Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

CD-MOJ: Contribuições para melhorias no sistema

Autor: Luciano dos Santos Silva
Orientador: Prof. Dr. Bruno César Ribas

Brasília, DF
2022

Luciano dos Santos Silva

CD-MOJ: Contribuições para melhorias no sistema

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Bruno César Ribas

Coorientador: Prof. Dr. John Lenon Cardoso Gardenghi

Brasília, DF

2022

Agradecimentos

Aos meus pais e irmão, que sempre me incentivaram e me apoiaram nos momentos difíceis, me dando forças para prosseguir em meu caminho.

Agradeço aos professores que, ao longo do curso, contribuíram com conhecimento e lições, não somente sobre a vida acadêmica, mas também de vida, moral e ética para minha formação profissional.

Aos meus orientadores, Prof. Dr. Bruno César Ribas e Prof. John Lenon Cardoso Gardenghi, pela oportunidade de retribuir os conhecimentos recebidos ao longo dos anos para a Universidade de Brasília e sua comunidade. Também agradeço pela dedicação e paciência com que conduziram o trabalho e compartilharam seu conhecimento.

Por fim, agradeço a todos meus amigos que começaram comigo essa jornada, aos que já terminaram-na e aos que, por algum motivo, não conseguiram finalizar. Todos tiveram importância em minha formação acadêmica.

Resumo

Juízes online são sistemas criados para testar soluções de problemas em código-fonte em competições de programação. Esses sistemas compilam, executam e comparam o resultado desse código com saídas pré-definidas. O *Contest Driven Meta Online Judge*, criado inicialmente com o intuito de facilitar o treinamento de equipes para a Maratona de Programação da UTFPR — Campus Pato Branco, mostrou-se capaz de fornecer suporte a grandes intensidades de tráfego e atua agora como apoio para as disciplinas de programação na Universidade de Brasília. Esse trabalho contribui com um acréscimo de uma área para esclarecimentos dentro do sistema, com a implementação de um novo mecanismo para verificar plágio nas submissões e com uma instância do CD-MOJ disponibilizada como um ambiente de desenvolvimento através de uma imagem *docker* e a documentação do sistema.

Palavras-chave: juízes online; competições de programação; plágio; submissões; desenvolvimento; esclarecimentos; documentação.

Abstract

Online judges are systems created to test solutions to problems in source code in programming competitions. These systems compile, run and compare the result of this code with predefined outputs. The Contest Driven Meta Online Judge created initially for the purpose of to facilitate the training of teams for the UTFPR's Programming Marathon — Pato Branco Campus, proved capable of providing support to high traffic intensities and now acts as a support for programming disciplines at the University of Brasilia. This work contributes with the addition of an area for clarification within the system, with the implementation of a new mechanism to check for plagiarism in submissions, and with an instance of CD-MOJ available as a development environment through a docker image and the documentation of the system.

Key-words: online judges; programming competitions; plagiarism; submissions; development; clarification; documentation.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Interface de programação de aplicações
CD-MOJ	Contest-Driven Meta Online Judge
CGI	Common Gateway Interface
HTML	Linguagem de Marcação de Hipertexto
JSON	JavaScript Object Notation
PHP	Linguagem de programação
PostgreSQ	Sistema gerenciador de banco de dados objeto relacional
Spoj	Sphere Online Judge
TCC1	Trabalho de Conclusão de Curso 1
TCC2	Trabalho de Conclusão de Curso 2
UFAM	Universidade Federal do Amazonas
UTFPR	Universidade Tecnológica Federal do Paraná

Sumário

1 Introdução

A Maratona de Programação é um evento da Sociedade Brasileira de Computação que existe desde o ano de 1996. A Maratona nasceu das competições classificatórias regionais para as finais mundiais do concurso de programação, o *International Collegiate Programming Contest*[1], e é parte da regional sul-americana do concurso (SBC. . . , 1996). É destinada a alunos de cursos de graduação ou pós-graduação inicial na área de computação. As equipes que participam da maratona são compostas por três estudantes que tentam resolver o maior número possível de problemas durante cinco horas.

Juízes online são bastantes utilizados nas maratonas de programação, nas quais equipes se reúnem para competir e resolver problemas de diversas dificuldades. A avaliação desses problemas é feita por submissões para um determinado sistema que executa o código e compara o resultado obtido com o resultado esperado já armazenado na plataforma.

O componente chave deste ambiente de *contest* é um sistema que verifica automaticamente as respostas das soluções apresentadas pelos participantes. O juiz online compara o resultado obtido ao que foi enviado, baseado nos resultados esperados de sua execução com conjuntos de testes predefinidos. Também é verificado se a solução não excede os limites de utilização de recursos (como tempo e memória) (WASIK et al., 2018).

Durante a concepção e implementação de sistemas de juízes online, muitos fatores importantes devem ser considerados. A questão chave é a segurança de tal sistema. O conceito que um juiz online assume é que o usuário submete a solução como o código-fonte que será avaliado na próxima etapa, geralmente em uma infraestrutura baseada em nuvem. Os desenvolvedores dos juízes online devem garantir que o sistema seja resistente a uma ampla gama de ataques como: forçar um alto tempo de compilação, modificar o ambiente de teste ou acessar recursos durante o processo de avaliação da solução (WASIK et al., 2018).

O *Contest Driven Meta Online Judge*(CD-MOJ), é um sistema de *online judge* robusto com capacidade de processar uma quantidade elevada de submissões. O CD-MOJ atua no apoio de disciplinas de programação, que requerem muitas horas de prática e uma avaliação rápida das soluções enviadas. Somente no segundo semestre de 2022, o CD-MOJ corrigiu 28.700 submissões em 59 listas, e desde o início do ano, foram 80.800 submissões em 126 listas.

[1] Se trata de uma competição mundial de programação entre universidades que ocorre anualmente. Disponível em: https://icpc.global/

1.1 Contribuições

O objetivo geral deste trabalho é contribuir com melhorias para o Juiz online *Contest-Driven Meta Online Judge*(CD-MOJ), de forma que seja possível fazer esclarecimentos durante um *contest* e também avaliar as soluções submetidas para evitar ocorrências de plágios. Além disso, garantir uma imagem *docker* para contribuir no desenvolvimento de futuras melhorias e uma documentação para auxiliar no desenvolvimento e uso do CD-MOJ.

As contribuições específicas feitas nesse trabalho são:

- Área para esclarecimentos;
- Imagem *docker* para instalação de instância do CD-MOJ como ambiente de desenvolvimento;
- Verificador de plágio JPlag[2] implantado;
- Documentação CD-MOJ;

1.2 Organização do Trabalho

Este trabalho está organizado da seguinte maneira: No Capítulo 2, será contextualizado toda a base utilizada como referência para aplicação das práticas. No Capítulo 3, será explicada a estrutura e arquitetura em que o CD-MOJ está baseado. No Capítulo 4, será descrito todos os resultados alcançados através desse trabalho. Finalmente, no Capítulo 5, será apresentada uma análise do que foi desenvolvido no trabalho e previsões para os futuros.

[2] JPlag é um sistema que encontra semelhanças entre vários conjuntos de arquivos de código-fonte. Disponível em: https://github.com/jplag/JPlag

2 Juízes online

Neste Capítulo serão apresentados os conceitos e conhecimentos necessários para o entendimento da proposta e a execução do trabalho. Serão apresentados esclarecimentos sobre o que é um juiz online e quais são as plataformas disponíveis atualmente.

2.1 Juízes online

Juízes online são sistemas responsáveis por compilar, executar e realizar avaliação automática dos códigos-fonte submetidos (WASIK et al., 2018). Geralmente, encontram-se embarcados em uma plataforma web com um repositório de problemas com variados desafios e dificuldades.

Juízes online testam algoritmos, que são procedimentos para resolver uma tarefa. É a ideia por trás de um software. Um algoritmo deve ser projetado para resolver um problema geral e bem especificado. O problema a ser resolvido é especificado descrevendo o conjunto completo de entradas e o retorno de cada instância do problema (PEREIRA, 2015 apud SKIENA; REVILLA, 2003).

Existem diversas plataformas com juízes onlines na internet e cada uma delas possui alguma funcionalidade que pode ser específica daquela plataforma. Alguns exemplos de plataformas são: BOCA (BOCA..., 2010), MOJO (CHAVES et al., 2013), *Beecrowd* (BEECROWD, 2022), SPOJ (SPHERE..., 2022), *CodeBench* (CODEBENCH, 2022) e DMOJ (DMOJ..., 2022).

2.1.1 BOCA

O BOCA (BOCA..., 2010) é um sistema de apoio a competições de programação e também com foco de ser usado no apoio de disciplinas que recorrem à submissão e à correção de trabalhos durante as aulas. Responsável por todo o gerenciamento de uma competição, se faz necessário uma *staff*, que são assistentes para desempenhar algumas funções necessárias que podem surgir ou já foram delegadas para aquela competição. A avaliação das submissões realizadas são feitas com auxílio de um juiz, com a possibilidade de analisar e corrigir os programas submetidos, definindo as respostas que o participante receberá, podendo ser individual ou para um time. O sistema é dividido em cinco partes a variar conforme a especialidade de cada usuário: time, juiz, administrador, *staff* e placar.

Desenvolvido com uma interface web, o BOCA é um sistema de entrega de exercícios, com autenticação, controle de tempo e disponibilização de resultados, tudo em tempo real (CAMPOS, 2004). O projeto foi construído com as tecnologias PHP e banco de dados

PostgreSQL, como o objetivo de ser flexível para poder haver trocas de tecnologia e que não causem grande impacto. O intuito da utilização do PHP foi para que o sistema fosse portável para todos ambientes em que a linguagem esteja disponível, com o PostgreSQL utilizado para armazenar e controlar a concorrência. O sistema é implantado em cima de um servidor de páginas, no qual os dados dos participantes são armazenados para que caso ocorra algum problema, os dados possam ser reutilizados em uma nova máquina designada para a competição.

2.1.2 Módulo de Integração com os Juízes Online (MOJO)

O MOJO (CHAVES et al., 2013) é um módulo integrado ao *moodle* (MOODLE..., s.d) cujo objetivo é contribuir com a melhoria do processo de ensino e aprendizagem em disciplinas de programação de computadores. Essa integração contribui para ocorrer a automatização dos processos de Elaboração, Submissão e Avaliação das atividades de programação propostas pelo professor (CHAVES et al., 2013).

2.1.3 *Beecrowd*

O *Beecrowd* (BEECROWD, 2022), originalmente chamado *URI Online Judge*, se trata de uma plataforma composta por um repositório com diversos problemas de programação, separados por tópicos e por dificuldade. Os problemas são avaliados de forma automática por um juiz online e, a partir da submissão, são postos em uma fila para serem compilados, executados e avaliados. A plataforma contém diversas funcionalidades, como uma área para maratonas, acadêmicos, instrutores, ranks e um fórum para discutir sobre problemas.

2.1.4 SPOJ e SPOJ Brasil

SPOJ é uma plataforma que contém um juiz online para a avaliação automática de códigos fontes enviados pelos usuários. Dispõe de um repositório com mais de 13.000 problemas para prática disponíveis 24 horas, em diversos idiomas, como inglês, polonês, vietnamita e português (SPHERE..., 2022).

Alguns dos recursos disponíveis no sistema do SPOJ são:

- Suporte para mais de 45 linguagens de programação e compiladores;
- Um sistema flexível de testes que suporta uma interação dinâmica com os programas submetidos e uma saída altamente personalizável dos resultados de avaliação;
- Gerenciamento de conteúdos intuitivos baseados no navegador, que permite aos usuários configurar os próprios *contests* e utilizar os problemas já disponíveis no sistema;

- Possibilidade de contribuir com novos problemas para o repositório.

2.1.4.1 SPOJ Brasil

O SPOJ Brasil é uma versão em português do SPOJ, que também realiza avaliações automáticas de códigos fontes, porém com um repositório de problemas voltado para problemas regionais, seletivas ou olimpíadas de programação em português. O SPOJ Brasil também conta com a possibilidade de adicionar novos problemas na plataforma. A submissão de problemas é feita através de um e-mail **`wander@ime.usp.br`** e deve conter o enunciado da prova, arquivos de entrada, arquivos de saída e soluções dos juízes (SPOJ. . . , 2022).

