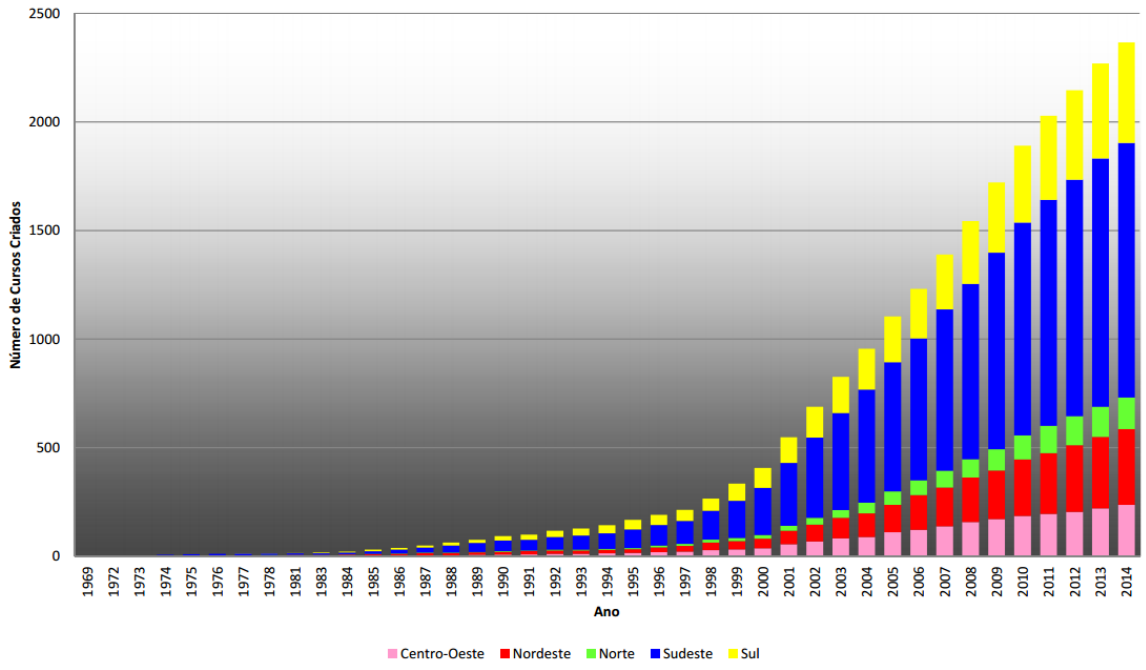
Sumário

1 Introdução

“O mal é o bem esperando para crescer”.
Autor desconhecido

Segundo o relatório[1] Brasscom (2014), no período de 2007 a 2009 houve cerca de 174 mil matrículas em cursos de Tecnologia da Informação e Comunicação (TIC), sendo que apenas 23 mil o concluíram neste mesmo período. Um dos motivos dessa diferença está no aumento dos cursos de graduação (FIG. 1) e também no aumento da evasão nesses cursos. Um dos motivos que levam a evasão é que o estudo desta área de conhecimento parece difícil e complexo (MCGETTRICK et al., 2005). Além disso, Brasscom (2014) também aponta a relação da demanda de profissionais de TIC em relação aos concluintes para o ano de 2014, sendo 78 mil vagas ofertadas no mercado para apenas 33 mil concluintes, portanto há um deficit de profissionais.

Figura 1 – Crescimento de cursos de TIC



Fonte: Nunes (2014)

Em uma análise realizada na Universidade Federal de Pernambuco (UFPB) por Duarte (2013b), a taxa de aprovação das disciplinas básicas de programação como: Estrutura de dados; Introdução a Programação; Linguagem de Programação 2, foi de 52%, 56% e 60% respectivamente, um valor não muito satisfatório. Nesse mesmo estudo, foi constatado um crescimento do número de matrículas nas disciplinas e uma diminuição na taxa de aprovação das mesmas. Além disso, foi constatado que: “Um a cada cinco alunos matriculados em

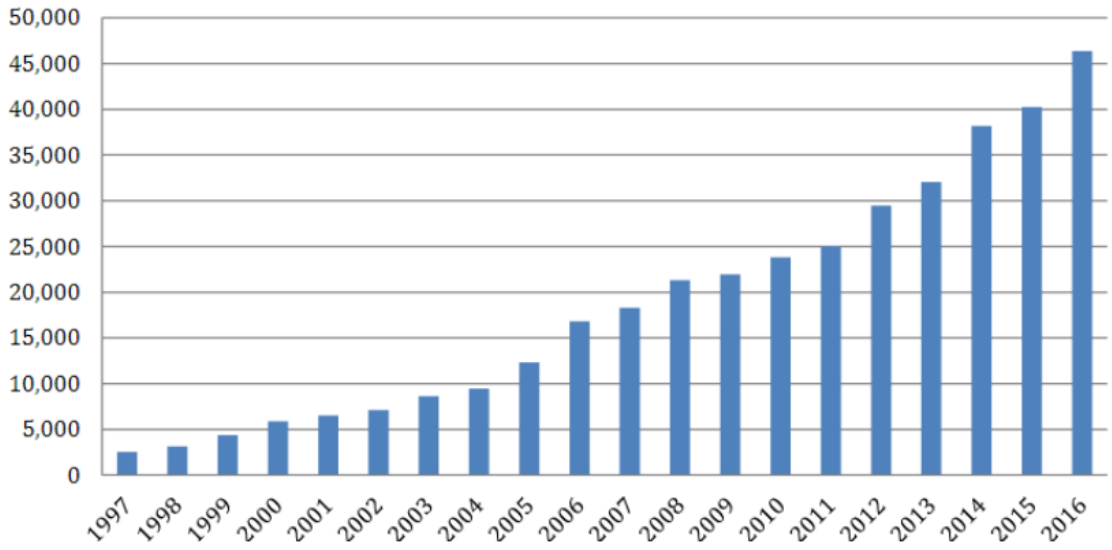
[1] Dados relativos à média de 8 estados brasileiros, sendo eles: BA, MG, RJ. PR, RS, DF, PE e SP.

uma disciplina do curso de Ciência da Computação da UFPB é reprovado por falta, o que geralmente ocorre por conta do abandono da disciplina.” (DUARTE, 2013a).

Existem vários fatores que podem levar ao abandono do curso de computação (DETERS et al., 2008), mas para fins práticos este trabalho concentra-se em apenas um, que é a falta de uma base bem desenvolvida de raciocínio lógico-matemático (RODRIGUES, 2013; PRIETCH; PAZETO, 2010). No estudo realizado por Rodrigues (2013) na Universidade do Rio Grande do Sul (UFRGS), 54% dos entrevistados considera bastante elevado o nível de dificuldade do curso - 8 ou mais em uma escala de 1 a 10 - sendo que 66% desses alunos estão insatisfeitos com seus próprios desempenhos e que 79% tem previsão de formar além do tempo ideal, o que implica em um desperdício de recursos da universidade. Já no estudo realizado por Prietch e Pazeto (2010), no primeiro ano do curso de licenciatura em informática da Universidade Federal do Mato Grosso (UFMT), quase metade da turma é reprovada na disciplina de Programação I, apesar de ser um caso isolado em uma instituição, é razoável pensar que isso também acontece em outros lugares. Segundo Deters et al. (2008), as disciplinas de “Algoritmos” e “Programação” possuem um dos maiores índices de reprovação nos cursos de tecnologia.

Ainda neste contexto, competições como maratona de programação tem aumentado o interesse dos alunos por um ambiente dinâmico de competição. Maratona de programação é um evento onde os participantes se reúnem em equipes para resolver problemas de programação dos mais diversos tipos. Esse evento envolve os participantes em uma competição que desenvolve o trabalho em equipe, habilidades de programação e o domínio de algoritmos (POUCHER, 2012).

Figura 2 – Crescimento da participação de estudantes no concurso ICPC



Fonte: ICPC (2017)

Exemplos de competições são o International Collegiate Programming Contest (ICPC), evento promovido pela ACM e a Maratona de Programação promovida pela Sociedade Bra-

sileira de Computação (SBC). O ICPC apresenta crescimento ao longo dos anos desde sua criação, o que pode ser visto na Figura 2.

Não há dúvida que competições acadêmicas podem servir como forte motivação para estudantes, provendo um incentivo para os seus estudos (OZTURK; DEBELAK, 2008). Essa motivação pode ser porque maratonas são divertidas ou por causa das viagens nacionais ou internacionais, que muitas das vezes são completamente subsidiadas pelo evento (TROTMAN; HANDLEY, 2008). Além disso, competições como o ICPC, oferecem prêmios em dinheiro para os vencedores e estágios ou oportunidades de trabalho para os medalhistas de ouro (GUSTO-KASHIN; ZAVARIN, 2017). Por outro lado, não estar preparado e não ser bem preparado pode desestimular os estudantes a participarem desse tipo de evento, que por sua vez remove os benefícios acima citados.

Com o interesse em treinamento e capacitação, surgiram ferramentas online que buscam auxiliar os estudantes a se preparar para competições. Estes sistemas apresentam como principais funcionalidades serem repositórios de problemas a serem resolvidos e terem mecanismos online automatizados para correção de soluções para estes problemas baseado em casos de testes.

Existem vários sistemas derivados do ambiente de maratona de programação utilizados para treinamento de competições como o SPOJ (2017), UVA (2017) e o URI (2017). O Uri apresenta funcionalidade interessante para o ensino, com interface para professores e alunos, e a maioria cumpre apenas sua função de treinamento de competições e nenhum deles prezam a colaboração e foco no ensino e classificação de problemas. Além disso, esses são sistemas fechados, o que impede a adição de funcionalidades, personalizações e possíveis melhorias pela comunidade.

1.1 Problema

Ambientes colaborativos tem se tornado importante devido a capacidade de prover o compartilhamento de informações, interação entre pessoas distantes e também na construção de ambientes de aprendizagem (BRITO; PEREIRA, 2004). Estes sistemas ao estilo Wikipédia e SourceForge criaram na web repositórios crescentes e confiáveis de conteúdo, uma demonstração que com as ferramentas corretas, é possível promover de forma coordenada o desenvolvimento de novos conhecimentos pela comunidade (SCHMITT, 2006).

Pensando em novas formas de aumentar o interesse na computação, aumenta-se o entendimento de que ferramentas colaborativas possam por exemplo melhorar a formulação de enunciados de problemas tornando os cada vez mais interessante aos alunos (GOMES; HENRIQUES; MENDES, 2008). Reformular os enunciados já existentes, ilustrações e formulação de casos de teste, de forma a criar um ambiente mais interessante para a aprendizagem, o que é uma alternativa a simplesmente criar novos problemas.

Embora existam repositórios de problemas para resolução, um professor que deseja buscar, aplicar ou adaptar os problemas às disciplinas não tem informações suficientes para fazê-lo com facilidade, nem mesmo contribuir para o enriquecimento dos mesmos, pois como

já mencionado, esses são sistemas fechados não focados na colaboração.

1.2 Objetivos

O objetivo geral deste trabalho é modelar uma arquitetura conceitual colaborativa, modular e baseado em reuso. É proposto que esta arquitetura faça a gestão de um repositório online de problemas, testes e soluções classificados e flexíveis baseado em conteúdos e taxonomias da comunidade, permitindo a submissão de soluções em um ambiente classificatório avaliativo.

Também, objetivam-se mais especificamente:

1. Permitir a integração do sistema com softwares avaliadores automáticos de problemas, permitindo a submissão e avaliações de soluções para os problemas disponíveis;
2. Sugerir e criar os primeiros critérios de ranqueamento de soluções;
3. Sugerir formas iniciais taxonômicas para classificação dos problemas e testes;
4. Sugerir ferramentas de avaliação das submissões baseado em ranqueamento;

1.3 Justificativa

Se por um lado competições melhoram habilidades de programação e motivam os alunos a estudarem por si só, a falta de uma base inicial de aprendizado pode comprometer essa motivação. Um sistema semelhante àquele que é utilizado em maratonas de programação, que auxilie no aprendizado nos períodos iniciais do curso pode ser usado para contribuir com a motivação dos estudantes a participarem de competições e por consequência um aprofundamento e continuidade por parte deles nos seus estudos.

Temos também que, na educação, uma das preocupações é justamente a forma de apresentação de objetos de aprendizagem, estes devem despertar a curiosidade do estudante sendo fundamentais para a motivação e interesse com o objetivo de aumentar o nível de absorção de conteúdo e consequentemente diminuir a reprovação em disciplinas.

Além disso, Moreira e Favero (2009) relatam a dificuldade de um professor avaliar exercícios de alunos em um curto período de tempo, isso pode ser contornado com uma ferramenta de avaliação automática onde se usa casos de testes predefinidos, melhorando o tempo de *feedback* com os alunos, que é fundamental para que o aluno consiga aprender (FRANCISCO, 2016).

Tem-se como hipótese que o desenvolvimento de um sistema colaborativo com foco na gestão e organização de problemas computacionais, suas soluções, casos de testes e uma classificação a partir de diversos critérios é um trabalho a ser desenvolvido e traria benefícios para o ensino de computação e à comunidade.

1.4 Organização do Trabalho

No Capítulo 1 temos a contextualização do problema, os objetivos, justificativa e resultados esperados. Apresenta-se no Capítulo 2 sistemas de autojulgamento com repositórios de problemas. Tem-se no Capítulo 3 a apresentação de trabalhos relacionados a sistemas de avaliação automática. É apresentado no Capítulo 4 cinco sistemas autojulgadores encontrados no estado da arte. O Capítulo 5 apresenta o desenvolvimento do trabalho. Por fim, no capítulo 6 apresenta-se as conclusões.

2 Juízes Online com Repositório de Problemas

“Como a perfeição de Deus sempre aumenta,
então todo mal tem um potencial latente de se tornar um bem”.
Gottfried Wilhelm Leibniz, adaptado

Há uma crescente demanda por ferramentas que ajudem no processo de aprendizagem nas mais diversas áreas de conhecimento, estudantes têm cada vez mais dificuldade com o tradicional estilo de ensino baseado na exposição de conteúdos em lousa ou slides (BEZ; FERREIRA; TONIN, 2013). Nesse caso, repositórios de problemas com um sistema de autojulgamento embutido se apresenta como uma dessas ferramentas que tem potencial de serem utilizadas no processo de aprendizagem, entretanto não são completas, tendo limitações. Este capítulo discorre sobre algumas dessas ferramentas bem como suas características.

2.1 Sistemas de autojulgamento

Podemos entender como Sistemas de Autojulgamento ou Juiz Online ou Sistema de Avaliação Automática, aplicações com o objetivo de gerir uma competição temporal, onde desafiantes tem um tempo determinado para resolver problemas computáveis. São submetidos códigos fonte, sendo estes corrigidos por um processo automatizado que consiste em compilação, execução e teste. O sistema em questão dever ter como característica o controle de submissões corretas e erradas a fim de pontuar e classificar usuários sobre um ranking. São sistemas utilizados em competições locais ou distribuídas como uma forma de coordenar eventos, pontuações e auxílio na avaliação (CAMPOS; FERREIRA, 2004).

Vamos apresentar a seguir os Juízes Online BOCA (BOCA, 2017) e DOMjudge (DOMJUDGE, 2017), ambos são software livre para autojulgamento e cumprem dentro de suas características o papel que lhe foram propostos. São também uma possibilidade software livre para adaptação em universidades, mas são restritos com relação à proposta deste trabalho onde se busca uma gestão de conhecimento, onde então o juiz online teria a tarefa de executar um conjunto de tarefas e dar um *feedback* adequado ao usuário.

