CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS TIMÓTEO

Vinnícius Henrique Magione Soares

MODELAGEM E GESTÃO DE PROBLEMAS PARA USO EM TREINAMENTOS DE MARATONAS E ADAPTAÇÕES PARA DISCIPLINAS DE COMPUTAÇÃO

Timóteo

2021

Vinnícius Henrique Magione Soares

MODELAGEM E GESTÃO DE PROBLEMAS PARA USO EM TREINAMENTOS DE MARATONAS E ADAPTAÇÕES PARA DISCIPLINAS DE COMPUTAÇÃO

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

2021

Vinnícius Henrique Magione Soares

MODELAGEM E GESTÃO DE PROBLEMAS PARA USO EM TREINAMENTOS DE MARATONAS E ADAPTAÇÕES PARA DISCIPLINAS DE COMPUTAÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais, campus Timóteo, como requisito parcial para obtenção do título de Engenheiro de Computação.

Trabalho aprovado. Timóteo, 19 de abril de 2021:

_____

Prof. Me. Aléssio Miranda Júnior
Orientador

_____

Prof. Dr. Lucas Pantuza Amorim
Professor Convidado

_____

Prof. Me. Odilon Correa da Silva
Professor Convidado

Timóteo
2021

MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
DEPARTAMENTO DE COMPUTAÇÃO E CONSTRUÇÃO CIVIL - TM

ATA Nº 35 / 2021 - DCCTM (11.63.05)

Nº do Protocolo: 23062.023692/2021-78

Timóteo-MG, 24 de maio de 2021.

(Assinado digitalmente em 24/05/2021 23:29)
ALESSIO MIRANDA JUNIOR
PROFESSOR ENS BASICO TECN TECNOLOGICO
DCCTM (11.63.05)
Matrícula: 1713470

(Assinado digitalmente em 25/05/2021 23:19)
LUCAS PANTUZA AMORIM
PROFESSOR ENS BASICO TECN TECNOLOGICO
DCCTM (11.63.05)
Matrícula: 2897411

(Assinado digitalmente em 26/05/2021 10:01)
ODILON CORREA DA SILVA
PROFESSOR ENS BASICO TECN TECNOLOGICO
DCCTM (11.63.05)
Matrícula: 2794495

Para verificar a autenticidade deste documento entre em https://sig.cefetmg.br/public/documentos/index.jsp informando seu número: **35**, ano: **2021**, tipo: **ATA**, data de emissão: **24/05/2021** e o código de verificação: **446bcefa63**

Dedico a
minha família e amigos, pelo incentivo e apoio constantes.

Agradecimentos

A Deus por ter me dado forças para superar obstáculos e saúde física e mental para estar vivo e em condições de realizar este trabalho.

A minha família e amigos por todo apoio que recebi ao longo dos anos, por me ajudarem das mais diversas formas.

À instituição pelas oportunidades que me foram proporcionadas no decorrer do curso e que ainda me proporciona através do aprendizado que continuo obtendo.

Aos professores pelas diversas formas de conhecimento que me foram passadas no decorrer de todas disciplinas que estudei.

Ao meu orientador, pelo suporte e pela oportunidade da elaboração e deste trabalho.

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

“A menos que modifiquemos a nossa maneira de pensar, não seremos capazes de resolver os problemas causados pela forma como nos acostumamos a ver o mundo”.
Albert Einstein

Resumo

Um grande índice de evasão em cursos de computação, somadas a uma baixa motivação por parte de alunos ao lidarem com conceitos abstratos de lógica de programação mostram dificuldades no aprendizado dessa área. Professores vêm necessitando cada vez mais de um evolução das formas de ensino para que, consigam não só atrair novos estudantes, mas também mantê-los interessados nos estudos e todo seu trabalho executado de forma cada vez mais prática. Competições de programação se mostram como um atrativo nesse meio, no qual alguns fundamentos de maratonas possam vir a ser utilizados em sala de aula e contribuir com a didática. Foi percebido, porém, que portais de treinamento de computação deixam a desejar quando se trata do aprendizado fora de competições. Surge, assim, a ideia de usar elementos dessas ferramentas em uma plataforma direcionada para o aprendizado do aluno em sala de aula, aproveitando do acervo de problemas já existente na comunidade, e que permita uma maior colaboração entre professores. Desenvolveu-se neste trabalho, utilizando a plataforma OutSytems, uma aplicação web de gestão de problemas para professores de computação com elementos de colaboração e integração com repositórios do GitLab. Como resultado, tem-se um desenvolvimento utilizando do controle de versionamento do gerenciador de repositório baseado em git, capaz de fazer uma gestão de problemas colaborativa. Por meio disso é possível que diferentes usuários consigam atuar em um mesmo problema, criando versões diferentes ou atuando juntos em uma mesma versão com características de autores diferentes.

Palavras-chave: Sistemas colaborativos, Sistema de controle de versões, Sistemas de avaliação *online*, OutSystems, GitLab.

Abstract

A high dropout rate in computer courses, coupled with low motivation on the part of students when dealing with abstract concepts of programming logic, show difficulties in learning in this area. Teachers are increasingly in need of an evolution of teaching methods so that they can not only attract new students, but also keep them interested in studies and all their work performed in an increasingly practical way. Programming competitions are an attraction in this environment, in which some fundamentals of marathons can be used in the classroom and contribute to didactics. It has been noticed, however, that computer training portals fall short when it comes to learning outside of competition. Thus, the idea arises of using elements of these tools in a platform aimed at the student's learning in the classroom, taking advantage of the collection of problems that already exist in the community, and that allows greater collaboration between teachers. In this work, using the OutSytems platform, a problem management web application for computing teachers with elements of collaboration and integration with GitLab repositories was developed. As a result, there is a development using the git-based repository manager versioning control, capable of collaborative problem management. Through this it is possible that different users can act on the same problem, creating different versions or acting together in the same version with characteristics of different authors.

Keywords: Collaborative systems, Version control systems, Automated Judge Systems, OutSystems, GitLab.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CETECOP	Collaborative Environment for Teaching Computer Programing
ICPC	*International Collegiate Programming Contest*
IOI	*International Olympiad in Informatics de Conteúdo*
OBI	Olimpíada Brasileira de Informática
ACM	*Association for Computing Machinery*
TI	Tecnologia da Informação
SPOJ	*Sphere Online Judge*
DEED	Diretoria de Estatísticas Educacionais
RPT	Repositórios de Problemas para Treinamento
API	*Application Programming Interface*
REST	*Representational State Transfer*
SRM	*Single Round Matches*

Sumário

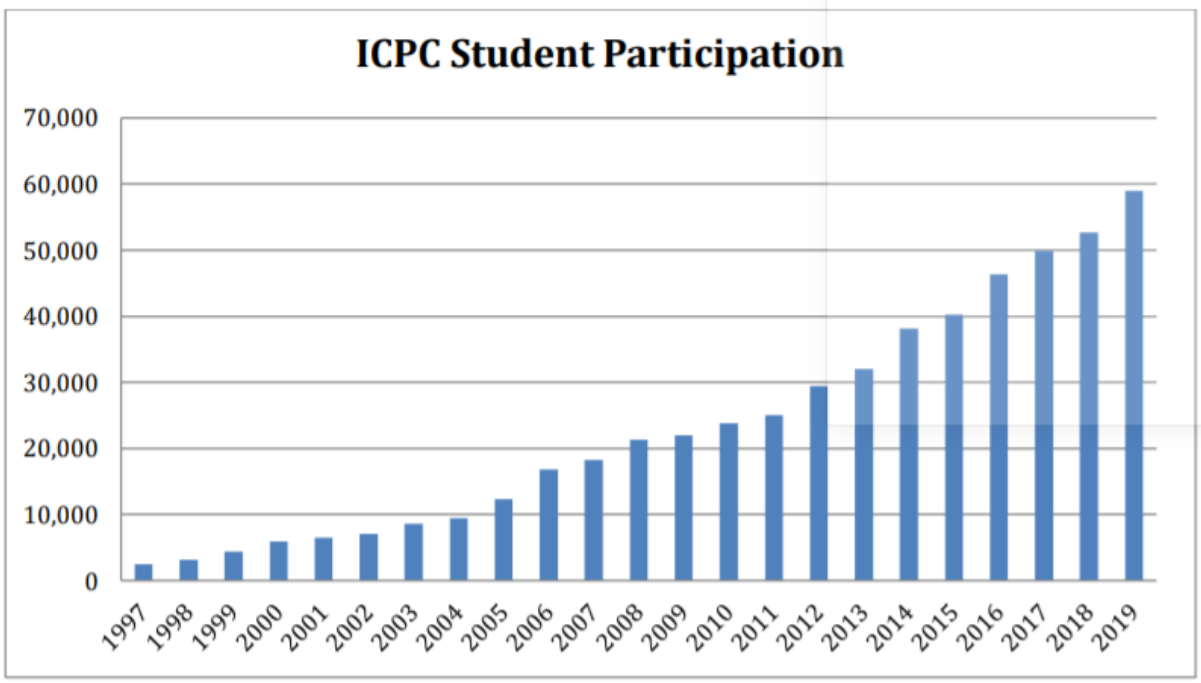
1 Introdução

A aprendizagem na área da computação envolve conhecimentos em conceitos abstratos, além de linguagens, exige alto nível de generalização, abstração e pensamento crítico (Gomes et al. 2008). Desse modo, é imprescindível que haja tanto dedicação por parte do aluno, quanto melhorias nos recursos voltados ao ensino (Rountree et al. 2004). São necessárias alternativas que tornem os estudos mais eficientes e contribuam com a diminuição do número de reprovações em disciplinas introdutórias do curso.

Diante deste cenário, onde estudantes apresentam dificuldades em continuar na área, é preciso elaborar métodos para atrair a atenção de alunos e mantê-los motivados a continuar estudando. As competições acadêmicas apresentam-se como um importante método para despertar a curiosidade dos alunos, por serem capazes de atraí-los e mantê-los interessados e motivados a estudar.

Competições de programação vêm atraindo mais participantes a cada ano. Segundo o relatório de dados da *International Collegiate Programming Contest* (ICPC) de 2020, este ano um total de 58963 competidores vindos de 3406 universidades de 104 países competiram em regionais da ICPC. Como pode ser visto na Figura 1, em 20 anos o número de participantes aumentou em cerca de 20 vezes entre os anos de 1997 e 2019 (ICPC 2020).

Figura 1 – Crescimento da participação de estudantes em competições da ICPC.



Fonte: (ICPC, 2018).

Existem diversas competições de programação, voltadas tanto para alunos do ensino

médio, como a *International Olympiad in Informatics* (IOI), uma das Olimpíadas Internacionais de Ciências, e a Olimpíada Brasileira de Informática (OBI), uma competição de programação realizada pela Sociedade Brasileira de Computação (SBC). Já para alunos do ensino superior, existe o ICPC, evento promovido pela ACM *(Association for Computing Machinery)* e a Maratona de Programação promovida pela Sociedade Brasileira de Computação no Brasil como parte da ICPC. Além destas, destacam-se também competições livres que ocorrem pela *internet*, realizadas pelos principais sites que disponibilizam problemas *online* com corretores automáticos, úteis como treino para outras competições, apesar de não oferecerem premiações.

Com o interesse pela área e pelas competições de programação aumentando, vêm sendo desenvolvidas ferramentas que podem auxiliar e contribuir com o aprendizado de estudantes e também podem ser usadas por aqueles que pretendem exercitar seus conhecimentos. Estes sistemas têm como objetivo servir de repositório de problemas para estudantes de programação. Além de possuírem mecanismos para correção *online* baseados em casos de testes predefinidos e auxilio para professores da área.

Dentre os portais mais utilizadas para treinamento e ensino de programação voltada para competições, pode-se citar a ACM-ICPC *Live Portal*, o acervo da ACM onde é possível encontrar os problemas utilizados nas competições regionais e finais da ICPC, a UVa *Online Judge*, Uri *online Judge* e a SPOJ (SPOJ 2021). Além de servirem como repositórios de problemas e soluções, realizam suas próprias competições de programação, e possuem classificação de pontos dos usuários que enviam suas respostas.

Estudantes e professores da área de computação podem se beneficiam ao utilizar tais sistemas, porém há parâmetros neles que podem ser melhorados. Estes sistemas, por não serem colaborativos e sendo independentes uns dos outros, possuem diferentes métricas, níveis de dificuldades e de classificação dos usuários e de seus problemas. Não há um espaço comum para discussões e enriquecimento da comunidade. Em contrapartida, ambientes colaborativos vêm se tornando cada vez mais importantes, devido a capacidade de proporcionar o compartilhamento de conhecimento e comunicação entre pessoas distantes (Brito e Pereira 2004).

Portanto, percebe-se que o aperfeiçoamento das ferramentas seria de grande valia para alunos, professores e interessados na área. O desenvolvimento de um framework colaborativo personalizável pode trazer facilidade ao aprendizado de estudantes. É possível também, um maior cooperativismo entre aqueles dispostos a discutir problemas e soluções de programação.

A ferramenta pode servir como um forte suporte didático, auxiliando o professor a gerir suas aulas, identificando as dificuldades específicas e o desempenho de cada aluno, realizando funções especificas de forma a facilitar seu trabalho, além de instigar o interesse do aluno pelos estudos em computação.

1.1 Justificativa

Cursos relacionados à computação vêm se tornando mais populares progressivamente, conforme a tecnologia e o mercado geram maior demanda por profissionais da Tecnologia da Informação (TI). De acordo com a Diretoria de Estatísticas Educacionais (DEED 2015) de 2005 a 2014, houve aumento no número de cursos de computação, de vagas oferecidas e no número de candidatos inscritos. O número de alunos inscritos para cursos na área de TI apresentou um crescimento de 162.655 em 2005, para 742.344 estudantes em 2014. Analisando ainda a relação de candidatos inscritos por vagas oferecidas, pode-se perceber que houve um aumento constante entre 2011 a 2014, como mostrado na Tabela 1.

Tabela 1 – Relação candidatos inscritos versus vagas oferecidas – 2005 a 2014.

Ano	Vagas Oferecidas (V)	Candidatos Inscritos (C)	Relação C/V
2005	102.424	162.655	1,59
2006	106.064	164.367	1,55
2007	197.282	304.298	1,54
2008	205.092	307.564	1,50
2009	278.708	376.675	1,35
2010	327.326	376.985	1,15
2011	238.673	511.510	2,14
2012	250.168	609.429	2,44
2013	248.411	650.234	2,62
2014	331.399	742.344	2,24

Fonte: Censo da Educação Superior (DEED, 2015).

Isto mostra que, nos últimos anos, um número cada vez maior de jovens tem se interessado por cursos como Ciências da Computação, Sistemas de Informação, Engenharia de Computação e diversos outros cursos de programação.

Entretanto, é comum que muitos destes alunos, que não possuem experiência na área, enfrentem dificuldades nas disciplinas introdutórias nesses cursos. Estas disciplinas têm como propósito fazer com que os alunos adquiram um conjunto de habilidades necessárias para que ao finalizar seu curso, estejam aptos a projetar programas e sistemas computacionais capazes de resolver problemas reais. Contudo, o número de concluintes de cursos na área de TI não apresentou crescimento constante, entre 2005 a 2014, conforme apresentado na Tabela 2.

Uma das possíveis razões para este alto número de desistências é o fato de existir uma grande dificuldade, por uma parte significativa dos alunos cursando disciplinas introdutórias, em compreender certos conceitos abstratos de programação e no desenvolvimento do raciocínio lógico/algorítmico (Rodrigues 2013). Além disto, tratam-se de disciplinas que requerem um maior suporte por parte do professor, que precisa conduzir de perto o trabalho dos alunos. Entretanto, prestar auxílio individual pode sobrecarregar o docente, de modo que, al-

Tabela 2 – Censo da educação superior – 2005 a 2014.

Ano	Cursos	Vagas Oferecidas	Candidatos Inscritos	Concluintes
2005	1.001	102.424	162.655	19.344
2006	1.028	164.367	164.367	19.886
2007	1.744	197.282	304.298	33.447
2008	1.813	205.092	307.564	34.392
2009	1.929	278.708	376.675	41.117
2010	1.988	327.326	376.985	40.302
2011	1.982	238.673	511.510	37.694
2012	2.105	250.168	609.429	38.372
2013	2.140	248.411	650.234	36.752
2014	2.184	331.399	742.344	37.347

Fonte: Censo da Educação Superior (DEED, 2015).

guns estudantes não consigam acompanhar o andamento do curso de forma satisfatória. Isto desestimula o aluno, gerando evasão e reprovação (Gomes et al. 2008).

Com o intuito de contribuir para a melhoria deste quadro, surgiram sistemas *online* que buscam, além de proporcionar treinamento, preparar o estudante para competições de programação. Estes sistemas têm como principal característica a capacidade de servirem como repositórios *online* de problemas a serem resolvidos, sendo dotados de ferramentas de correção automáticas para estes problemas. Entretanto, por mais que as plataformas atuais possam atender aos estudantes, ainda não apresentam funcionalidades que despertem o interesse dos professores. Dessa forma, melhorias podem ser feitas para que os docentes passem a considerar tais ferramentas como auxílio didático em suas aulas.

A seguir foram avaliados parâmetros de oito das mais utilizadas plataformas de auxilio e resolução de problemas de computação. Foi estudado o diferencial que cada uma destas plataformas oferece do ponto de vista tanto do estudante quanto do professor como usuários. Os principais pontos de cada uma das ferramentas analisadas são apresentados na Tabela 3.

Dentre os estudados, nenhum sistema possui uma classificação satisfatória para seus problemas pensando em um plano didático. Mesmos aqueles que possuem grande número de categorias para distinção, não apresentam separação realmente eficiente que facilite a busca por parte dos estudantes e a definição e classificação de novos problemas por parte de professores em disciplinas.

Outra carência analisada é que, mesmo que estas plataformas possuam repositório de problemas e juízes *online* eficientes, estes realizam apenas a função de corrigir e avaliar as soluções apresentadas por cada usuário. Não existem métodos que possam auxiliar no aprendizado do usuário que pode apresentar dificuldades para resolver determinado problema. Alguns dos sistemas estudados possuem plataformas de ensino com vídeo aulas, material de apoio e

Tabela 3 – Análise de sistemas de juízes *online*.

Online Judge	SPOJ	URI	UVa	Project Euler	TopCoder	CodingBat	Coderbyte	HackrRank
Número de problemas	20000+	700+	4300+	600+	70+	300+	200+	300+
Capacidade de gerir competições	Sim	Sim	Sim	Não	Sim	Não	Não	Sim
Possui plataforma de aprendizado	Não	Não	Não	Não	Sim	Sim	Sim	Sim
Linguagens de programação compatíveis	60+	12	6	Qualquer	Qualquer	2	10	50+
Número de categorias de problemas	5	9	15	1	4	17	26	18
Possui fórum de discussão	Sim	Sim	Não	Sim	Sim	Não	Sim	Não
Possui suporte para professor	Não	Sim	Não	Não	Não	Não	Sim	Não
Possui Ranqueamento	Sim	Sim	Sim	Sim	Não	Não	Não	Não

Fonte: Elaborado pelo autor.

fórum de discussões, porém tais funcionalidades podem ser melhoradas, ou reformuladas de forma a melhorar o processo de aprendizado do aluno.

Problemas bem formulados e categorizados de forma a facilitar o aprendizado de alunos, além de sua utilização por professores são oferecidos no contexto das competições acadêmicas. Competições as quais são capazes ainda de incentivar a autonomia na busca por conhecimentos por parte do estudante.

Os problemas e os dados normalmente são públicos. Entretanto, da forma que o aluno que busca, dificilmente encontrará suporte adequado ou uma base inicial para seus estudos. Há ainda a dificuldade por parte de professores que optaram por usar repositórios de problemas em encontrar e classificar os que o atendam seus requisitos didáticos de forma eficiente.

Um projeto, sendo bem organizado, que recebe apoio de diversos colaboradores, pode apresentar resultados melhores do que se fosse realizado de forma individual. Isto se deve ao fato de que as ideias passam por um processo de depuração, mediante sugestões e críticas (Turoff e Hiltz 1982).

Portanto, este trabalho se justifica pela necessidade de um sistema que seja colaborativo, além de personalizável. Busca-se desenvolver um ambiente de fácil acesso, que possa receber a colaboração de professores e ofereça soluções de maneira personalizada, que se apresente como um espaço para uma maior interação, crescimento e enriquecimento intelectual da comunidade.

1.2 Objetivo geral

Considerando o problema apresentado, que se relaciona ao fato de as plataformas existentes não possuírem meios de tornarem seu uso personalizável e colaborativo, objetiva-se com este trabalho, modelar a camada de interação com o professor como usuário do framework conceitual colaborativo personalizável baseado em reuso CETECOP *(Collaborative*

Environment for Teaching Computer Programing).