2.1.5 *CodeBench*

O *CodeBench* (CODEBENCH, 2022) é um juiz online desenvolvido na Universidade Federal do Amazonas - UFAM com o propósito de automatizar a correção dos exercícios de programação. Nele, os professores podem disponibilizar exercícios de programação para seus alunos, que podem codificar soluções e submetê-las através da interface do sistema. Assim que o aluno submete uma solução para um certo exercício, o *CodeBench* informa se ela está correta ou não (GADELHA, 2016).

O *CodeBench* conta com um ambiente integrado de desenvolvimento, que possibilita a resolução dos problemas na própria plataforma. Também é disponibilizado, com esse ambiente, um terminal Linux em que o participante pode acessar um diretório específico onde um dado exercício fica disponível. Outros recursos também estão presentes, como a aplicação de conceitos de gamificação, disponibilização de materiais didáticos e de um banco de exercícios, detecção de plágio e a possibilidade de troca de mensagens entre os usuários do sistema.

2.1.6 DMOJ

DMOJ (DMOJ. . . , 2022) é um juiz online moderno e um repositório de problemas de programação *open-source*, que disponibiliza dados dos problemas, submissões, usuários e *contests* em tempo real. É disponibilizada uma documentação que possui informações que possibilitam a configuração de novas instâncias do site e do juiz online do DMOJ e explicações para o consumo dos seus serviços através de *endpoints*. Os serviços do DMOJ são expostos através de uma JSON API, e os respectivos *endpoints* são:

- **/api/v2/contests** - Retorna a lista de *contests*;
- **/api/v2/contest/<contest key>** - Detalha um *contest* específico;
- **/api/v2/participations** - Detalha um participante de um *contest* específico;

- **/api/v2/problems** - Retorna uma lista de problemas do DMOJ;
- **/api/v2/problem/<problem code>** - Detalha um problema específico;
- **/api/v2/users** - Lista os usuários de uma organização;
- **/api/v2/user/<user username>** - Detalha um usuário específico;
- **/api/v2/submissions** - Detalha as submissões conforme o usuário ou problema;
- **/api/v2/submission/<submission id>** - Detalha uma submissão especifica;
- **/api/v2/organizations** - Retorna uma lista de organizações;
- **/api/v2/languages** - Retorna a descrição de uma linguagem;
- **/api/v2/judges** - Retorna os Juízes em execução.

2.2 Comparação

São diversas as plataformas que trabalham com o sistema de juízes online, cada uma possui uma especificidade. O BOCA foi desenvolvido visando servir como um sistema de apoio a competições de programação e disciplinas que recorrem à submissão e correção de trabalhos durante a aula. Não somente o BOCA, como também o MOJO, foram desenvolvidos focados em contribuir com o aprendizado em disciplinas de programação, sendo o MOJO uma integração de juiz online com o *moodle*. O *CodeBench* é uma plataforma que conta com um juiz online e é voltada para a aprendizagem, disponibilizando materiais didáticos e conceitos de gamificação dentro da plataforma. *Beecrowd* é uma plataforma que, além de conter um repositório de problemas e um juiz online, contém outras funcionalidades que diversificam seu público alvo. O SPOJ é uma plataforma robusta, com um acervo com mais de 13.000 problemas, e que disponibiliza seu conteúdo em diversos idiomas, além do suporte para mais de 45 linguagens de programação e de possibilidade de contribuição. E, por fim, DMOJ, que diferentemente das outras plataformas, é um sistema *open-source* na qual seus serviços são expostos por *endpoints* e podem ser consumidos externamente.

3 *Contest Driven Meta Online Judge*

O *Contest Driven Meta Online Judge* ou CD-MOJ é um *online judge* desenvolvido pelo Prof. Dr. Bruno César Ribas, inicialmente com o intuito de facilitar o treinamento das equipes para a Maratona de Programação da Universidade Tecnológica Federal do Paraná(UTFPR) — Campus Pato Branco e também tornar possível a resolução de problemas das plataformas do SPOJ-BR e URI *online judge*. Atualmente CD-MOJ atua como uma plataforma de apoio para disciplinas de programação na Universidade de Brasília(UnB) e conta com um repositório de problemas para as disciplinas de Algoritmos e Programação de Computadores, Compiladores, Estrutura de Dados I e II, Fundamentos de Arquitetura de Computadores e Fundamentos de Sistemas Operacionais. A atual *interface* do CD-MOJ é demonstrada na Figura 1.

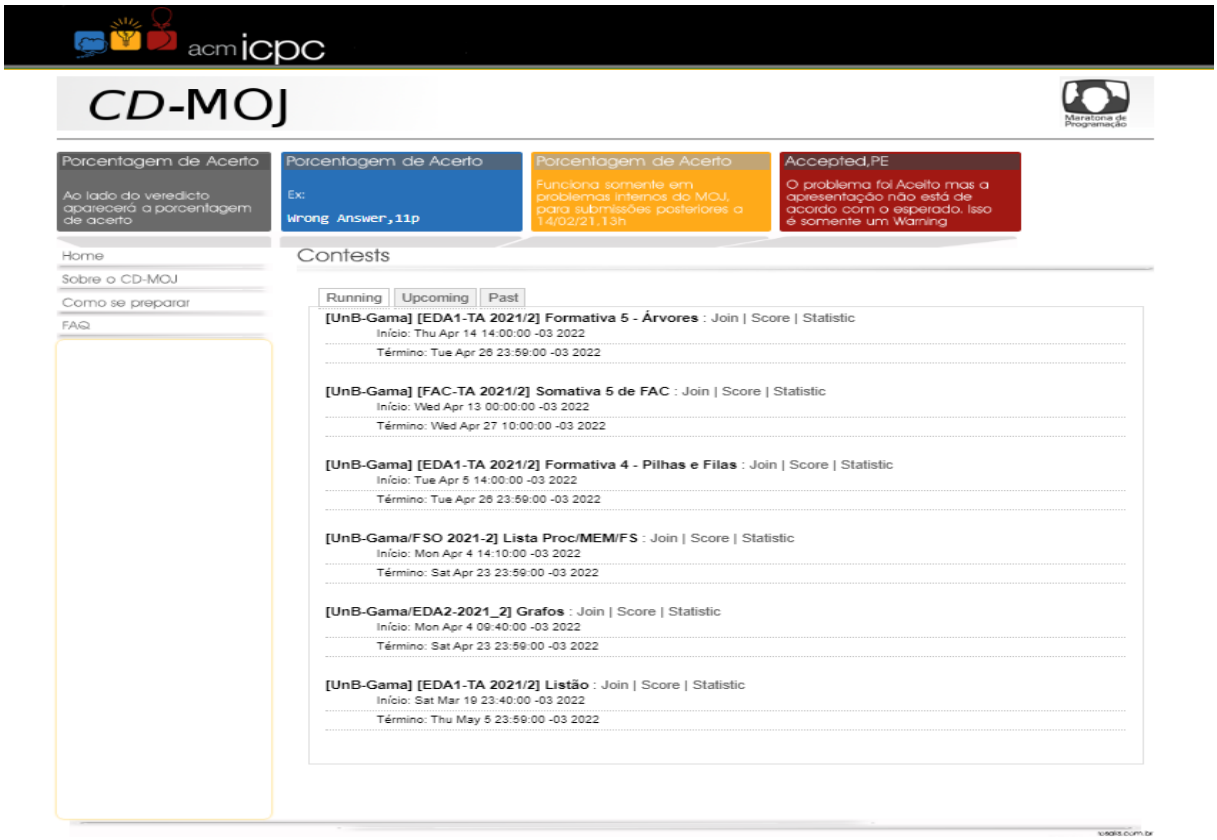

Figura 1 – Página inicial do CD-MOJ(CD-MOJ. . . , 2013)

O CD-MOJ pode disparar as submissões de código-fonte de problemas para outros juízes online e executar seu próprio módulo para compilar, executar e avaliar. A execução do sistema é realizada da maneira mostrada na Figura 2.

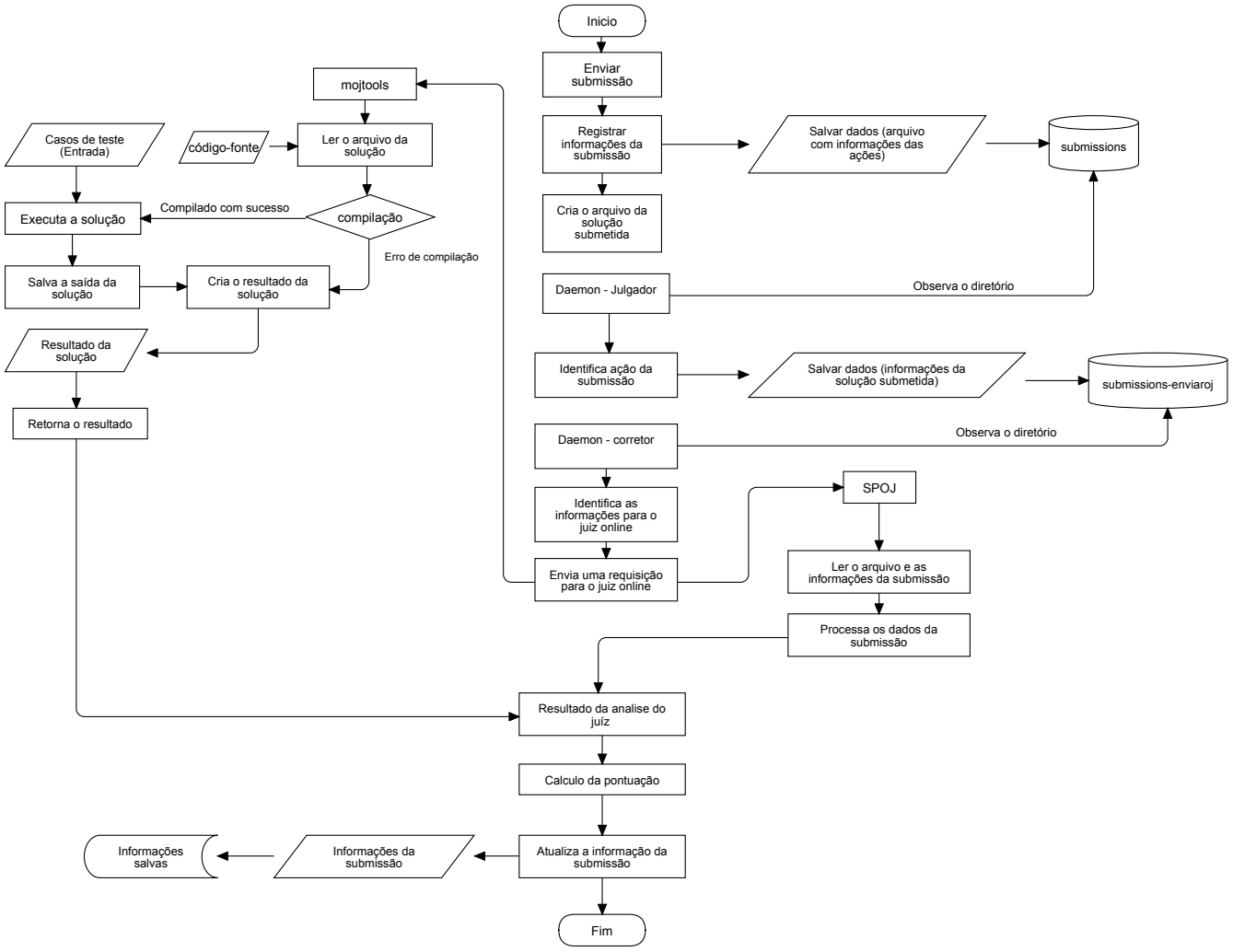

Figura 2 – Fluxograma do processo de julgamento automático do sistema de juiz online

O processo de julgamento automático do sistema começa quando o participante realiza uma submissão, onde as informações dessa submissão são salvas através de um arquivo criado no diretório **`submissions`**, esse diretório é observado pelo *daemon* do julgador que irá identificar o arquivo criado e verificar as informações contidas e a partir desse momento identificar que se trata de uma submissão, o arquivo da solução é salvo dentro do diretório **`submissions-enviaroj`** observado pelo outro *daemon* responsável pelo corretor. As informações da submissão são identificadas pelo corretor e uma requisição é disparada para o juiz online identificado. A submissão pode ser redirecionada para dois juízes online que são: o SPOJ e o próprio módulo de correção do CD-MOJ que é chamado de *mojtools*[1]. Para o SPOJ é realizada uma requisição para a API na qual vai ser realizada os processos para avaliação da submissão e após isso o resultado será retornado para que o cálculo da pontuação seja feito e as informações da submissão sejam atualizadas para

[1] *mojtools* se trata do módulo responsável por compilar, executar e avaliar as submissões do próprio CD-MOJ. Disponível em: https://github.com/cd-moj/mojtools

serem entregues ao participante. Para o *mojtools* são enviados os dados da submissão e logo após é feita a compilação do código-fonte enviado como solução, se a compilação for concluída com sucesso a solução é executada com casos de teste de entrada e após a execução a saída dessa solução é salva, assim como se houver erro de compilação também é salvo e retornado como um resultado da submissão. Quando o resultado da análise de algum juiz online é retornado, será feito o cálculo da pontuação e as informações da submissão são atualizadas para serem apresentadas ao participante.