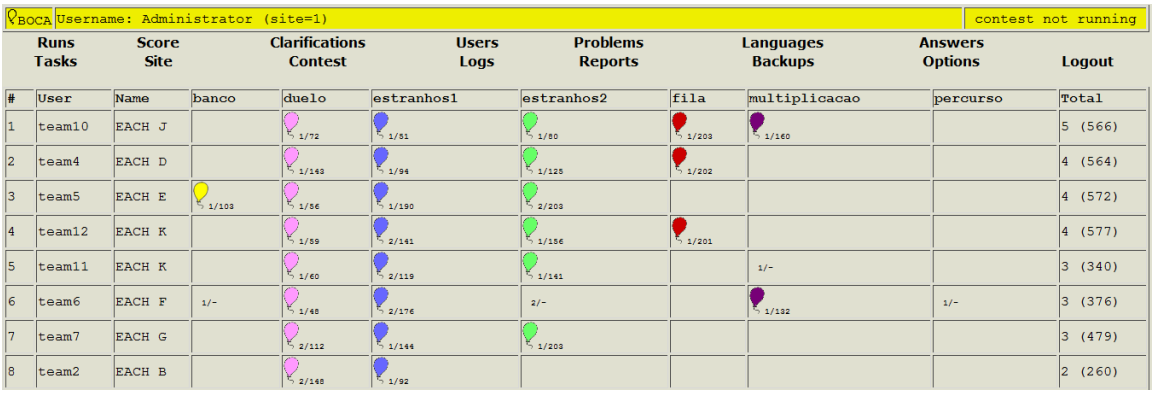
2.1.1 Juiz Online BOCA

O Brasil participa do evento de programação do ICPC desde 1996, com o crescente número de participantes nas fases regionais, foi desenvolvido em 2002 o sistema Boca para auxiliar no julgamento das competições.

Neste sistema a equipe pode submeter a solução do problema e estas soluções são comparadas com casos de testes, arquivos com entradas de dados a serem lidos pelo algoritmo feito pela equipe e os arquivos de saídas, para comparar com as saídas do programa. No

final dessa comparação, o sistema gera uma resposta dizendo se a solução está correta, senão é informado o erro, como erro de compilação, saída mal formatada ou tempo de execução excedido (CAMPOS; FERREIRA, 2004).

Figura 3 – Visualização do placar geral de uma competição



Fonte: SELETIVA de SI para a Maratona de Programação (2013)

Neste trabalho foi usado a versão 1.5.8 e de acordo com o repositório do código fonte é desenvolvido por uma comunidade de poucos desenvolvedores e com um volume baixo de atualizações, além de pouca documentação para ampliação de suas funcionalidades.

Apesar de um volume baixo de atualizações, apresenta de forma especialista todas as características relevantes para o seu objetivo, justificando seu uso em competições e treinamentos oficiais, como: cadastro de problemas; ajustes de características para cada caso de teste; compatível com linguagens de programação diferentes; configuração de diferentes papeis de usuário; segurança com relação às soluções e feedback adequado tanto ao usuário que submete uma solução durante um evento e os juízes, como por exemplo o placar da competição, visto na Figura 3.

Uma limitação importante e não adequada aos propósitos deste trabalho é que uma mesma instância instalada não gerencia eventos (contests) simultâneos, e cada evento não pode ultrapassar a barreira da mudança de um dia para o outro (Francisco, 2016). A criação de instancias independentes requer mais ainda um repositório de problemas desacoplado do BOCA, pois instancias diferentes não compartilham seus problemas. Deve ser investigado no futuro, o impacto que a necessidade de criar instâncias tem no aumento da complexidade de se usar o BOCA como um módulo no sistema que propomos.

2.1.2 Juiz Online DOMjudge

O DomJudge, ilustrado parcialmente na Figura 4, é um sistema de autojulgamento para maratonas de programação desenvolvido pela Study Association A-Eskwadraat na Universidade Utrecht. É um projeto de código fonte aberto iniciado em 2004 para sub-regionais do ICPC, sendo utilizado a partir de 2012 nas finais desse mesmo evento de competição (DOMJUDGE, 2017). Assim como o BOCA, seu objetivo se restringe a ser um sistema de autojulgamento, entretanto apresenta uma comunidade de colaboradores maior e atualizações

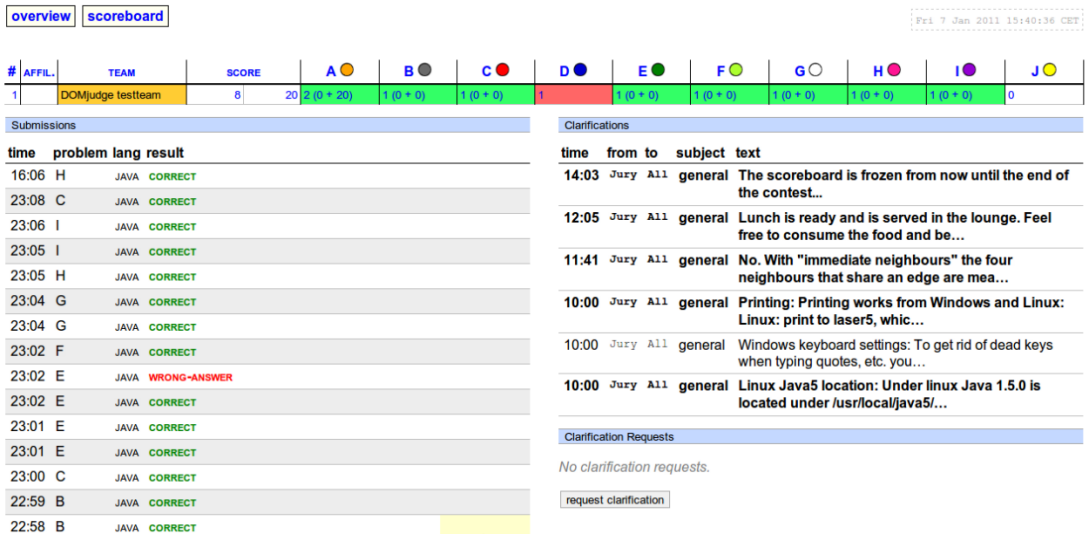
mais constantes.

Faremos agora um pequeno relato comparativo com o BOCA sobre a ótica dos objetivos deste trabalho, sendo que um diferencial direto e importante do DOMjudge é já permitir a gerência e execução de vários contests simultâneos com duração a escolha do administrador. Este diferencial acompanha uma série de melhorias como uma interface mais adequada para acompanhamento das equipes que podem ser divididas em campeonatos, uma arquitetura escalável e parcialmente modularizada (o módulo que compila e executa as submissões das soluções, o Judgehost, pode ser executado em várias máquinas distintas).

Uma desvantagem considerável é a menor quantidade de parâmetros para execução individual dos limites de execução dos casos de testes, isto reduz o controle de um possível professor ou juiz a um ajuste fino da execução das soluções.

Na arquitetura (CETECOP) que será proposta seria desejável um módulo que tivesse capacidades de autojulgamento. O DOMjudge se apresenta como um candidato interessante a compor uma parte da solução. Notamos que ele fornece API's (*RESTful*) de comunicação adequadas para que ele possa ser adaptado para ocupar as lacunas do Contest Manager e Judge System (detalhados no Capítulo 6) o que pode gerar uma economia na escrita de código quando precisarmos usar suas funcionalidades.

Figura 4 – Visualização da pontuação por uma equipe



Fonte: DOMjudge (2017)

2.2 Repositório de problemas para treinamento

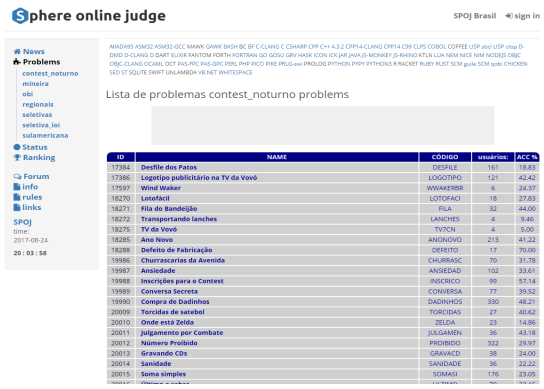
Repositórios de Problemas para Treinamento (RPT) são sistemas que concentram e organizam problemas da comunidade de forma a facilitar o uso por alunos que desejam treinar para competição. Apesar desta definição restrita, todos os mais famosos RPT's que serão citados incluem funções adicionais como integração com algum tipo de Sistema de Autojulgamento, entretanto, como são softwares de código fechado não é possível inferir qual a tecno-

logia de integração utilizada. Devido a isso, não é possível discutir este ponto, justificando o capítulo acima sobre sistemas de autojulgamento como BOCA e DOMjudge.

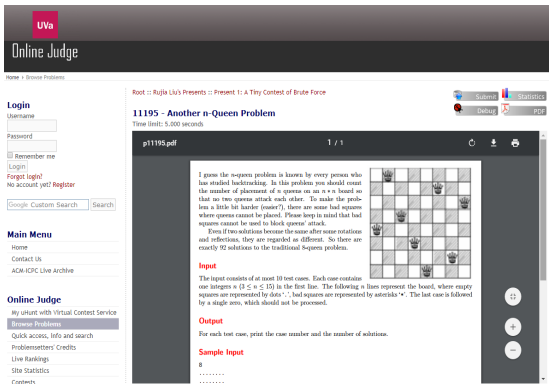
2.2.1 SPOJ e UVa

SPOJ (Figura 5a) e Uva (Figura 5b) são sistemas com funcionalidades muito parecidas, o primeiro é coordenado pelo Instituto de Matemática e Estatística da USP e possui problemas apenas no idioma português, já o segundo é da Universidade de Valladolid da Espanha com problemas principalmente em Inglês e Espanhol. Ambos possuem rank de usuários, mas sem subdivisões como por exemplo rank de universidades, turmas e áreas de conhecimento. Uma diferença relevante é que o SPOJ classifica seus problemas principalmente por eventos de programação (regionais, sul-americana, OBI, mineira) e o UVa tem classificações mais completas, como por autores e livros de maratona de programação.

Figura 5 – Exemplo de uma tela dos sistemas SPOJ e UVa



(a) Tela do SPOJ com uma relação de problemas



(b) Tela do UVa mostrando de um problema

Fonte: SPOJ (2017)

Fonte: UVA (2017)

O SPOJ permite uma atuação da comunidade em um formato próximo a um fórum de discussão sobre problemas e soluções, além da classificação de problemas usando tags (tópicos). Estas funcionalidades apresentam traços colaborativos importantes para a solução, o que é desejável em uma arquitetura de gestão de problemas.

2.2.2 URI

Dentre os citados o URI é o mais recente, criado em 2012 pela Universidade Regional Integrada. Possui um repositório de problemas baseado em oito categorias (FIG. 6) e níveis de dificuldade. e um módulo (URI Acadêmico) exclusivo para professores (BEZ; FERREIRA; TONIN, 2013).

Sua interface apresenta critérios de usabilidade mais adequados para um sistema de acompanhamento e sua proposta já prevê uma interface multilíngue sendo que inclusive os problemas também possuem sua versão em português e inglês, podendo ser adotado por comunidades estrangeiras.

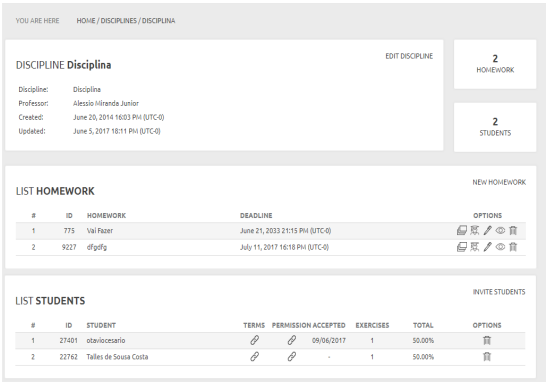
Figura 6 – Categorias do URI Judge

Fonte: URI (2017)

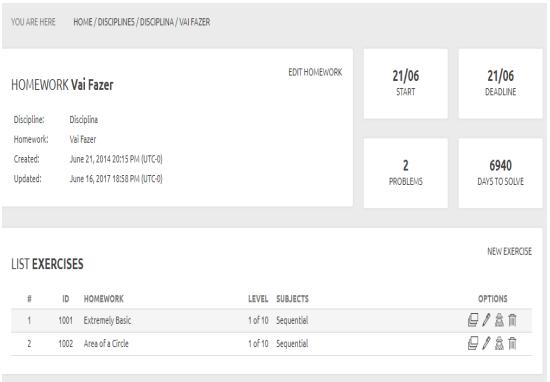
Do ponto de vista de busca, classificação e organização, tem uma diversidade de critérios que atendem os usuários na busca de problemas por nível de dificuldade e classificação por tipos de problemas em áreas da computação. Um ponto a ser questionado é que toda esta organização de conhecimento é feita por uma equipe interna, reduzindo a colaboração orgânica da comunidade e adição de novos problemas.

O módulo acadêmico é onde o professor ou treinador pode acompanhar seus alunos com funcionalidades interessantes e desejáveis como: visualizar o código fonte dos alunos, analisar possíveis plágios com um sistema identificador automático, verificar o ranqueamento de problemas por tempo de execução, comparar alunos com universidades e outros usuários; visualização da porcentagem de casos de testes falhos e o histórico de submissões dos alunos (BEZ; FERREIRA; TONIN, 2013; SELIVON; BEZ; TONIN, 2015). O acompanhamento do progresso dos alunos é feito através de uma interface de criação de disciplinas através de convites e a criação de listas de exercícios com tempo resolução definido.

Figura 7 – Exemplo de uma tela do módulo URI Acadêmico



(a) Tela de configuração de uma disciplina



(b) Tela de uma tarefa e seus problemas

Fonte: URI (2017)

Na Figura 7a temos a tela do URI Acadêmico relativo a uma disciplina, bem como a listas de tarefas e a lista de estudantes. A Figura 7b mostra uma tarefa especifica de uma disciplina, e nela temos os exercícios ou problemas a serem solucionados pelos estudantes.

Apesar de funcionalidades bem desenhadas para apoiar os professores e treinadores no ensino através do módulo acadêmico e criar formas de guiar o aluno a se aprimorar, a solução URI não leva em consideração critérios colaborativos e de personalização. Além disso, por ser um software fechado e não ter uma API de integração, o URI torna seu uso pouco adaptável em vários requisitos.

Existe também dificuldades em traduzir um problema existente para outra língua, sugerir novos problemas, novas temáticas, controlar os casos de testes para uma determinada disciplina ou requisito, discutir de forma separada tópicos entre alunos e professores, criar novas classificações de problemas, etc.

3 Trabalhos sobre Juízes Online

“Se cair sete vezes, levante oito”.
Monja Coen

No trabalho realizado por Alonso e Miranda (2014), é proposto a modelagem e implementação de módulos para treinamento de maratona de programação, um destes módulos é uma ferramenta de autojulgamento. Apesar desse autor estudar ferramentas existentes que fazem isso, como o Domjudge e outros, ele afirma que seria custoso recodificar essas ferramentas, pensamento contrário ao adotado nesta monografia, que aproveita-se uma ferramenta de autojulgamento existente para evitar desperdício de energia construindo algo que já existe.