1.3 Objetivos específicos

- Detectar os principais requisitos esperados por estudantes e professores para este tipo de sistema;
- Permitir a colaboração entre diferentes professores ou gestores de problemas em um mesmo exercício;
- Permitir a integração do sistema com softwares avaliadores automáticos de problemas, permitindo a submissão e avaliações de soluções para os problemas disponíveis;

1.4 Limitações do trabalho

Dentro os módulos propostos da arquitetura conceitual CETECOP, será desenvolvido apenas o gerenciador de problemas *(ProblemManager)*. Poderá ser utilizado posteriormente os módulos de gerenciamento de usuários *(UserPerformanceManager)* e o modulo de julgamento *(JudgeSystem)* para a integração do sistema.

1.5 Estrutura do trabalho

Este trabalho é dividido em 6 capítulos, sendo o primeiro para a introdução, onde é descrita a contextualização do problema, sua justificativa, os objetivos do trabalho, e resultados esperados. Para o segundo capítulo é apresentado o levantamento bibliográfico, relatando os conceitos essenciais. No terceiro capítulo são apontados os principais trabalhos correlatos. O quarto capítulo é composto pela metodologia abordada neste trabalho e os materiais utilizados para seu desenvolvimento. Finalmente no quinto capítulo é apresentado o desenvolvimento do trabalho, no sexto o estudo de caso realizado e no sexto a suas conclusões.

2 Fundamentos Históricos, Teóricos e Metodológicos

2.1 Competições de programação

Competições de programação são eventos nos quais os participantes devem apresentar soluções de problemas relacionados à computação, lógica e matemática em um determinado período de tempo. Existem dois formatos de competições de programação: de curto e de longo prazo. Para competições de curto prazo cada rodada dura de uma a três horas, enquanto para as de longo prazo podem durar de alguns dias a até mais de um mês.

Estes eventos são reconhecidos e hospedados por *sites* de treinamentos especializados, além de empresas como Google (Google 2021) e Facebook (Facebook 2021). Os principais sistemas que disponibilizam problemas *online* com corretores automáticos costumam também realizar competições periodicamente (Combéfis e Wautelet 2014).

2.2 Repositório de problemas para treinamento

Sistemas que armazenam e organizam problemas de diversos níveis de dificuldade são chamados Repositórios de Problemas para Treinamento (RPT). São utilizados por alunos que desejam treinar para competições de programação ou aqueles que desejam estudar e praticar seus conhecimentos. Tais plataformas são utilizados em conjunto com sistemas de autojulgamento, permitindo assim que o estudante obtenha de forma fácil um determinado problema que deseja resolver. Os problemas armazenados nestes repositórios podem ser problemas desenvolvidos pela comunidade ou vindos de competições de programação anteriores.

2.3 Sistemas de autojulgamento

Sistemas de autojulgamento, também conhecidos como sistemas de avaliação automática ou juízes *online* são *softwares* que oferecem testes e correção automática de problemas de linguagem de programação, através de compilação, execução e casos de testes, dado o código fonte (Kurnia e Cheang 2001). São amplamente usados em competições de programação e também para treinamento, sendo que muitos destes sistemas possuem a capacidade de gerir suas próprias competições.

Os problemas podem apresentar restrições, sejam de tempo de resolução, de limite de memória ou restrições de segurança. A saída do código fonte apresentado é capturada pelo sistema e comparada com a resposta e então, o sistema retorna o resultado. Sistemas de autojulgamento apresentam *rankings*, onde são exibidos os usuários com as maiores pontuações, incluindo números de soluções aceitas e menor tempo de execução para determinados problemas.

A seguir serão apresentados dois juízes *online*, o BOCA (BOCA 2018) e DOMJudge (DOMJudge 2021), sendo estes, *softwares* livres, possibilitando assim possíveis adaptações. Posteriormente serão mostrados também exemplos de juízes *online* de código fechado os quais também possuem seus próprios repositórios de problemas, mais especificamente o TopCoder (TopCoder 2021), SPOJ (SPOJ 2021), URI (URI 2021) e UVa (UVa 2021).

2.3.1 Juiz *Online* BOCA

De forma a promover uma forma fácil de gerir as competições, foi desenvolvido em 2002 o BOCA, um sistema baseado nas regras da ICPC e distribuição Linux baseada em maquinas virtuais, conhecidas como Maratona Linux (Campos e Ferreira 2004). Foi criada inicialmente com base na Red Hat Linux (Red Hat 2021), e desenvolvida de forma que não necessite de um especialista em distribuição Linux para sua instalação.

As correções são realizadas de forma online. Desta forma os participantes enviam suas resoluções e elas são corrigidas trazendo o resultado o mais rápido possível. Isto permite que competidores possam refazer as questões que apresentarem erros. Diferentemente de uma prova comum, onde estudantes possuem apenas uma chance de fazer cada questão, em competições de programação é comum que seus participantes possuam inúmeras chances de realizar os problemas. A única restrição neste caso é o limite de tempo para realização.

O sistema pode apresentar respostas distintas para cada competição ou edições diferentes, tornando assim útil a padronização destas respostas no sistema a ser desenvolvido. Na Tabela 4 são apresentads sete tipos de respostas comuns para a resolução de um problema.

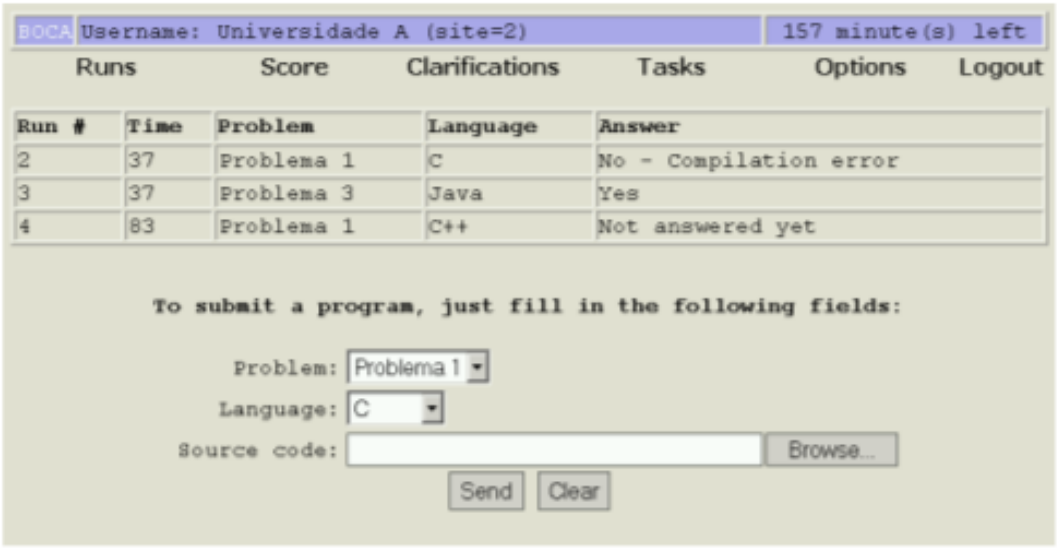
Tabela 4 – Respostas possíveis para resolução de problemas do BOCA.

Resposta	Descrição
YES	O código foi devidamente compilado, executado e gerou a resposta esperado para o problema.
NO: Wrong answer	Indica que o código compilou e executou, porém não apresentou a resposta esperada.
NO: Time-limit Exceeded	Indica que o código compilou e executou, porém gastou mais tempo de execução do que o permitido para o problema dado.
NO: Runtime Error	Indica que o código compilou porém exibiu um erro em sua execução dado uma entrada dos juízes.
NO: Compilation Error	Indica que o código não compilou propriamente no computador dos juízes.
NO: Presentation Error	Indica que o código compilou, executou e apresentou o resultado esperado, porém com problemas de visualização, como por exemplo letras maiúsculas/minúsculas e acentos.
NO: If possible, contact staff	Esta resposta indica que o participante deve entrar em contato com os funcionários para tratar do erro, se for possível.

Fonte: BOCA Online Contest Administrator: Manual para times – versão Outubro de 2011.

É possível observar três tipos de respostas dadas pelo sistema na Figura 2. A primeira apresenta erro de compilação, a segunda foi corrigida e aceita, e a terceira ainda não possui resposta. A imagem ilustra a interface que o BOCA retorna ao usuário no momento de submissão de seus problemas.

Figura 2 – Interface de submissões para um determinado time do BOCA.



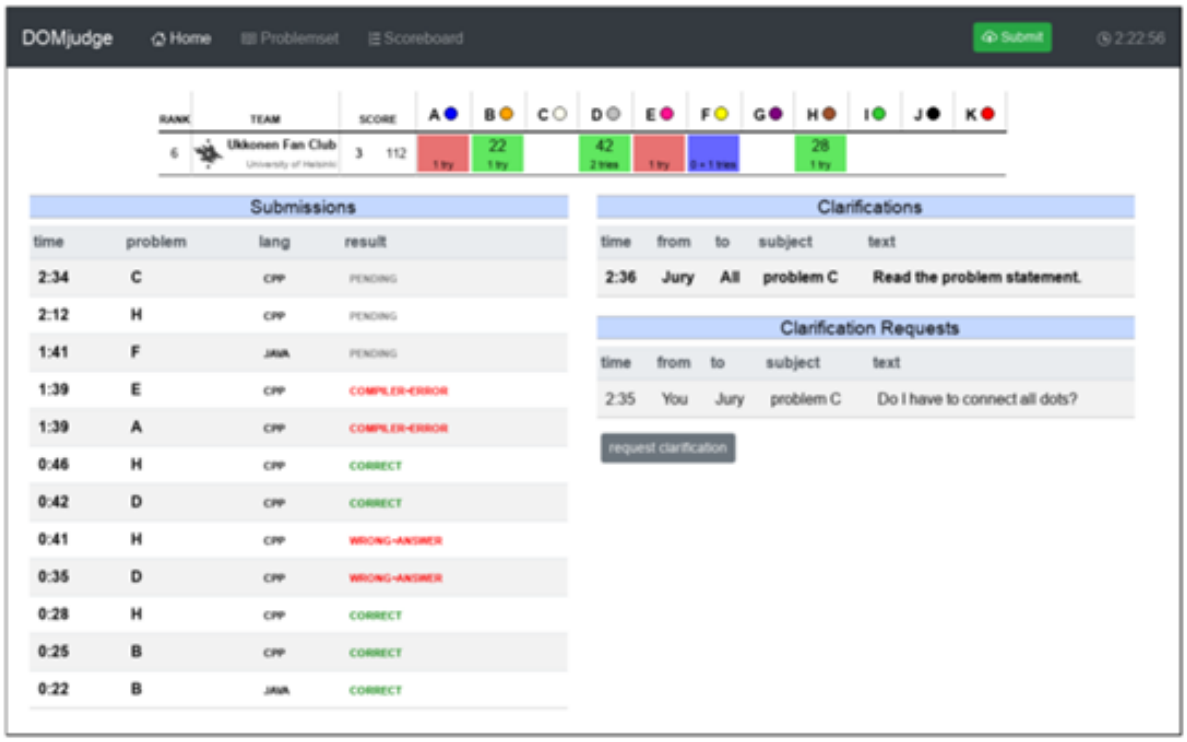
Fonte: (CAMPOS, 2004).

2.3.2 Juiz Online DOMJudge

DOMJudge é um sistema de juiz *online* automatizado utilizado nas competições de programação ICPC da ACM. É usado em competições regionais e também em finais mundiais. É um sistema com interfaces *web* tanto para os times participantes quanto para os juízes escrito em PHP (PHP 2021), Shell-script (Goebel 2003) e C/C++, utilizando um servidor MySQL (MYSQL 2021). É um *software* de código aberto, tornando fácil, portanto adaptações e colaborações da comunidade.

Da mesma forma que o BOCA, o juiz *online* DOMJudge possui uma lista de respostas possíveis para a entrega de soluções dos participantes. No entanto, esta lista apresenta variações de retorno por não haver um padrão a ser seguido. São apresentadas algumas destas possibilidades. A Figura 3 ilustra a interface que o DOMJudge retorna aos usuários como página de visão geral.

Figura 3 – Interface de submissões para um determinado time do BOCA.



Fonte: (DOMJudge, Manual para Times, 2018).

O DOMJudge apresenta ainda uma tela de visualização onde é possível ver o placar em tempo real dos competidores, tornando fácil o acompanhamento de cada time na competição, além da relação de problemas resolvidos por equipe. Esta pode ser visualizada na Figura 4.

Figura 4 – Interface do placar de pontos do DOMJudge.

Fonte: (DOMJudge, Manual para Times, 2018).

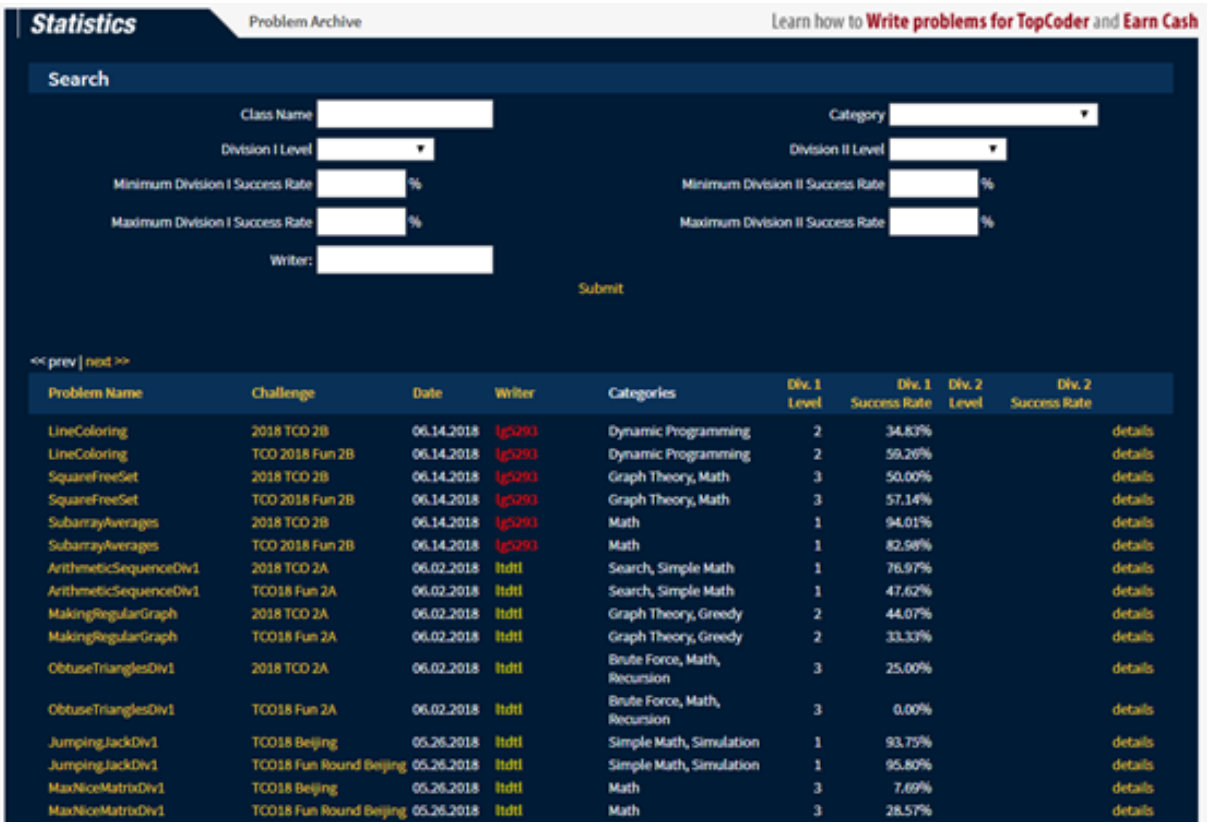
O DOMJudge apresenta uma interface simples para os times participantes. Porém, para os juízes e administradores ele possui mais funcionalidades, como por exemplo a possibilidade de julgar novamente algum problema e exibir informações detalhadas. Existe ainda uma API *(Application Programming Interface)* REST *(Representational State Transfer)* para que seja possível criar extensões para o DOMJudge, tornando-o assim uma boa opção para compor os módulos de *Contest Manage*r e *Judge System*.

2.3.3 TopCoder

A TopCoder é uma empresa de financiamento colaborativo, que hospeda e administra competições *online* de programação. A mais conhecida é a TopCoder *Open Tournament*, uma competição que envolve *design*, *Data Science* e programação. É também organizada uma série de eventos regionais, além de pequenas competições conhecidas como SRM *(Single Round Matches)* realizadas quinzenalmente ou a cada semana.

A ferramenta possui um repositório de problemas *online*, visto na Figura 5, o qual apresenta quatro tipos de modalidades, desafios de *design*, desafios de desenvolvimento, desafios de *data Science*, e programação competitiva.

Figura 5 – Arquivo de Problemas do TopCoder.



Problem Name	Challenge	Date	Writer	Categories	Div. 1 Level	Div. 1 Success Rate	Div. 2 Level	Div. 2 Success Rate	
LineColoring	2018 TCO 2B	06.14.2018	lg5293	Dynamic Programming	2	34.83%			details
LineColoring	TCO 2018 Fun 2B	06.14.2018	lg5293	Dynamic Programming	2	59.26%			details
SquareFreeSet	2018 TCO 2B	06.14.2018	lg5293	Graph Theory, Math	3	50.00%			details
SquareFreeSet	TCO 2018 Fun 2B	06.14.2018	lg5293	Graph Theory, Math	3	57.14%			details
SubarrayAverages	2018 TCO 2B	06.14.2018	lg5293	Math	1	94.01%			details
SubarrayAverages	TCO 2018 Fun 2B	06.14.2018	lg5293	Math	1	82.98%			details
ArithmeticSequenceDiv1	2018 TCO 2A	06.02.2018	ltdtl	Search, Simple Math	1	76.97%			details
ArithmeticSequenceDiv1	TCO18 Fun 2A	06.02.2018	ltdtl	Search, Simple Math	1	47.62%			details
MakingRegularGraph	2018 TCO 2A	06.02.2018	ltdtl	Graph Theory, Greedy	2	44.07%			details
MakingRegularGraph	TCO18 Fun 2A	06.02.2018	ltdtl	Graph Theory, Greedy	2	33.33%			details
ObtuseTrianglesDiv1	2018 TCO 2A	06.02.2018	ltdtl	Brute Force, Math, Recursion	3	25.00%			details
ObtuseTrianglesDiv1	TCO18 Fun 2A	06.02.2018	ltdtl	Brute Force, Math, Recursion	3	0.00%			details
JumpingJackDiv1	TCO18 Beijing	05.26.2018	ltdtl	Simple Math, Simulation	1	93.75%			details
JumpingJackDiv1	TCO18 Fun Round Beijing	05.26.2018	ltdtl	Simple Math, Simulation	1	95.80%			details
MaxNiceMatrixDiv1	TCO18 Beijing	05.26.2018	ltdtl	Math	3	7.69%			details
MaxNiceMatrixDiv1	TCO18 Fun Round Beijing	05.26.2018	ltdtl	Math	3	28.57%			details

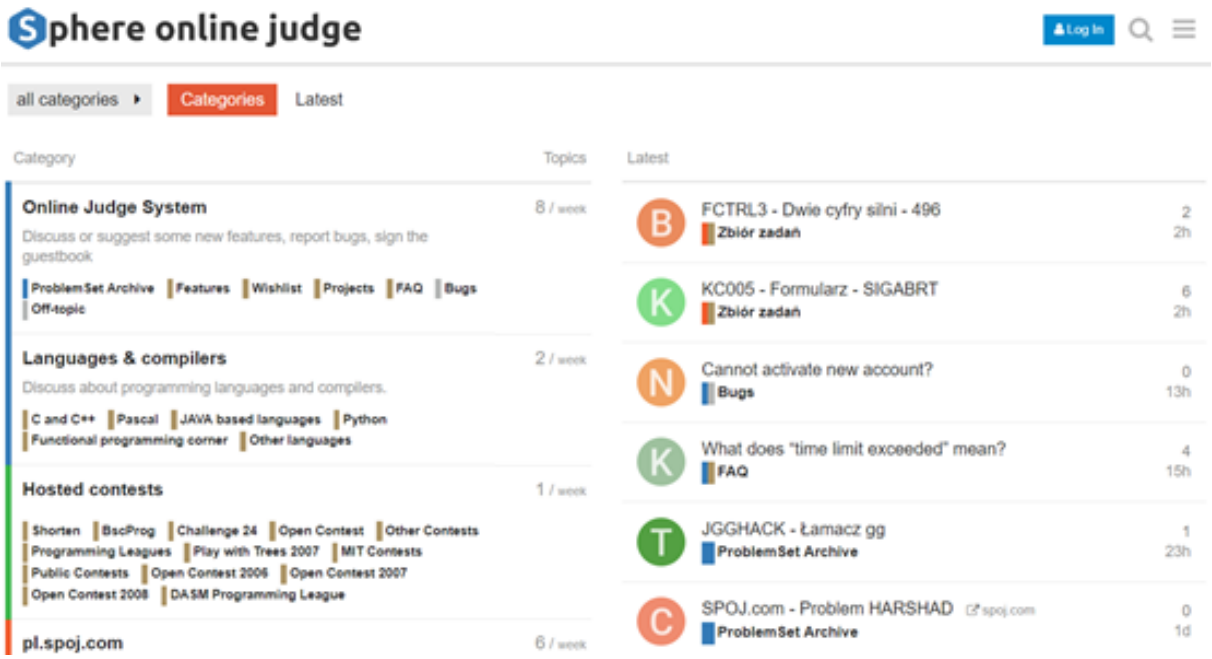
Fonte: (TopCoder, Problem Archive, 2018).

