Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

CD-MOJ: Implementação do módulo de treinamento livre

Caio Martins Ferreira

TRABALHO DE CONCLUSÃO DE CURSO

BACHARELATO ENGENHARIA DE SOFTWARE

Brasília

2024

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA

CD-MOJ: Implementação do módulo de treinamento livre

Caio Martins Ferreira

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.

Orientador: Prof. Dr. Bruno César Ribas

Brasília
2024

FICHA CATALOGRÁFICA

Martins Ferreira, Caio.
CD-MOJ: Implementação do módulo de treinamento livre / Caio Martins Ferreira; orientador Bruno César Ribas. -- Brasília, 2024.
60 p.

Trabalho de Conclusão de Curso (Bacharelato Engenharia de Software) -- Universidade de Brasília - UnB, 2024.

1. CD-MOJ. 2. Online Judge. 3. Open source. 4. Engenharia de Software. 5. Competições de Programação.I. Ribas, Bruno César, orient. II. Título.

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA

CD-MOJ: Implementação do módulo de treinamento livre

Caio Martins Ferreira

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.

Trabalho aprovado. Brasília, 26 de junho de 2024:

Prof. Dr. Bruno César Ribas
Orientador

Prof. Dr. John Lenon Cardoso Gardenghi
Examinador interno

Prof. Dr. Daniel Saad Nogueira Nunes
Examinador externo

Resumo

Este trabalho tem como foco propostas de melhorias para o sistema do CD-MOJ, um juiz online utilizado em disciplinas do ensino superior e maratonas de programação, focando em melhorias de *back-end*, *front-end* e documentação, com ênfase no módulo de treinamento livre, um espaço para que os usuários tenham acesso as questões independente da participação em *contests* . A implementação desse módulo traz benefícios significativos para estudantes e professores, como visualização clara das questões, navegação facilitada, acesso rápido a questões categorizadas por *tags*, acompanhamento de submissões e desempenho, e atualização automática de questões. Professores podem monitorar o desempenho dos alunos individuais para melhor planejar atividades conforme necessidades identificadas, etiquetar questões por conceitos abordados e bloquear questões específicas para manter a segurança.

Palavras-chave: CD-MOJ. Online Judge. Open source. Engenharia de Software. Competições de Programação.

Abstract

This paper presents proposals for enhancing the CD-MOJ system, an online judge used in higher education disciplines and programming marathons, focusing on backend, frontend, and documentation enhancements, with emphasis on the free training module, a space where users have access to questions regardless of participation in contests. The implementation of this module brings significant benefits to students and teachers, such as clear question visualization, easy navigation, quick access to questions categorized by tags, submission and performance tracking, and automatic question updates. Teachers can monitor individual student performance to better plan activities based on identified needs, label questions by covered concepts, and block specific questions to maintain security.

Keywords: CD-MOJ. Online Judge. Open source. Software Engineering. Programming Competitions.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

BOCA	BOCA Online Contest Administrator
CD-MOJ	Contest-Driven Meta Online Judge
CGI	Common Gateway Interface
GOMS	Goals, Operators, Methods and Selections rules
GUI	Graphical User Interfaces
HTML	HyperText Markup Language
IHC	Interação Humano-Computador
ISO	International Organization for Standardization
MIT	Massachusetts Institute of Technology
SPOJ	Sphere Online Judge
TCC	Trabalho de conclusão de curso
UI	User Interface
UnB	Universidade de Brasília
UTFPR	Universidade Tecnológica Federal do Paraná
UX	User Experience
WIMP	Windows, Icons, Menus e Pointers

Sumário

1 Introdução

As competições de programação têm conquistado um espaço crescente tanto no ambiente acadêmico quanto no mercado de trabalho. Elas desempenham um papel crucial no desenvolvimento de habilidades fundamentais para programadores, desafiando-os a compreender problemas, identificar requisitos e restrições, aplicar análise e raciocínio lógico, planejar estratégias eficazes e realizar testes rigorosos em suas soluções, considerando aspectos de tempo e uso adequado dos recursos disponíveis. Dessa forma, as competições de programação desempenham um papel importante no aprendizado e na preparação dos estudantes (Pereira, 2015), os colocando de frente com um fluxo simplificado de entender, resolver e testar.