O autojulgador desenvolvido por Alonso e Miranda (2014) contempla apenas as linguagens Java, C e C++, mas possui um mecanismo de segurança desenvolvido para evitar ciclos infinitos nas correções automáticas. Esse sistema também permite a criação de problemas que são corrigidos de forma manual para o professor, o compartilhamento de trechos de código para todos os usuários e o acesso à problemas das finais do ICPC. A comunicação se dá por correio eletrônico quando um problema é solucionado pelo usuário, quando o treinador corrige o mesmo e também quando este publica uma nova informação no sistema.

Na dissertação de Francisco (2016), foi feito um levantamento bibliográfico sobre características, funcionalidades e exemplos de ferramentas de autoavaliação, sobre as formas de classificar a dificuldade de um problema e sobre softwares de detecção de plágio em algoritmos. Para medir a dificuldade foi proposto um algoritmo que mede a altura de uma árvore formada a partir do conjunto e subconjuntos de códigos aninhados (como while, if, etc.) do programa, quanto maior a árvore maior a dificuldade. Além da árvore para representar o algoritmo foi utilizado grafos. Outra forma de classificar a dificuldade foi por assuntos como estruturas de repetição, matriz, string, etc. Os experimentos realizados foram na disciplina introdutória de programação, de forma a não abordar outros assuntos mais complexos. Para detecção de plágio foi proposto o algoritmo Distância de Edição desenvolvido para analisar algoritmos escritos em C.

Francisco (2016) utilizou o sistema Boca para gerenciar exercícios nas turmas e a partir das soluções feitas pelos alunos foi aplicado o algoritmo detector de plágio. O resultado não é exato e sim baseado em probabilidade, o que muitas vezes fica para o professor identificar manualmente se ocorreu plágio ou não. Entretanto foi constatado uma tendência de alunos identificados com alto índice de plágio serem reprovados.

A dissertação de Xavier (2011) envolve a continuação do trabalho de mestrado de Pacheco (2010), um sistema de autojulgamento. Além de discorrer sobre o sistema já implementado, comparando com avaliadores automáticos existentes, é identificado as limitações do mesmo. Umas dessas limitações, ou funcionalidades não contempladas, é a análise estática de código e detecção de plágio. Para isso Xavier (2011) relata o estado da arte desses dois

tópicos e propõe a implementação do primeiro usando diversas métricas, e o segundo é utilizado o Crot, uma API (Application Programming Interface) do moodle para detecção de plágio. Foi desenvolvido também uma API, que permite a adição de outros detectores de plágio

O sistema de Pacheco (2010) foi feito usando a Plataforma Moodle juntamente com o avaliador Domjudge. Xavier (2011) adicionou a linguagem SQL ao DOMjudge, mostrando a capacidade deste de aceitar novas linguagens além daquelas que ele vem configurado por padrão. Depois de testar os módulos implementados, Xavier (2011) aponta como trabalhos futuros, a criação de novas métricas e novas linguagens contempladas pelo analisador estático de código, e o melhoramento da forma como o *feedback* é dado aos estudantes, como o uso de notificações.

A dissertação de Rocha (2011) propõe um avaliador automático para a disciplina de computação gráfica. A arquitetura proposta permite um professor criar um problema com uma estrutura base, tipo um modelo a ser seguido, para os alunos usarem na solução. A correção da solução aceita programas criados no Visual Studio usando o OpenGL. Quando o aluno submete o exercício feito, o sistema compila, executa e tira *printscreen* do cenário criado, a partir daí o professor pode dar a sua avaliação, portanto o processo automatiza a execução dos exercícios, e não uma avaliação com casos pré-definidos.

4 Materiais e Método de Pequisa

“O mesmo chão que você cai é
aquele que te da suporte para levantar”.
Monja Coen

Segundo Wazlawick (2009) esta pesquisa é exploratória, e segundo Ander-Egg (1978 apud LAKATOS; MARCONI, 1995) ela se classifica como aplicada. É classificada como uma pesquisa exploratória pois, a partir de um problema abstrato, ainda se procura passos e formas para a criação de um método eficaz para resolução do mesmo. A seguir é apresentado os passos necessários para a realização deste trabalho:

1. Analisar ferramentas de autojulgamento e repositórios de problemas em busca de características desejáveis e limitações, priorizando softwares livres;
2. Analisar e compreender arquitetura da ferramenta de autojulgamento escolhida para uso e adaptação;
3. Modelar o sistema proposto de forma a contemplar as funcionalidades requeridas;
4. Implementar um protótipo com as principais funcionalidades;

Para o desenvolvimento do protótipo é utilizado o JavaServer Faces (JSF) juntamente com o framework PrimeFaces. A plataforma de desenvolvimento é a IDE Eclipse, utilizando-se o Maven para gerenciar automaticamente as bibliotecas dependentes. O banco de dados utilizado é o MySql.

5 Desenvolvimento

“Os erros ficam no passado, tente de novo
e o acerto irá durar a eternidade”.
Autor desconhecido

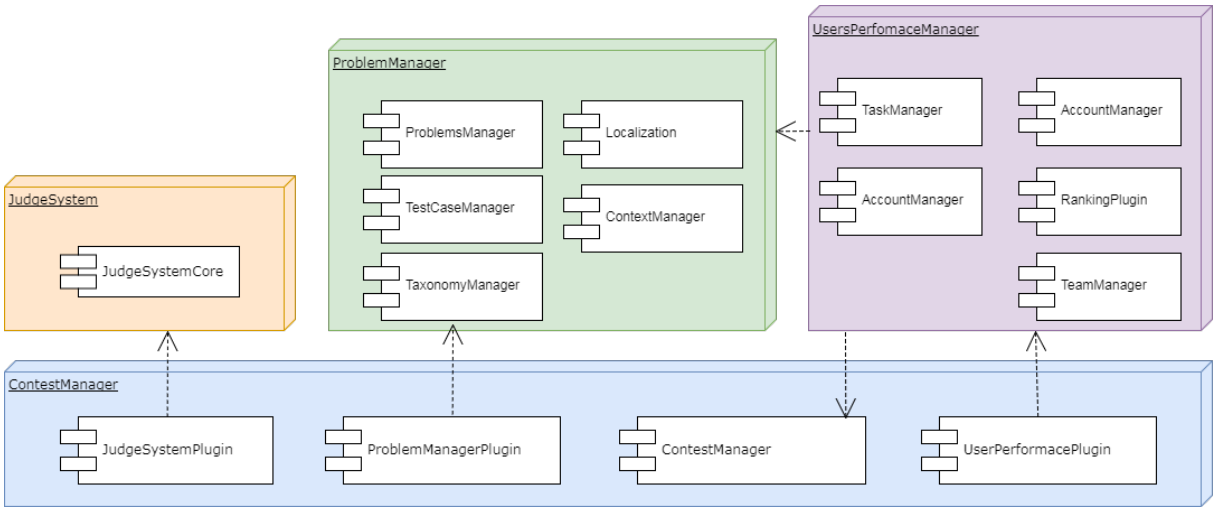
Este capítulo trata sobre o sistema colaborativo para ensinar programação de computadores e treinamento para competições. É também proposto taxonomias inicias para problemas, diagramas de modelagem e é apresentado um protótipo funcional como estudo de caso.

5.1 Arquitetura Conceitual (CETECOP)

A proposta da arquitetura colaborativa e conceitual do CETECOP (Collaborative Environment for Teaching Computer Programing), é ser uma plataforma web composta por módulos independentes, onde professores possam gerir conhecimento ligados à problemas e soluções computacionais e alunos possam de forma guiada ter seus desempenhos medidos em relação a ele mesmo e à diversas esferas de alunos pelo mundo.

Os módulos Judge System, Contest Manager, Problem Manager e User Performace Manager foram desenhados para funcionar de forma escalável e cada um tem um objetivo específico delimitado para que possa ser substituído ou evoluído isoladamente. A Figura 8 ilustra os quatro módulos sugeridos para compor a solução e seus componentes e a seguir é feito a descrição e algumas relações com os sistemas propostos.

Figura 8 – Modelo Conceitual CETECOP



Fonte: Elaborada pelo autor

5.1.1 Judge System

Esse é um módulo independente, responsável por receber uma solução e os dados de um problema, executar os procedimentos e testes para garantir se a submissão é correta ou rejeitar indicando possíveis erros (erro de compilação, erro de execução, tempo limite excedido e resposta errada).

Ele é composto por um único componente Judge System Core que é responsável por este objetivo e com os seguintes requisitos: ser escalável, receber soluções em quaisquer linguagens de programação; receber e executar todos os tipos de casos de testes baseado em entrada e saída de dados; aceitar ou rejeitar testes baseados em parâmetros individuais ou de conjunto como tempo de execução, uso de memória e tipos de compilação; e resposta com estatísticas abertas de todo o processo de teste e execução como memória utilizada, bibliotecas compiladas.

Não exatamente de forma modular, mas parte dos requisitos deste módulo estão incluído em todos os trabalhos citados anteriormente, muito embora somente no DOMjudge e Boca temos informações sobre sua implementação pois os outros são sistemas fechados.

5.1.2 Problem Manager

O Problem Manager tem como objetivo gerenciar de forma colaborativa o conhecimento ligado às problemáticas, suas soluções e casos de testes, de forma a ser uma base de dados viva. Dele são excluídas responsabilidades como testar soluções (Judge System) ou verificar contests.

É constituído de cinco componentes que são: Problem Manager (gestão de problemas); Test Case Manager (gestão dos casos de testes e suas particularidades); Taxonomy Manager (controle de categorias e suas métricas de atribuição); Context Manger (gerência de temáticas de enunciados e suas comunidades); e Localization (Cada idioma pode ter particularidades na organização de problemas).

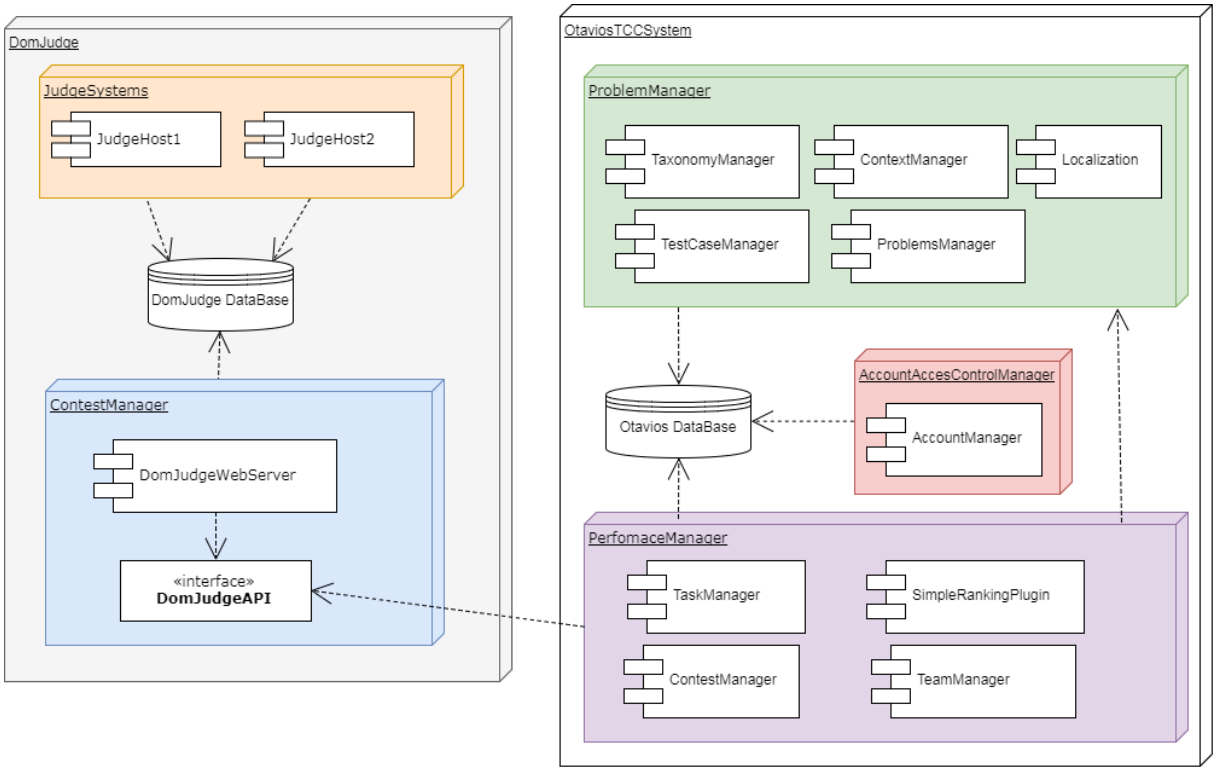
As principais funcionalidades apresentadas por este módulo são:

1. Problemas: cadastro; classificação;
2. Casos de testes: cadastro e atribuição com cada problema; parâmetros de execução; limites de tempo de execução; limites de memória; requisitos de linguagens e compilação; critérios de solução;

De alguma forma estas funcionalidades básicas estão disponíveis em praticamente todos os sistemas citados, muito embora no caso do URI, SPOJ e UVa o usuário tem barreiras para fazer o cadastro de forma simplificada.

Como características que vão além das básicas podemos citar: a internacionalização dos problemas (presente parcialmente no URI); a definição de enunciados temáticos, pois embora um determinado problema tenha um enunciado, é possível criar variações de forma a

Figura 9 – Diagrama de Implementação



Fonte: Elaborada pelo autor

atrair os alunos sem ao mesmo mudar os parâmetros de entrada e saída; definir colaborativamente classificações personalizadas e da comunidade de forma que os problemas possam ser agrupados sobre taxonomias em diferentes contextos, como assuntos em uma disciplina ou tópicos de algoritmos (parcialmente presente no SPOJ na forma de tags, e no URI não existe a colaboração, embora há sugestão de classificadores).

5.1.3 Contest Manager

Definiremos um contest como uma atividade avaliativa, uma lista de exercícios de disciplina e uma competição com tempo curto ou longo. Em qualquer destes contextos são nos contests que os usuários (alunos) interagem com o sistema submetendo soluções e alimentando-o com os dados de performance.

O Contest Manager é o módulo central que controla a execução de eventos simultâneos e faz integração com os outros 3 módulos: Judge System para solicitar a execução de correção de uma submissão e os dados de retorno; Problem Manager pois os contests são compostos por problemas e dependem das informações; e User Performace Manager para alimentar dados sobre as soluções que complementam o perfil do usuário.

5.1.4 User Perfomance Manager

O User Performace Manager concentra as métricas de um usuário, equipes ou turmas e é constituído por quatro componentes: Account Manager (gerenciador logins; dados

pessoais; papéis de responsabilidade entre professores e alunos; relacionamentos e suas estatísticas); Raking Plugin (interface para plugins acopláveis que apresentam novas métricas e critérios nas avaliações de comparação entre usuários ou times); Task Manager (controle responsável por informar os usuários por atividades previstas e executadas; informações de comunicação do sistema e fluxos de controle); Team Manager (gestão dos tipos de times, turmas e colaborações sobre diversas óticas de controle como indicativos de características e perfis).

Este módulo tem comunicação com o Problem Manager para obter informações sobre problemas e com o Contest Manger para obter informações sobre as tarefas executadas. Esse sistema também deve comunicar com a API do gerenciador de competição, de forma a poder criar *contests* customizados, como por exemplo em uma lista de exercícios, poder atribuir pesos diferentes a cada problema.

Dentre as ferramentas analisadas, o módulo acadêmico do URI apresenta uma parte restrita das funcionalidades previstas, o painel não possui parametrização ou personalização.