2.3.4 *Sphere Online Judge*

O SPOJ *(Sphere Online Judge)* é um sistema centrado em torno de um juiz online. Assim como as outras ferramentas descritas, ele serve como avaliador automático de problemas de programação resolvidos por usuários. Possui um suporte para mais de 60 linguagens de programação, incluindo C/C++, Pascal, Perl, Python, Ruby, Hasckell, Ocaml. Além disso, está disponível em inglês, polonês, vietnamita e português (SPOJ 2021).

Na Figura 6 é mostrado um fórum de discussões onde é possível diferentes usuários se comunicarem através de tópicos públicos.

Figura 6 – Fórum de discussão de problemas do SPOJ.



Fonte: (Discuss SPOJ, 2018).

A ferramenta apresenta, ainda, um sistema de testes flexível que suporta a interação dinâmica com os programas submetidos e saídas personalizáveis dos resultados de avaliação. Uma característica interessante para este trabalho é a capacidade de gerenciamento intuitivo de conteúdo, que permite que os usuários criem seus próprios concursos rapidamente (em torno de minutos) e façam uso de tarefas já disponíveis no sistema (SPOJ 2021).

2.3.5 UVa *Online Judge*

Hospedado pela Universidade de Valhadolide na Espanha, o UVa *Online Judge* é um juiz *online* automático criado em 1995 que possui um repositório com mais de 4300 problemas. Assim como as outras plataformas já descritas, o UVa realiza suas próprias competições, possibilitando a resolução de problemas de programação nas linguagens ANSI C, C++, Pascal, Java, C++ e Python (UVa 2021). Há também um *site* relacionado ao UVa, chamado de UVa *Toolkit*, onde o usuário pode testar possíveis entradas em soluções já aceitas para diversos problemas presente no UVa.

Uma diferença do UVa em relação aos outros juízes online é a forma de classificação dos problemas, onde estes podem ser organizados por autores, maratonas de programação anteriores ou mesmo por livros.

A Figura 7 apresenta a interface do UVa relativa à forma de organização dos problemas na plataforma.

Figura 7 – Organização de Problemas do UVa *Online Judge*.

Fonte: (UVa Online Judge, 2018).

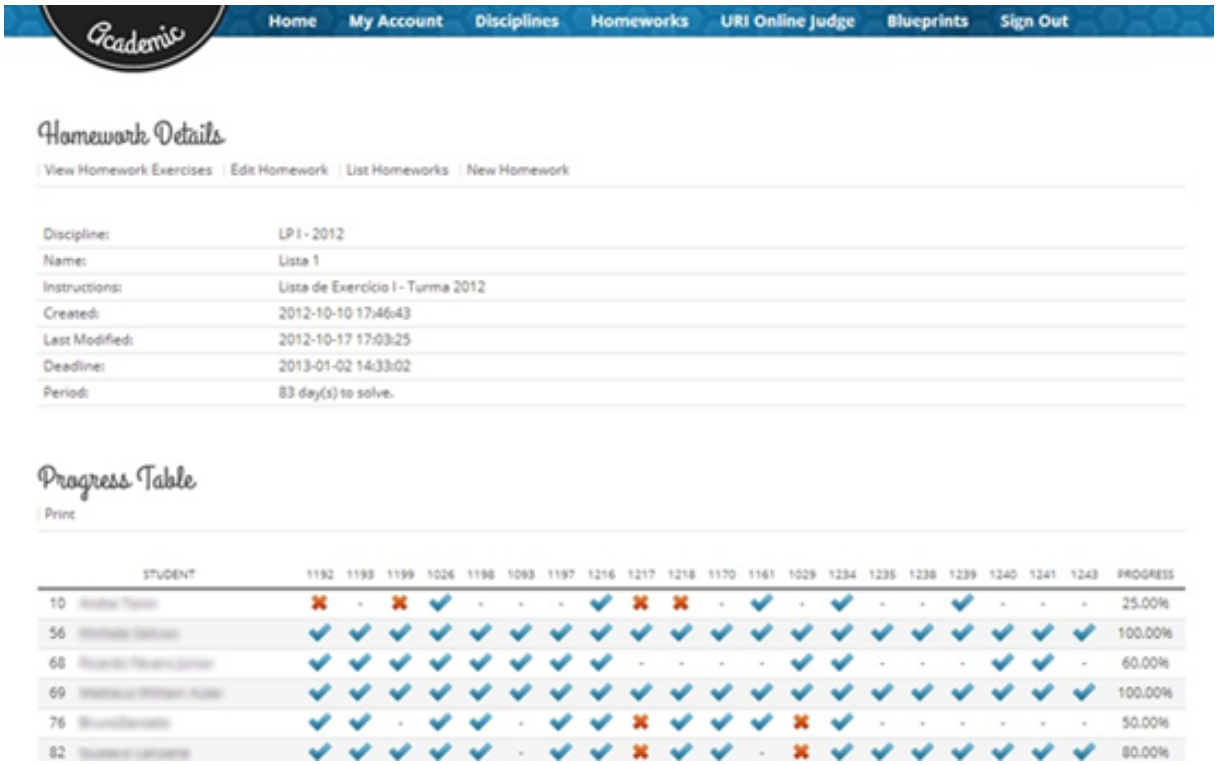
2.3.6 URI *Online Judge*

Um projeto desenvolvido na URI (Universidade Regional Integrada, Brasil) em 2012, o URI *Online Judge* é um sistema *web* nos moldes da ICPC que além de permitir que seus usuários resolvam problemas de computação, possibilita também que professores acompanhem o progresso de seus alunos no decorrer do curso (URI 2021).

É uma plataforma que pode ser utilizada tanto pelo aluno quanto pelo professor ou treinador, auxiliando assim não só o treinamento para competições de competição, mas também o aprendizado em cursos de computação. Para isso é utilizado o modulo URI *Online Judge Academic*. Este módulo apresenta funcionalidades interessantes, como visualizar o código fonte dos alunos, histórico de submissão do aluno, sistema identificador de possíveis plágios, além de ferramentas para comparação entre os usuários e o ranqueamento geral (Bez e Tonin 2014).

Neste módulo, representado pela Figura 8, é possível ainda a criação de grupos e disciplinas. O professor ou treinador podem auxiliar seus alunos que possuírem maiores dificuldades, enquanto os outros são livres para explorar o sistema e novos problemas (Bez e Tonin 2014).

Figura 8 – Organização de Problemas do UVa Online Judge.



Fonte: (UVa Online Judge, 2018).

As funcionalidades descritas nos juízes *online* apresentados, permitem uma maior interação dos usuários entre si e um maior enriquecimento do conteúdo presente nestes últimos sistemas. São parâmetros interessantes a serem considerados no desenvolvimento do módulo de interação com o usuário.

2.4 Ferramentas de ensino *online*

Existem também outras ferramentas *online* que oferecem suporte ao aprendizado de programação. Estas são focadas no ensino de programação e não necessariamente voltadas para competições. Podem apresentar repositórios de problemas, mas não apresentam um juiz online que analise a resolução seguindo o estilo de competições de programação. Serão apresentados sistemas que oferecem funcionalidades interessantes didaticamente, que podem auxiliar os estudantes que irão utilizar a ferramenta.

2.4.1 *Project* Euler

Project Euler é um *site* de resolução de problemas matemáticos de diversos níveis de dificuldade que requer conhecimento em programação por parte do usuário. Uma das funcionalidades do *site* é que após ser resolvido o problema, ganha-se acesso a um fórum de discussões sobre o mesmo, onde é permitido a discussão da solução e o compartilhamento da resposta com outros usuários (Project Euler 2021).

Na Figura 9 pode-se ver a página inicial do acervo de problemas disponível no *Project* Euler.

Figura 9 – Arquivo de problemas do Project Euler.

ID	Description / Title	Solved By
1	Multiples of 3 and 5	759354
2	Even Fibonacci numbers	609964
3	Largest prime factor	436759
4	Largest palindrome product	388676
5	Smallest multiple	396869
6	Sum square difference	399242
7	10001st prime	341752
8	Largest product in a series	288553
9	Special Pythagorean triplet	291521
10	Summation of primes	267362

Fonte: (Project Euler, About, 2018).

2.4.2 CodingBat

CodingBat é um *site* livre que permite o treinamento nas linguagens Java e Python. É voltado para iniciantes, apresentando diversos problemas a serem resolvidos, que se encontram separados por categorias como lógica: vetores, funções, recursão, entre outras. Através de um editor já presente no *site* é possível a resolução dos problemas de forma direta no site, dispensando a instalação de ferramentas (CodingBat 2021).

2.4.3 Coderbyte

Coderbyte se assemelha ao CodingBat, que possibilita a resolução de problemas de programação diretamente em seu próprio editor *online*. Apresenta as opções de nove linguagens de computação disponíveis, além de prover tutorias, vídeos e cursos introdutórios (Coderbyte 2021).

2.5 Sistemas de controle de versões

Um sistema de controle de versões, também conhecido como sistema de controle de versionamento, do inglês VCS (*version control system*) ou ainda SCM (*source code management*) é um *software* que tem como finalidade gerenciar diferentes versões de um documento qualquer no seu processo de desenvolvimento. Tais sistemas são utilizados com maior frequência no desenvolvimento de *software* para controlar suas diferentes versões, além de registrar o histórico e desenvolvimento dos códigos fontes e da documentação.

Dentre os mais comuns, pode-se citar das soluções livres o *CVS, Mercurial, Git* e *SVN*, e das soluções comerciais o *SourceSafe, TFS, PVCS* e *ClearCase*. No desenvolvimento de *software* livre, em 2015 o mais utilizado foi o Git (Stack Overflow 2015), que possui repositórios no GitHub. Algumas empresas também adotam o Git, como por exemplo a Microsoft (Microsoft 2017).

2.5.1 Termos utilizados

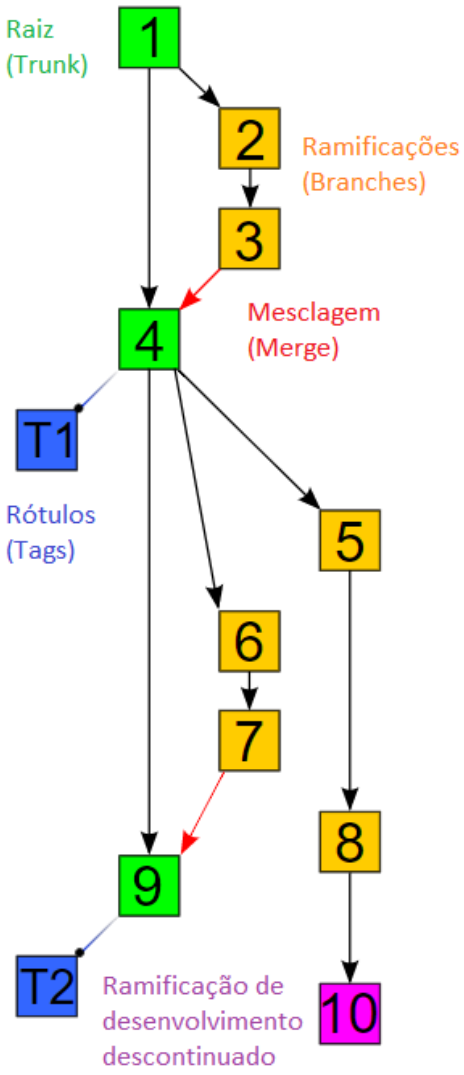
A seguir é apresentada uma lista de termos comumente utilizados em sistemas de controle de versões seguidos por uma breve explicação sobre o mesmo (Gist GitHub 2021).

- Atualização (*Update*): Acontece quando atualiza na cópia local novas informações vindo do servidor, sendo estas prováveis mudanças feitas por outro desenvolvedor;
- Baixar (*Checkout, Check-out*): É utilizado quando não existe cópia local do projeto, e assim é necessário baixa-lo por inteiro do servidor;
- Conflito (*Conflict*): Acontece quando há uma alteração simultânea, entre uma atualização e uma submissão, de um mesmo documento por usuários diferentes;
- Cópia local (*Working copy, Working áre*a): É usualmente uma pasta no sistema operacional do desenvolvedor que possui a cópia da versão mais recente do projeto;
- Efetivar, Submeter (*Commit, Submit, Check-in*): É quando acontece o envio das alterações da cópia local ao servidor;
- Marcação (ie): É correspondente ao processo de dar nome a um determinado estado do repositório;
- Mesclar (*Merge, Integration*): É quando acontece a união de um mesmo documento alterado por mais de um desenvolvedor ao mesmo tempo. É feito uma comparação das diferenças e mesclando mantendo as duas alterações;
- Mesclação reversa (*Reverse Integration*): Acontece quando uma ramificação é mesclada a raiz;
- Pedido de Mesclagem (*Merge request*): É uma solicitação a um desenvolvedor especifico para a mesclagem de um documento o qual outro desenvolvedor realizou alterações;
- Raíz (*Trunk, Head*): É o caminho inicial que não se quebrou em ramificações;

- Ramificação (*Branch*): Surge quando a linha de desenvolvimento se quebra em duas ou mais linhas.

A Figura 10 representa um exemplo de visualização do histórico de um determinado projeto utilizando um sistema de controle de versões.

Figura 10 – Exemplo de visualização do histórico de um projeto utilizando sistema de controle de versões.



Fonte: Desenvolvido pelo Autor.

2.5.2 Git

Um *software* livre, o Git é distribuído sob os termos da versão 2 da GNU *General Public License*. Foi projetado inicialmente para o desenvolvimento do kernel Linux, mas foi eventualmente adotado por vários outros projetos. Todos diretórios de trabalho do Git atuam como repositórios com total habilidade de acompanhamento de revisões, não dependendo de acesso a uma rede ou servidor central, além de um histórico completo destas (Git 2021).

O Git possui duas estruturas de dados, sendo estas um índice mutável e um banco de dados de objetos de acréscimo imutável. O índice, que provê informações sobre o diretório de trabalho e a próxima revisão a ser cometida, serve como ponto de conexão entre o banco de dados de objetos e a árvore de trabalho.

O banco de dados de objetos armazena quatro tipos de objetos:

- *Blob*: Conteúdo de um arquivo. Estes objetos não possuem metadados;
- *Tree* (árvore): Equivalente a um diretório, armazenam uma lista de nomes de arquivos, cada um podendo conter *bits* correspondentes ao tipo, nome do *blob*, nome da árvore, ligação simbólica, além do conteúdo de diretório que pertence a este nome. Estes objetos descrevem o estado da árvore de diretório;
- *Commit* (entrega): Liga as árvores de objetos junto a um histórico. Possui o nome de uma árvore de objetos, sua datação, uma mensagem de *log*, e os nomes de zero ou mais objetos pai do *commit*;
- *Tag* (rótulo): Involucro que referencia outros objetos e pode armazenar metadados adicionais. É usado comumente para armazenar assinaturas digitais de objetos *commit* que correspondem àquele release de dados que estão sendo monitorados pelo Git.

2.5.3 GitHub

O GitHub é uma plataforma de hospedagem e controle de versão de código fonte utilizando o Git. Ele possibilita que desenvolvedores, utilitários ou qualquer tipo de usuário cadastrado contribuam em projetos privados ou abertos, de qualquer lugar do mundo. É uma plataforma amplamente usada por programadores para compartilhamento de projetos, onde outros usuários podem contribuir, ou mesmo para divulgação de seus trabalhos (GitHub 2021).

O número de usuários do GitHub chega a mais de 36 milhões, contribuindo em projetos comerciais ou pessoais. Hoje a plataforma abriga mais de 100 milhões de projetos (GitHub 2021). O GitHub oferece também suporte a um suporte de organização que é utilizado por aqueles que desejam uma escala maior para seus projetos.

2.5.4 GitLab

O GitLab é um gerenciador de repositórios de *software* baseado em Git. Possui suporte a Wiki, além de gerenciamento de tarefas (GitLab 2021). Esta ferramenta funciona de maneira similar ao GitHub, porem o diferencial é que o GitLab permite que desenvolvedores armazenem seus códigos e projetos em servidores próprios, não necessitando da utilização de servidores de terceiros. É um *software* livre, distribuído pela licença MIT (Olanoff 2011).

O código da plataforma foi originalmente escrito em Rubi, e posteriormente outras partes foram reescritas em Go. É utilizado por mais de 100000 organizações, incluindo a Marinha do Brasil, IBM, NASA, Alibaba, CERN e SpaceX (Degeler 2014).

3 Estado da Arte

Este capítulo apresenta alguns trabalhos encontrados onde autores apresentam modelos e pesquisas semelhantes as feitas neste projeto. Tais trabalhos disponibilizam funcionalidades desejáveis para o desenvolvimento da plataforma proposta, apresentada neste trabalho.

3.1 Pacheco (2010)

A dissertação de Pacheco (Pacheco 2010) propõe um sistema de avaliação automática, que contenha uma componente central suficientemente flexível para permitir sua integração com outras plataformas como o Moodle, SIGEX e sistemas de eBooks. Para isto, foi realizado um estudo sobre os sistemas de autojulgamento mais conhecidos, e então foi feita uma análise de como estes sistemas são utilizados no meio acadêmico e quais boas práticas devem ser seguidas para o desenvolvimento de um sistema afim.

O sistema desenvolvido apresentou duas funcionalidades inovadoras, sendo estas a possibilidade de definir casos de testes com atributos, tais como mensagens de feedback personalizadas e um mecanismo de código esqueleto.

3.2 Xavier (2011)

Dando continuidade ao trabalho de mestrado de Pacheco (Pacheco 2010), por este não ter sido considerado como preparado para ser utilizado ativamente, Xavier (Xavier 2011) faz uma análise do sistema de avaliação automática desenvolvido, comparando-o com avaliadores automáticos já existentes, além de um estudo sobre módulos avançados para sistemas deste tipo a fim de remover as carências desta ferramenta.

Entre as funcionalidades avançadas ausentes, identificadas como necessárias para o sistema, pode-se citar a detecção de plágio e a análise estática da qualidade do código, sendo a primeira focada na motivação do aluno, para que este possa aprender por si só, e a última priorizando a melhoria das técnicas e estilo de programação dos estudantes. A detecção de plágio é proposta fazendo uso de diversas métricas, enquanto para a análise estática foi empregado o uso de uma API (*Application Programming Interface*) do Moodle chamada Crot, além da criação de uma nova API, que permite a integração com possíveis novos detectores de plágio para seu sistema.

3.3 Quadros (2012)

No trabalho de Quadros (QUADROS 2012) é feito um estudo de casos para analisar o aprendizado de Espanhol utilizando um Sistema de atividades virtual, chamado Livemocha. Este é um *site* colaborativo que permite a interação entre alunos e professores ao redor do mundo, possibilitando que ambos possam contribuir para a comunidade. Neste trabalho,

Quadros (QUADROS 2012) identificou que os sujeitos integrados à comunidade virtual, e com orientação presencial, apresentaram maior motivação em relação a sua aprendizagem. Entretanto, notou-se ainda que conforme as atividades se repetiam e as interações presenciais foram reduzidas, houve um aumento do desinteresse por parte dos alunos.

3.4 Alonso e Miranda (2014)

É descrito no projeto (Alonso e Miranda 2014) o desenvolvimento de módulos de treinamento para participantes da competição de programação da ACM, a ICPC. É apresentado um estudo dos mais importantes mecanismos de aprendizado que resulta na elaboração de uma ferramenta de testes automáticos de problemas de computação, além de uma funcionalidade de segurança que define tempos limite de execução. O sistema incorpora ainda novas funcionalidades, como o suporte para compartilhamento de trechos de códigos e desafios propostos nos moldes das finais da ICPC.

Os autores, após analisarem as ferramentas já existentes como o já citado neste artigo, DOMJudge (ELDERING e WERTH. 2014), OpenJudgeSystem (NIKOLAY e TODOR 2014), MoodleOnlineJudgee (Zhigang 2014) e ACMDHU-OnlineJudge (UNIVERSITY 2009), decidem por não rescreverem todos módulos do projeto, abordagem diferente da adotada por este trabalho.

4 Materiais e Métodos

4.1 Tipo de pesquisa

Esta pesquisa se classifica como exploratória, pois segundo Wazlawick, entram aqui estudos de casos, para a partir de um problema abstrato, ocorrer a realização do levantamento bibliográfico sobre o objeto estudado e, dessa forma, buscar métodos e critérios eficazes para a resolução do mesmo (WAZLAWICK 2009).