Para isso, são utilizados os juízes online, sistemas que disponibilizam exercícios de programação e permitem a construção de códigos-fonte, que são submetidos e corrigidos automaticamente utilizando casos de teste pré-cadastrados. Normalmente são utilizados como ferramentas de apoio pedagógico, pois reduzem a carga de trabalho para correção de exercícios e oferecem *feedback* rápido aos aprendizes.

É importante ressaltar que muitos professores utilizam sistemas de correção de algoritmos semelhantes aos utilizados em maratonas de programação para auxiliar no ensino de programação e algoritmos. A utilização de juízes online tem se mostrado fundamental para verificar as soluções apresentadas por alunos, pois com eles é possível acelerar o processo de avaliação, o que dinamiza e torna mais rápido, resultando em um melhor progresso no desenvolvimento de habilidades de programação.

Do ponto de vista dos alunos, é possível proporcionar uma maneira eficiente e automatizada de verificar suas submissões. Esses sistemas oferecem um ambiente onde os usuários podem enviar seus códigos, receber *feedback* imediato e aprimorar suas habilidades de programação de forma prática e objetiva.

Nesse contexto, destaca-se o *Contest-Driven Meta Online Judge* (CD-MOJ), desenvolvido pelo professor Bruno Ribas, é um exemplar notável. Este ambiente de código aberto processa um grande volume de submissões e é usado para apoiar disciplinas de programação que exigem muitas horas de prática e avaliações rápidas das soluções dos alunos. O CD-MOJ oferece suporte aos professores na criação, adaptação e reutilização de problemas de programação, permitindo uma personalização abrangente para atender às necessidades específicas das disciplinas. Utilizando uma abordagem baseada em casos de teste, o CD-MOJ julga uma submissão como correta e otimizada quando atende a todas as soluções requeridas pelo juiz online, sendo elas os resultados esperados do algoritmo e o consumo de recursos computacionais. Em suma, suas funcionalidades são: envio e correção de exercícios, criação de *contests*, pontuação em tempo real e quadro de dúvidas.

O objetivo deste trabalho é promover e facilitar a utilização, configuração, manutenção e compreensão do CD-MOJ, por meio da criação de uma documentação clara e precisa destinada a usuários, desenvolvedores e administradores. A organização do fluxo de informação é estruturada em perfis de usuário, adaptados às necessidades de cada público-alvo. Para garantir a fácil manutenção e *deploy* da documentação, foram utilizados *markdown e MkDocs* (Christie, 2014), com publicação via *GitHub Pages*. Essas contribuições visam organizar a documentação do CD-MOJ de maneira eficiente, melhorar a usabilidade do sistema e facilitar o processo de ensino e aprendizado.

A decisão entre o uso de *Makefile e Script Bash* foi ponderada, com preferência pelo *Makefile*. O projeto necessita de dependências complexas e regras claras de construção, o *Makefile* (IBM, 2021) permite a definição de variáveis, instalação de pacotes, alteração de *tokens* nos arquivos de instalação de cliente e servidor, além da configuração do Apache.

O módulo de treinamento foi implementado com o objetivo de incentivar a resolução independente de problemas do banco de questões do CD-MOJ, trazendo benefícios tanto para estudantes quanto para professores. Os principais benefícios incluem a visualização clara e navegação facilitada pelas questões, a categorização de questões por *tags* permitindo que os usuários foquem em áreas específicas, e o monitoramento de submissões e desempenho para a avaliação dos estudantes. Além disso, houve uma simplificação do processo de atualização do banco de questões.