Dado o modelo conceitual acima e o estudo dos sistemas de autoavaliação, faz-se a proposta da seguinte implementação mostrada na Figura 9. Foi escolhido o sistema Domjudge em vez do Boca pois ele atende os requisitos de um sistema autojulgador e se apresenta como uma alternativa de fácil implantação .

5.2 Taxonomias iniciais para problemas

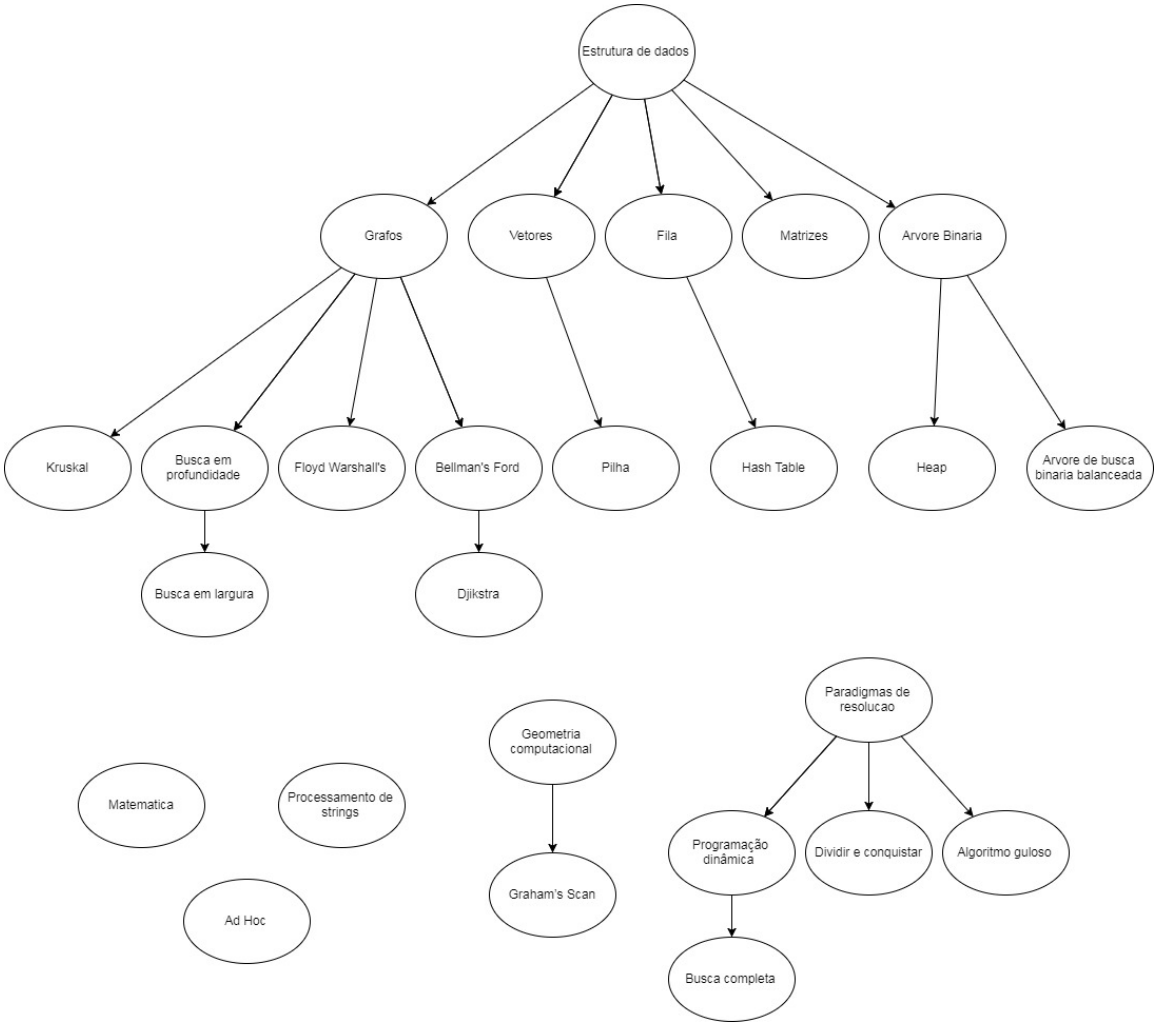
Taxonomias é uma forma de classificar objetos em categorias. Em um contexto de programação, os problemas podem ser classificados por técnicas, por exemplo. Baseado no livro Competitive Programming de Steven e Felix (2010), propomos as categorias presentes na Figura 10, que por coincidência, as categorias gerais são iguais a do URI, entretanto o URI não possui subcategorias. Como pode haver a necessidade de novas classificações, haverá no sistema a possibilidade de novos cadastros de taxonomia por parte do ator professor no sistema.

5.3 API de integração com o Domjudge

Para a integração com qualquer sistema de avaliação automática, existe funções e comportamentos comuns, os quais podem ser generalizados através de uma API (Interface de Programação de Aplicação). Neste trabalho é proposto alguns desses métodos através da interface Automated Judge Interface, a saber:

- void cadastrarEvento(Evento evento);
- void cadastrarProblema(Versao versao, Evento evento);
- void cadastrarEquipe(Equipe equipe);
- void cadastrarCasosDeTeste(List<CasoDeTeste> casosDeTeste);

Figura 10 – Taxonomias iniciais



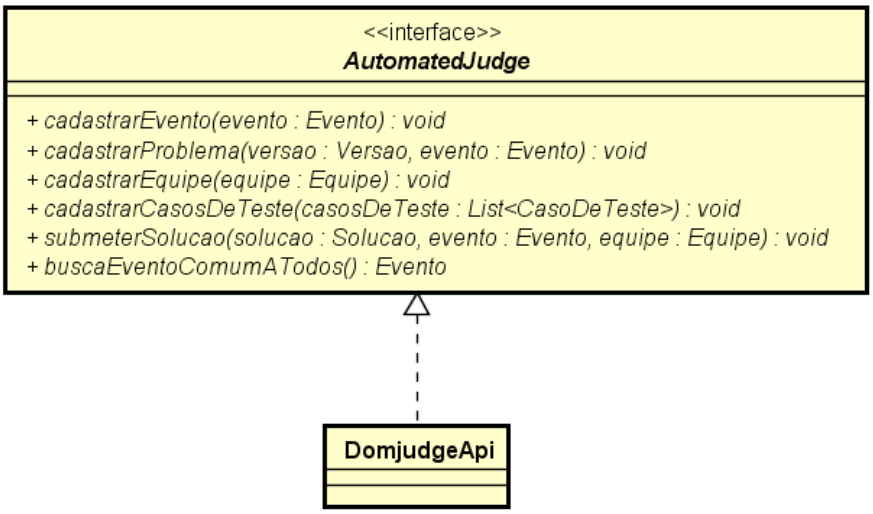
Fonte: Elaborada pelo autor

- void submeterSolucao(Solucao solucao, Evento evento, Equipe equipe);
- Evento buscaEventoComumATodos();

Muitos desses métodos são autoexplicativos, tirando o buscaEventoComumATodos. Esse método gera um evento (contest) se não existir e o retorna para ser usado por todos usuários do sistema. Isso remove o custo de ser ter que criar um evento para cada submissão de solução, ou fazer um evento particular para cada usuário (em um contexto de disciplinas, cada disciplina poderá ter um evento associado).

Ao escolhermos um avaliador automático para integração, deve ser criado uma classe que implemente esse métodos. Essa implementação vai depender da arquitetura do sistema avaliador. Neste trabalho implementamos essa interface para funcionar com o Domjudge. Dentre os métodos citados, apenas o cadastrarEvento(); e buscaEventoComumATodos(); não foi implementado, foi utilizado um evento fornecido pelo próprio Domjudge. Há ainda métodos a serem definidos na interface, como por exemplo um que obtenha resultados de uma solução

Figura 11 – Automated Judge Interface

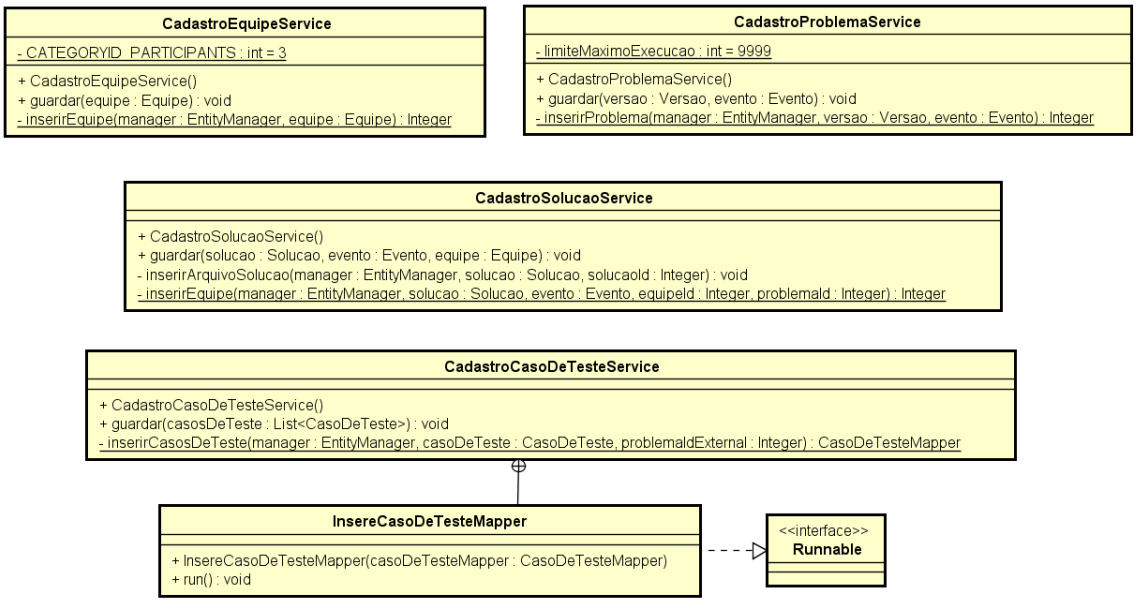


Fonte: Elaborada pelo autor

submetida. Apesar de ele não ter sido definido, ele foi implementado externamente.

Na Figura 11, a classe que implementa esses métodos é a DomjudgeApi, através da Automated Judge Interface. A DomjudgeApi utiliza serviços, que implementam propriamente dito esses métodos. Essas classes de serviço são mostradas na Figura 12.

Figura 12 – Serviços utilizados pelo DomjudgeApi



Fonte: Elaborada pelo autor

5.4 Protótipo funcional como estudo de caso

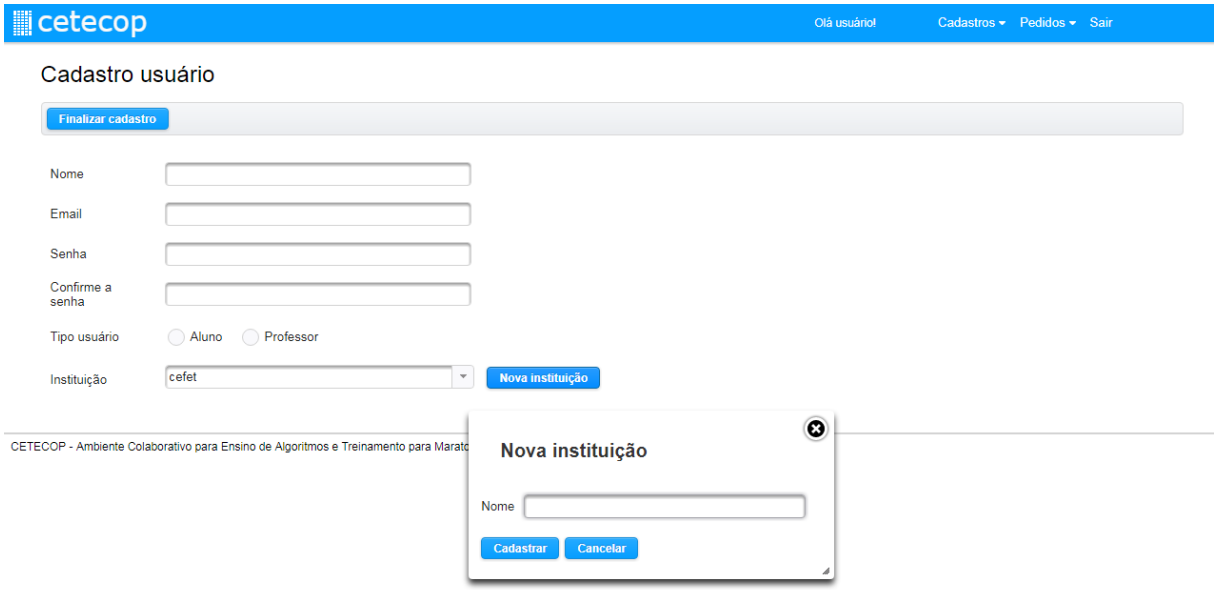
Foi desenvolvido um protótipo funcional para testar a integração com o Domjude bem como simular uma sequência comum de passos encontrados em sistemas de avaliação online, de modo a ter-se um sistema capaz de ser prontamente utilizado para o treinamento de alunos para competições.

A implementação realizada deste protótipo envolve:

- O cadastro de usuários (Figura 13) e login.
- O cadastro dos dados do problema, onde se informa o nome, enunciado, classificações, etc (Figura 14).
- O cadastro dos casos de teste, que tem os conjuntos de dados de entrada e saída (Figura 15).
- Os limites de execução, que permite limitar o tempo de acordo com uma classificação (Figura 16).
- A busca de problemas por alunos (Figura 17).
- A visualização da informação do problema, que envolve enunciado, nome e entradas para teste (Figura 18).
- A submissão de soluções por alunos, para estes submeterem os algoritmos realizados (Figura 19).
- O acompanhamento do resultado das soluções é visto pelo ranqueamento global (Figura 20).

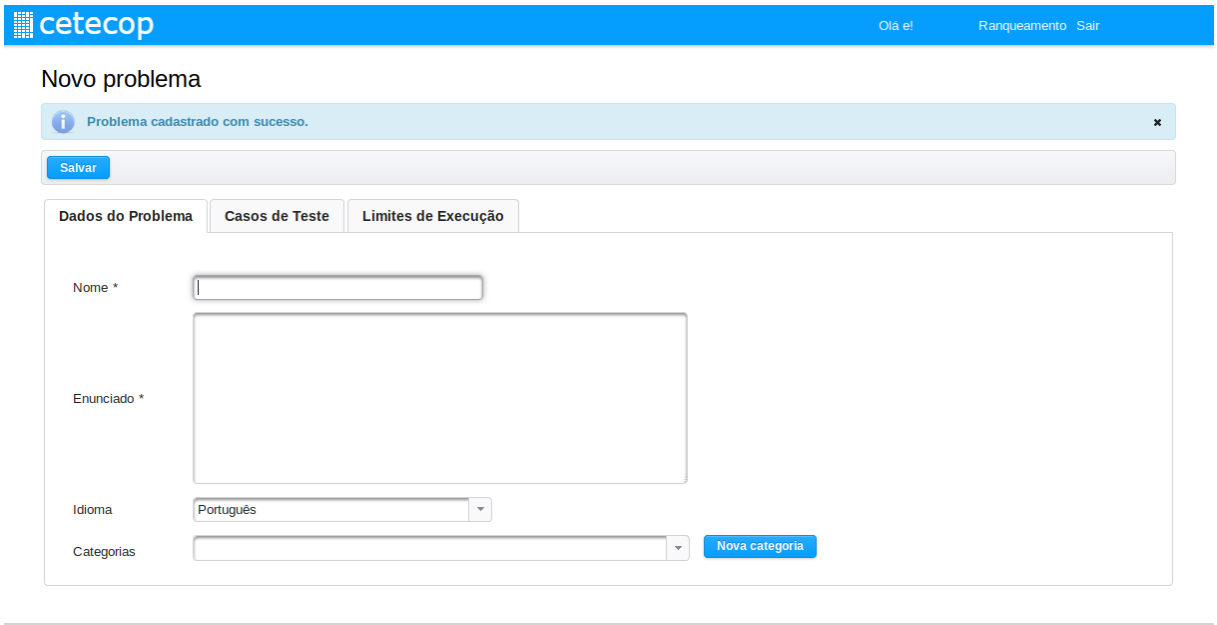
Além disso, ao salvarmos informações na base de dados do domjudge, há a necessidade de buscarmos novamente essas informações, entretanto a priori não temos uma equivalência entre os dados salvos com o banco utilizado pelo CETECOP. Dessa forma foi utilizado o conceito de mapeadores, que relacionam os dois bancos. Com isso quando há a necessidade de buscar a solução de um problema, por exemplo, recorremos aos mapeadores para encontrarmos o problema equivalente na base do domjudge. Algumas classes desses mapeadores podem ser encontradas no Apêndice A .