4.2 Métodos

A seguir são apresentados os passos necessários para a realização deste trabalho:

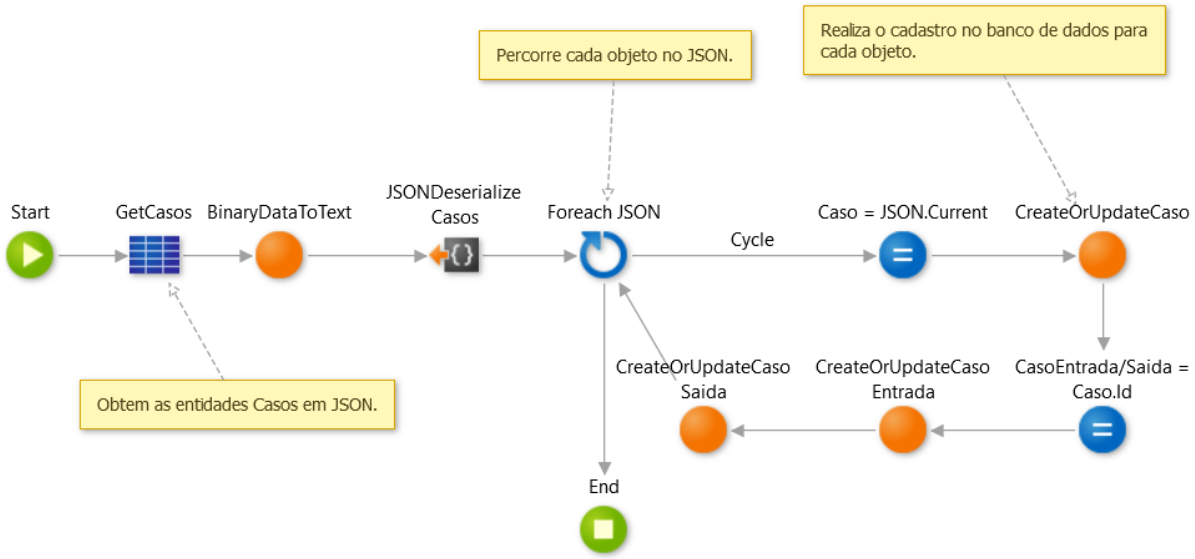
- Analisar ferramentas de autojulgamento e repositórios de problemas em busca de características desejáveis e limitações, priorizando *softwares* livres;
- Analisar os módulos já prontos do CETECOP para permitir uma integração eficiente com o modulo a ser desenvolvido;
- Desenvolver o módulo proposto de forma a contemplar as funcionalidades requeridas;
- Realizar a integração do módulo desenvolvido com os demais módulos.

4.3 Materiais e ferramentas

Para o desenvolvimento do protótipo utilizado na validação do projeto foi feito um sistema *web* utilizando a plataforma OutSystems (OutSystems 2018).

Esta é uma plataforma de desenvolvimento *low-code* que permite desenvolver rapidamente aplicativos de maneira visual. Através dos componentes que a ferramenta fornece é possível construir telas, seja para aplicações *web* ou aplicativos, fluxos lógicos, conexão com bancos de dados e consumir serviços REST/SOAP e SAP. É possível também a integração com componentes criados pela comunidade, ou mesmo próprios criados pelo desenvolvedor. Na Figura 11 é possível visualizar um exemplo de fluxo lógico referente ao cadastro de um novo problema na sistema desenvolvido.

Figura 11 – Exemplo de fluxo lógico: Importação de dados via JSON.



Fonte: Elaborado pelo Autor.

A principal vantagem da utilização de tal plataforma é a agilidade no desenvolvimento oferecido, possibilitando a criação de páginas completas de cadastro, alteração, pesquisa e listagem com apenas o arrastar de ícones.

5 Desenvolvimento

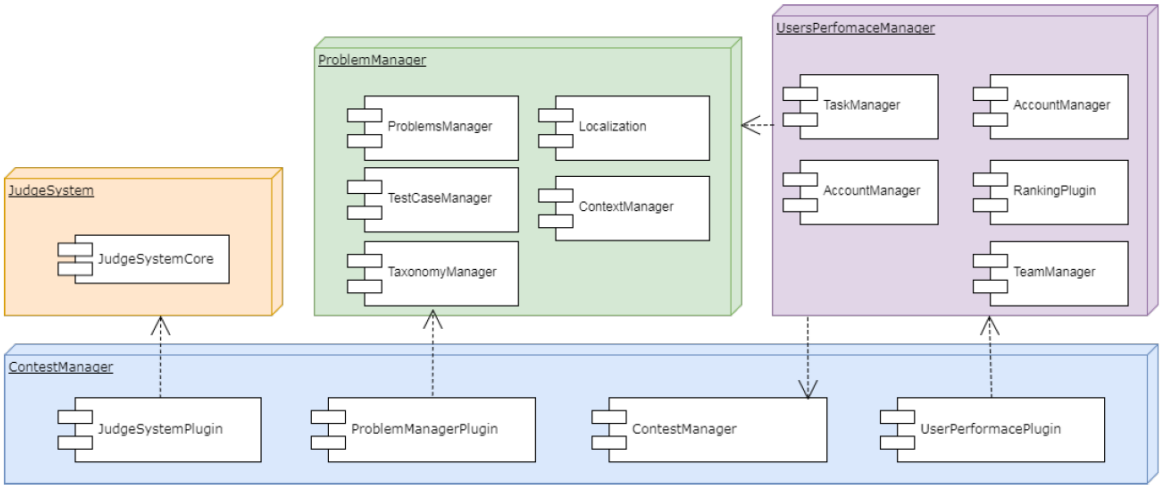
Este capitulo é voltado para a definição e desenvolvimento do módulo de problemas do sistema colaborativo para ensino e programação. São propostas taxonomias iniciais para problemas, funcionalidades esperadas, diagramas de modelagens e por fim, apresentado um protótipo funcional como estudo de caso deste trabalho.

5.1 Arquitetura Conceitual (CETECOP)

Foi proposto inicialmente um modelo conceitual da arquitetura do sistema CETECOP *(Collaborative Environment for Teaching Computer Programing)*. Desenvolvido pelo ex-aluno Otávio Cesário Oliveira, do curso de Engenharia de Computação do CEFET-MG Campus VII (Oliveira 2017). Esta arquitetura contempla a divisão entre 4 módulos que juntos integrariam o sistema completo. Este trabalho irá assim focar no desenvolvimento e estudo de um dos módulos propostos nesta arquitetura, o *ProblemManager*. A proposta da arquitetura colaborativa e conceitual do CETECOP, 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 a diversos outros alunos pelo mundo.

A Figura 12 apresenta a arquitetura incluindo os quatro módulos sugeridos para compor a solução e seus componentes, o *Judge System*, *Contest Manager*, *Problem Manager* e *User Manager*. Estes foram desenhados para funcionar seguindo um modelo escalável, além de possuir objetivos específicos e delimitados, para que assim possam ser substituídos caso necessário ou evoluídos separadamente.

Figura 12 – Arquitetura Conceitual CETECOP.



Fonte: (Oliveira 2017)

5.1.1 *Judge System*

É o módulo responsável por fazer o julgamento das respostas dos problemas resolvidos. Ele analisa o código fonte enviado e baseado em suas métricas ele define se a submissão esta correta ou não. Atualmente é utilizado o juiz do DOMJudge.

5.1.2 *Problem Manager*

Ferramenta específica para gerir os problemas, o versionamento de cada um destes e sua colaboração. É o modulo aprofundado neste artigo.

5.1.3 *Contest Manager*

É responsável pela criação e gerência de conteúdos do sistema. Ele gere competições criadas, além de outras entidades como, por exemplo, disciplina, material e tópico. Ele funciona como uma especie de MOODLE para o aluno. MOODLE (*Modular Object-Oriented Dynamic Learning Environment*) é um *software* livre, de apoio à aprendizagem, executado num ambiente virtual.

5.1.4 User Performance Manager

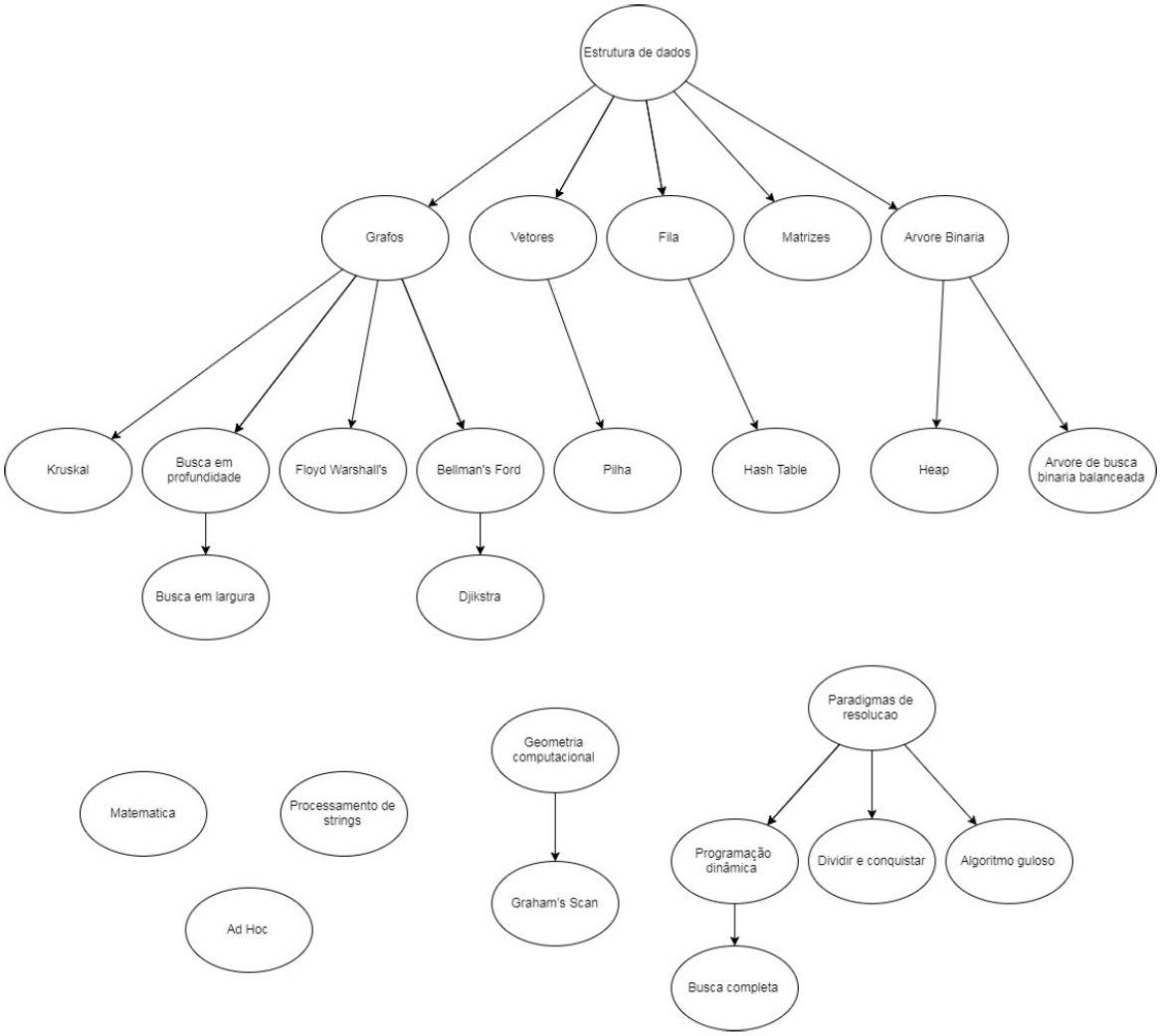
É a parte de interação com os usuário do sistema, sejam estes os alunos, professores ou outros atores do sistema. Neste projeto parte deste módulo é feito em OutSystems, pois sua totalidade não é necessária para o trabalho de pesquisa.

5.2 Taxonomias iniciais para problemas

Por definição, taxonomia é uma ciência ou forma de classificação de objetos em categorias. Considerando o ambiente de maratonas de programação ou mesmo estudos de computação, os problemas podem ser classificados de diversas formas. Pode-se citar classificação por dificuldade, por disciplina especifica da computação estudada, ou por técnicas utilizadas na resolução do problema.

É proposta assim uma classificação inicial para o sistema, baseado no livro *Competitive Programming* de (STEVEN 2010), representada pela Figura 13. Esta taxonomia utilizada é semelhante a estudada no sistema URI, porém este possui uma classificação limitada. Assim, havendo a necessidade de novas categorias ou mesmo subcategorias, o sistema irá possibilitar um novo cadastro desta.

Figura 13 – Taxonomias iniciais para problemas do CETECOP.



Fonte: (Oliveira 2017)

5.3 Funcionalidades do Sistema

Após analisar as ferramentas semelhantes já existentes foi feita uma listagem das principais funcionalidades que se deseja que o sistema proposto contemple.

- Saídas e correções padronizadas - Adotar ou criar um padrão de saídas, pois cada juiz *online* apresenta sua própria saída que pode ser diferente de outro juiz *online*;
- Classificação de problemas - Poder classificar o problema não só por dificuldade, mas por tipo, conteúdo, disciplina, área, caso de teste, autores e origem;
- Detector de plágio - Permitir a fácil identificação de cópias e plágios em resoluções de um problema em determinado grupo ou turma criada por um professor;

- Sistema de ranqueamento atrativo e comparação entre usuários - Não ser apenas uma escada onde cada usuário ocupa um degrau, mas permitir que mesmo usuários em níveis mais baixos sintam-se satisfeitos com cada conquista;
- Ambiente para discussão - Fórum onde é possível compartilhar ideias e soluções especificas para algum problema. Não necessariamente permitindo a cópia de solução de um problema como todo, mas possibilitando uma fácil comunicação entre os usuários do sistema;
- Mecanismo de *chat*, ou troca de mensagens - Permitir a comunicação entre diferentes usuários dentro do sistema, de forma que um aluno possa se comunicar com o professor quando necessário, ou mesmo com colegas;
- Editor interno *online* - Permitir a programação direta via o sistema, dispensando a instalação de qualquer outra ferramenta;
- Material didático de apoio - Possibilitar que os alunos possuam uma material de base para seu aprendizado sem ter que sair do sistema;
- Capacidade de gerir competições internas de forma fácil e eficiente - Permitir que alunos e professores possam criar suas próprias competições e maratonas de programação em minutos.

5.4 Funcionalidades para o Professor

O professor é o ator responsável pelo gerenciamento e utilização da maior parte do módulo de problemas. Este possui capacidades e funcionalidades especificas que irão lhe prestar suporte no ensino, seja em sala de aula ou a distância, além de ferramentas especificas que irão permitir um melhor controle de suas aulas e atividades ministradas.

Na Tabela 5 é proposta uma lista de funcionalidades esperadas para o professor.

Tabela 5 – Funcionalidades esperadas para o professor.

Funcionalidade	Descrição
Criação de grupos.	O professor tem a capacidade de definir grupos que podem servir como turmas ou disciplinas lecionadas.
Gerenciamento de grupos.	O professor pode incluir ou remover integrantes de grupos, além de poder incluir e editar detalhes específicos deste grupo.
Criação e compartilhamento de atividades.	O professor pode criar atividades específicas e customizadas, servindo como listas de exercícios, provas e etc. Estas atividades são salvas, podendo assim serem utilizadas em diferentes grupos, ou mesmo compartilhadas com outros usuários específicos.
Cadastro de material de apoio.	O professor pode fazer *upload* de materiais de apoio em seus grupos, onde os integrantes destes podem fazer *download* e consultarem.
Visualização da correção de atividades.	O professor pode visualizar a correção completa de respostas das atividades feitas por integrantes de seus grupos, podendo analisar também o código fonte utilizado como solução.
Visualização do histórico de submissão dos integrantes de seus grupos.	O professor pode monitorar os integrantes de seus grupos, podendo ver com detalhes o status da submissão de cada atividade.
Visualização do progresso de cada integrante em seus grupos.	O professor pode ver detalhes do desenvolvimento de cada integrante de seu grupo, incluindo nota atual, notas anteriores, gráficos de progressão do desempenho geral e em tópicos específicos.
Submissão de exercício.	O professor pode fazer a submissão de um novo exercício proposto para o sistema, onde este pode vir a ser utilizado por outros professores e usuários do sistema.

Fonte: Elaborado pelo Autor.

5.5 Dicionário de Palavras

É comum sistemas de controles de versões possuírem termos próprios referente aos nomes de ações e entidades de seu ambiente. Algo que pode ser comum para usuários experientes destas plataformas, mas que para um novo usuário inexperiente pode não ser completamente intuitivo. Para isto, o projeto proposto irá criar um dicionário de palavras o qual irá traduzir estas ações específicos para uma linguagem de comandos que mesmo usuários leigos em sistemas de controle de versões irão compreender.

5.6 Organização de problemas

Atualmente cada problema utilizado em maratonas de programação ou mesmo em exercícios e provas em salas de aulas possuem poucos casos de adaptações. São tratados como exercícios fechados, onde são utilizados apenas em uma disciplina ou situações especificas. É proposto uma nova organização de exercícios que permite uma restruturação destes problemas, possibilitando que estes possam ser utilizados em diferentes áreas de conhecimentos da programação.

A organização idealizada é baseada em três diferentes níveis de casos de testes, no-

meados de A, B e C. Um problema com caso de teste C é o mais básico possível, onde o aluno poderia resolver o problema da forma que conseguisse, mesmo esta resolução sendo menos eficiente. Já um problema de nível B apresenta um grau de dificuldade um pouco maior, necessitando de uma resolução mais robusta, com possíveis algoritmos de ordenação específicos ou armazenamento mais otimizado, por exemplo. Porem este problema ainda assim pode ser lento, e para isto tem o caso de teste A, onde para resolve-lo é necessário além dos outros fatores, uma habilidade matemática mais apurado.

Um determinado problema, com o mesmo enunciado, pode apresentar distintas resoluções, cada uma correspondendo a um caso de teste especifico. Um determinado aluno pode assim resolver o exercício da maneias mais simples que conseguir, e passar no caso de teste C, assim conseguindo a competência necessária para aquele exercício. Porem para resolver o problema nos outros níveis de dificuldade, será exigido do aluno conhecimentos adicionais.

5.7 API de integração com DOMJudge

A API (Interface de Programação de Aplicação) DOMJudge é uma implementação da API do Concurso ICPC. O juiz *online* fornece uma API REST em JSON que pode ser usada para consultar o estado da competição, bem como executar certas ações, como enviar soluções, solicitar julgamento de submissões e controlar o estado da competição.

O funcionamento da API DOMJudge é feito de forma que ela acessa dados fornecidos por um CCS (Sistema de controle de competições) ou um CDS (Servidor de dados de competições), e pode ser usada por uma variedade de clientes. Dentre os mais comuns podem ser citados placares externos„ *softwares* de analise de competições, fornecendo encaminhamento de submissões, ou mesmo clientes internos, servindo de interface entre o servidor CCS e as instâncias de juízes.

Esta API destina-se a ser útil em qualquer configuração de competição no estilo ICPC. É feita para incorporar e substituir o JSON Scoreboard, a interface REST de busca de código-fonte e a interface inicial Contest (DOMJudge 2021). Essa interface REST é especificada em conjunto com um *feed* de evento JSON, que fornece todas as alterações a essa interface como eventos no estilo CRUD (*Create, Read, Update e Delete*).

5.8 Integração com GitLab

A integração do protótipo é feita através de chamadas a API REST do próprio GitLab. No modulo ProblemManager foi criado funções que acessam diretamente esta API passando o Id do projeto no GitLab, e parâmetros de entrada correspondentes da função e retornando um valor de resposta, seja este uma mensagem, em caso de chamadas POST, PUT e DELETE ou uma estrutura montada a partir de um JSON, em caso de GET.

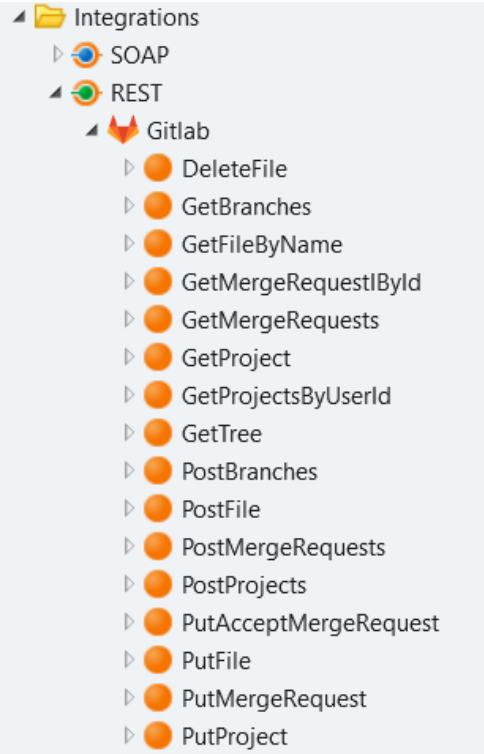
Na Tabela 6 é listado todas as funções, ou como são chamadas no OutSystems, *Server Actions*, criadas para a integração com a API do GitLab, e na Figura 14 a lista de funções de integração à API REST do GitLab.

Tabela 6 – Lista de funções de chamadas a API do GitLab.