3.1 Estrutura

Common Gateway Interface(CGI) é um mecanismo padrão de serviço de invocação que os servidores Web suportam para fornecer conteúdo dinâmico, páginas HTML criadas dinamicamente para responder a consultas/solicitações de usuários ou para executar scripts em segundo plano. Aplicações CGI escritas em linguagens compiladas como C ou interpretadas como Perl e Shell Script, adicionam ao servidor padrão Web um serviço de acesso à página HTML com um conjunto de funções do site, desde *gateways* de banco de dados até processamento de pedidos no comércio eletrônico (VENKITACHALAM; CHIUEH, 1999).

A estrutura do CD-MOJ se baseia no *Common Gateway Interface*(CGI), onde o usuário realiza uma requisição através de um *browser* para o servidor, onde será executado um *script*. Terminada a execução, os dados são retornados ao servidor e servidos ao *browser*. Essa estrutura é demonstrada na Figura 3.

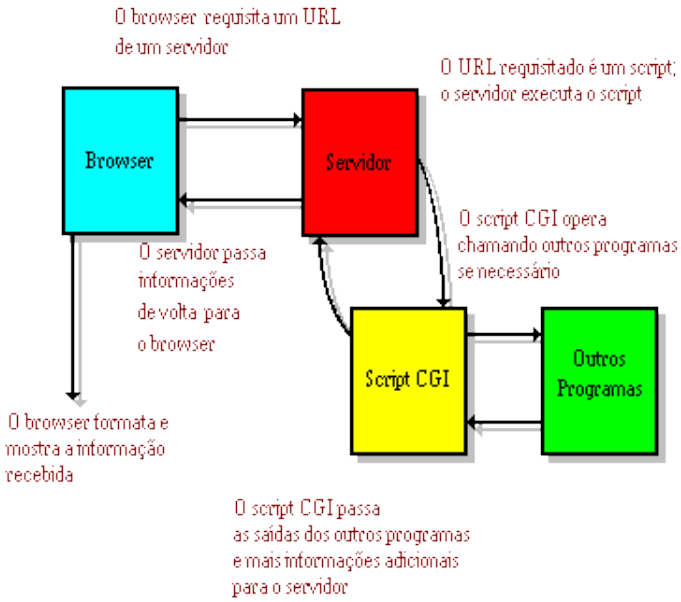

Figura 3 – Visão lógica do *Common Gateway Interface* (PORTO, 2022)

São diversas as interações necessárias para que o sistema execute conforme o planejado. Apesar de ser um sistema simples, a comunicação entre os *scripts* possui uma complexidade alta devido a muitas interações e dependências. As interações e dependências entre cada um dos scripts são descritas na Figura 4.

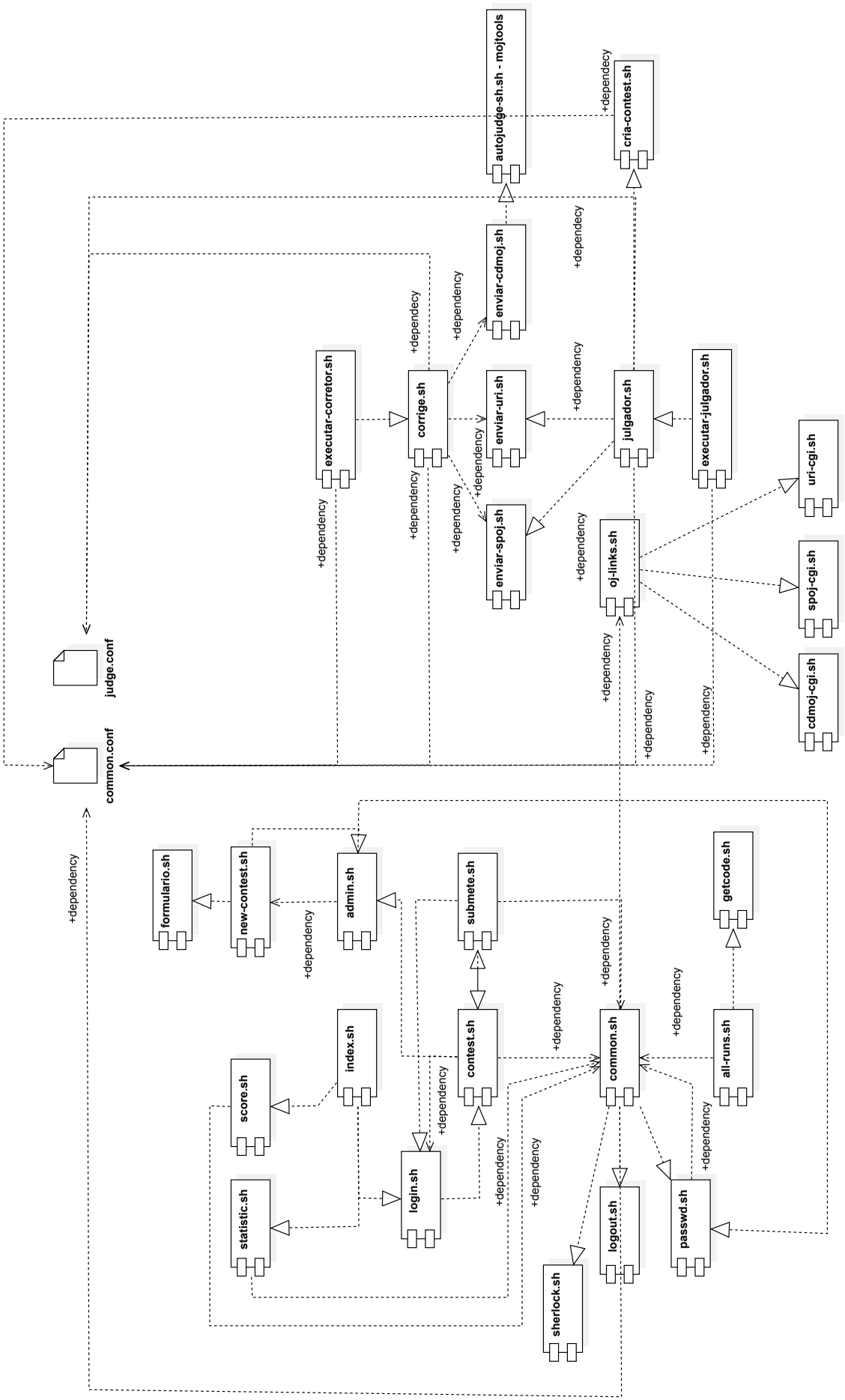

Figura 4 – Diagrama de componentes CD-MOJ

A Tabela 1 descreve os *scripts* que tratam a comunicação com as APIs externas e internas do CD-MOJ.

Tabela 1 – Descrição dos *scripts* de comunicação entre APIs

Componente	**Descrição**
`cdmoj-cgi.sh`	*Script* responsável por manipular todos os links referentes à API do CD-MOJ
`enviar-cdmoj.sh`	*Script* responsável por verificar o login, ativar o módulo juiz do CD-MOJ e obter o resultado da submissão
`enviar-spoj.sh`	*Script* responsável por verificar o login, enviar envios para a API SPOJ e obter o resultado
`enviar-uri.sh` [**Obsoleto**]	*Script* responsável por verificar o login, enviar submissões para a API URI e obter o resultado
`oj-links.sh`	*Script* responsável por redirecionar cada problema específico para sua própria API
`spoj-cgi.sh`	*Script* responsável por manipular os links referentes à API SPOJ
`uri-cgi.sh` [**Obsoleto**]	*Script* responsável por manipular os links referentes à API URI

A Tabela 2 descreve os *scripts* no diretório *judge*, que contém os principais *scripts* para o funcionamento do CD-MOJ.

Tabela 2 – Descrição dos *scripts* do diretório *judge*

Componente	**Descrição**
`corrige.sh`	Módulo responsável por corrigir todas as submissões pendentes
`julgador.sh`	Módulo responsável por julgar submissões, computar tentativas e penalidades. Acionar o *script* para criar *contests*, adicionar novos usuários através do bot mojinho

A Tabela 3 descreve os *scripts* dos *daemons* responsáveis por acionar os *scripts* da Tabela 2.

Tabela 3 – Descrição dos daemons

Componente	**Descrição**
`executar-corretor.sh`	*Daemon* responsável por disparar o *script* do módulo de correção de submissões
`executar-julgador.sh`	*Daemon* responsável por disparar o *script* do módulo de julgamento de submissões

A Tabela 4 descreve os arquivos de configuração que são utilizados para executar o CD-MOJ.

Tabela 4 – Descrição dos arquivos de configuração

Componente	**Descrição**
`common.conf`	Arquivo raiz de configuração
`judge.conf`	Arquivo com algumas variáveis para as APIs

A Tabela 5 descreve o módulo do juiz do CD-MOJ e o *script* de criar *contest*.

Tabela 5 – Descrição dos demais *scripts* do CD-MOJ

Componente	**Descrição**
`autojudge/mojtools`	Módulo responsável pelo juiz CD-MOJ
`cria-contest.sh`	Módulo responsável por criar um *contest*

A Tabela 6 descreve os *scripts* do diretório CGI-BIN responsáveis pelas operações no *client-side*.

Tabela 6 – Descrição dos *scripts* do diretório CGI-BIN

Componente	**Descrição**
`admin.sh`	Página de administrador do CD-MOJ
`all-runs.sh`	Apresenta todas as submissões para o administrador do *contest*
`common.sh`	*Script* responsável por todas as ações em comum do fluxo do CD-MOJ
`contest.sh`	Página principal dos *contests*, responsável por redirecionar para o login e mostrar os problemas do *contest* em execução
`formulario.sh`	Formulário para criação de novos *contests*
`getcode.sh`	*Script* responsável por coletar todas as respostas dos problemas e disponibilizá-las para visualização
`index.sh`	Home Page
`login.sh`	*Script* responsável por manipular o login e suas operações
`logout.sh`	*Script* responsável por manipular o logout e suas operações
`new-contest.sh`	Pagina de criação de novos *contests*
`passwd.sh`	*Script* responsável por alterar a senha
`score.sh`	*Script* responsável por computar a pontuação do *contest*
`sherlock.sh`	Módulo responsável pela detecção de plágio
`statistic.sh`	*Script* responsável por calcular as estatísticas do *contest* em execução
`submete.sh`	*Script* responsável por executar as submissões

3.1.1 Modelo de implantação

O modelo de implantação é composto por duas camadas, que correspondem com o modelo seguido pelo CGI. São elas: o servidor, responsável pelos componentes onde ocorrem a execução de todos os *scripts* e pela persistência de dados do sistema, o CGI-BIN, que renderiza o *website* e por fim, a camada do cliente, que apresenta as informações para o usuário através do *browser*. O modelo de implantação é descrito na Figura 5.