Figura 13 – Cadastro de Usuários



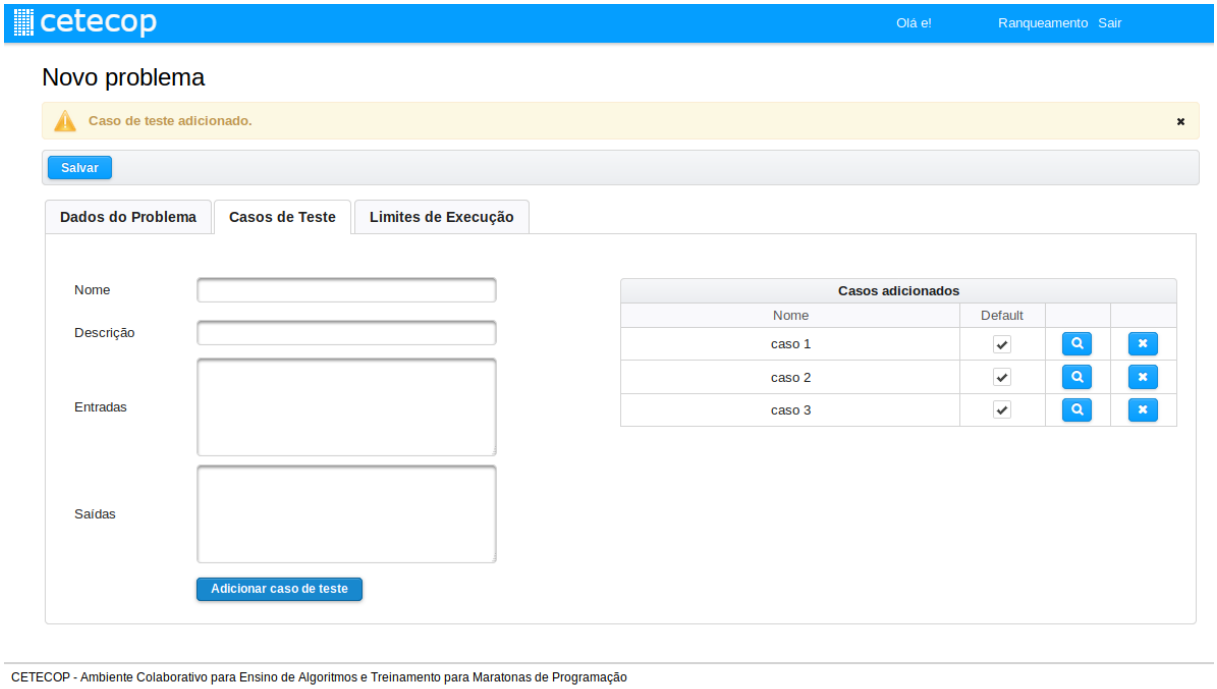
Fonte: Elaborada pelo autor

Figura 14 – Dados do problema



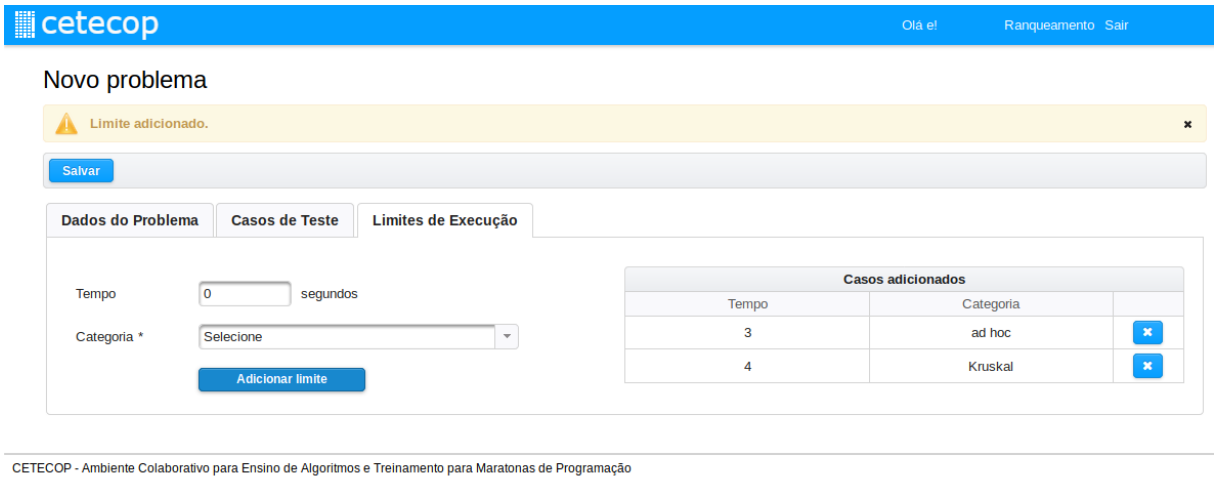
Fonte: Elaborada pelo autor

Figura 15 – Casos de Teste



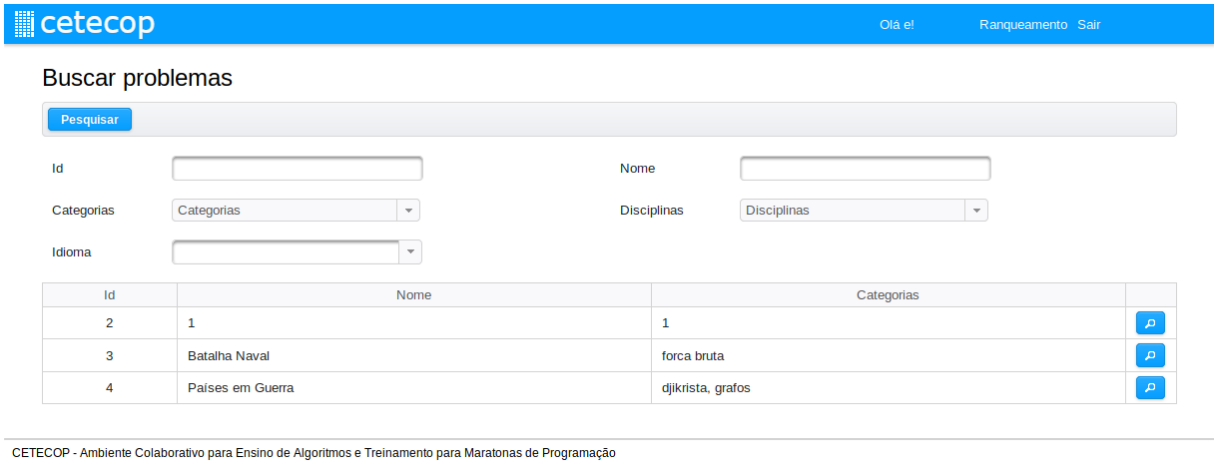
Fonte: Elaborada pelo autor

Figura 16 – Limites de Execução



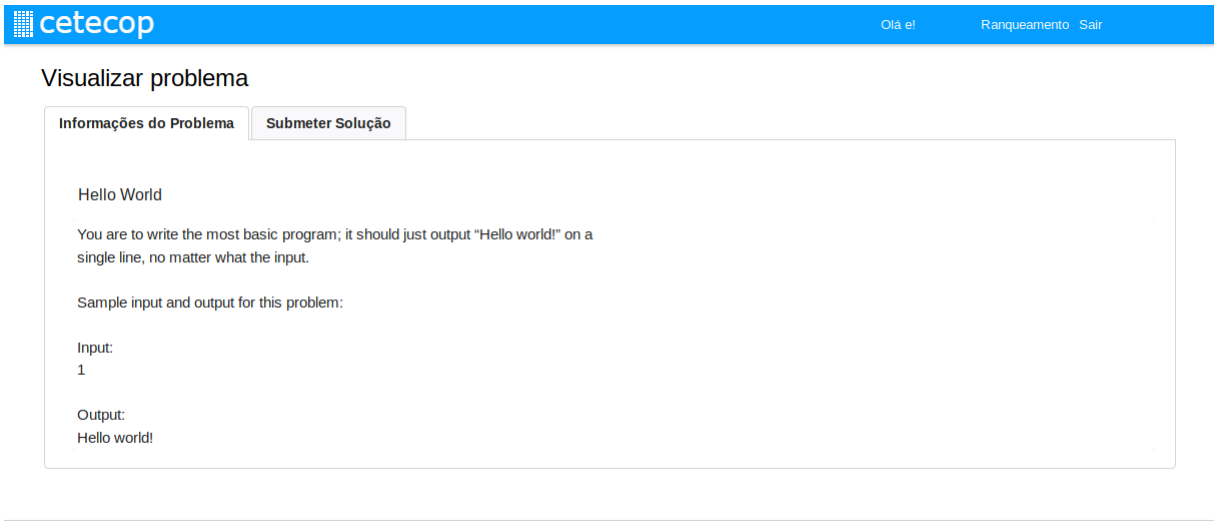
Fonte: Elaborada pelo autor

Figura 17 – Busca de Problema



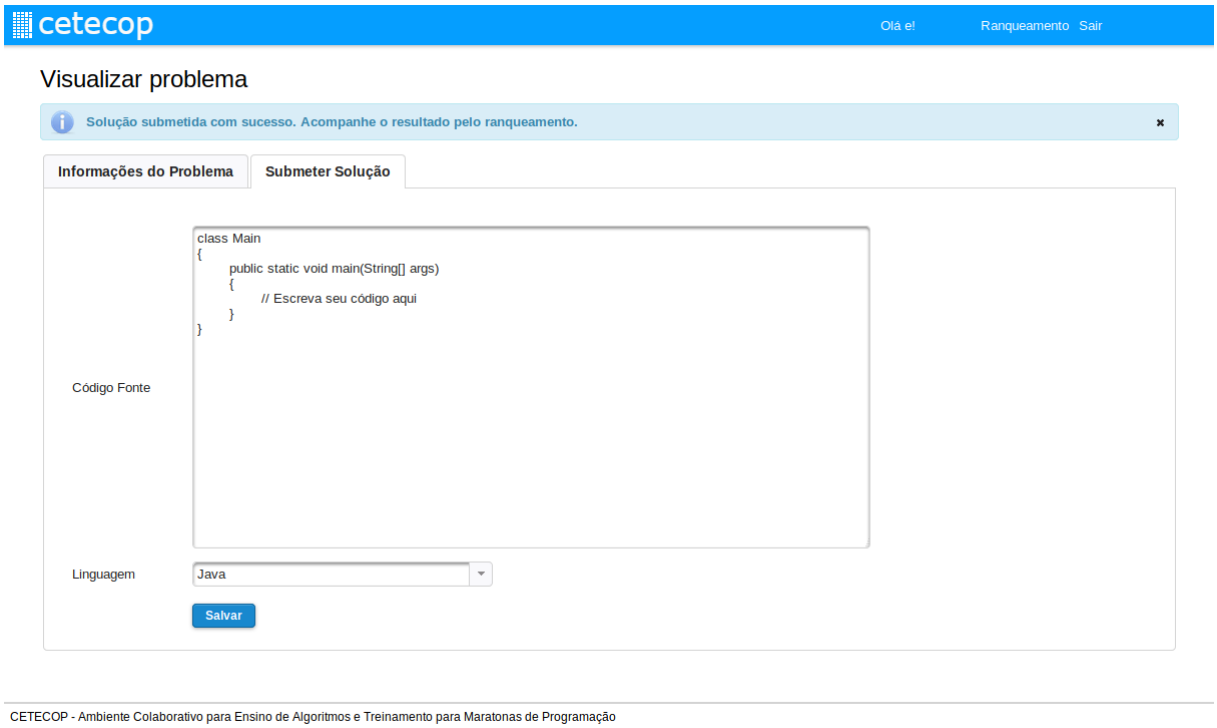
Fonte: Elaborada pelo autor

Figura 18 – Visualização de Problema



Fonte: Elaborada pelo autor

Figura 19 – Submissão de Problema



Fonte: Elaborada pelo autor

Figura 20 – Ranqueamento ou Resultado Global

Id solucao	Id problema	Nome do problema	Programador	Resultado
1	2	1	e	na fila
2	2	1	e	na fila
3	2	1	e	na fila
4	2	1	e	na fila
5	2	1	e	na fila
6	2	1	e	correct
7	2	1	e	no-output
8	2	1	e	correct
9	2	1	e	correct
10	5	Hello World	e	na fila

Fonte: Elaborada pelo autor

6 Conclusão

“Penso, logo existo”.
René Descartes

O ensino através de ferramentas virtuais tem ganhado espaço e adeptos, no mesmo sentido, em que a colaboração e formação de comunidades que compartilham e constroem o conhecimento. Dentro do ensino de computação e lógica de programação, existem vários desafios e paradigmas entre o aluno e o saber, mas os ambientes virtuais por competições continuam sendo atrativos.

Apesar de existirem ferramentas com foco no treinamento de competições, estas estão distantes de outros modelos que já existem no ensino virtual. Dentro do ambiente de colaboração e liberdade de conhecimento, as soluções são fechadas e dão pouca abertura à própria comunidade que cria sistemas.

Propomos a primeira versão da arquitetura conceitual de CETECOP que apresenta requisitos colaborativos não presentes nas principais ferramentas existentes. Em especial, dentro do conceito prático, notamos que o DOMjudge apresenta uma estrutura modular e fornece API’s de comunicação adequadas para que ele possa ser adaptado para ocupar as lacunas do Contest Manager e Judge System o que pode gerar uma economia na escrita de código quando precisarmos usar suas funcionalidades.

Além disso, foi proposto taxonomias iniciais para problemas bem como a criação de uma API integradora com o avaliador automático Domjudge. Foi implementado também o ranquemaneto global para visualização de resultados.

Com esse trabalho, espera-se que instituições acadêmicas utilizem os resultados encontrados no ensino de programação, uma vez que a base de dados de problemas gerado pouparia o esforço do professor na criação de listas de exercícios. Os alunos poderão ser beneficiados devido a um *feedback* rápido e um melhor acompanhamento da disciplina.