Server Action	Descrição
DeleteFile	Deleta um arquivo de um projeto.
GetFile	Retorna um arquivo de um diretório.
GetBranches	Retorna as ramificações de um repositório.
GetFileByName	Retorna um arquivo filtrado por nome.
GetMergeRequestIById	Retorna uma requisição de união específica.
GetMergeRequests	Retornar todas requisições de união.
GetProject	Retorna um projeto específico.
GetProjectsByUserIdAndNome	Retorna todos projetos de um usuário filtrado por Id e Nome.
GetTree	Retorna uma árvore de um repositório.
PostBranches	Cria uma nova ramificação em um repositório.
PostMergeRequests	Cria uma nova requisição de união em um repositório.
PostProjects	Cria um novo projeto para um usuário.
PutAcceptMergeRequest	Altera o status da requisição de união para aceita.
PutFile	Altera um arquivo em um repositório.
PutMergeRequest	Altera informações de uma requisição de união.
PutProject	Altera informações de um projeto.

Fonte: Elaborado pelo Autor.

Figura 14 – Lista de funções de integração REST ao GitLab no módulo ProblemManager.



Fonte: Elaborado pelo Autor.

As principais chamadas utilizadas neste projeto são aquelas relacionadas ao problemas. Para aumentar a segurança do sistema, impedindo um acesso direto do usuário aos

dados do sistema, funções no OutSystems, chamadas de *Server Actions*, definidas como privadas, foram criadas para utilizar destas chamadas e exibir um retorno conforme o tipo da chamada que será enviado para o sistema.

Abaixo é descrito como funcionam cada uma destas chamadas, para o que e como foram utilizadas no sistema proposto.

5.8.1 DeleteFile

Para se excluir um arquivo específico de um diretório do GitLab através de chamadas REST é utilizado o terminal `DeleteFile`. Este recurso é utilizado quando se deseja fazer a exclusão de um problema. Cada problema é salvo como um projeto que contém um arquivo JSON em seu diretório, assim para excluir o problema, antes de excluir o projeto em si, é feita a exclusão do arquivo em seu interior. O terminal da API utilizado é o seguinte:

```
DELETE /projects/:id/repository/files/:file_path
```

Foi criado pelo autor uma função no OutSystems que utiliza esta chamada, de nome ExcluirProblemaGitLab. Na Figura 15 é apresentado o seu fluxo.

Figura 15 – Fluxo da função ExcluirProblemaGitLab.

Fonte: Elaborado pelo Autor.

5.8.2 GetBranches

Uma branch no GitLab são as ramificações utilizadas neste trabalho. Varias ramificações podem existir para para diferentes arquivos em um projeto do GitLab. Quando é necessário listar todas as branchs de um arquivo, ou no caso do projeto, todas as ramificações de um problema, é utilizado a chamada `GetBranches`. Através dela será retornado todas as ramificações de um projeto especifico no GitLab, definido por um identificador. O terminal da API utilizado é o seguinte:

```
GET /projects/:id/repository/branches
```

Foi criado pelo autor uma função no OutSystems que utiliza esta chamada, de nome ListarBranchesGitLab. Na Figura 16 é apresentado o seu fluxo.

Figura 16 – Fluxo da função ListarBranchesGitLab.

Fonte: Elaborado pelo Autor.

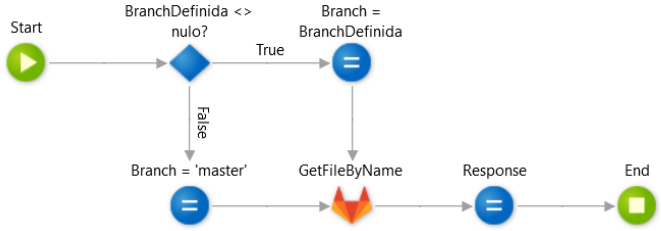
5.8.3 GetFileByName

A chamada `GetFileByName` é utilizada quando é feito o retorno de um problema. Cada problema é salvo como um projeto que contem um arquivo em sua raiz. É neste arquivo, no formato JSON, que contém as informações do problema. Para obter tais informações é necessário a utilização da chamada GetFileByName, que irá retornar uma responsta com as informações e o conteúdo daquele arquivo. O terminal da API utilizado é o seguinte:

```
GET /projects/:id/repository/files/:file_path
```

Foi criado pelo autor uma função no OutSystems que utiliza esta chamada, de nome ObterArquivoPorNomeGitLab. Na Figura 17 é apresentado o seu fluxo.

Figura 17 – Fluxo da função ObterArquivoPorNomeGitLab.



Fonte: Elaborado pelo Autor.

5.8.4 GetMergeRequestIById

Um usuário do tipo professor pode alterar um problema de outro autor, fazendo modificações em um determinado parâmetro ou informação de um problema e, assim, é criada uma ramificação daquele problema. O problema permanece intacto, porém uma nova versão modificada é criada e é enviada para o autor uma requisição de união do problema original com esta nova ramificação, que ele pode aceitar ou não. Para obter as informações de uma requisição específica de união entre um problema e uma ramificação, é utilizado a chamada `GetMergeRequestIById`. O terminal da API utilizado é o seguinte:

```
GET /projects/:id/merge_requests/:merge_request_iid
```

Foi criado pelo autor uma função no OutSystems que utiliza esta chamada, de nome ObterMergeRequestPorIdGitLab. Na Figura 18 é apresentado o seu fluxo.

Figura 18 – Fluxo da função ObterMergeRequestPorIdGitLab.

Fonte: Elaborado pelo Autor.

5.8.5 GetMergeRequests

Cada problema é salvo como um arquivo em um projeto, e arquivos pode sofrer alterações. Através do sistema proposto, tais alterações só irão alterar o problema original caso o autor deste aceite. Antes que o autor aceite, as alterações serão salvas como ramificações. Será então criado uma solicitação de união, entre o ramo e o problema original. E assim, para o autor de um problema, é possível ver a lista de solicitações de união de um problema e suas ramificações. Para obter a listagem de todas requisições de união de um determinado problema, é utilizado a chamada `GetMergeRequests`. O terminal da API utilizado é o seguinte:

```
GET /projects/:id/merge_requests
```

Foi criado pelo autor uma função no OutSystems que utiliza esta chamada, de nome ListarMergeRequestsGitLab. Na Figura 19 é apresentado o seu fluxo.

Figura 19 – Fluxo da função ListarMergeRequestsGitLab.

Fonte: Elaborado pelo Autor.

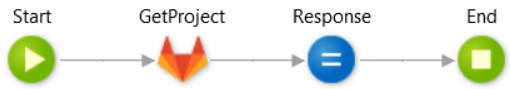
5.8.6 GetProject

Um projeto é um diretório onde pode existir vários arquivos no GitLab. Cada problema é salvo no GitLab como um novo projeto, contendo um arquivo, de nome idêntico ao do projeto, correspondendo ao nome do problema, e contendo informações detalhadas deste. Para se obter um problema, antes de obter o arquivo específico deste, é necessário obter o projeto correspondente. Para isto é utilizado a chamada `GetProject`. O terminal da API utilizado é o seguinte:

```
GET /projects/:id
```

Foi criado pelo autor uma função no OutSystems que utiliza esta chamada, de nome ObterProjetoGitLab. Na Figura 20 é apresentado o seu fluxo.

Figura 20 – Fluxo da função ObterProjetoGitLab.



Fonte: Elaborado pelo Autor.

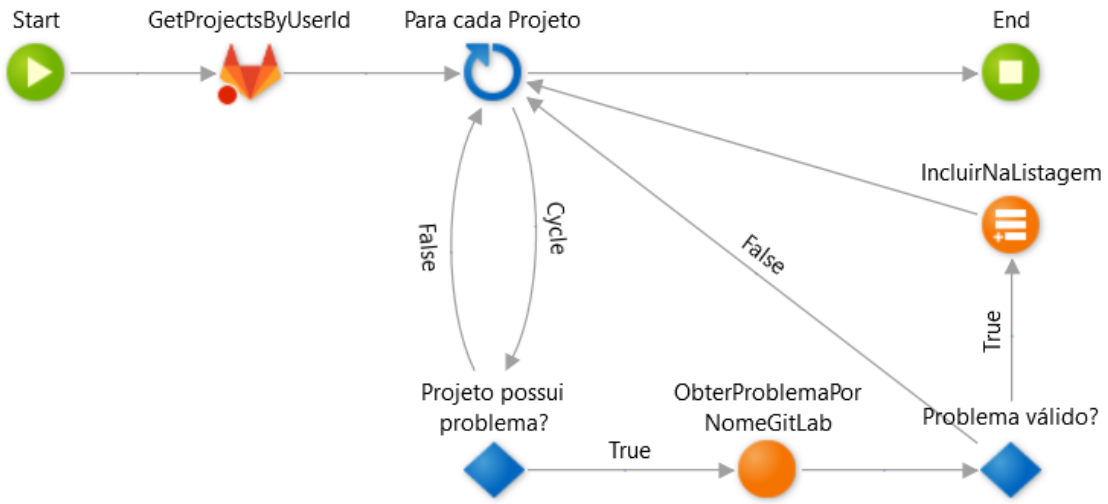
5.8.7 GetProjectsByUserId

Para a listagem de projetos de um determinado usuário é utilizado a chamada `GetProjectsByUserI` Estes projetos correspondem aos problemas, que serão exibidos em uma tabela ou lista do sistema proposto. O terminal da API utilizado é o seguinte:

```
GET /projects/:id
```

Foi criado pelo autor uma função no OutSystems que utiliza desta chamada, de nome ListarProblemasGitLab. Esta função possui validações adicionais para verificar se realmente existe o arquivo JSON do problema dentro do projeto. Isto serve para não trazer todo e qualquer projeto do usuário, e sim apenas aqueles que correspondem a problemas do sistema proposto. Cada problema de projetos válidos são então adicionados a uma listagem que será retornada através da função. Na Figura 21 é apresentado o seu fluxo.

Figura 21 – Fluxo da função ListarProblemasGitLab.



Fonte: Elaborado pelo Autor.

5.8.8 GetTree

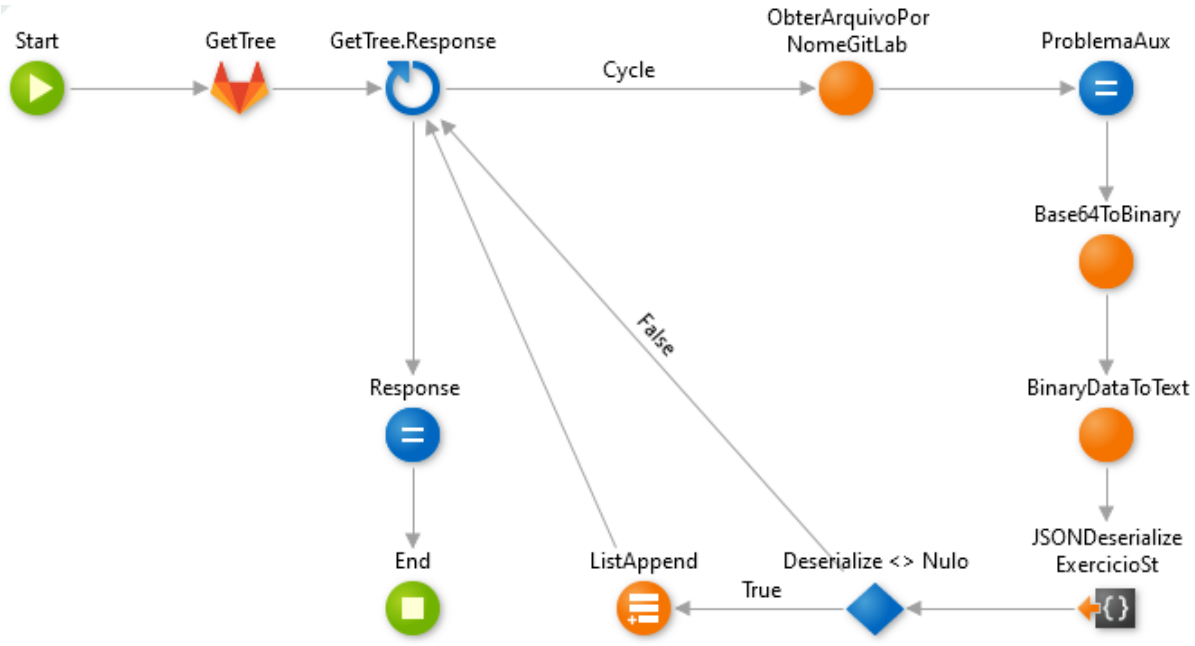
A chamada `GetTree` é utilizada para fazer a listagem dos problemas salvo em um repositório. Como, por padrão adotado neste trabalho, cada projeto possui apenas um arquivo, o retorno será sempre apenas um arquivo de problema.

Para fazer a listagem dos arquivos salvos em um determinado repositório do GitLab é utilizado o terminal:

```
GET /projects/:id/repository/tree
```

Foi criado pelo autor uma função no OutSystems que utiliza desta chamada, de nome ListarProblemasExerciciosGitLab. Nele é feita uma chamada a função REST GetTree que retorna uma lista contendo o nome e dados dos Problemas salvos no repositório. Para cada um destes problemas é feito uma busca que irá retornar o conteúdo JSON. É feito uma conversão de Base64 para texto e então ele é desserializado. caso o retorno desta desserialização seja diferente de nula, é feita uma inclusão na lista de problemas que será utilizada como retorno. Na Figura 22 é apresentado o seu fluxo.

Figura 22 – Fluxo da função ListarProblemasExerciciosGitLab.



Fonte: Elaborado pelo Autor.

5.8.9 PostBranches

Um usuário pode criar uma nova ramificação a partir de um arquivo um projeto no GitLab, que irá corresponder a um problema no sistema proposto. Esta ramificação será uma alteração no problema original que o usuário pode solicitar que se una com o problema original, caso o autor aceite ou não esta união. Para criar uma nova ramificação é utilizada a chamada `PostBranches`. O terminal da API utilizado é o seguinte:

```
POST /projects/:id/repository/branches
```

Foi criada pelo autor uma função no OutSystems que utiliza esta chamada, de nome InserirBranchGitLab. Na Figura 23 é apresentado o seu fluxo.

Figura 23 – Fluxo da função InserirBranchGitLab.

Fonte: Elaborado pelo Autor.

5.8.10 PostFile

Como já descrito anteriormente, um problema é salvo como um novo projeto no GitLab, que por sua vez possui um arquivo JSON contendo as informações deste, que será salvo utilizando a chamada `PostFile`, que inclui um novo arquivo em um projeto. O terminal utilizado é o seguinte:

```
POST /projects/:id/repository/files/:file_path
```

É necessário passar o diretório com o nome do arquivo na requisição, assim como o id do projeto, como um parâmetro. O arquivo assim será salvo no Projeto do GitLab com o nome passado no terminal, e este nome será aquele usado para pesquisas de listagem ou obtenção de um problema específico no sistema proposto.

Foi criada pelo autor uma função no OutSystems que utiliza esta chamada, de nome InserirProblemaProjetoGitLab. É criado um novo projeto que irá retornar um identificador único. Este identificador é salvo como parâmetro em uma estrutura de dados contendo as informações do problema. Em seguida é criado o arquivo através da chamada PostFile, contendo todas essas informações do problema. Na Figura 24 é apresentado o seu fluxo.

Figura 24 – Fluxo da função InserirProblemaProjetoGitLab.

Fonte: Elaborado pelo Autor.

5.8.11 PostMergeRequests

Merge Requests são as requisições de união do GitLab. Quando uma nova ramificação é criada o autor da ramificação pode solicitar uma união entre a ramificação e o arquivo original. Um problema pode possuir várias solicitações de união. Para se criar uma através da API REST do GitLab usa-se a chamada `PostMergeRequests`. O terminal utilizado é o seguinte:

```
POST /projects/:id/merge_requests
```

Foi criada pelo autor uma função no OutSystems que utiliza esta chamada, de nome InserirMergeRequestGitLab. Na Figura 25 é apresentado o seu fluxo.

Figura 25 – Fluxo da função InserirMergeRequestGitLab.

Fonte: Elaborado pelo Autor.

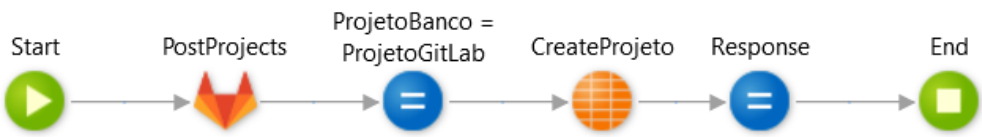
5.8.12 PostProjects

Foi visto que antes de se criar um arquivo em um diretório do GitLab é necessário criar um novo projeto, onde este irá estar. A chamada utilizada para se criar projetos é a `PostProjects`. Quando um usuário cria um novo problema no sistema proposto é criado também um novo projeto, de nome correspondente ao título do problema é criada para um usuário no GitLab. O terminal utilizado é o seguinte:

```
POST /projects/user/:user_id
```

Foi criada pelo autor uma função no OutSystems que utiliza esta chamada, de nome InserirProjetoGitLab. Nesta função, após ser criado o novo projeto no GitLab, é criado também uma cópia do Projeto no banco de dados, que serve para validações extra do sistema proposto. Na Figura 26 é apresentado o seu fluxo.

Figura 26 – Fluxo da função InserirProjetoGitLab.



Fonte: Elaborado pelo Autor.

5.8.13 PutAcceptMergeRequest

Após um usuário criar uma nova ramificação, ele pode solicitar ao autor do arquivo original que seja feita a união deste com seu ramo. O autor pode então aceitar ou não a união. Caso aceite, o arquivo original irá conter características de ambas versões e será uma nova versão, mesclada de sua versão anterior com a nova ramificação. Para alterar o status de uma requisição de união através de API REST do Gitlab é utilizado a chamada `PutAcceptMergeRequest`. O terminal utilizado é o seguinte:

```
PUT /projects/:id/merge_requests/:merge_request_iid/merge
```

Foi criada pelo autor uma função no OutSystems que utiliza esta chamada, de nome AceitarMergeRequestGitLab. Na Figura 27 é apresentado o seu fluxo.

Figura 27 – Fluxo da função AceitarMergeRequestGitLab.

Fonte: Elaborado pelo Autor.

5.8.14 PutFile

Quando é necessário atualizar um problema salvo no projeto do GitLab é utilizado uma requisição PUT, com a chamada `PutFile`. Ela funciona de forma semelhante ao PostFile, sendo que a única diferença é que ela irá alterar um arquivo já existente de um diretório do GitLab, e não criar um novo. O terminal utilizado é o seguinte:

```
PUT /projects/:id/repository/files/:file_path
```

Foi criada pelo autor uma função no OutSystems que utiliza esta chamada, de nome AlterarProblemaProjetoGitLab. Assim como no PostFile, é necessário passar o diretório com o nome do arquivo na requisição como parâmetro, porém como o projeto que aquele problema reside já existe, não há a necessidade de criar um novo projeto, e sim apenas obter o projeto correspondente. Na Figura 28 é apresentado o seu fluxo.

Figura 28 – Fluxo da função AlterarProblemaProjetoGitLab.

Fonte: Elaborado pelo Autor.

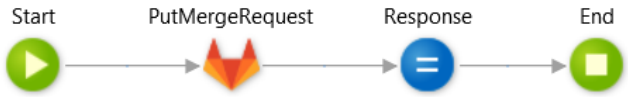
5.8.15 PutMergeRequest

Uma requisição de união possui parâmetros como “comentário” e “descrição”, que podem ser deixados pelo autor da ramificação para o autor do arquivo original. Assim torna-se necessário, em alguns cenários, que seja possível editar algum dos parâmetros de uma requisição de união, para poder atualizá-las. A chamada utilizada para isto é a `PutMergeRequest`, que permite alterar não apenas o status da requisição, mas a requisição como toda. O terminal utilizado é o seguinte:

```
PUT /projects/:id/merge_requests/:merge_request_iid
```

Foi criada pelo autor uma função no OutSystems que utiliza esta chamada, de nome AlterarMergeRequestGitLab. Na Figura 29 é apresentado o seu fluxo.

Figura 29 – Fluxo da função AlterarMergeRequestGitLab.



Fonte: Elaborado pelo Autor.

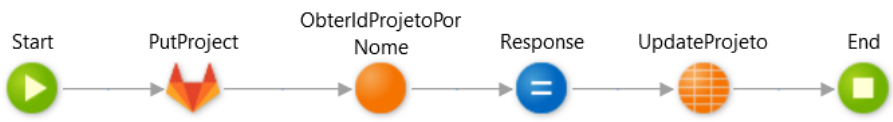
5.8.16 PutProject

Tendo em vista que um problema é salvo tanto como um novo projeto como um arquivo deste, ao alterar um problema é necessário alterar não apenas o arquivo, mas também o projeto em si, atualizando seu nome. É necessário assim utilizar da chamada `PutProject`, que tem como finalidade atualizar um projeto em um repositório do GitLab. O terminal utilizado é o seguinte:

```
PUT /projects/:id
```

Foi criada pelo autor uma função no OutSystems que utiliza esta chamada, de nome AlterarProjetoGitLab. Da mesma forma que se cria uma referência ao projeto no banco de dados quando um novo projeto no GitLab é criado, ao atualizar um projeto a sua referência no banco é atualizada também. Na Figura 30 é apresentado o seu fluxo.