A refatoração da documentação e a implementação do módulo de treinamento foram realizadas seguindo os princípios da Interação Humano-Computador (Oliveira, 2015) . Diretrizes de contribuição foram estabelecidas, incluindo a elaboração de *templates* para padronização de *commits e issues*. A adoção de práticas da engenharia de software visa melhorar a qualidade e a eficiência do desenvolvimento, bem como promover uma experiência de usuário mais intuitiva e satisfatória.

Neste trabalho, exploraremos vários de conceitos e tecnologias importantes para a compreensão e aprimoramento do CD-MOJ. Para isso, no Capítulo 2, apresentaremos as definições essenciais que fundamentam a problemática e as propostas abordadas. O Capítulo 3 trará uma revisão de trabalhos relacionados, oferecendo uma contextualização sobre a temática discutida. No Capítulo 4, são detalhadas as contribuições feitas ao CD-MOJ, incluindo a documentação de seus recursos, funcionalidades e práticas para usuários, desenvolvedores e administradores, além da implementação de um módulo de treinamento. Por fim, o Capítulo 5 apresentará a conclusão, destacando os resultados das contribuições realizadas e sugerindo direções para trabalhos futuros.

2 Referencial teórico

Para um melhor entendimento deste trabalho, esta seção tratará das definições de conceitos importantes que compõem a problemática e proposta abordadas.

2.1 Juízes Online

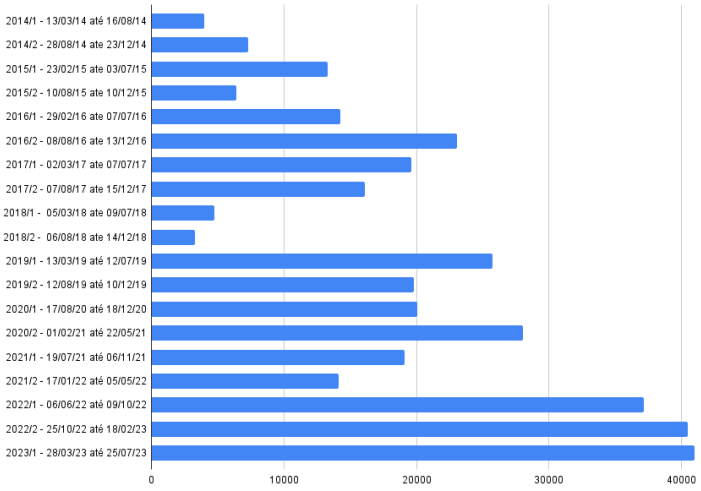
Os juízes online são projetados para testar algoritmos criados para solucionar problemas definidos por um conjunto completo de entradas e suas respectivas saídas esperadas (Pereira, 2015). Esses sistemas automáticos desempenham um papel crucial na avaliação de códigos-fonte, abrangendo etapas essenciais como compilação, execução e verificação dos resultados obtidos (Wasik Maciej Antczak, 2018). Assim os juízes verificam os códigos ao submete-los em conjuntos de testes predefinidos. Além disso, também são responsáveis por garantir que as soluções estejam dentro dos limites de recursos estabelecidos, como tempo de execução e uso de memória. Existem diversas opções e cada uma delas possui funcionalidades específicas. Alguns exemplos de plataformas são:

2.1.1 CD-MOJ

O CD-MOJ (Ribas, 2013) é um juiz online direcionado a competições de programação. Ele atua tanto como um *online judge* ao compilar, executar e verificar soluções, quanto como um *meta online judge* ao despachar códigos para outros juízes online. A principal motivação para a implementação deste sistema foi facilitar o treinamento das equipes para a Maratona de Programação da UTFPR - Campus Pato Branco, permitindo o uso de problemas do SPOJ-BR e do URI-OJ (atualmente Beecrowd), que possuem enunciados em português.

Houve um aumento substancial na utilização do CD-MOJ desde sua implementação na UnB campus Gama. Foram realizadas mais de 34 mil submissões durante o semestre de 2023/1 em mais de 60 *contests* com exercícios de programação. A Figura 2.1 demonstra o aumento de uso de acordo com os semestres letivos da UTFPR e UNB, possuindo o seu ápice no semestre 2023/1.