Espera-se também, que os alunos tenham um contato maior desde o início do curso com sistemas de avaliação online, que são semelhantes ao da maratona de programação, deixando os mais familiarizados com tais sistemas de forma a aumentar a sua competitividade.

6.1 Trabalhos Futuros

Devido a grande complexidade do sistema, este trabalho focou primeiramente no desenvolvimento de um protótipo que envolva o módulo de gerenciador de problemas. Os outros módulos, como o sistema de acompanhamento, poderão ser continuados por outros alunos futuramente. Quando todo o sistema estiver completo, poderá ser realizado testes nas próprias disciplinas do curso, de forma a avaliar o sistema, podendo usar até mesmo questionários de

validação. Além disso, novos critério de ranqueamento devem ser propostos e implementados, aprimorando-se o desenvolvido.

Há ainda a necessidade de modificar os protótipos de interfaces criados, deixando-os responsivos, de forma a permitir que o sistema possa ser acessado por dispositivos móveis, e não somentes desktops. Para alunos poderem codificar pelo próprio sistema, deve-se usar um analisador estático de código para auxilia-los na códificação.

Tem-se em aberto para desenvolvimento, a criação de versões de problemas pela comunidade e a colaboração através da classificação de problemas. Deve-se ser realizado ainda, a criação do sistema de acompanhamento, através do módulo User Performance Manager, de forma a professores poderem gerenciar alunos em disciplinas e avalia-los.

Referências

ALONSO, A. T.; MIRANDA, H. L. H. *Módulos para el entrenamiento de los concursantes del ACM-ICPC*. 2014. Dissertação (Monografia) — UCLV, 2014. Citado na página 22.

ANDER-EGG, E. *Introducción a las técnicas de investigación social: para trabajadores sociales*. 7. ed. [S.l.]: Humanitas, 1978. Citado na página 24.

BEZ, J. L.; FERREIRA, C. E.; TONIN, N. A. URI Online Judge Academic: A Tool for Professors. *Proceedings of the 2013 International Conference on Advanced Ict and Education*, v. 33, p. 763–766, 2013. Citado nas páginas 16, 19 e 20.

BOCA. *BOCA Online Contest Administrator*. 2017. Disponível em: <https://www.ime.usp.br/~cassio/boca/>. Acesso em: 30 ago. 2017. Citado na página 16.

BRASSCOM. *O Mercado de Profissionais de TI no Brasil*. 2014. Disponível em: <http://alexandre.ci.ufpb.br/retencao-cc-2/>. Acesso em: 14 mai. 2017. Citado na página 11.

BRITO, R. F.; PEREIRA, A. T. C. Um estudo para ambientes colaborativos e suas ferramentas. *Congresso Nacional de Ambientes Hipermídia para Aprendizagem*, 2004. Citado na página 13.

CAMPOS, C.; FERREIRA, C. Boca: um sistema de apoio a competições de programação. *Workshop de Educaçao em Computaçao; Anais do Congresso da SBC*, 2004. Citado nas páginas 16 e 17.

DETERS, J. I. et al. *O Desafio de Trabalhar com Alunos Repetentes na Disciplina de Algoritmos e Programação*. 2008. Disponível em: <http://www.proativa.virtual.ufc.br/sbie/CD_ROM_COMPLETO/workshops/workshop2/ODesafiodeTrabalharcomAlunosRepetentesna.pdf>. Acesso em: 8 mai. 2017. Citado na página 12.

DOMJUDGE. *DOMjudge - Introduction*. 2017. Disponível em: <https://www.domjudge.org/intro>. Acesso em: 26 jun. 2017. Citado nas páginas 16 e 17.

DUARTE, A. N. *As disciplinas que mais retém alunos em um curso de computação*. 2013. Disponível em: <http://alexandre.ci.ufpb.br/retencao-cc-2/>. Acesso em: 24 jun. 2017. Citado na página 12.

DUARTE, A. N. *O que está acontecendo com nossos alunos?* 2013. Disponível em: <http://alexandre.ci.ufpb.br/retencao-cc/>. Acesso em: 24 jun. 2017. Citado na página 11.

FRANCISCO, R. E. *Juiz Online no Ensino de CS1: Requisitos, Dificuldade de Problemas e Plágio em Código-Fonte*. 2016. Dissertação (Mestrado) — UFG, 2016. Citado nas páginas 14 e 22.

GOMES, A.; HENRIQUES, J.; MENDES, A. Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. *Educação, Formação & Tecnologias*, v. 1, p. [93–103], 2008. Disponível em: <http://www.eft.educom.pt/index.php/eft/article/view/23>. Citado na página 13.

GUSTOKASHIN, M.; ZAVARIN, S. *ACM/ICPC: Why Do Students Need Programming Contests?* 2017. Disponível em: <https://cs.hse.ru/en/HERB/gustokashin_zavarin>. Acesso em: 25 jun. 2017. Citado na página 13.

ICPC. *ICPC Fact Sheet*. 2017. Disponível em: <https://icpc.baylor.edu/worldfinals/pdf/Factsheet.pdf>. Acesso em: 25 jun. 2017. Citado na página 12.

LAKATOS, E. M.; MARCONI, M. A. *Metodologia Cientifica*. 2. ed. [S.l.]: Atlas, 1995. Citado na página 24.

MCGETTRICK, A. et al. Grand Challenges in Computing: Education—A Summary. *The Computer Journal*, v. 48, n. 1, p. 42–48, 2005. Citado na página 11.

MOREIRA, M.; FAVERO, E. Um ambiente para ensino de programação com feedback automático de exercícios. *Workshop Sobre Educação em Computação*, p. 429–438, 2009. Disponível em: <http://csbc2009.inf.ufrgs.br/anais/pdf/wei/st01_02.pdf>. Citado na página 14.

NUNES, D. J. *Educação Superior em Computação Estatísticas*. 2014. Disponível em: <http://www.sbc.org.br/documentos-da-sbc/summary/133-estatisticas/1007-estatisticas-da-educacao-superior-2014>. Acesso em: 14 mai. 2017. Citado na página 11.

OZTURK, M. A.; DEBELAK, C. Affective Benefits from Academic Competitions for Middle School Gifted Students. *Gifted Child Today*, SAGE PublicationsSage CA: Los Angeles, CA, v. 31, n. 2, p. 48–53, apr 2008. Disponível em: <http://journals.sagepub.com/doi/10.4219/gct-2008-758>. Citado na página 13.

PACHECO, P. *Computer-Based Assessment System for E-learning Applied to Programming Education*. 2010. Dissertação (Mestrado) — FEUP, 2010. Citado nas páginas 22 e 23.

POUCHER, B. Giving Students the Competitive Edge. *Commun. ACM*, ACM, New York, NY, USA, v. 55, n. 8, p. 5, aug 2012. Disponível em: <http://doi.acm.org/10.1145/2240236.2240237>. Citado na página 12.

PRIETCH, S. S.; PAZETO, T. A. Estudo sobre a evasão em um curso de licenciatura em informática e considerações para melhorias. *Escola Regional da Bahia*, 2010. Citado na página 12.

ROCHA, L. B. *Sistema de apoio à submissão e avaliação de trabalhos académicos com componentes gráficas*. 2011. Dissertação (Mestrado) — FEUP, 2011. Citado na página 23.

RODRIGUES, F. S. *Estudo sobre a evasão no curso de Ciência da Computação da UFRGS*. 2013. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/77275/000896280.pdf>. Acesso em: 24 jun. 2017. Citado na página 12.

SCHMITT, M. A. R. Dificuldades apresentadas pelo modelo wiki para a implementação de um ambiente colaborativo de aprendizagem. *Renote*, 2006. Disponível em: <http://seer.ufrgs.br/renote/article/view/14175/8103>. Citado na página 13.

SELIVON, M.; BEZ, J. L.; TONIN, N. A. URI Online Judge Academic: Integracão e Consolidação da Ferramenta no Processo de Ensino/Aprendizagem. *Anais do 23 Workshop sobre Educação em Computação (WEI 2015)*, n. CSBC, 2015. Citado na página 20.

SPOJ. *SPOJ Online Judge*. 2017. Disponível em: <http://br.spoj.com.com.br>. Acesso em: 30 ago.. 2017. Citado nas páginas 13 e 19.

STEVEN, H.; FELIX, H. *Competitive Programming*. [S.l.]: Lulu, 2010. Citado na página 28.

TROTMAN, A.; HANDLEY, C. Programming contest strategy. *Computers & Education*, v. 50, n. 3, p. 821–837, apr 2008. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0360131506001357>. Citado na página 13.

URI. *URI Online Judge*. 2017. Disponível em: <https://www.urionlinejudge.com.br/>. Acesso em: 30 ago.. 2017. Citado nas páginas 13 e 20.

UVA. *UVA Online Judge*. 2017. Disponível em: <http://uva.onlinejudge.org>. Acesso em: 30 ago.. 2017. Citado nas páginas 13 e 19.

WAZLAWICK, R. S. *Metodologia de Pesquisa para Ciência da Computacao*. [S.l.]: Elsevier, 2009. Citado na página 24.

XAVIER, J. C. *Computer-Based Assessment System for E-learning Applied to Programming Education*. 2011. Dissertação (Mestrado) — FEUP, 2011. Citado nas páginas 22 e 23.

Apêndices

APÊNDICE A – Documentação do Projeto

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS VII - UNIDADE TIMÓTEO

Documentação de Desenvolvimento de Projeto de Software

Ambiente Colaborativo para Ensino e Aprendizagem de Programação de Computadores e Treinamento para Competições

Aluno: Otavio Cesário Oliveira

Timóteo - MG
01/10/2017

1. Lista de Requisitos

Prioridade	Tipo	Nome da função	Descrição
Alta	1	Cadastro aluno e cadastro professor	Alunos e professores devem se cadastrar no sistema
Alta	2	Buscar problemas	Buscar problemas por categorias
Media	2	Visualizar rank	Visualizar diversos tipos de rank, inclusive o acompanhamento de uma lista de exercício por um professor
Alta	1	Cadastro de problemas	Professores podem cadastrar problemas
Media	1	Cadastro disciplina	Professores podem criar disciplinas
Media	1	Cadastro de lista	Professores podem criar lista de exercícios
Media	1	Submeter solução	Alunos submetem solução de um problema para avaliação
Alta	3	Login	Usuários fazem login no sistema
Alta	1	Classificar problema	Professores e alunos podem atribuir classificações a um problema
Alta	1	Criar classificação de problema	Professores e alunos podem criar novas taxonomias
Alta	1	Avaliar classificação	Professores e alunos podem avaliar classificação de um problema
Alta	1	Editar problema	Professor pode editar um problema criando novas versões
Media	1	Cadastrar aluno em disciplina	Professor convida alunos para participarem de uma disciplina
Alta	1	Excluir classificação	Administradores do sistema podem excluir classificações do sistema

Tipo: 1 – Cadastro 2 – Relatórios/Consultas 3 – Controle de Acesso

2. Requisitos de Software

2.0 Regras de negócio

Política de classificação RN01	
Descrição	Somente os alunos que tiverem resolvido um problema poderão classifica-los.

2.1 Descrição dos Atores

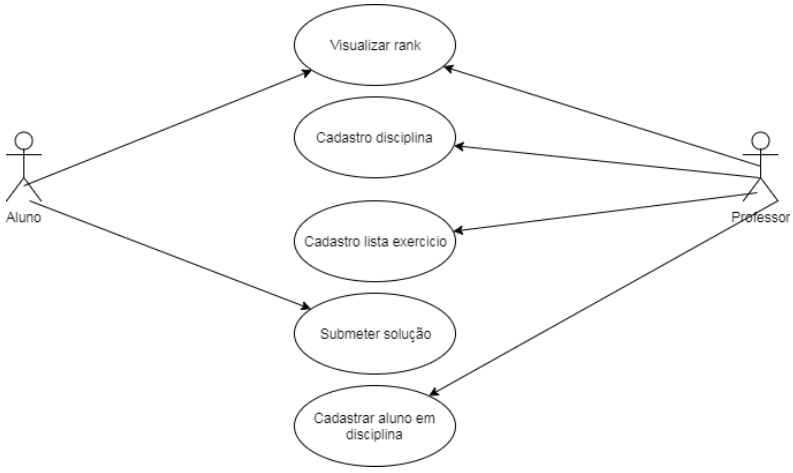
Num.	Nome	Descrição	Freqüência de Uso	Proficiência em Informática
1	Administrador	Os administradores são os usuários que têm acesso ao módulo administrativo. Eles têm como objetivo realizar tarefas de característica gerencial, como por exemplo excluir classificações	Diária	Alta

		duplicadas.		
2	Professor	Coordena a criação de listas de exercícios, cadastro de problemas, cadastro de classificações, etc.	Diária	Alta
3	Aluno	Submete soluções de problemas e participa na classificação de problemas, etc.	Alta	Alta