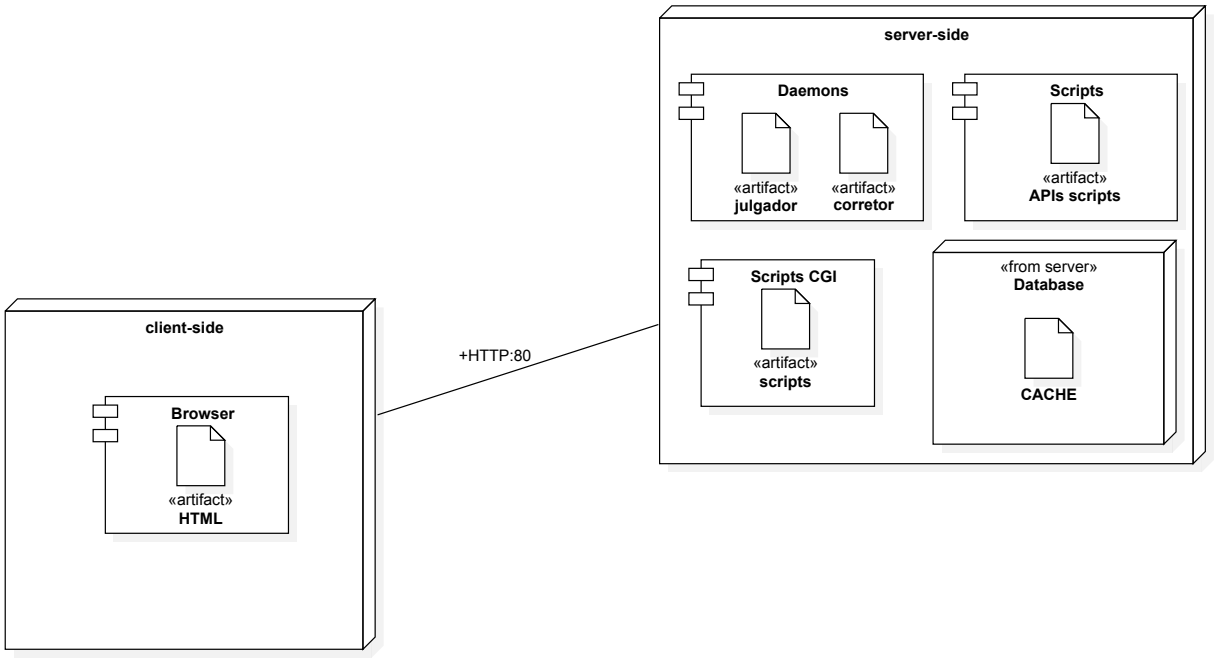

Figura 5 – Diagrama de implantação CD-MOJ

3.2 *Daemons*

Existem dois módulos responsáveis pela execução das duas principais funcionalidades do CD-MOJ, que são os *daemons* executados no servidor.

O *daemon* que tem a ação de executar o corretor é responsável por realizar um *loop* que verifica se há um novo arquivo de submissão no diretório `submissions-enviaroj`. Quando um novo arquivo é identificado, o *script* `corrige.sh` é disparado. Assim que é disparado, é são realizadas validações e a partir desse momento, os redirecionamentos para os juízes específicos de cada problema são executados e os resultados dos problemas pendentes são obtidos.

Quando o usuário submete um problema para o juiz, a submissão é salva dentro do diretório `submissions-enviaroj` especificado pelo *script* responsável por tratar das

submissões dentro do CD-MOJ. O *daemon* que executa o corretor fica observando o diretório até que uma submissão seja realizada e no momento em que é uma nova submissão é identificada, o *daemon* aciona o *script* `corrige.sh`. As validações e verificações dos dados são realizadas e se todas forem aceitas, o problema é enviado para a API específica. Ao retornar com o resultado, o mesmo é devolvido e salvo no *database*. Essa sequência é demonstrada no diagrama da Figura 6.

O *daemon* responsável pela execução do julgador também realiza um *loop* que fica aguardando novas submissões. A partir da detecção de uma nova submissão, o *script* `julgador.sh` é disparado, realizando todas as validações para atualizar as pontuações de um *contest*, computar as tentativas, acertos e penalidades, criar *contests* e adicionar um novo usuário ao *contest* através do bot mojinho[2].

O *daemon* responsável por executar o julgador fica esperando os resultados serem salvos e, após esse processo, é acionado o *script* `julgador.sh`. Ocorrerão validações e verificações para identificar os fluxos representados na Figura 7, e tais fluxos podem ter a possibilidade de adicionar um novo usuário pelo bot mojinho, realizar um *login*, criar um *contest*, rejulgar um problema, além de gravar soluções no *database* e computar as pontuações das listas, provas ou *contests*. Esses passos são demonstrados pelo diagrama de sequência da Figura 7.

A fundamentação acerca da estrutura e arquitetura em que o CD-MOJ está baseado atualmente é de extrema importância para o entendimento do funcionamento do sistema, e contribui para a clarificação de dúvidas sobre os componentes do sistema e suas funções desempenhadas durante a execução dele. Isso possibilita que uma documentação mais extensiva possa ser desenvolvida a partir do conteúdo apresentado neste capítulo.

[2] Mojinho é um bot do telegram que se comunica com o sistema do CD-MOJ. Onde é possível baixar soluções submetidas, obter *logs* das submissões, participar de alguma lista ou *contest* e trocar a senha de usuário.

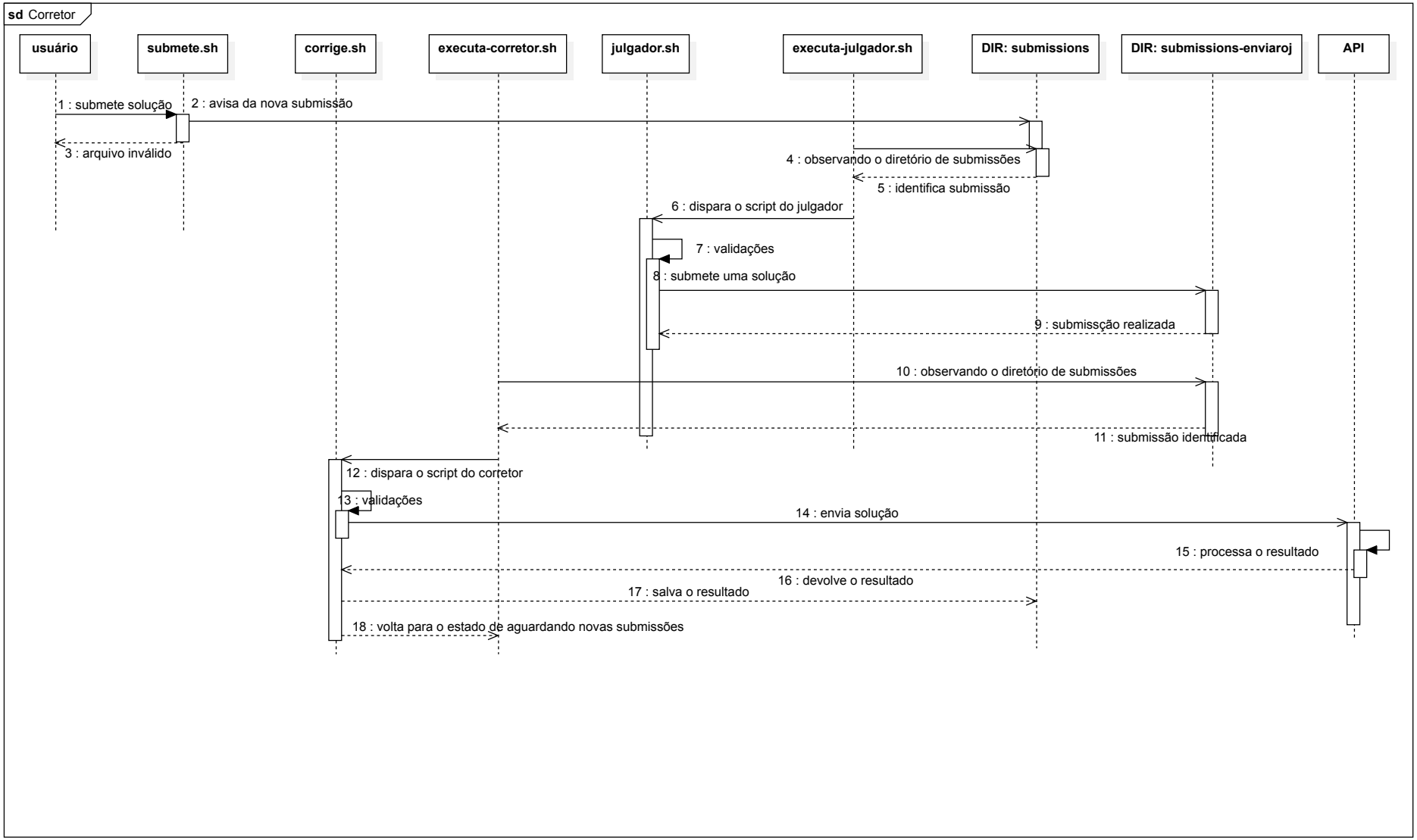

Figura 6 – Diagrama de sequência do corretor

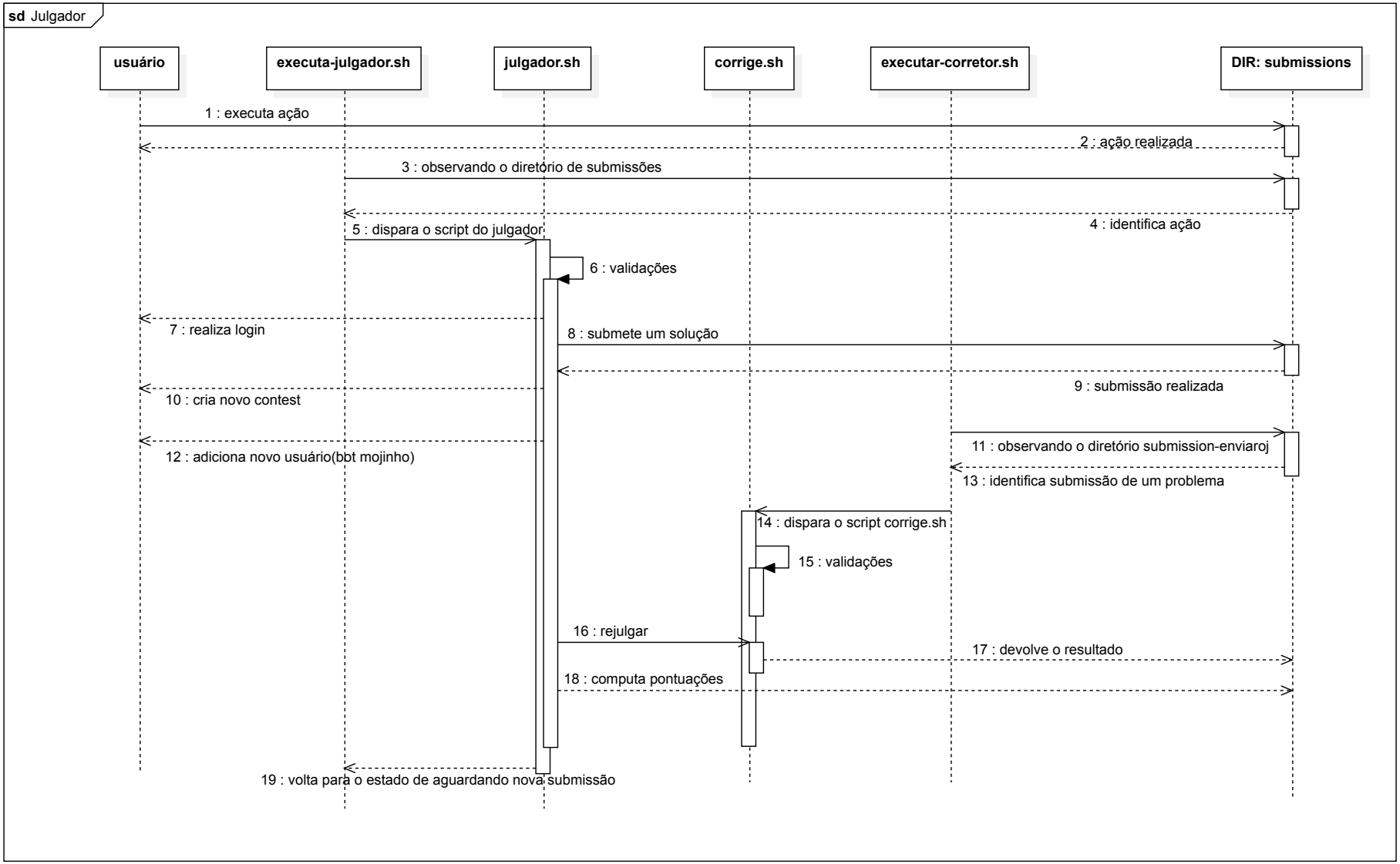

Figura 7 – Diagrama de sequência do julgador

4 Resultados

Neste Capítulo serão apresentadas as explicações e os resultados que foram obtidos no decorrer do desenvolvimento deste trabalho. Serão apresentados as seguintes Seções, cada uma representando um resultado obtido: Documentação do CD-MOJ, Implantação de área para soluções de dúvidas, Atualização do verificador de plágio para JPlag e Disponibilização de imagem *docker* para instalação de instância do CD-MOJ como ambiente de desenvolvimento.