Figura 2.1 – Gráfico de submissões no CD-MOJ por semestre



A implementação, administração e cuidados são feitos pelo professor Bruno Ribas e por estudantes em projetos de extensão ou trabalhos de conclusão de curso. O sistema é dividido em:

Tabela 2.1 – Partes do CD-MOJ

Contest	Visualizar as questões do *contest* com seu enunciado e realizar o envio da tentativa de resposta.
Formulario de *contest*	Área para os administradores criarem novos *contests*.
Score	Visualizar o placar atualizado da competição contendo todos os participantes.
Jplag	Ferramenta para detecção de plágio.
Clarification	Área para que os usuários possam enviar suas dúvidas ou para que os administradores possam fazer esclarecimentos.
Login	Iniciar a sessão.
Logout	Encerrar a sessão.
Trocar senha	Trocar a senha de login.

Fonte: (Ribas, 2013)

2.1.2 BOCA

O BOCA (Campos; Ferreira, 2004) (BOCA Online Contest Administrator) é um sistema de apoio a competições de programação e também é usado no apoio de disciplinas que recorrem à submissão e correção de trabalhos durante as aulas no Instituto de Informática (INF) autônomo da UFG (Universidade Federal de Goiás);

É responsável por gerenciar todas as etapas de uma competição, exigindo uma equipe para desempenhar funções necessárias específicas para cada competição. As submissões são avaliadas pelo juiz online, que analisa e corrige os programas submetidos, definindo as respostas que os participantes receberão. O sistema é dividido em:

Tabela 2.2 – Partes do BOCA

Problems	Visualizar as questões da prova com seu enunciado.
Runs	Submeter as suas soluções e receber as respostas dos juízes.
Score	Visualizar o placar atualizado da competição contendo todos os participantes.
Clarifications	Realização de perguntas aos juízes acerca de um problema da prova, mostrando essa pergunta para todos que acessarem.
Tasks	Permite enviar arquivos para impressão e clicar no botão S.O.S. que pode ser usado para chamar a ajuda da equipe de staff.
Options	Mostra as informações: Username, User full name e User description do time. Também permite alterar a senha da equipe.

Fonte: (Campos; Ferreira, 2004)

2.1.3 SPOJ

SPOJ (SPOJ, 2013) é uma plataforma que contém um juiz online para avaliação automática de códigos fonte enviados pelos usuários. Ela dispõe de um repositório com mais de 13.000 problemas para prática em diversos idiomas. A plataforma suporta mais de 45 linguagens de programação e oferece a possibilidade de contribuir com novos problemas para o repositório. Alguns de seus diferenciais são:

Tabela 2.3 – Diferenciais SPOJ

Sistema flexível	Permite uma interação com as respostas submetidas e uma saída condizente com o exercício.
Contribuição aberta	Permite contribuir com novos problemas para o repositório.
Gerenciamento de conteúdos	Permite que os usuários criem contests utilizando os problemas existentes no sistema.
Diversidade de linguagens de programação	Conta com mais de 45 linguagens de programação.

Fonte: (SPOJ, 2013)

2.1.4 Beecrowd

O Beecrowd (beecrowd, 2023) é uma plataforma composta por um repositório com diversos problemas de programação, separados por tópicos e dificuldade. A plataforma possui funcionalidades como uma área para maratonas, acadêmicos, instrutores, rankings e um fórum para discussão de problemas;

2.2 Ferramentas

2.2.1 Bash

Durante o uso do computador, os usuários interagem indiretamente com um "*shell*", um componente presente em sistemas operacionais como *GNU/Linux, iOS, Microsoft Windows* e outros. O *shell* desempenha a função de receber comandos digitados pelos usuários, traduzi-los em uma forma compreensível para o núcleo do computador e executar as tarefas correspondentes. No caso específico do CD-MOJ, ele é desenvolvido utilizando a linguagem de programação interpretada Bash, que foi criada e disponibilizada pela *Free Software Foundation* em 1989 (Stallman, 2000). O Bash é o interpretador padrão usado nos sistemas GNU/Linux e representa um tipo comum de *shell* textual, que oferece uma linguagem de comunicação para interagir com o sistema operacional.