Figura 30 – Fluxo da função AlterarProjetoGitLab.

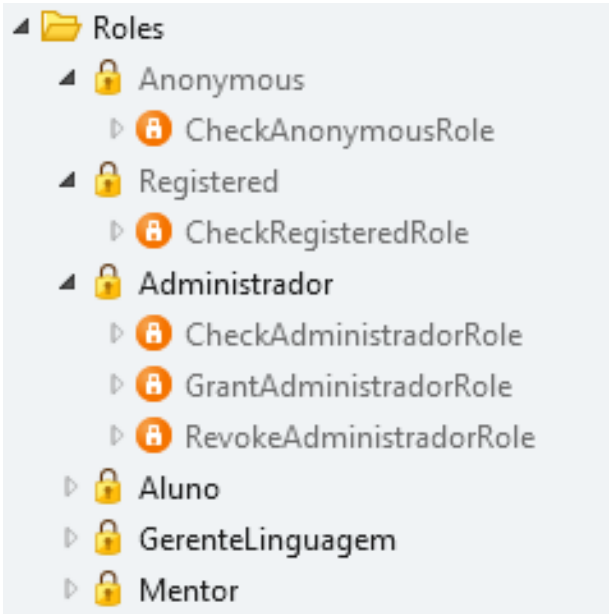


Fonte: Elaborado pelo Autor.

5.9 Atores do sistema

Um ator é um humano ou entidade que interage com a plataforma para executar um determinado trabalho. Ele especifica um papel executado por um usuário que interage com o sistema. Estes papeis são definidos como *Roles* no OutSystems. Para cada uma são criadas, automaticamente, funções correspondentes que tem como finalidade fazer o gerenciamento daquela *role*, como por exemplo, concede-la a um usuário, checar se o usuário pertence a este perfil, ou mesmo revogar o acesso a ele, como demonstrado na Figura 31.

Figura 31 – Roles de acesso do sistema.



Fonte: Elaborado pelo Autor.

A seguir serão apresentados os atores presente no sistema proposto.

5.9.1 Aluno

O ator com menor número de permissões é representado comumente por um aluno. O objetivo deste consiste em realizar as atividades propostas pelo mentor. Possui também a capacidade de desenvolver atividades disponíveis na plataforma por conta própria, assim acumulando pontos e conquistas, além de possibilitar a criação de um sistema de ranqueamento. O usuário poderá, também, visualizar uma lista de *feedbacks* sobre seu desempenho no decorrer das atividades feitas.

5.9.2 Mentor

Um usuário do sistema que possui mais privilégios que o usuário comum. Este é representado pelo professor. Através de filtros ele pode selecionar exercícios para uma lista de atividades, além de criar modelos de provas as quais pode enviar para usuários específicos ou de suas turmas.

5.9.3 Administrador

Administradores do sistema possuem o maior número de privilégios. Este tipo de usuário é responsável por cadastro específicos, além de gerenciar e verificar dados de uma forma mais ampla.

5.9.4 Gerente de linguagem

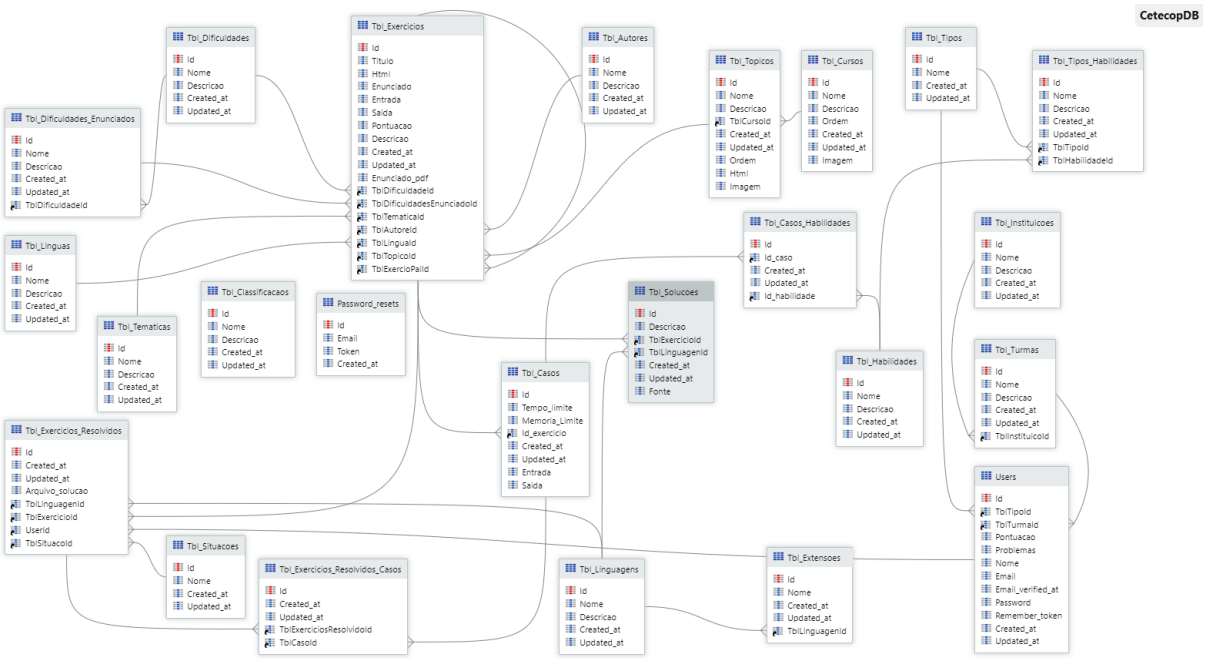
Responsável por fazer a curadoria dos problemas, vários usuários podem ser Gerentes de linguagem. São usuários classificados internamente como competentes no julgamento de problemas, e em suas formulações. É o usuário responsável por gerenciar os problemas existentes, além de poder incluir novos.

5.10 Modelo de dados

O modelo do de banco de dados proposto visa contemplar as principais funcionalidades esperadas do sistema, permitindo um cadastro de problemas, representado pela entidade Exercício, e esta possuindo níveis de dificuldade para cada enunciado, temáticas, autores, linguagens de programação, tópicos e exercícios pai. Isto permitirá que o problema possua ligações com as especificações esperadas pelo professor, tornando mais detalhada a sua busca e mais direcionada o seu uso.

Na Figura 32 é apresentado o diagrama de classe do modelo proposto para o banco de dados, gerado na plataforma OutSystems com todas entidades do sistema considerando o gestor de problemas e um protótipo do módulo do usuário.

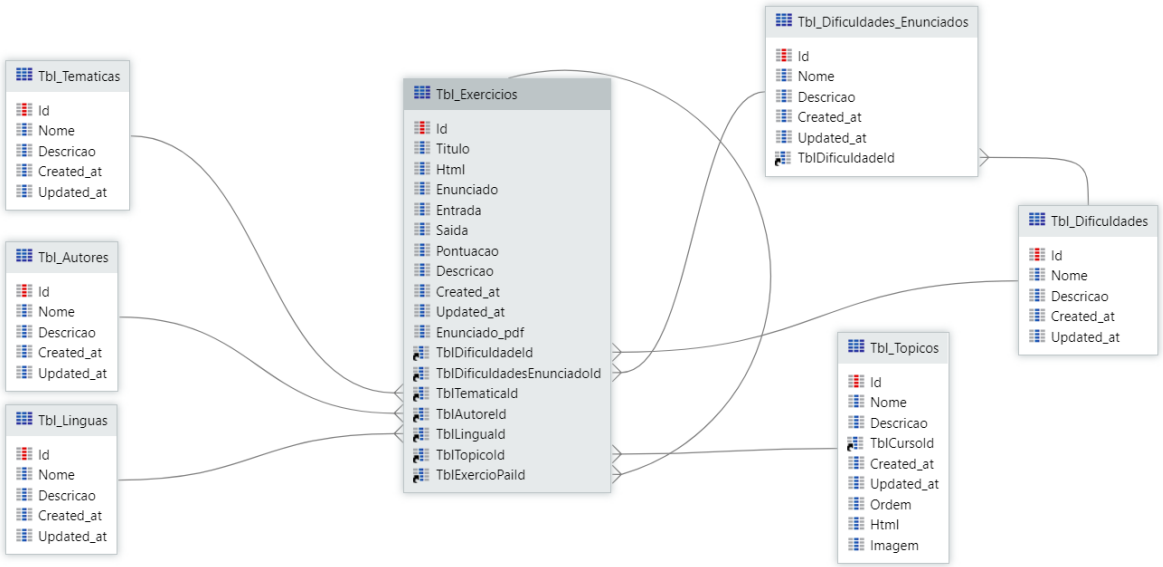
Figura 32 – Modelo estrutural do banco de dados do sistema como um todo.



Fonte: Elaborado pelo Autor.

O trabalho no entanto foca na gestão de exercícios, possuindo assim um modelo de dados mais simplificado e direcionado. Na Figura 33 é apresentado este modelo de dados do gestor de problemas.

Figura 33 – Modelo estrutural do banco de dados do gestor de problemas.



Fonte: Elaborado pelo Autor.

5.11 Protótipo funcional

Para a validação do projeto é proposto um sistema *web* que irá atuar como o gerenciador de problemas, onde o professor poderá organizar seus exercícios e turmas. O protótipo deste sistema é dividido em dois módulos. Em OutSystems, módulos são onde a interface do usuário e a lógica de negócio são desenvolvidas.

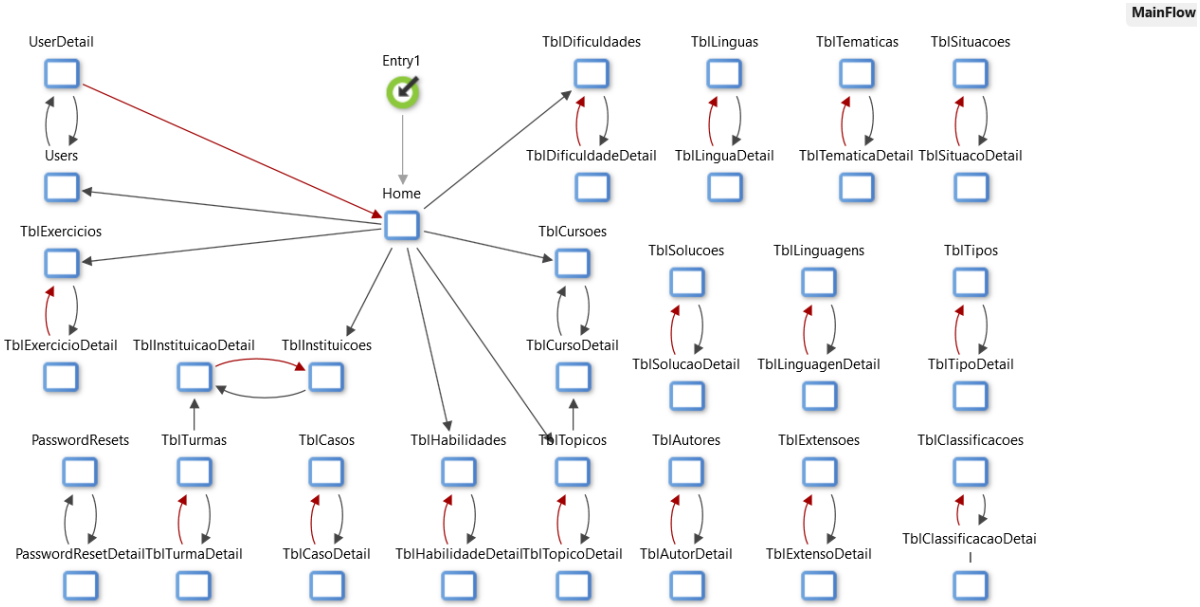
O primeiro modulo criado é o ProblemManagerCore, que atuará como servidor, e ficará responsável por armazenar as entidades, e funções de acesso ao banco. O segundo será o ProblemManager, que irá atuar como cliente e irá contemplar as páginas *web* e as chamadas ao servidor.

O ProblemManagerCore não possui interface gráfica, servindo apenas de comunicação com o banco. Ele contém 26 entidades, e 90 funções de comunicação com o servidor que são utilizadas pelo cliente. Já o *ProblemManager*, modulo este de interação com os usuários, possui interface gráficas.

Na Figura 34 é apresentado o fluxo das telas de cadastros do sistema, mostrando o caminho que o usuário do sistema pode percorrer ao ser direcionado por cada uma das páginas, além de seu conteúdo e funcionalidades. Estas são páginas *web* que realizam o cadastro em um banco pessoal do desenvolvedor na nuvem do OutSystems. Neste fluxo de telas apresentado, as páginas que não possuem ligação direta ainda podem ser acessadas através de menus laterais ou superiores. Já aquelas com ligações podem ser acessadas diretamente via

botões.

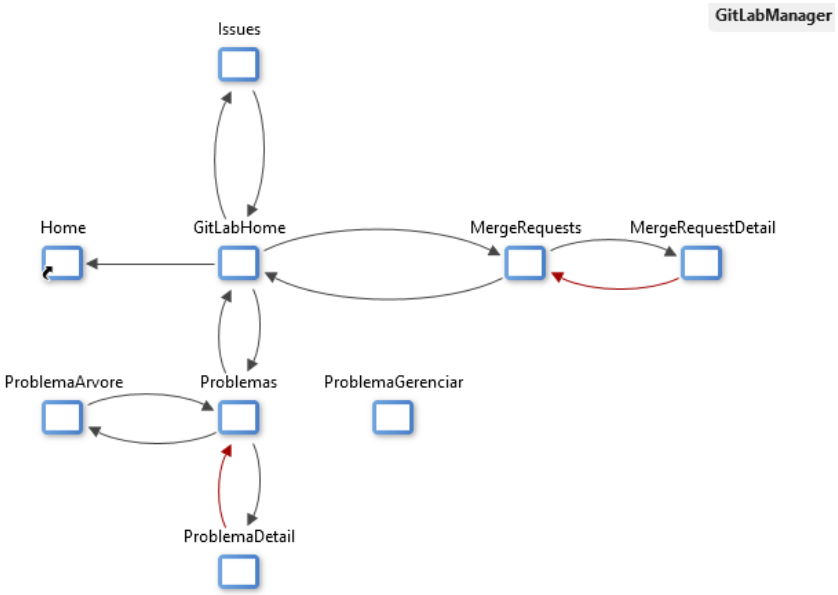
Figura 34 – Fluxo de telas de cadastros do sistema.



Fonte: Elaborado pelo Autor.

Existe também um segundo fluxo, onde é tratado o cadastro, listagem e gerenciamento dos Problemas. Este reside em um fluxo separado pois seu cadastro é feito de forma diferente, armazenado em um projeto do GitLab ao invés do banco pessoal, como é mostrado na Figura 35.

Figura 35 – Fluxo de telas do gerenciamento de problemas.

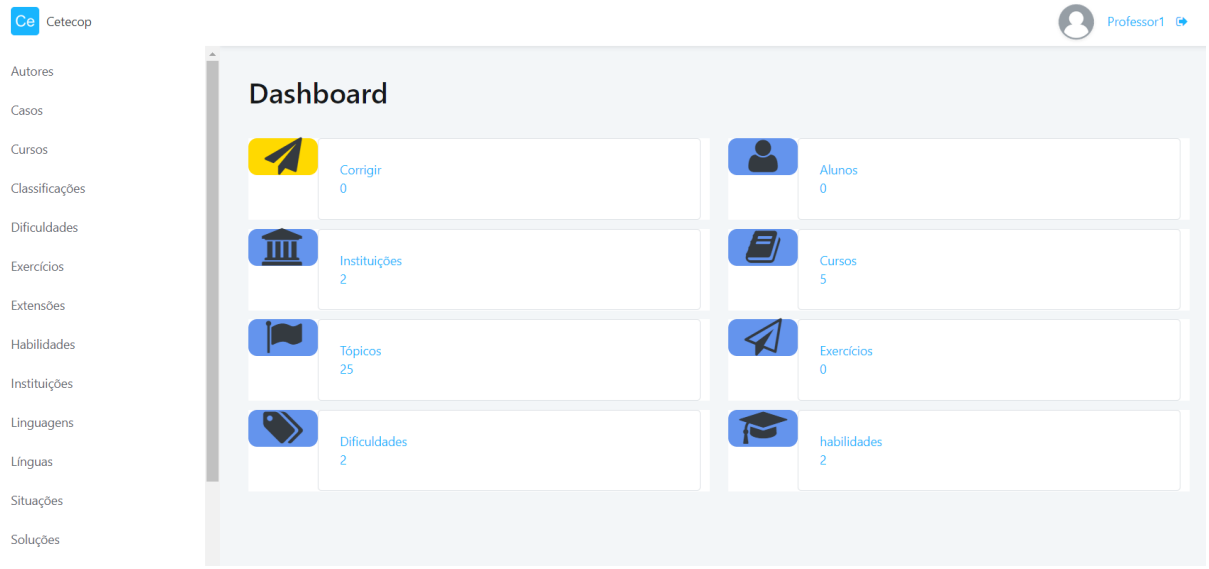


Fonte: Elaborado pelo Autor.

Ao logar no sistema o usuário é direcionado a página inicial *Home*, onde então são

apresentados *links* para outras telas com conteúdos referentes a cadastros, listagens, e ações distintas correspondentes a funcionalidades especificas do sistema, como pode ser visto na Figura 36.

Figura 36 – Tela inicial do módulo do professor.

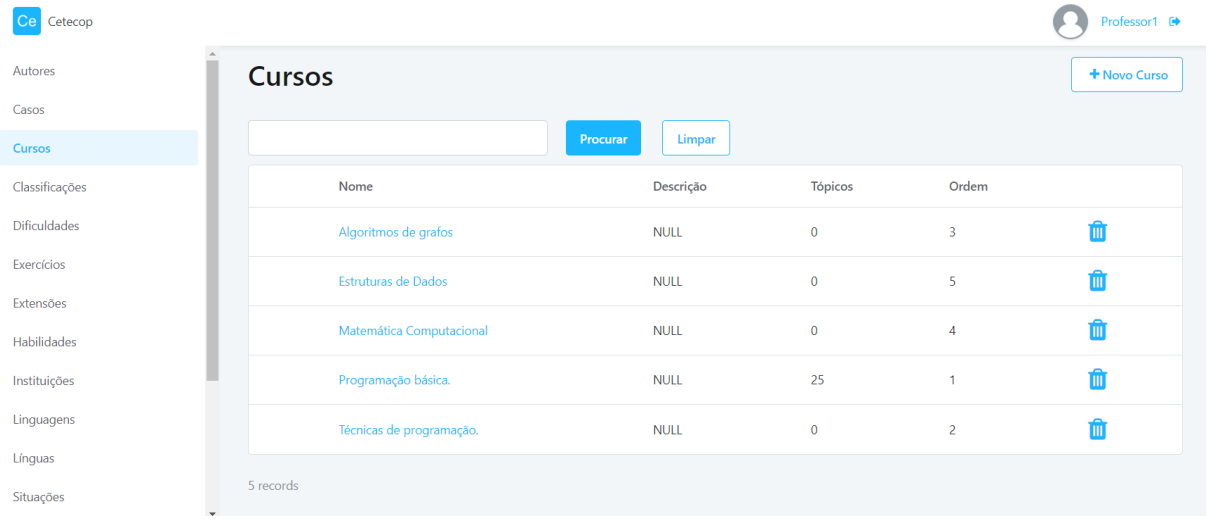


Fonte: Elaborado pelo Autor.

5.11.1 Listagem e Cadastros

O cadastro e listagem do sistema são feitos de forma semelhantes para a maioria das entidades, sendo diferente apenas para a entidade “Problema”. Para todas outras entidades existe uma tela de listagem com tabela, filtros, botão de exclusão e de inclusão de um novo cadastro. Na Figura 37 é apresentada a página de listagem de cursos.