4.1 Documentação do CD-MOJ

Foram documentadas informações importantes para o entendimento do funcionamento do sistema do CD-MOJ e também as que auxiliam o usuário que desejar instalar o CD-MOJ e executá-lo na sua própria máquina. A documentação é separada em três seções: Site, *Judge* e Formato de arquivo com *contest*.

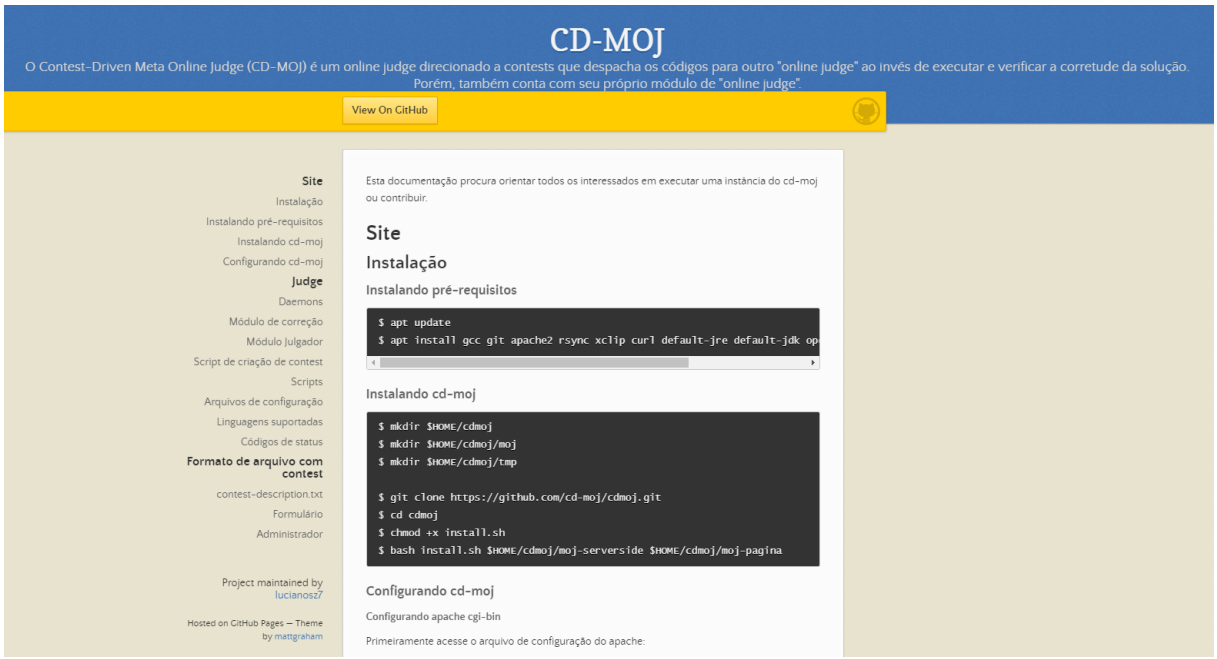

Figura 8 – Site da documentação do CD-MOJ

A primeira seção é dedicada a auxiliar a quem tem interesse de instalar um instância do CD-MOJ e configurá-la de forma que consiga executar todas suas funcionalidades. Por sua vez, a segunda seção tem como objetivo explicar o funcionamento do *judge* que seriam as partes relacionadas com os *daemons*, módulos de correção e julgador, *script*

responsável por criar os *contests*, os *scripts* que tratam as requisições com os outros juízes online, arquivos de configuração, as linguagens suportadas pelo CD-MOJ e os códigos de *status*. A última seção é responsável por explicar os formatos aceitos para a criação dos *contests*, os meios de criação de um *contest* e a criação de um usuário administrador para que os usuários possam ter o privilégio para enviar um arquivo com *contest* ou acessar o formulário.

O repositório dessa documentação é encontrado dentro da organização do CD-MOJ[1] no Github, e a documentação está aberta para contribuições e pode ser acessada através da URL https://cd-moj.github.io/cd-moj.docs. Para contribuir, é necessário fazer um *fork* do repositório e logo após feitas as mudanças, fazer um *pull request* para o repositório de origem.

4.2 Implantação de área para soluções de dúvidas

A área para soluções de dúvidas chamada *clarification* foi implementada, se baseando em duas abas *clarification* e respostas, para que os usuários possam enviar suas dúvidas ou para que os administradores possam fazer esclarecimentos.

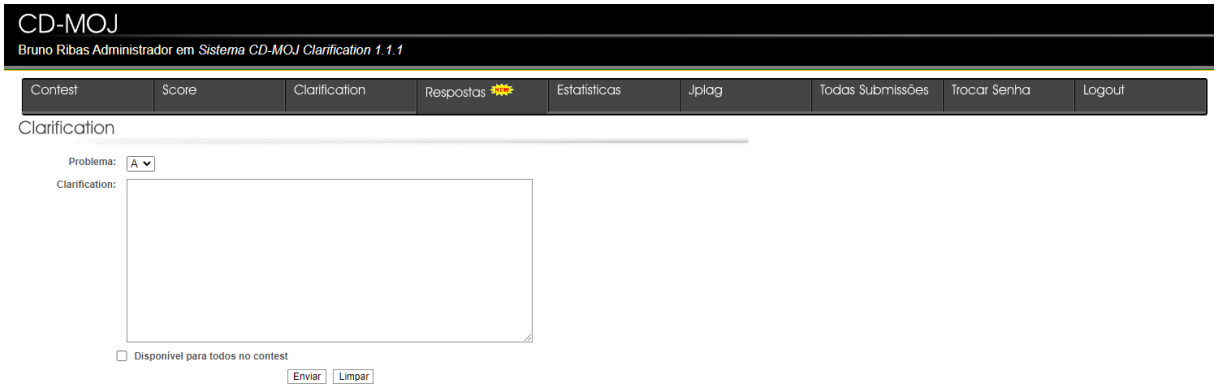

Figura 9 – Visão de um usuário comum para a área de *clarification*

Após enviar uma dúvida ou realizar um esclarecimento, é possível ir até a aba respostas e ser redirecionado para a página de respostas. Lá, pode-se visualizar a pergunta feita e se houve uma resposta. Para os administradores e monitores, a página é apresentada de acordo com Figura 11, e para os participantes comuns, a página é demonstrada conforme a Figura 10.

[1] Organização contendo todos os repositórios do CD-MOJ. Disponível em: https://github.com/cd-moj

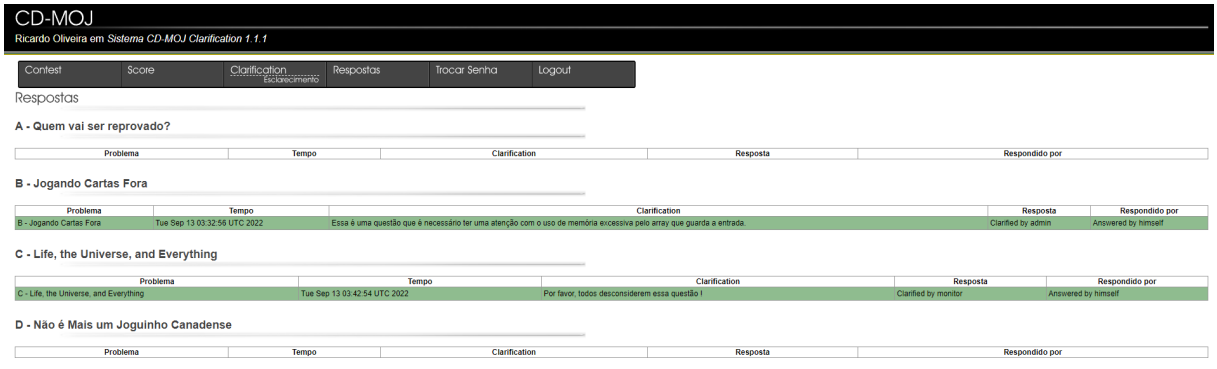

Figura 10 – Visão de um usuário comum sem dúvidas para a área de respostas

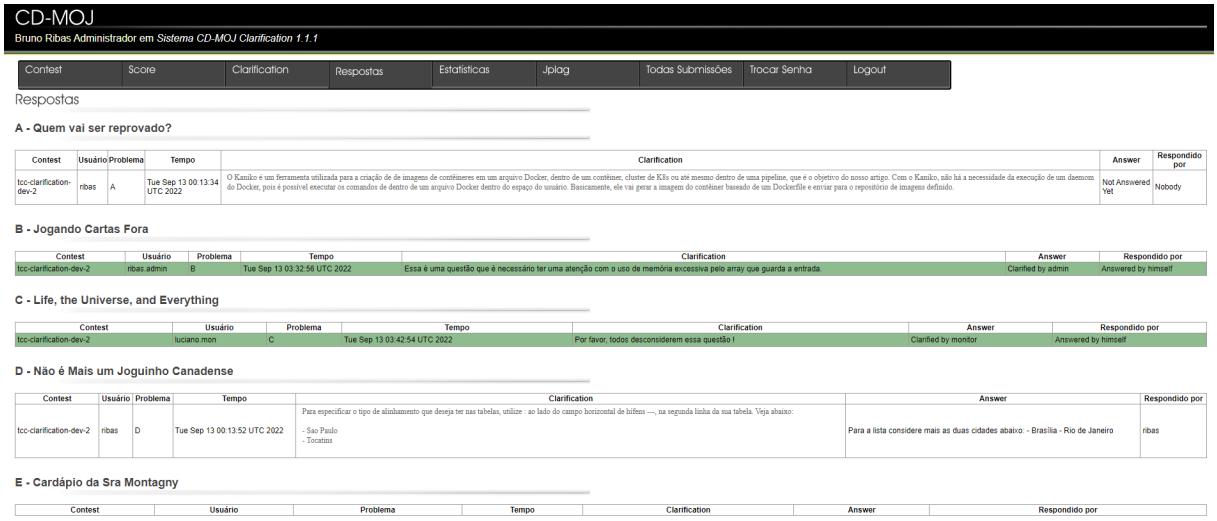

Figura 11 – Visão de um usuário administrador ou monitor para a área de respostas

Somente administradores e monitores podem responder e acessar a área de resposta, através de um link da própria pergunta do participante. A página acessada pelo link apresenta um formulário com uma caixa de texto preenchida com a dúvida, e outro em branco para a resposta, com a opção de disponibilizar a resposta de forma global para todos os participantes do *contest*, como demonstrado pela Figura 12.