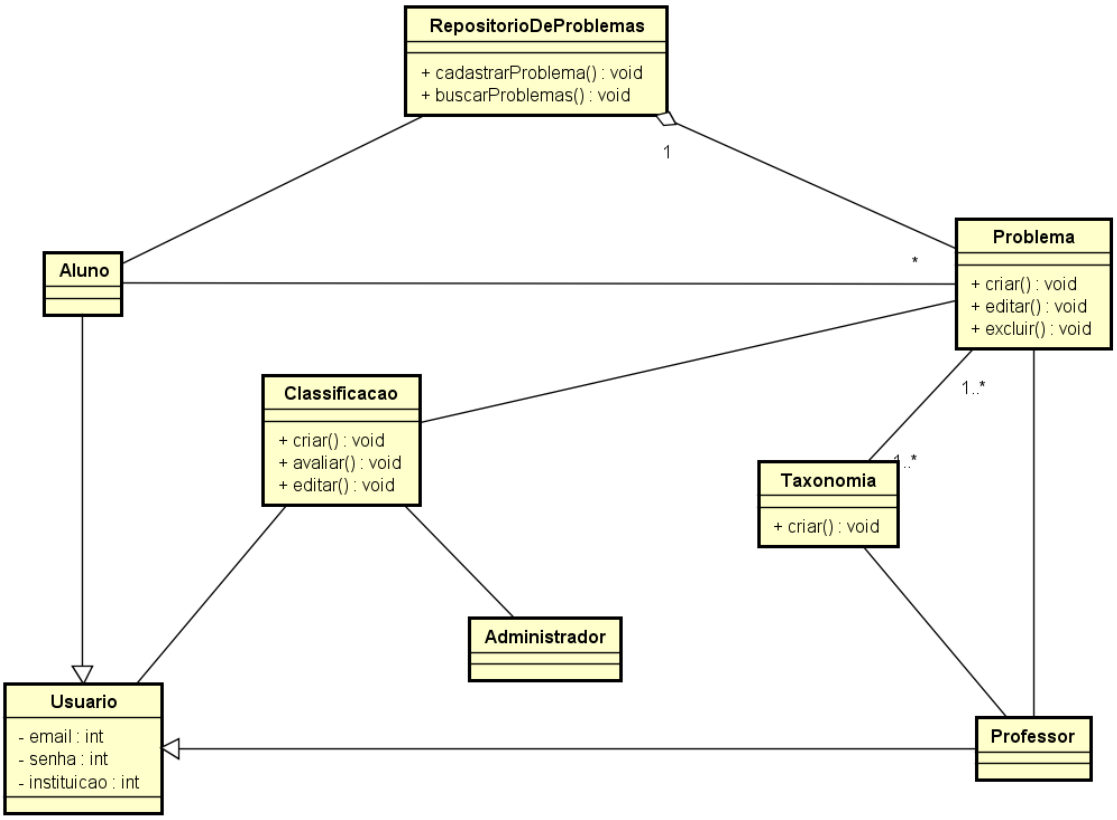
2.2 Casos de Uso

Gerenciador de Problemas

Sistema de Acompanhamento



2.4 Diagrama de Classes



3. Diagrama de Objetos

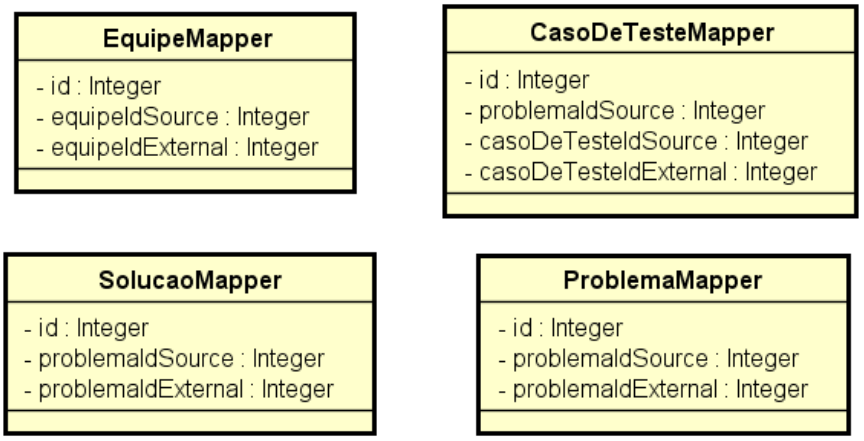


Figura 1 Diagrama dos Mapeadores

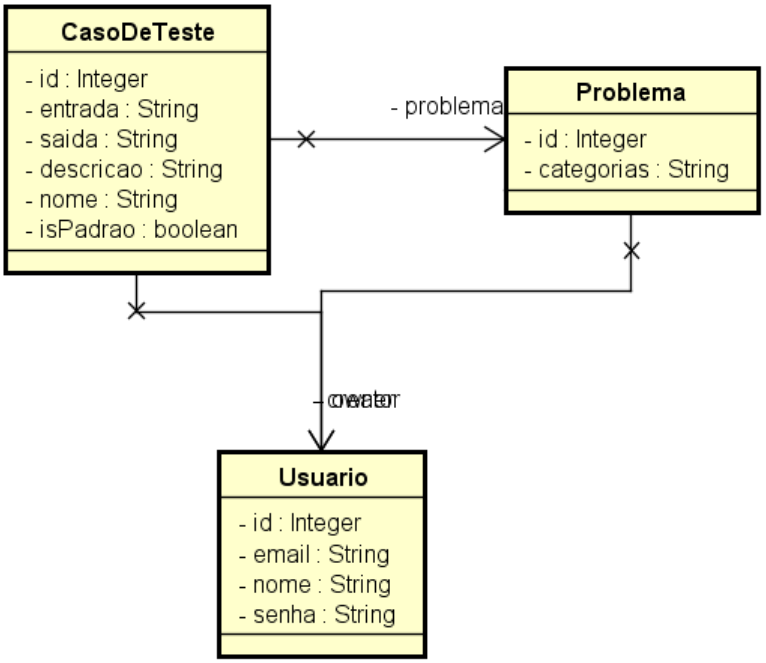


Figura 2 - Diagrama do Caso de Teste

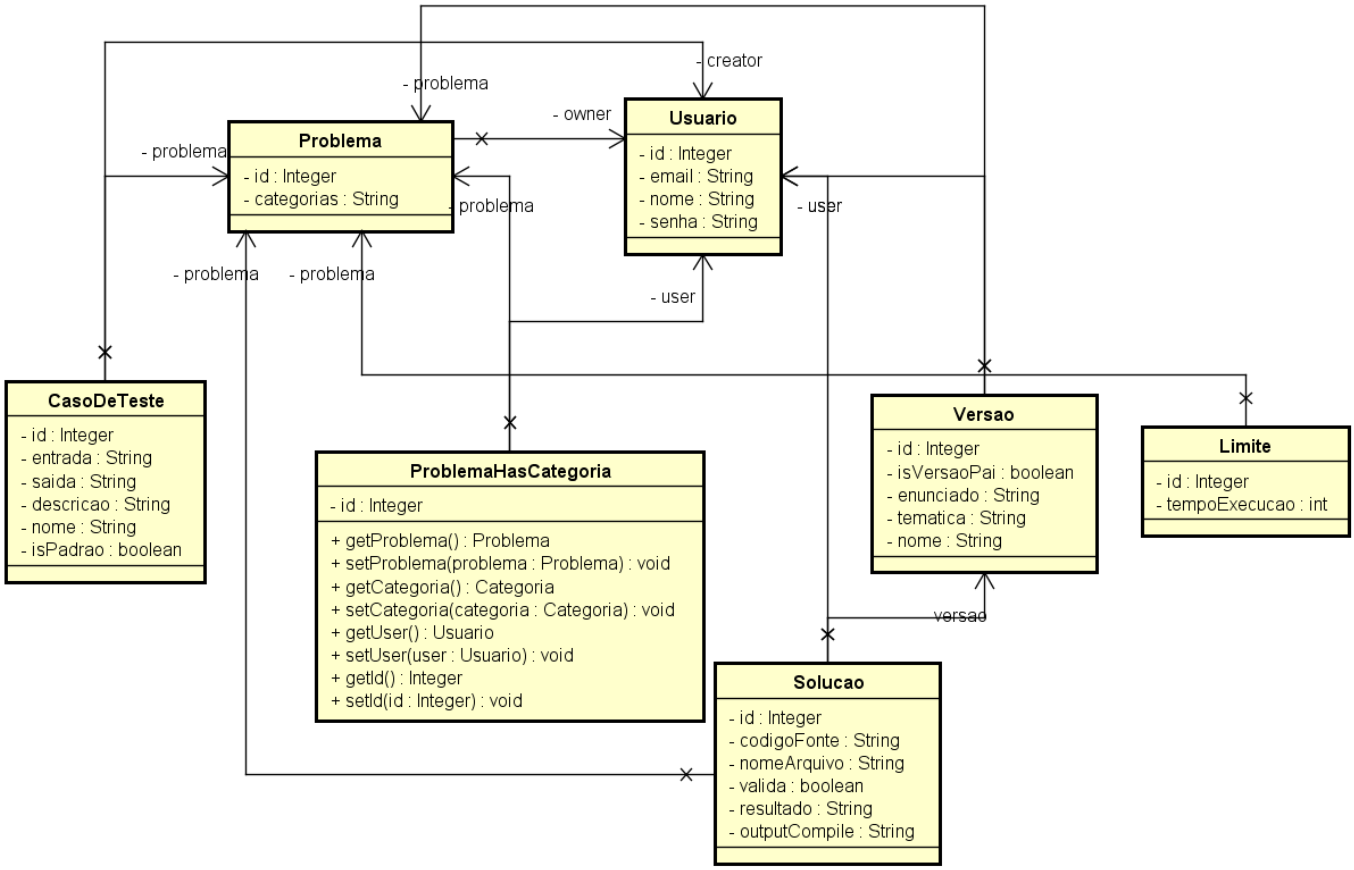


Figura 3 - Diagrama do Problema

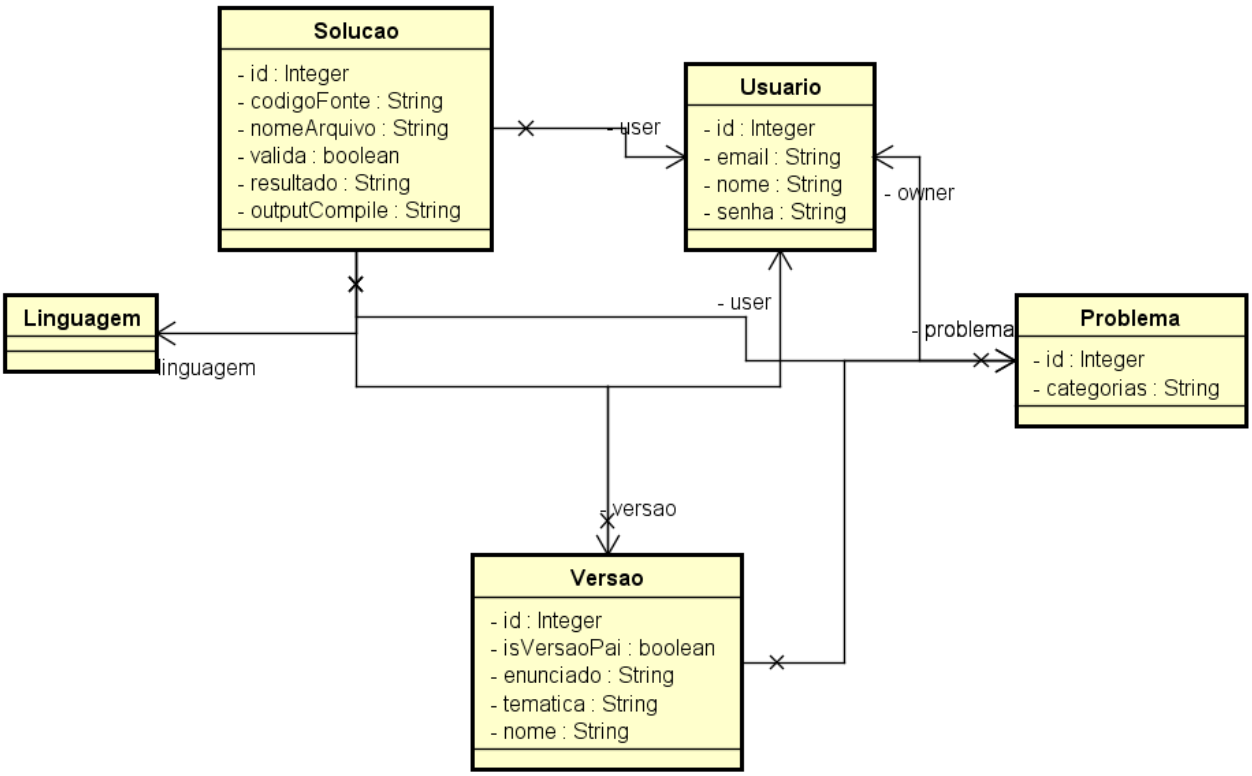
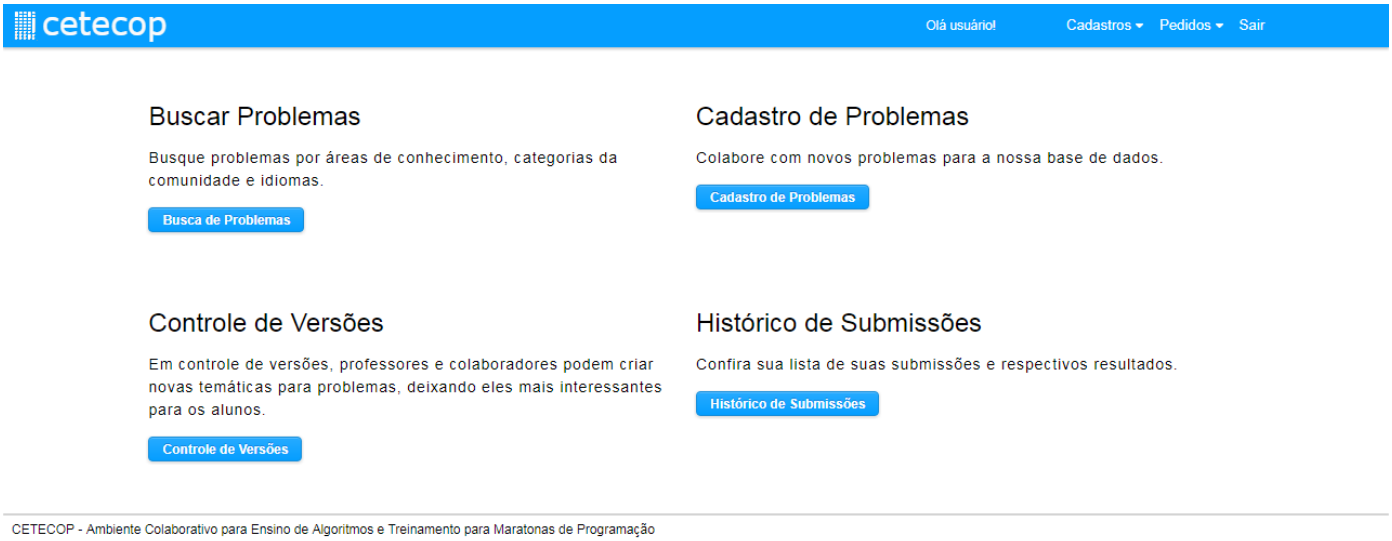
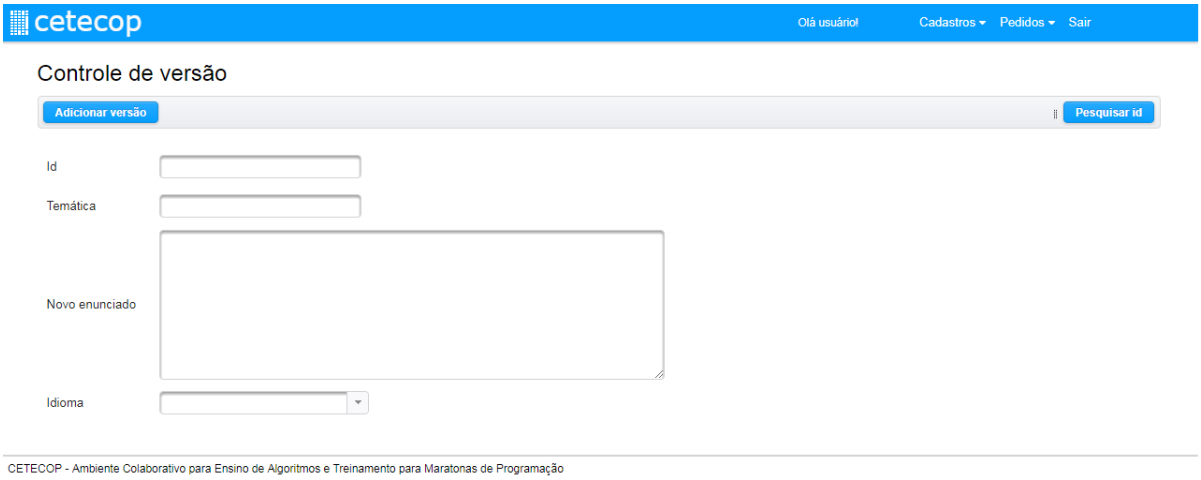