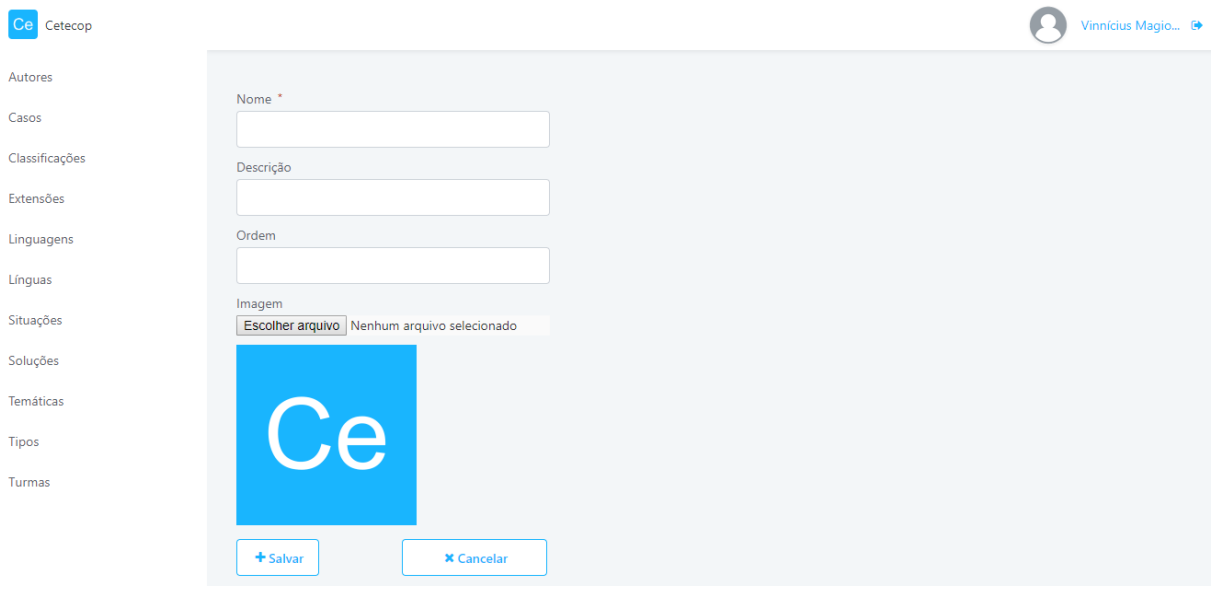
Figura 37 – Tela de listagem de cursos cadastrados.



Fonte: Elaborado pelo Autor.

Ao clicar no nome da entidade o sistema, ou no botão “Novo”, o sistema é direcionado para uma página de inclusão ou alteração, como é demonstrado na Figura 38. Caso a entidade em questão tenha sido selecionada, a página irá ter como funcionalidade a alteração dos dados da mesma. Caso contrário será feita a inclusão de uma entidade nova correspondente ao formulário preenchido.

Figura 38 – Tela de alteração ou inclusão de curso.

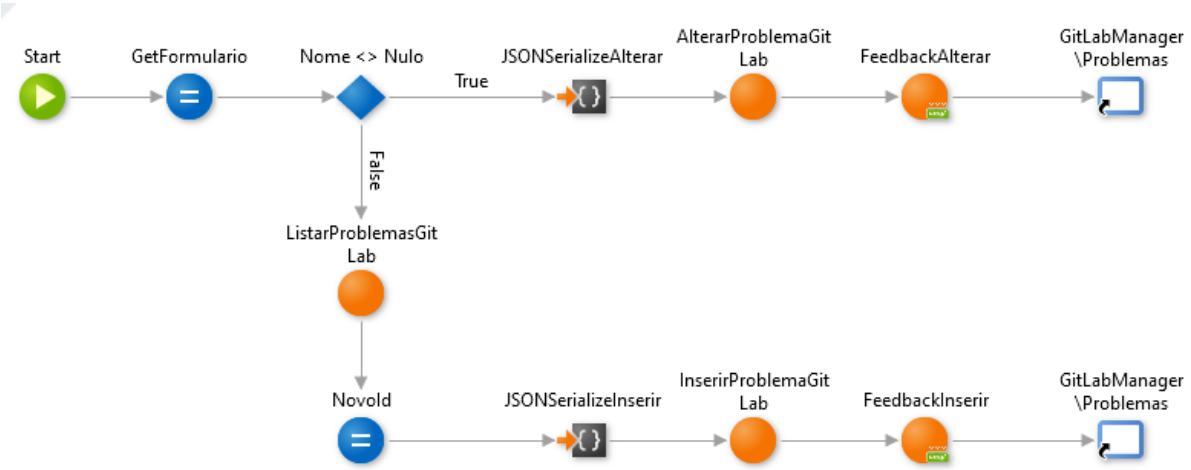


Fonte: Elaborado pelo Autor.

5.11.2 Cadastro de Problemas

O cadastro e listagem de problemas foi implementado seguindo um modelo diferente das outras entidades do sistema. Os problemas são salvos como arquivos JSON em um projeto do GitLab. No processo de cadastro é feita uma serialização e na listagem uma desserialização deste JSON criado a partir da entidade cadastrada, como visto na Figura 47.

Figura 39 – Função de salvar um novo Problema.



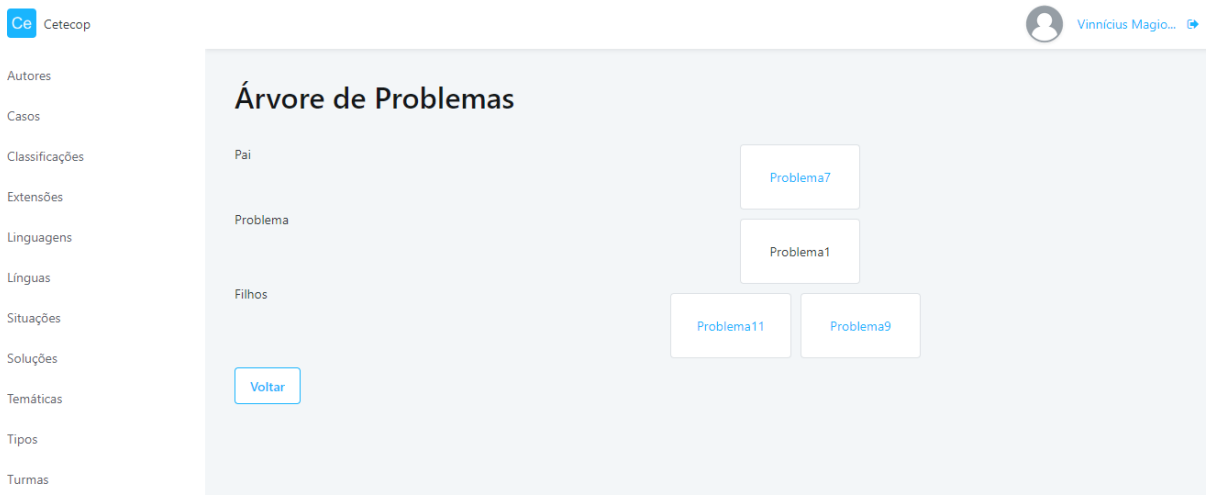
Fonte: Elaborado pelo Autor.

Ao salvar um problema é verificado primeiro se o problema já existe ou não, para determinar se irá incluir um novo, ou se irá alterar um já existente. Caso o problema não exista, é feita uma listagem de todos outros problemas já salvos no GitLab para criar um novo identificador ainda não utilizado. Em seguida, a entidade é serializada e então incluída no projeto do GitLab através da função InserirProblemaGitLab. Caso o problema já exista, ele é salvo com o identificador que ele já esta utilizando, assim apenas sendo serializado e salvo através da função AlterarProblemaGitLab. Ao final de ambos os fluxos é retornada uma mensagem de retorno para o usuário.

5.11.3 Árvore de Problemas

Cada problema possui a opção de exibir a sua árvore, informando em formato de cartões o seu exercício pai, do qual ele veio, e a lista de exercícios filhos que ele gerou. Ao clicar no problema pai ou em algum dos filhos o usuário é direcionado para a árvore daquele problema. Um exemplo disto pode ser observado na Figura 40.

Figura 40 – Tela da árvore de um determinado problema.



Fonte: Elaborado pelo Autor.

6 Estudo de Caso

Este capítulo apresenta um experimento de implementação da arquitetura proposta. Com os conceitos apresentados no referencial teórico e o desenvolvimento, construiu-se uma plataforma efetiva e funcional para aplicar para professores e alunos. Com isso pretende-se demonstrar a utilização das tecnologias na prática e validar todo o processo desenvolvido na fase conceitual.

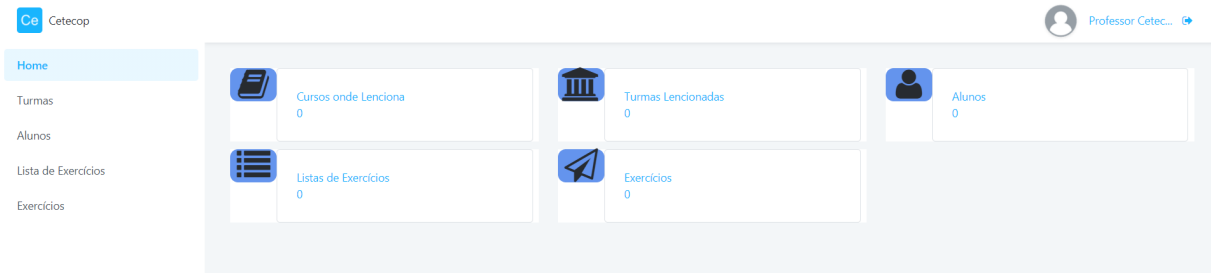
O experimento consiste em um Mínimo Produto Viável (MVP) contendo as funcionalidades descritas neste trabalho. Para a validação do mesmo, a seguir serão apresentadas as seguintes funcionalidades em execução.

- Tela principal do professor;
- Lecionar turmas e listagem de turmas lecionados;
- Listagem de alunos matriculados em turmas do professor;
- Tela do gestor de problemas;
- Criação de um novo problema;
- Criação de uma nova ramificação do problema;
- Visualizar ramificações de um problema;
- Solicitar união entre ramificação e o problema original;
- ela de gerenciamento de um problema;
- Incluir e excluir temáticas em um problema;
- Aceitar ou recusar união de uma ramificação com um problema;
- Visualizar árvore de relação do problema.

6.1 Página Principal do Professor

Após ser feito o cadastro em uma tela de novo usuário, informando o tipo de usuário, sendo este professor, e de terem sido feitas as validações de *login*, o usuário é direcionado para a página principal do sistema, representada pela Figura 41. Nesta tela são apresentadas para os usuários suas principais funcionalidades, além de contadores informando a quantidade de cada entidade relacionada a ele já salva no sistema. Neste estudo de caso, por ser o primeiro login deste usuário, este não possuirá nada em seu nome ainda, assim mostrando um zero em todas informações na página principal.

Figura 41 – *Dashboard* do Professor.



Fonte: Elaborado pelo autor.

6.2 Listagem e Lecionar Cursos

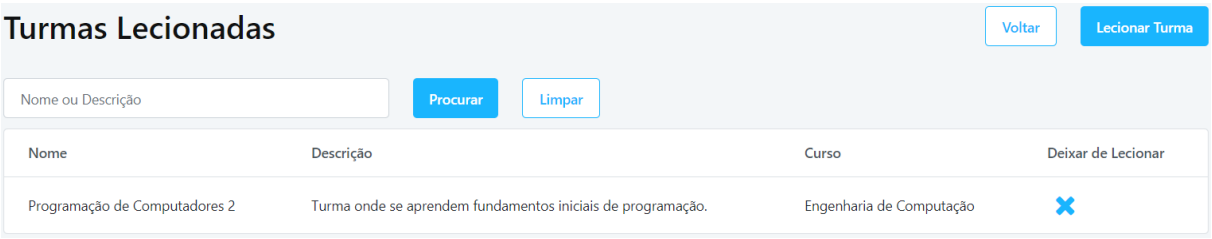
Para associar “Cursos” ao professor, basta associar uma turma a ele, pois esta já é cadastrada sempre associada a um curso em específico naquela instituição que o usuário se cadastrou. Já para associar uma turma, basta clicar em “Turmas” lecionadas que irá para uma página com uma listagem das turmas que o professor logado leciona. Como o usuário é novo no sistema ainda não existirá nenhuma turma na listagem. Para lecionar uma nova turma deve-se clicar no botão “Lecionar Turma”, e assim será direcionado para uma nova página onde, após filtrar por curso, o professor pode selecionar a turma a qual irá lecionar, como representada pela Figura 42.

Figura 42 – Lecionar Turma.

Fonte: Elaborado pelo autor.

Após clicar em salvar o usuário será direcionado novamente para a listagem de turmas lecionadas. Porém agora na listagem terá a turma selecionada incluída na tabela, como mostrado na Figura 43.

Figura 43 – Listagem de turmas lecionadas pelo professor.



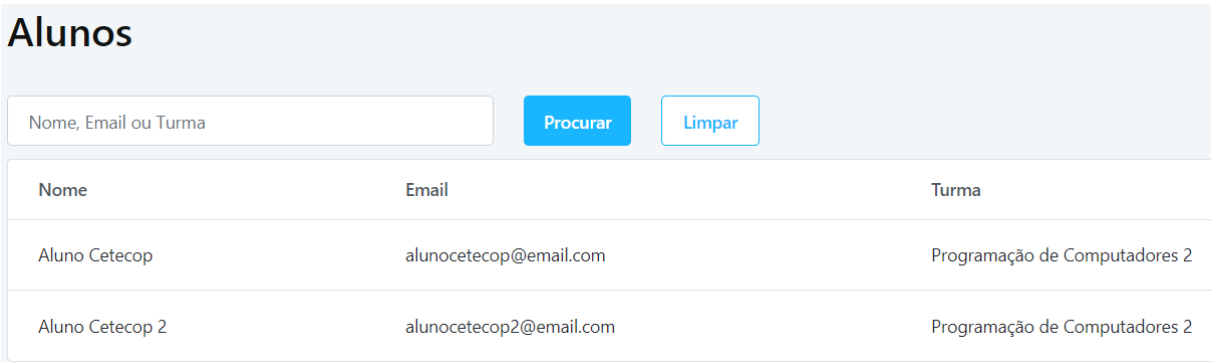
Fonte: Elaborado pelo autor.

Ao retornar para a página principal, seja clicando no botão voltar ou pelo menu a esquerda, os valores correspondentes ao número de cursos onde se leciona e de cursos lecionados passam a considerar a nova turma selecionada para se lecionar. Caso esta turma já possua alunos, os valor correspondente ao número de alunos que o professor possui já será atualizado também. Porém, como nenhum foi cadastrado ainda nem incluído em nenhuma turma, o valor permanece como zero.

6.3 Listagem de Alunos Matriculados em Turmas do Professor

A inclusão de alunos em turmas é possível de duas formas. A primeira é o próprio usuário do tipo aluno se matricular em uma determinada turma, já a segunda é o próprio administrador daquela "Instituição" incluir o aluno em uma turma. Para este estudo de caso, dois novos usuários do tipo "Aluno" foram cadastrados e matriculados na turma que o usuário do professor havia selecionado para lecionar. Assim, ao retornar para a listagem de alunos daquele professor são exibidos os dois novos alunos, como mostrado na Figura 44.

Figura 44 – Listagem de "Aluno" que cursam "Turmas" de um "Professor".



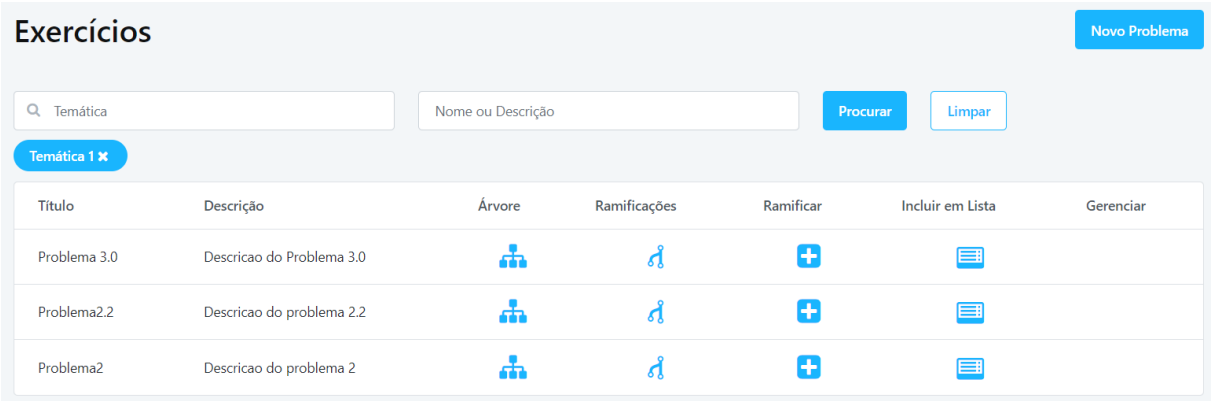
Fonte: Elaborado pelo autor.

6.4 Gestor de Problemas do Professor

Para gerenciar problemas o professor pode clicar em "Problemas" na tela principal ou clicar em "Problemas" no menu a esquerda. Nesta página, representada na Figura 45, o usuário pode pesquisar por exercícios, através de filtros por nome, descrição, ou por temáticas.

É possível que se pesquise por quantas temáticas o usuário desejar, possibilitando assim uma filtragem mais específica. Ele poderá utilizar os problemas retornados em suas turmas através de listas de exercícios.

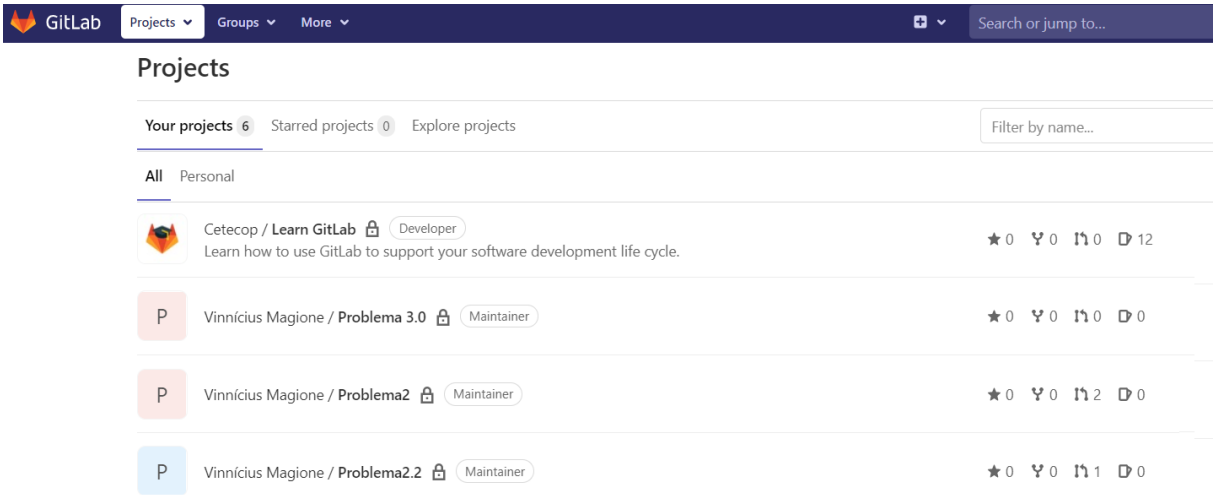
Figura 45 – Listagem de Problemas na tela do gestor de problemas.



Fonte: Elaborado pelo autor.

Como descrito neste anteriormente, tais problemas não são salvos em banco de dados, e sim como arquivos JSON, cada um salvo em um projeto no GitLab. Assim, o retorno da pesquisa traz apenas aqueles salvos no repositório de arquivos. Esta listagem condiz com os projetos salvos que estão no formato de um problema, isto é, possuem um arquivo JSON com o mesmo nome do título do projeto e que possua as informações do problema. Na Figura 46 é apresentado a lista de problemas salvos como projetos no GitLab.

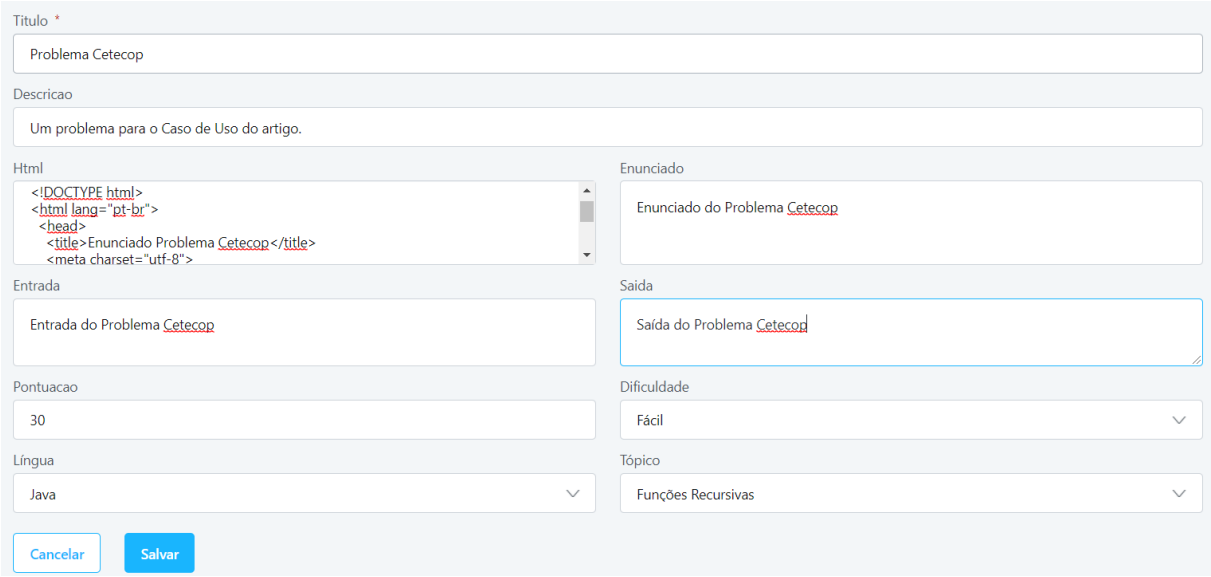
Figura 46 – Listagem de Projetos no GitLab



Fonte: Elaborado pelo autor.