Figura 12 – Visão de um usuário com privilégios de administrador para a área de *clarification*

O desenvolvimento dessa funcionalidade foi realizado com auxílio da imagem *docker* disponibilizada na Seção 4.4, juntamente com algumas obras da literatura para auxiliar no desenvolvimento com *shell script*. A ferramenta *ShellCheck*[2] foi utilizada para dar apoio e para que melhores práticas para o desenvolvimento sejam adotadas, pois se trata de uma ferramenta que fornece avisos e sugestões para scripts bash/sh.

4.2.1 Arquitetura da solução para área de dúvidas

A solução foi desenvolvida se baseando na comunicação dentro do sistema através de arquivos criados para armazenar as *clarifications* e as respostas e também disparar um comando para o módulo julgador.

[2] Se trata de uma ferramenta de análise estática para scripts de shell. Disponível em: https://www.shellcheck.net/

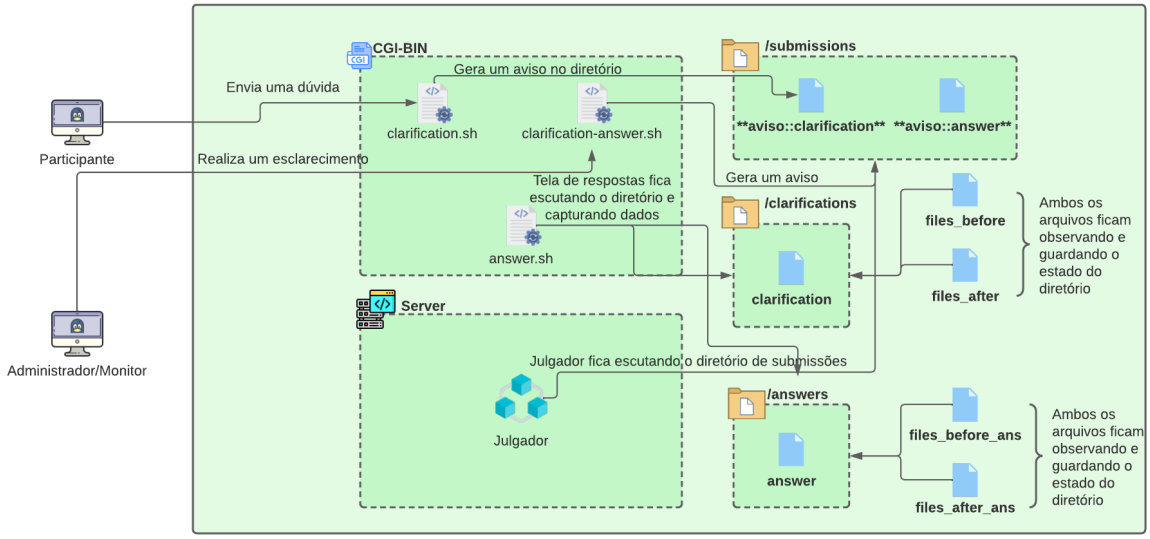

Figura 13 – Arquitetura da solução da área de dúvidas

Os arquivos gerados pelas submissões são armazenados dentro dos diretórios *answers* e *clarifications*, que por sua vez ficam armazenados dentro de outro diretório chamado *messages*. Dois arquivos chamados *files_after* e *files_before*, responsáveis por guardar o estado dos diretórios de *clarification*, são modificados sempre que há uma nova submissão. Tanto os diretórios quanto os arquivos são criados no momento em que o script cria-contest.sh é acionado para dar origem a um novo *contest*. Para monitorar as respostas dos administradores e monitores, também são criados dois arquivos *files_after_ans* e *files_before_ans* no momento em que os participantes realizam o primeiro login no sistema. Esses arquivos se localizam dentro da pasta $LOGIN.d de cada usuário.

4.3 Atualização do verificador de plágio para JPlag

Através da atualização do verificador de plágio, que foi a mudança do Sherlock para JPlag (JPLAG, 1996), foi possível disponibilizar o JPlag como uma ferramenta interna dentro do sistema CD-MOJ onde os administradores podem acessar por meio de uma aba JPlag que substituiu a antiga aba chamada Sherlock.

Figura 14 – Página de verificação de plágio de um *contest* que ainda não foi utilizado o JPlag

Inicialmente na página são demonstrados os enunciados dos problemas em branco. logo acima há um seletor que contém um range de linguagens disponíveis para que o JPlag faça análise. As linguagens disponibilizadas são C/C++, Csharp, Java, Python, Char, Text, Scheme e a opção *all* disponibilizada para fazer análise de todas as linguagens.

Figura 15 – Seletor com as linguagens disponíveis para análise

Após a análise realizada, são apresentadas tabelas para cada problema, que podem ser apresentadas separadamente por linguagem, e são compostas por três colunas que simbolizam a primeira e a segunda submissão e a porcentagem de plágio identificada. Cada identificador das tabelas possui um link para visão de modo geral dos problemas identificados daquela linguagem específica e um link para a página específica do caso na coluna de taxa de plágio. A porcentagem é um link que redireciona o administrador.

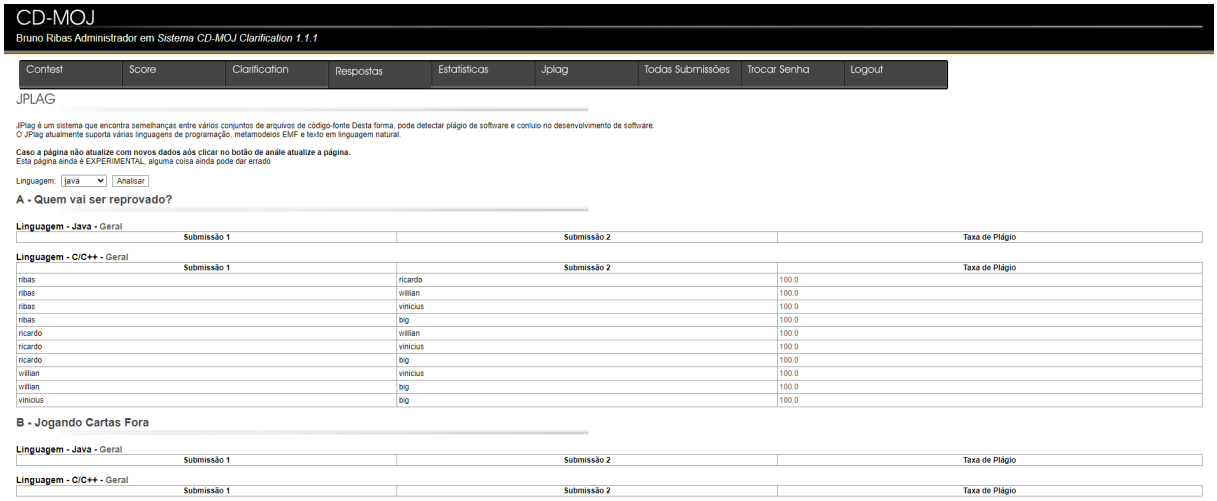

Figura 16 – Página de verificação de plágio de um contest após utilização do jplag

4.3.1 Arquitetura da solução para implantação do JPlag

A solução adotada se baseia nos padrões que são utilizados dentro do CD-MOJ, isso é, quando o usuário clica em analisar, é realizada um requisição *post* que é capturada e seus dados são processados. Logo após, é criado um arquivo para sinalizar ao módulo julgador que há um novo comando a ser executado. A partir do momento em que é identificado o comando, são iniciadas verificações para saber se já houve o *download* do JPlag, e então, a linguagem em que o usuário solicitou a análise é identificada. Caso seja para todas as linguagens, será feito um loop no *array* contendo todas e executando o *software* para cada uma delas, criando uma pasta separando cada uma.

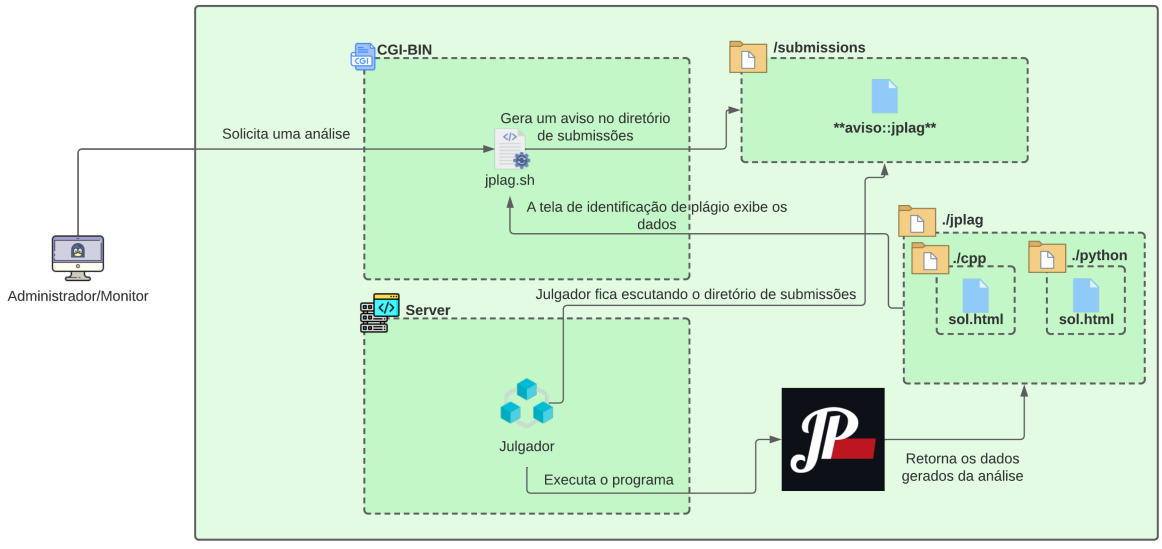

Figura 17 – Arquitetura da solução para implantação do JPlag

Após todos os passos descritos, a saída do programa é salva dentro da pasta JPlag. Dentro dela, foram criadas pastas específicas para cada linguagem. Cada pasta disponibiliza arquivos html/css para renderização das páginas de demonstração das comparações das soluções enviadas para o CD-MOJ.

4.4 Disponibilização de imagem docker para instalação de instância do CD-MOJ como ambiente de desenvolvimento

Para a criação da imagem *docker* para desenvolvimento, foi usada como base a imagem bitnami/minideb[3] baseada no sistema operacional Debian, onde são instaladas as dependências que são necessárias para executar o sistema do CD-MOJ. Após as instalações, é executado um script responsável por instalar e copiar as configurações para que o apache consiga executar o sistema. Dentro da imagem se localiza, também, um usuário administrador já configurado para que o desenvolvedor possa realizar o *upload* do *contest*.

A imagem é disponibilizada através de um repositório que fica dentro do repositório *Docker hub*, que armazena as imagens e possui um comportamento semelhante ao Github. Também é possível acessar o repositório onde estão contidos os arquivos usados para originar a imagem, possibilitando a contribuição futura de novos interessados pelo projeto. Pode ser acessado através da URL `https://hub.docker.com/r/lucianosz7/cdmoj`, e para contribuições, utiliza-se a URL `https://github.com/cd-moj/cd-moj.docker` e cria-se um *fork*.[4] Logo após todas as melhorias estiverem prontas, se realiza um *pull request* para a origem.