Figura 4 - Diagrama da Solução

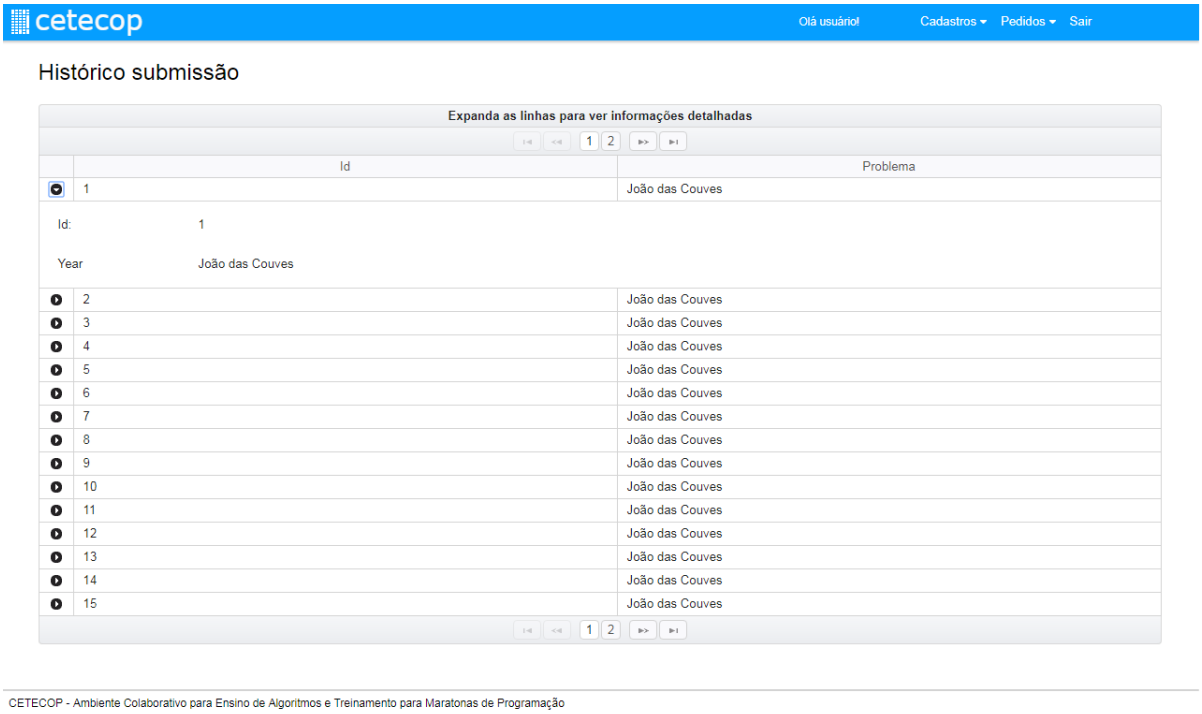
4. Projeto de Interface



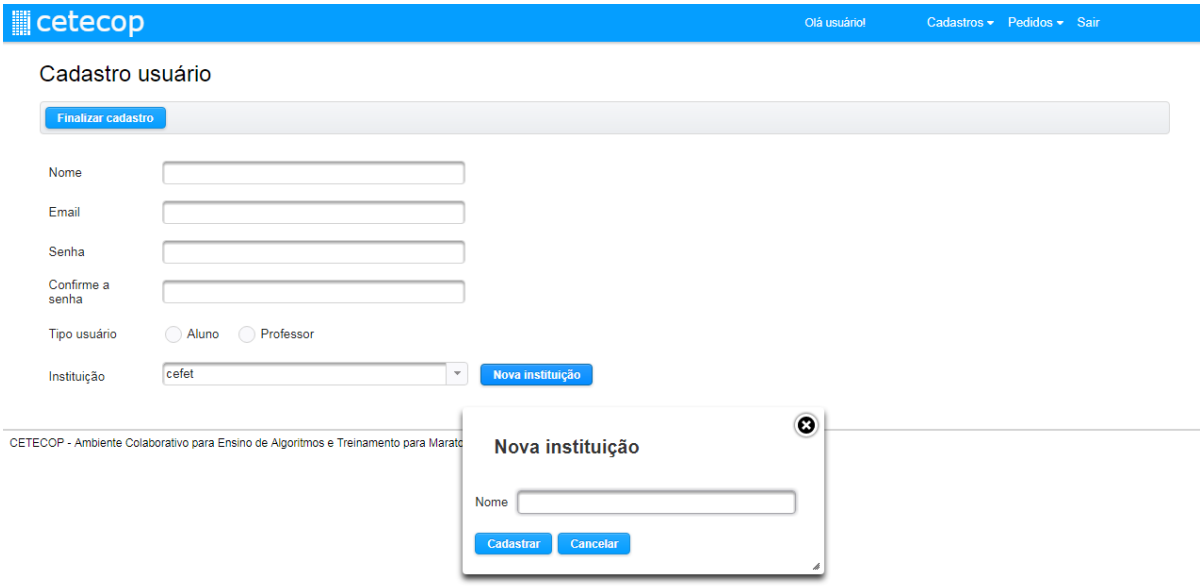
Tela Inicial



Controle de Versão

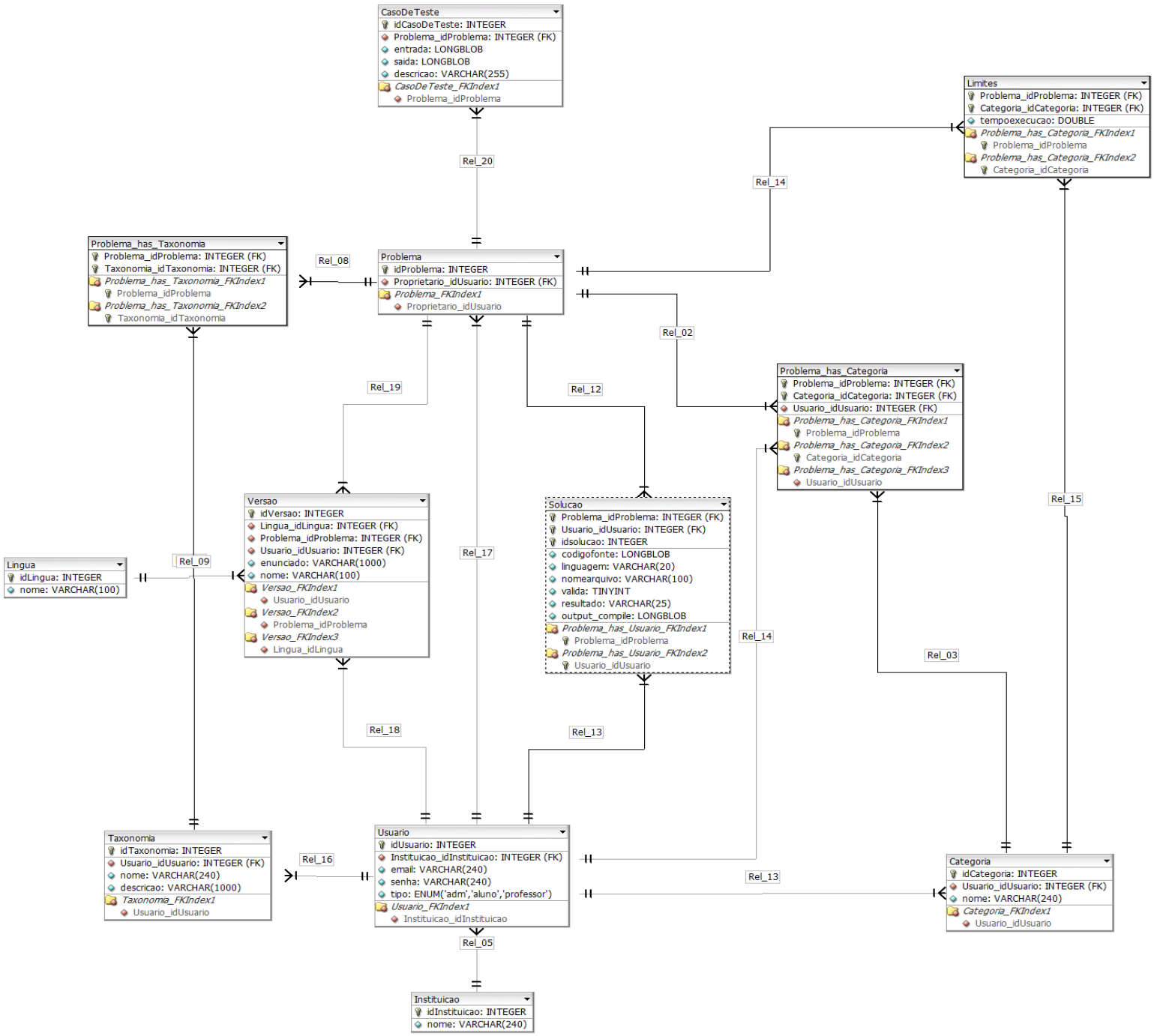


Histórico Submissão



Cadastro de Usuário

5. Modelo de Análise – Projeto de Dados - DER



APÊNDICE B – Código Fonte

O código fonte desenvolvido neste trabalho se encontra no seguinte endereço eletrônico: https://github.com/otavio-cesar/cetecop.

APÊNDICE C – Instalação e Configuração do Domjudge

A documentação do Domjudge fornece o manual de instalação, entretanto alguns passos não se apresentam sequencialmente, o que pode dificultar quando ele for instalado pela primeira vez. Além disso os SO's utilizados são Debian e RedHat, apresentamos a variante Ubuntu, que é bem semelhante ao Debian. Ao final de execução dos seguintes passos, tem-se o Domjudge configurado e pronto para testes com a linguagem Java.

1. sudo apt-get update
2. sudo apt-get install apache2
3. sudo apache2ctl configtest
4. sudo systemctl restart apache2
5. sudo apt-get install mysql-server
6. sudo apt-get install curl php libapache2-mod-php php-mcrypt php-mysql php-cli
7. sudo apt-get install phpmyadmin apache2-utils
8. sudo nano /etc/apache2/apache2.conf (Adicionar no final do arquivo apache2.conf: Include /etc /phpmyadmin /apache.conf)
9. sudo service apache2 restart
10. sudo chmod -R 777 domjudge-5.2.0 (Depois de extrair o domjudge-5.2.0.tar.gz, aplique o comando na pasta gerada)
11. sudo apt-get install libcgroup-dev libcurl4-gnutls-dev libcurl4-nss-dev libcurl4-openssl-dev libjsoncpp-dev
12. ./configure (Execute este comando dentro do diretório domjudge-5.2.0, sem o sudo.)
13. sudo make all
14. sudo make install-{domserver,judgehost,docs} (Irá instalar domjudge em /opt)
15. sudo ./dj_setup_database -u root -r install (Arquivo encontra-se no diretório / opt/ domjudge/ domserver/ bin)
16. sudo nano /etc/apache2/apache2.conf (Adicionar no final do arquivo apache2.conf: Include / opt/ domjudge /domserver/ etc/ apache.conf)

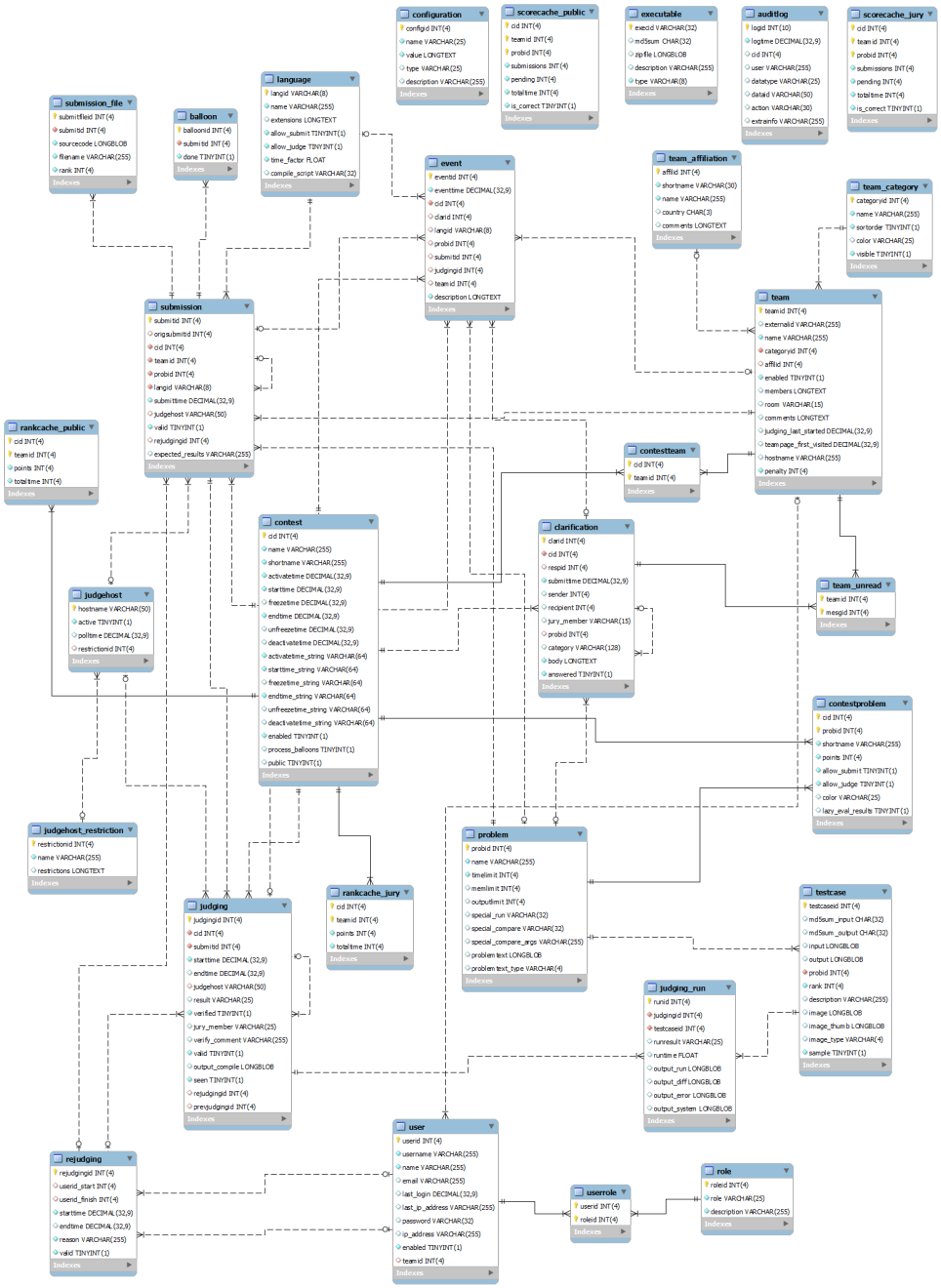
17. sudo service apache2 restart (A interface web do domjuge em http:// localhost/ domjudge deve estar funcionando. Em http:// localhost/ domjudge/ jury o usuario e senha é admin e admin, respectivamente.)
18. sudo useradd -d /nonexistent -g nogroup -s /bin/false domjudge-run
19. sudo groupadd domjudge-run
20. sudo nano / etc/ default/ grub (Adicione os parametros "cgroup_enable=memory"e "swaccount=1"na atribuição da variável GRUB_CMDLINE_LINUX_DEFAULT)
21. sudo update-grub
22. sudo reboot (Reinicie a máquina)
23. sudo apt-get install openjdk-8-jre-headless openjdk-8-jdk-headless
24. sudo nano /opt/domjudge/judgehost/etc/judgehost-config.php (Editar a última linha, trocando true para false. Isso desativa o chroot, é uma alternativa rápida e não recomendada de instalação. Para o outro modo, ver documentação oficial do DomJudge.)
25. sudo ./judgedaemon (Execute este arquivo que se encontra dentro do diretório / opt/ domjudge/ judgehost/ bin/)

Anexos

ANEXO A – Diagrama de Entidade de Relacionamento do Sistema DomJudge

Diagrama de entidade de relacionamento utilizado na análise e engenharia reversa do sistema Domjudge.

Figura 21 – DER do DOMJudge



Fonte: Elaborada pelo autor