Caso o professor clique no botão “Novo Problema”, na página de problemas, ele será encaminhado para uma nova página de cadastro, representada na Figura 47, onde poderá criar um novo problema que será salvo diretamente no GitLab em um arquivo JSON em um novo projeto com o mesmo nome.

Figura 47 – Cadastro de um novo Problema



Fonte: Elaborado pelo autor.

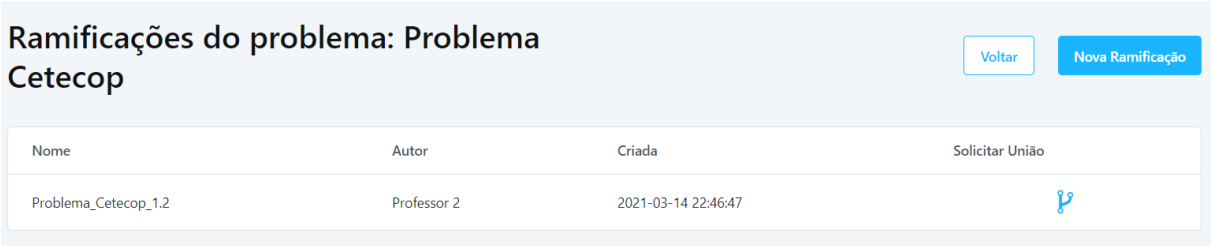
Ao clicar no botão “Salvar”, o novo problema é salvo no GitLab e o usuário é direcionado para a listagem de problemas novamente, e caso seja feita uma nova pesquisa, será possível encontrar o problema recém cadastrado. Como o usuário logado foi quem criou o problema ele é considerado seu autor. Sendo autor de um problema, o título deste problema, na listagem, aparece como um *link*, onde caso seja clicado é permitido editar os dados cadastrados deste problema.

O usuário pode, também, criar um novo problema a partir de um problema já existente. Ao clicar no ícone de “Ramificar”, uma nova página é aberta, semelhante a página de editar o problema, porém para o usuário que não é autor do problema só é permitido salvar como um novo problema ou salvar como uma ramificação daquele problema selecionado. Desta forma o problema original não será sobrescrito por outro usuário que não seja o próprio autor.

Ao salvar como um novo problema o sistema irá funcionar de forma semelhante a criar um novo problema do zero. Apenas irá salvar o problema novo com a autoria deste usuário. Caso seja salvo como uma ramificação, será criado uma nova ramificação no GitLab no mesmo projeto do problema original.

Para este estudo foi criado um segundo usuário do tipo Professor, e este criou uma nova ramificação do problema criado pelo primeiro professor. Ao retornar para a listagem de ramificações do problema esta nova ramificação é exibida, representada na Figura 48. Para o usuário autor desta ramificação é exibido a opção de solicitar união do problema original com a ramificação criada. Ao clicar neste ícone é criada uma nova solicitação de união no projeto do GitLab correspondente a aquele problema.

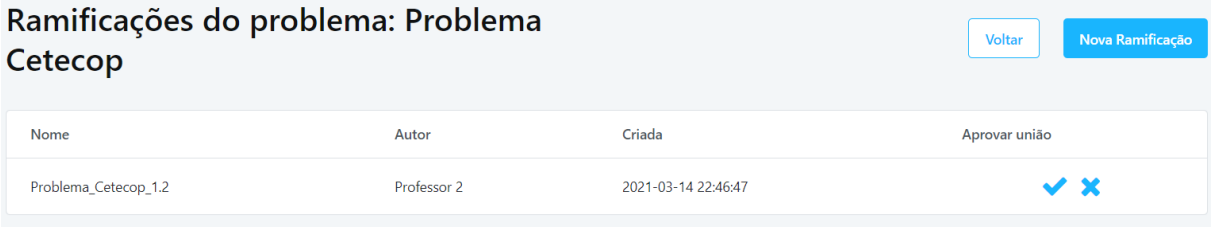
Figura 48 – Solicitações de união de um Problema



Fonte: Elaborado pelo autor.

Já para o usuário autor do problema, na listagem de ramificações aparece ícones de aprovar ou rejeitar uniões de ramificação que possuem solicitações de união com aquele problema original correspondente, como mostrado na Figura 49.

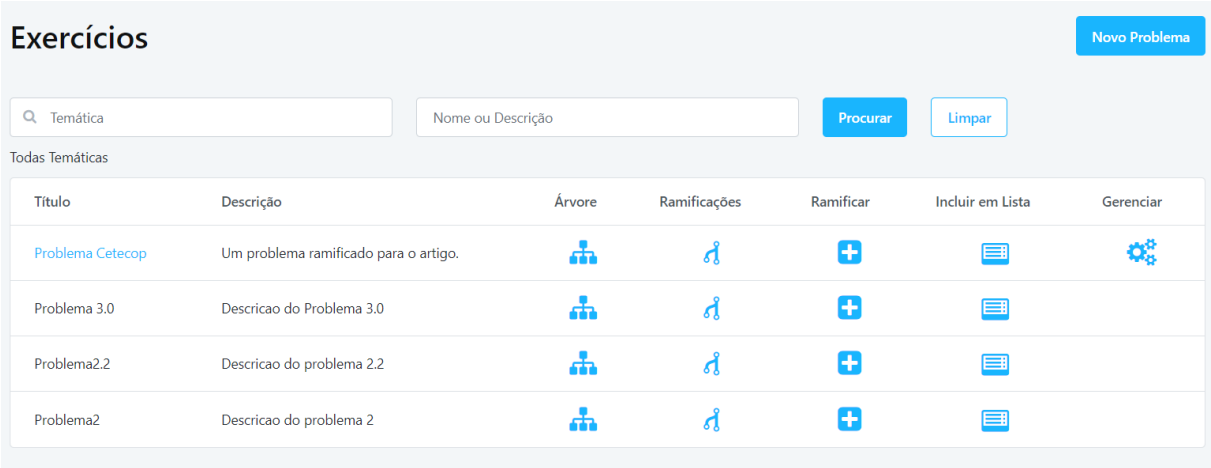
Figura 49 – Ramificações de um Problema



Fonte: Elaborado pelo autor.

Caso o autor do problema aceite a união daquela ramificação com o problema original o GitLab irá realizar a união, e assim o problema original irá adotar as alterações que existia naquela ramificação. No exemplo a seguir, após o usuário professor, que é autor do problema original, aceitar a união, o problema criado anteriormente no estudo, chamado de Problema Cetecop, passa a possuir uma descrição diferente da cadastrada por ele, sendo esta descrição uma criada na ramificação pelo segundo usuário de teste.

Figura 50 – União de Problemas.



Fonte: Elaborado pelo Autor.

7 Considerações Finais

Através da plataforma OutSystems o desenvolvimento de um protótipo funcional pode ser desenvolvido como proposto. A agilidade fornecida permitiu um menor tempo investido no desenvolvimento do sistema, em relação a um desenvolvimento de plataformas *web* usando programação convencional. Isto permitiu um maior foco no estudo do cenário do problema, nas tecnologias envolvidas no artigo e também dos resultados obtidos por esta dissertação.

Sobre o GitLab, este se mostrou eficiente no gerenciamento de arquivos além de possuir um controle robusto de versionamento. Seu uso, através de sua API vasta e detalhada, em conjunto com o sistema desenvolvido, mostrou ser possível a interação de duas pessoas em um mesmo arquivo, ficando registrado o histórico do desenvolvimento e alterações.

A utilização de um gerenciador de repositório como ferramenta para armazenamento de dados se mostrou como uma alternativa viável a utilização de bancos de dados comuns para alguns cenários. Para dados que possuam uma maior possibilidade de serem modificados por mais de uma pessoa em um determinado ambiente, o armazenamento comum pode não apresentar histórico de acesso e de modificação de um arquivo de uma maneira detalhada o suficiente. Isso torna difícil identificar qual usuário alterou qual parte de um arquivo, por exemplo. Em contra partida, através de uma ferramenta como o Git e GitLab, não só cada alteração fica registrada, mas também quem a fez e qual foi a alteração. Isto traz uma maior segurança e confiabilidade, além de possibilitar uma maior cooperação entre diferentes pessoas em um mesmo documento.

Além disto, a utilização de uma ferramenta gráfica para fazer o controle das ações do GitLab se mostrou interessante. Gerenciadores de versões possuem termologias próprias, além de todo um fluxo de funcionamento que um usuário leigo podem não se sentir confortável de utilizar em qualquer tarefa. Já com uma ferramenta que faça tudo isso para o usuário através de uma interface mais simples, isto pode incentivar o uso da tecnologia, diminuindo a curva de aprendizado e tornando sua utilização mais chamativa e prática.

7.1 Trabalhos Futuros

A seguir são apresentadas algumas propostas de trabalhos futuros. Estas sugestões visam melhorar o desenvolvimento da arquitetura e suas funcionalidades, dando continuidade ou executando novas implementações não abordadas neste trabalho. Portanto, recomenda-se:

- Estudo e desenvolvimento dos outros módulos propostos na arquitetura;
- Realização da integração do modulo proposto com os módulos remanescentes;
- Estudo da arquitetura e do sistema completo;

- Estudo mais aprofundado dos recursos do Git e GitLab para desenvolvimento de outras tecnologias;
- Estudo e utilização do GitLab em projetos diferentes, visando uma maior validação da ideia proposta também para outros cenários.

Referências

ALONSO, A. T.; MIRANDA, H. L. H. Módulos para el entrenamiento de los concursantes del acm-icpc. Santa Clara, Cuba, p. 101, 2014. Disponível em: <https://dspace.uclv.edu.cu/handle/123456789/1192>. Acesso em: 06 abril. 2021. Citado na página 36.

BEZ, J. L.; TONIN, N. A. Uri online judge academic: A tool for algorithms and programming classes. p. 149–152, Aug 2014. Disponível em: <https://ieeexplore.ieee.org/document/6926445>. Acesso em: 06 abril. 2021. Citado na página 29.

BOCA. *BOCA Online Contest Administrador*. 2018. Disponível em: <https://www.ime.usp.br/~cassio/boca/>. Acesso em: 17 jun. 2018. Citado na página 23.

BRITO, R. F. D.; PEREIRA, T. A. C. Um estudo para ambientes colaborativos e suas ferramentas. 2004. Disponível em: <http://www.avaad.ufsc.br/moodle/prelogin/publicarartigos/artigos04/ronnieconahpa.pdf>. Acesso em: 06 abril. 2021. Citado na página 17.

CAMPOS, C.; FERREIRA, C. Boca: um sistema de apoio a competições de programação. In: *Workshop de Educação em Computação*. [S.l.]: Sociedade Brasileira de Computação, 2004. Citado na página 23.

CODERBYTE. *Coderbyte*. 2021. Disponível em: <https://coderbyte.com/organizations>. Acesso em: 13 abr. 2021. Citado na página 31.

CODINGBAT. *CodingBat*. 2021. Disponível em: <https://codingbat.com/about.html>. Acesso em: 13 abr. 2021. Citado na página 31.

COMBéFIS, S.; WAUTELET, J. Programming trainings and informatics teaching through online contests. v. 8, p. 21–34, 07 2014. Citado na página 22.

DEED. *Diretoria de Estatísticas Educacionais. Censo da educação superior*. 2015. Disponível em: <http://inep.gov.br/informacao-da-publicacao/-/asset_publisher/6JYIsGMAMkW1/document/id/493780>. Acesso em: 17 jun. 2018. Citado na página 18.

DEGELER, A. Gitlab is building a business with 0.1% of paying customers. 2014. Disponível em: <https://thenextweb.com/insider/2014/06/04/github-rival-gitlab-building-business-just-0-1-paying-customers/>. Acesso em: 30 mai. 2019. Citado na página 34.

DOMJUDGE. *DOMJudge. Manual para times.* 2021. Disponível em: <https://www.domjudge.org/docs/team-manual.pdf>. Acesso em: 17 jun. 2018. Citado nas páginas 23 e 44.

ELDERING, K. J. J.; WERTH., T. Domjudge [online]. 2014. Disponível em: <https://thenextweb.com/insider/2014/06/04/github-rival-gitlab-building-business-just-0-1-paying-customers/>. Acesso em: 30 mai. 2019. Citado na página 36.

FACEBOOK. *Facebook*. 2021. Disponível em: <https://www.facebook.com/pg/facebook/about/>. Acesso em: 01 mar. 2021. Citado na página 22.

GIST GITHUB. *Gist GitHub*. 2021. Disponível em: <https://gist.github.com/juliosilvacwb/9e31e6876d62dbfc8210485ee1deca3a>. Acesso em: 27 mar. 2021. Citado na página 32.

GIT. *Git*. 2021. Disponível em: <https://git-scm.com/about>. Acesso em: 27 mar. 2021. Citado na página 33.

GITHUB. *GitHub, About*. 2021. Disponível em: <https://github.com/about>. Acesso em: 01 mar. 2021. Citado na página 34.

GITLAB. *GitLab, Features*. 2021. Disponível em: <https://about.gitlab.com/features/>. Acesso em: 01 mar. 2021. Citado na página 34.

GOEBEL, G. Na introduction to shell programming. 2003. Disponível em: <http://www.faqs.org/docs/air/tsshell.html>. Acesso em: 01 nov. 2018. Citado na página 24.

GOMES, A. et al. Aprendizagem de programação de computadores: dificuldades e ferramentas de suporte. *Revista Portuguesa de Pedagogia*, v. 42, p. 161–179, 07 2008. Citado nas páginas 16 e 19.

GOOGLE. *Google*. 2021. Disponível em: <https://www.google.com.br/about/>. Acesso em: 01 mar. 2021. Citado na página 22.

ICPC. *ICPC Fact Sheet*. 2020. Disponível em: <https://icpc.global/newcms/worldfinals/pdf/Factsheet.pdf>. Acesso em: 27 mar. 2021. Citado na página 16.

KURNIA, A. L. A.; CHEANG, B. Online judge. *Comput. Educ.*, Elsevier Science Ltd., GBR, v. 36, n. 4, p. 299–315, maio 2001. ISSN 0360-1315. Disponível em: <https://doi.org/10.1016/S0360-1315(01)00018-5>. Citado na página 22.

MICROSOFT. *The largest Git repo on the planet*. 2017. Disponível em: <https://devblogs.microsoft.com/bharry/the-largest-git-repo-on-the-planet/>. Acesso em: 01 mar. 2021. Citado na página 32.

MYSQL. *MYSQL*. 2021. Disponível em: <https://www.mysql.com/>. Acesso em: 01 mar. 2021. Citado na página 24.

NIKOLAY, D. V. I. K. K.; TODOR, S. Open judge system (ojs) [online]. 2014. Disponível em: <https://github.com/NikolayIT/OpenJudgeSystem2014>. Acesso em: 17 jun. 2018. Citado na página 36.

OLANOFF, D. Ship it faster and cheaper – gitlab is github for your own server. 2011. Disponível em: <https://thenextweb.com/apps/2011/10/13/ship-it-faster-and-cheaper-gitlab-is-github-for-your-own-servers/>. Acesso em: 30 mai. 2019. Citado na página 34.

OLIVEIRA, O. C. Ambiente colaborativo para ensino e aprendizagem de programação de computadores e treinamento para competições. 2017. Disponível em: <https://drive.google.com/drive/folders/1OyKTOec_B1X7J1JAd1e0fnlT9k0bf8xG>. Acesso em: 01 mar. 2021. Citado nas páginas 39 e 41.

OUTSYSTEMS. *OutSystems*. 2018. Disponível em: <https://https://www.outsystems.com/>. Acesso em: 01 nov. 2021. Citado na página 37.

PACHECO, P. Computer-based assessment system for e-learning applied to programming education. dissertação (mestrado) — feup. 2010. Citado na página 35.

PHP. *PHP*. 2021. Disponível em: <http://php.net/>. Acesso em: 1 mar. 2021. Citado na página 24.

PROJECT EULER. *Project Euler*. 2021. Disponível em: <https://projecteuler.net/>. Acesso em: 13 abr. 2021. Citado na página 30.

QUADROS, G. de. As novas tecnologias no processo de aprendizagem de idiomas. 2012. Disponível em: <http://www.ucs.br/etc/conferencias/index.php/anpedsul/9anpedsul/paper/viewFile/1310/919>. Acesso em: 13 abr. 2018. Citado nas páginas 35 e 36.

RED HAT. *Red Hat Linux*. 2021. Disponível em: <https://www.redhat.com/pt-br>. Acesso em: 1 mar. 2021. Citado na página 23.

RODRIGUES, F. S. Estudo sobre a evasão no curso de ciência da computação da ufrgs. Porto Alegre, p. 82–91, 2013. Disponível em: <https://lume.ufrgs.br/handle/10183/77275>. Acesso em: 06 abril. 2021. Citado na página 18.

ROUNTREE, N. et al. Interacting factors that predict success and failure in a cs1 course. In: . New York, NY, USA: Association for Computing Machinery, 2004. (ITiCSE-WGR '04), p. 101–104. ISBN 9781450377942. Disponível em: <https://doi.org/10.1145/1044550.1041669>. Citado na página 16.

SPOJ. *SPOJ Brasil*. 2021. Disponível em: <http://br.spoj.com.com.br>. Acesso em: 13 abr. 2018. Citado nas páginas 17, 23, 27 e 28.

STACK OVERFLOW. *Stack Overflow Developer Survey*. 2015. Disponível em: <https://insights.stackoverflow.com/survey/2015#tech-sourcecontro>. Acesso em: 01 mar. 2021. Citado na página 32.

STEVEN, H. F. H. Competitive programming. [s.l.]: Lulu. 2010. Citado na página 40.

TOPCODER. *TopCoder.* 2021. Disponível em: <https://www.topcoder.com/company/>. Acesso em: 17 jun. 2018. Citado na página 23.

TUROFF, M.; HILTZ, S. R. Computer support for group versus individual decisions. *IEEE Trans. Commun.*, v. 30, p. 82–91, 1982. Citado na página 20.

UNIVERSITY, D. Dhu online judge [online]. 2009. Disponível em: <http://acm.dhu.edu.cn/2014>. Acesso em: 13 abr. 2018. Citado na página 36.

URI. *URI Online Judge*. 2021. Disponível em: <https://www.urionlinejudge.com.br/judge/en/login>. Acesso em: 17 mar. 2021. Citado nas páginas 23 e 29.

UVA. *UVa Online Judge*. 2021. Disponível em: <http://uva.onlinejudge.org>. Acesso em: 13 abr. 2021. Citado nas páginas 23 e 28.

WAZLAWICK, R. S. Metodologia de pesquisa para ciência da computação. [s.l.]. 2009. Citado na página 37.

XAVIER, J. C. Computer-based assessment system for e-learning applied to programming education. 2011. dissertação (mestrado) — feup. 2011. Citado na página 35.

ZHIGANG, S. Moodle online judge [online]. 2014. Disponível em: <https://github.com/hitmoodle/moodle-local_onlinejudge2014>. Acesso em: 13 abr. 2018. Citado na página 36.

8 Apêndice

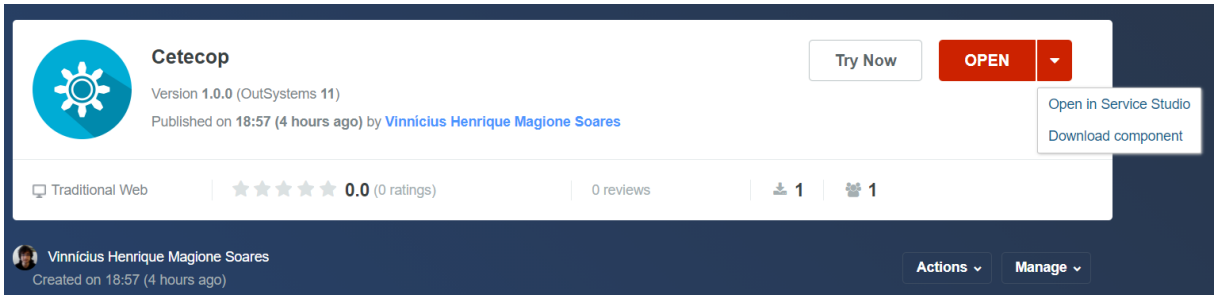
8.1 Código Fonte

A plataforma OutSystems não disponibiliza o código fonte gerado de suas aplicações, porém é possível importar e exportar projetos. Ela possui também um repositório de módulos e componentes, conhecida como Forge, onde é possível que a comunidade publique seus projetos para que outros desenvolvedores também possam usufruir.

Foi publicado na Forge a aplicação desenvolvida neste trabalho, utilizada para o estudo de caso. Seu nome é CETECOP. Para acessa-la acesse a url:

<https://www.outsystems.com/forge/Component_Overview.aspx?ProjectId=10594>

Figura 51 – Plataforma Cetecop publicada na Forge.



Fonte: Elaborado pelo autor

Para poder baixar a aplicação é necessário ter uma conta no OutSystems e ter a plataforma na versão 11 ou superior instalada.