Para fazer um *docker pull*[5] da imagem, basta executar o comando **`docker pull lucianosz7/cdmoj:1.2.0`** e logo após executar a imagem através do comando **`docker run -it -p 80:80 lucianosz7/cdmoj:1.2.0 bash`**. Após executar a imagem, o usuário estará dentro do *container* do sistema e poderá executar o sistema CD-MOJ. É necessário executar os *daemons* para que o sistema possa ser executado plenamente, através dos comandos:

```
$ bash moj-serverside/daemons/executar-corretor.sh
$ bash moj-serverside/daemons/executar-julgador.sh
```

Esses são os passos básicos para execução da instância do CD-MOJ. Há outros passos, que são basicamente dicas para melhorar o desenvolvimento. Não é necessário construir um novo *container* a cada execução da imagem, sendo possível executar o *con-*

[3] Disponível em: https://hub.docker.com/r/bitnami/minideb

[4] *Fork* é uma cópia de um repositório. Disponível em: https://docs.github.com/pt/get-started/quickstart/fork-a-repo

[5] ***`docker pull`*** é o comando responsável por baixar a imagem ou repositório do Docker hub. Disponível em: https://docs.docker.com/engine/reference/commandline/pull/

tainer que foi utilizado anteriormente e continuar o trabalho. Para visualizar os *containers* já criados, basta executar o comando **`docker ps -a`**, logo após a identificação daquele que estava em uso. Em seguida, utilizar os seguintes comandos **`docker start CONTAINER_ID`** para iniciar o *container*, e **`docker attach CONTAINER_ID`** para anexar os fluxos de entrada, saída e erro padrão local a um *container* em execução. Caso o usuário queira executar a instância em mais de uma janela, é preciso executar o seguinte comando **`docker exec -it CONTAINER_ID bash`**.

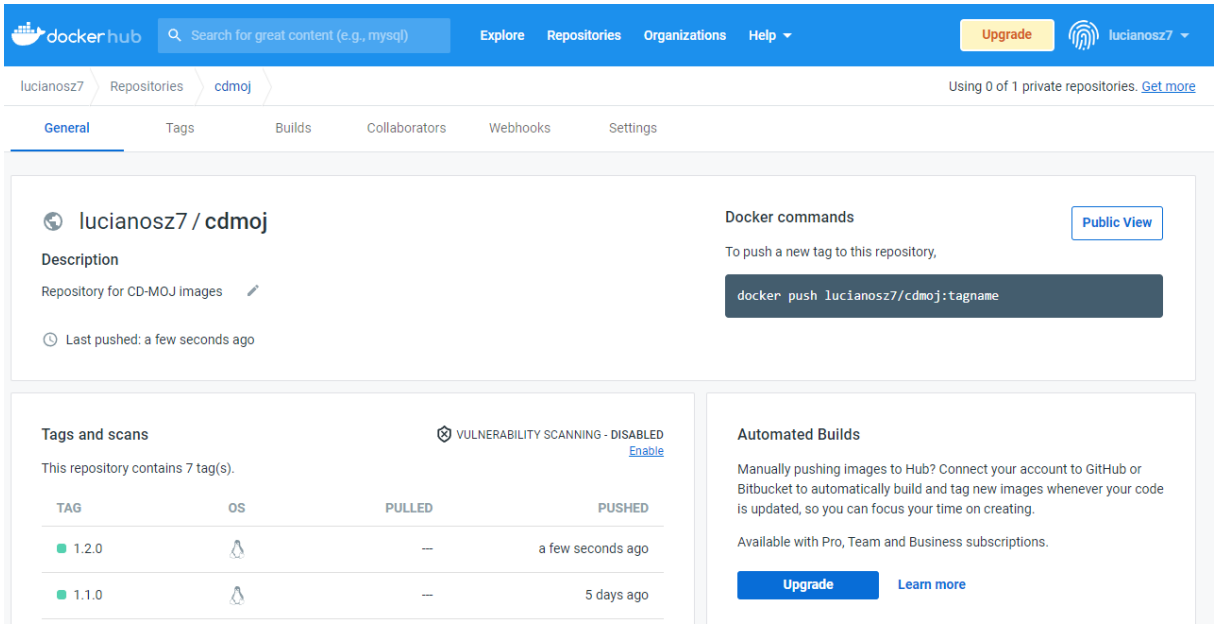

Figura 18 – Repositório no docker hub do CD-MOJ

4.5 Estrutura atual do CD-MOJ

Após as contribuições feitas através deste trabalho, foram adicionados novos recursos e também foram modificados alguns comportamentos dentro do sistema, sendo assim, é necessário atualizar as tabelas com os novos recursos e os diagramas de componentes e sequência do julgador.

Com o acréscimo dos *scripts* referentes à área de esclarecimentos e à substituição do *script* responsável pelo verificador de plágio, foi necessário reescrever o diagrama de componentes referente ao CGI. Os *scripts* que foram adicionados são: clarification.sh, clarification-answer.sh, answer.sh e jplag.sh. Segue o diagrama referente ao CGI representado pela Figura 19.

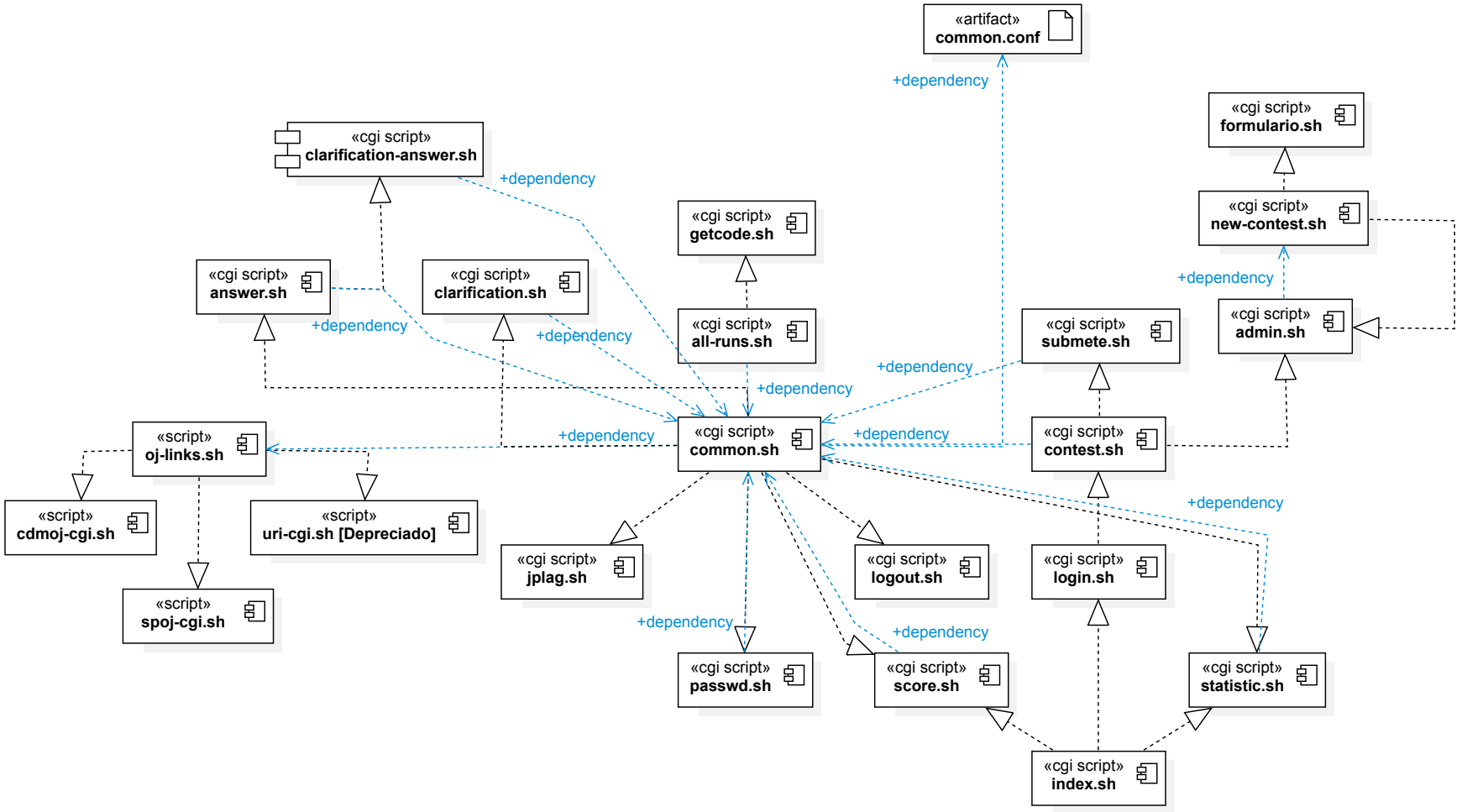

Figura 19 – Diagrama de componentes do CGI

Diferente do diagrama mostrado pela Figura 4, na Seção 3.1, o diagrama de componente foi divido em duas partes para uma melhor visualização. No lado do servidor, não houve adições de novos *scripts*, portanto o comportamento segue sendo o mesmo.

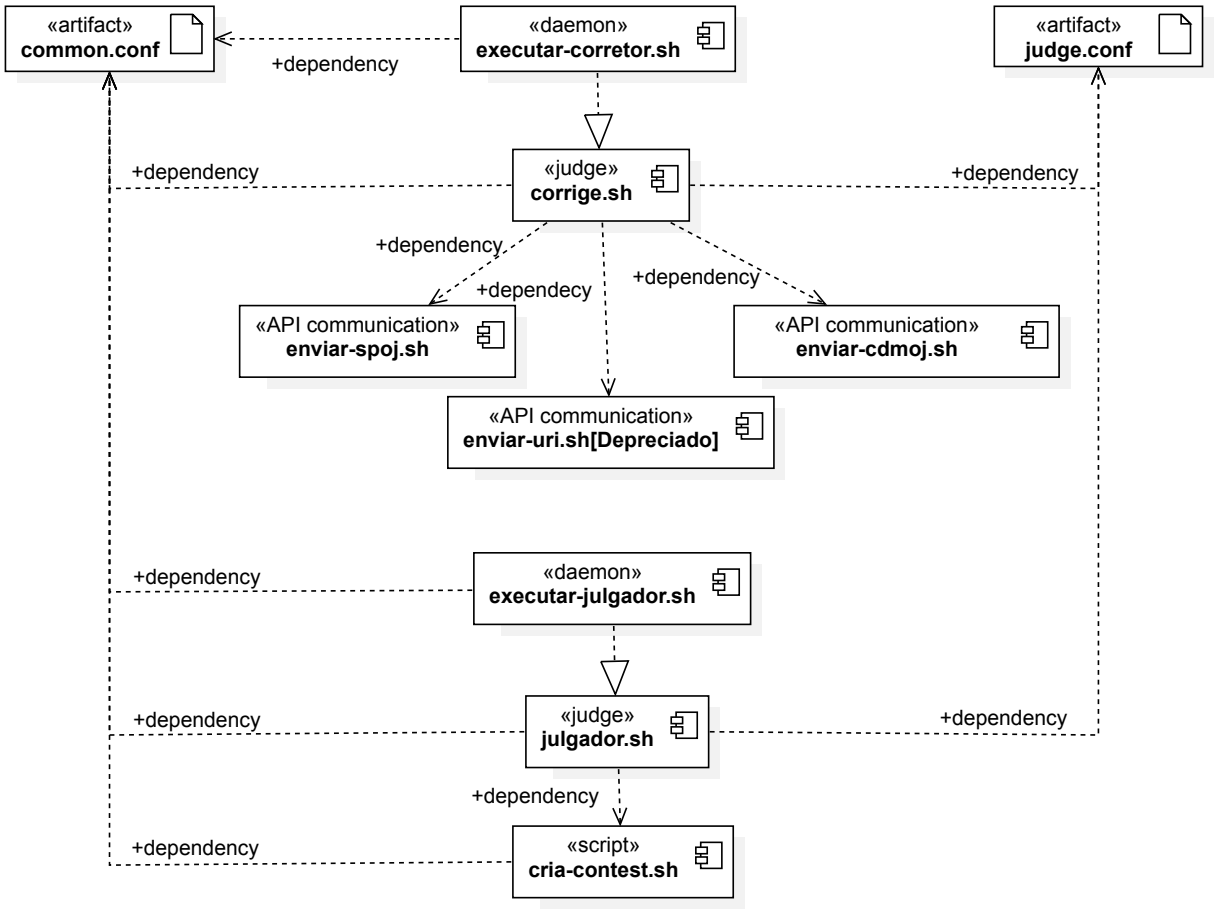

Figura 20 – Diagrama de componentes do servidor

Com as mudanças citadas, também houve alterações na Tabela 6, que descreve os *scripts* do diretório CGI-BIN responsáveis pelas operações no *client-side*. As atualizações da tabela incluem a adição dos quatro novos *scripts*.

Tabela 7 – Descrição dos *scripts* do diretório cgi-bin

Componente	**Descrição**
`admin.sh`	Página de administrador do CD-MOJ
`all-runs.sh`	Apresenta todas as submissões para o administrador do *contest*
`answer.sh`	Página de respostas, responsável por processar as *clarifications* e respostas
`clarification.sh`	Página de esclarecimento ou *clarification*, responsável por disparar a ação e processar a requisição para criação de uma nova *clarification*
`clarification-answer.sh`	Página de resposta, responsável por disparar a ação para criação de uma nova resposta
`common.sh`	*Script* responsável por todas as ações em comum do fluxo do CD-MOJ
`contest.sh`	Página principal dos *contests*, responsável por redirecionar para o login e mostrar os problemas do contest em execução
`formulario.sh`	Formulário para criação de novos *contests*
`getcode.sh`	*Script* responsável por coletar todas as respostas dos problemas e disponibilizá-las para visualização
`index.sh`	Home Page
`jplag.sh`	*Script* verificador de plágio responsável por disparar uma ação para analisar as soluções de códigos fontes submetidas.
`login.sh`	*Script* responsável por manipular o login e suas operações
`logout.sh`	*Script* responsável por manipular o logout e suas operações
`new-contest.sh`	Pagina de criação de novos *contests*
`passwd.sh`	*Script* responsável por alterar a senha
`score.sh`	*Script* responsável por computar a pontuação do *contest*
`sherlock.sh` [**Obsoleto**]	Módulo responsável pela detecção de plágio
`statistic.sh`	*Script* responsável por calcular as estatísticas do *contest* em execução
`submete.sh`	*Script* responsável por executar as submissões

Também houve mudanças no comportamento do módulo julgador, com o acréscimo de dois novos comandos. Um dos comandos é o ***answer***, responsável por identificar uma resposta e se ela é global ou não, e após esse processamento, a mesma pode ser mostrada para todos os usuários ou para somente um específico. Já o outro comando, ***jplag***, é responsável por disparar a ação para ser feita a análise de plágio das submissões, com a conclusão desses dados ficando disponíveis para o administrador visualizar. Esses comportamentos citados podem ser visualizados através do diagrama de sequência representado pela Figura 21.

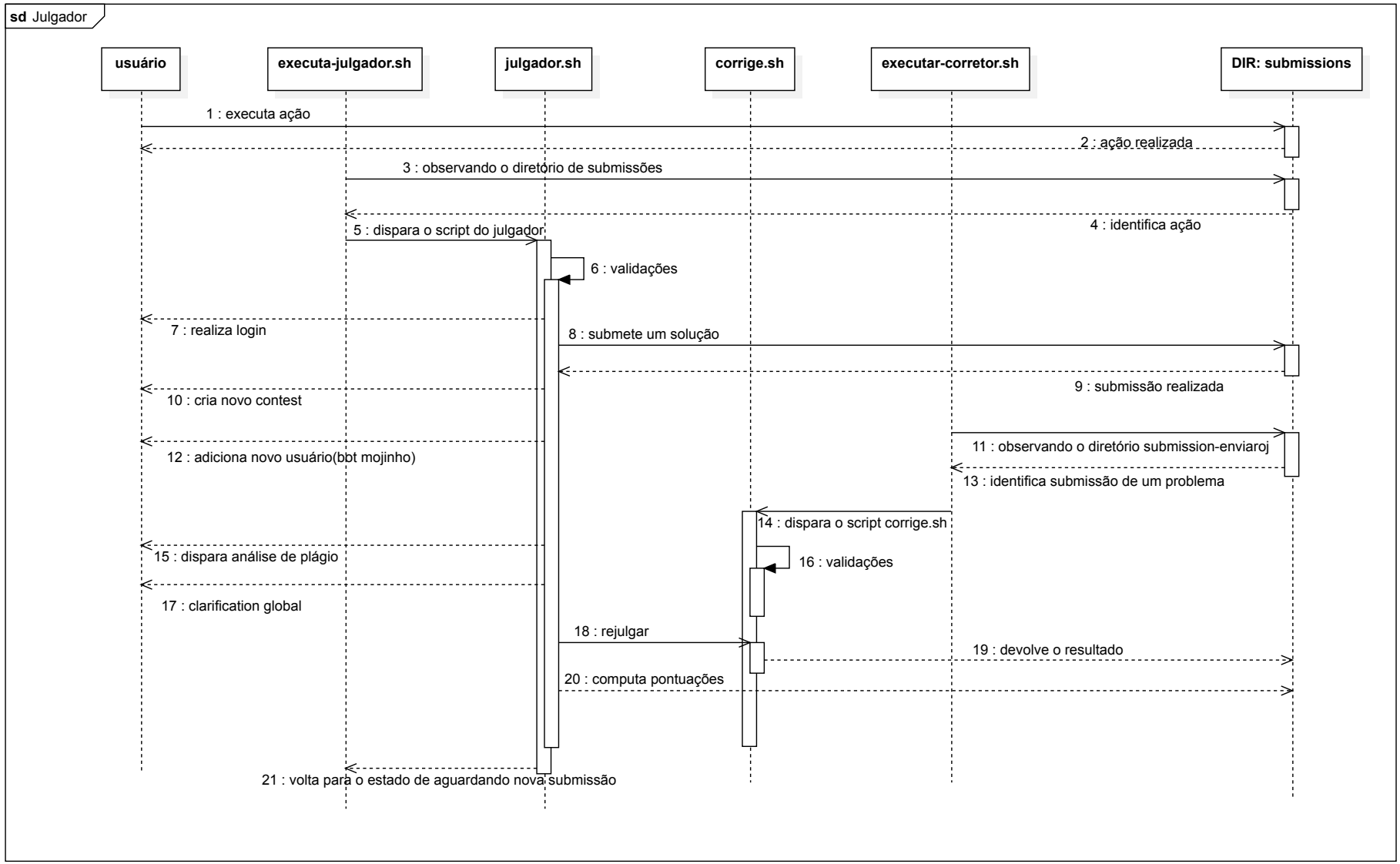

Figura 21 – Diagrama de componentes do CGI

5 Conclusão

O objetivo geral desse trabalho foi contribuir com melhorias para o sistema do CD-MOJ, utilizado em larga escala para aplicação de listas de atividades como reforço do aprendizado das disciplinas de programação. Entendendo algumas das necessidades apontados pelo desenvolvedor e mantenedor do CD-MOJ, o Prof. Dr. Bruno César Ribas, foi possível elencar prioridades para quais melhorias seriam melhores recebidas no atual momento do sistema.

A documentação desenvolvida com o objetivo de servir como um guia para possíveis futuras contribuições é um meio de facilitar a entrada de novos contribuidores para o projeto. Os conhecimentos adquiridos durante o desenvolvimento desse trabalho foram escritos e disponibilizados com expectativas de que possam ser usados futuramente e possam facilitar o entendimento sobre o sistema.

Ainda pensando em aumentar mais ainda o nível de contribuintes para o projeto, uma imagem *docker* que disponibiliza uma instância do CD-MOJ pronta para o desenvolvimento de novas funcionalidades e a manutenção do sistema foi desenvolvida com o objetivo de auxiliar novos desenvolvedores que contribuirão futuramente com o projeto.

A implementação do *clarification* foi pensada de modo que sejam feito esclarecimentos mais rápidos e fáceis e que possa ser levada tanto para os participantes quanto para os administradores e monitores de uma lista de exercícios, prova ou até mesmo uma competição.

Identificar plágio é uma maneira de manter um sistema de pontuação justo, para que outros não apresentem soluções de terceiros como suas próprias soluções, e para que os usuários dessa prática sejam disciplinados. A implantação da ferramenta JPlag contribui para que a identificação dessa prática seja feita de forma rápida e eficaz dentro do sistema do próprio CD-MOJ.

5.1 Trabalhos futuros

Há algumas ideias que podem ser adotadas para trabalhos futuros dentro do sistema do CD-MOJ. A segurança de um sistema é um esforço contínuo que deve estar sempre sendo reforçado e para a continuação desse esforço é importante restringir o acesso a dados sensíveis para reforçar a segurança do sistema. A outra, é que a disponibilização das funcionalidades do CD-MOJ como serviços web possam ser feitas através de requisições por serviços de terceiros e que consigam capturar informações a partir dessas ações.

Referências

BEECROWD. 2022. <https://www.beecrowd.com.br/judge/en/login>. (Accessed on 04/21/2022). Citado 2 vezes nas páginas 11 e 12.

BOCA Online Contest Administrator. 2010. <https://www.ime.usp.br/~cassio/boca/>. (Accessed on 04/21/2022). Citado na página 11.

CAMPOS, F. 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 11.

CD-MOJ - Contest Driven Meta Online Judge. 2013. <https://moj.naquadah.com.br/about.shtml>. (Accessed on 04/22/2022). Citado 2 vezes nas páginas 5 e 15.

CHAVES, J. et al. Uma ferramenta baseada em juízes online para apoio às atividades de programação de computadores no moodle. *RENOTE*, v. 11, 12 2013. Citado 2 vezes nas páginas 11 e 12.

CODEBENCH. 2022. <https://codebench.icomp.ufam.edu.br/>. (Accessed on 04/21/2022). Citado 2 vezes nas páginas 11 e 13.

DMOJ: Modern Online Judge. 2022. <https://dmoj.ca/>. (Accessed on 04/21/2022). Citado 2 vezes nas páginas 11 e 13.

GADELHA, L. G. e David Fernandes e B. Juiz online como ferramenta de apoio a uma metodologia de ensino híbrido em programação. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, v. 27, n. 1, p. 140, 2016. ISSN 2316-6533. Disponível em: <https://www.br-ie.org/pub/index.php/sbie/article/view/6694>. Citado na página 13.

JPLAG. 1996. <https://github.com/jplag/JPlag/wiki>. (Accessed on 04/22/2022). Citado na página 30.

MOODLE - Open-source learning platform | Moodle.org. s.d. <https://moodle.org/?lang=pt_br>. (Accessed on 04/22/2022). Citado na página 12.

PEREIRA, T. G. Utilização de juízes eletrônicos e problemas oriundos da maratona de programação no ensino de programação da faculdade unb gama: Um estudo de caso. In: . [s.n.], 2015. Disponível em: <https://bdm.unb.br/handle/10483/11315>. Citado na página 11.

PORTO, F. de Engenharia da Universidade do. *CGI - Common Gateway Interface*. 2022. Disponível em: <https://web.fe.up.pt/~goii2000/M9/cgi.htm>. Citado 2 vezes nas páginas 5 e 17.

SBC Maratona de Programação. 1996. <http://maratona.sbc.org.br/sobre22.html>. (Accessed on 09/16/2022). Citado na página 9.

SKIENA, S. S.; REVILLA, M. *Programming Challenges: The Programming Contest Training Manual*. Berlin, Heidelberg: Springer-Verlag, 2003. ISBN 0387001638. Citado na página 11.

SPHERE Online Judge (SPOJ). 2022. <https://www.spoj.com/>. (Accessed on 04/20/2022). Citado 2 vezes nas páginas 11 e 12.

SPOJ Brasil (SPOJ). 2022. <https://br.spoj.com>. (Accessed on 04/21/2022). Citado na página 13.

VENKITACHALAM, G.; CHIUEH, T. cker. High performance common gateway interface invocation. In: *Proceedings 1999 IEEE Workshop on Internet Applications (Cat. No.PR00197)*. [S.l.: s.n.], 1999. p. 4–11. Citado na página 17.

WASIK, S. et al. A survey on online judge systems and their applications. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 51, n. 1, jan 2018. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3143560>. Citado 2 vezes nas páginas 9 e 11.