2.2.2 AWK

O awk surgiu em 1977 como uma linguagem de programação simples voltada para a manipulação de texto e números de forma igualmente eficiente. Criado com o propósito de ser uma linguagem de *script* que se integra com as ferramentas Unix, ele segue a filosofia de executar cada programa de forma especializada. Com apenas algumas linhas de código, o awk permite lidar com cálculos simples e processamento de dados de forma eficaz. Seu programa é uma sequência de padrões e ações que indicam o que procurar nos dados de entrada e como agir quando eles são encontrados. Ao examinar conjuntos de arquivos de texto, o awk busca por linhas que correspondam a qualquer um dos padrões definidos. Quando uma correspondência é encontrada, a ação associada é executada. Os padrões podem selecionar linhas com base em expressões regulares, operações de comparação em strings, números, campos, variáveis e elementos de *array*. As ações, por sua vez, podem realizar uma variedade de processamentos nas linhas selecionadas, oferecendo uma flexibilidade significativa e podem realizar processamento arbitrário em linhas selecionadas; (Aho Brian W. Kernighan, 1988)

2.2.3 Cron

Em sistemas UNIX ou Linux, o processo do *daemon* cron é executado continuamente. Um *daemon*, geralmente, é um programa que executa serviços em segundo plano, e o cron serve como um agendador de tarefas. Um *script* de controle de execução é responsável por iniciar o *daemon* cron durante a inicialização do sistema. O *daemon* cron carrega as entradas da tabela cron do sistema ou do usuário individual na memória.

Uma entrada *crontab* é uma linha formatada em um arquivo *crontab* que especifica o minuto, hora, dia, dia da semana e/ou dia do mês em que uma determinada tarefa deve ser executada. O *daemon* cron verifica, com base em sua programação predefinida, se alguma das entradas na tabela cron deve ser executada.

Cada usuário em um sistema pode ter seu próprio arquivo *crontab* para definir suas tarefas específicas. No entanto, um administrador do sistema pode restringir as capacidades dos usuários individuais de acordo com as políticas de segurança estabelecidas.

É importante observar que um *script* executado pelo cron não opera no mesmo ambiente de *shell* que um comando digitado diretamente no prompt. Quando um usuário faz login em uma máquina e interage com o *prompt* do *shell*, diversas variáveis precisam ser definidas para configurar a sessão interativa do *shell*. No entanto, o cron não é executado a partir de uma sessão interativa.

2.2.4 Makefile

Um arquivo *Makefile* (IBM, 2021) é um *script* que contém um conjunto de regras e comandos utilizados pelo programa *make* para automatizar o processo de compilação e construção de projetos. O *make* é uma ferramenta de construção que gerencia as dependências entre arquivos fonte e alvos, permitindo compilar apenas os arquivos que foram modificados desde a última compilação.

Um *Makefile* consiste em regras, cada uma delas com um alvo e uma lista de dependências. Ao se executar o comando *make* no diretório que contém o *Makefile*, o *make* procura a primeira regra e executa os comandos associados para construir o alvo especificado.

2.2.5 JPlag

JPlag (JPlag, 1996) é uma ferramenta de detecção de similaridade de código-fonte. Ela foi desenvolvida para identificar casos de plágio em programas de computador, comparando o código-fonte de diferentes projetos e determinando o grau de similaridade entre eles. Ele é capaz de analisar códigos escritos em várias linguagens de programação, incluindo Java, C, C++, C#, Go, Kotlin, Python, R, Rust, Scala, Scheme, Swift, EMF.

Essa ferramenta é amplamente utilizada na garantia da originalidade e integridade do trabalho acadêmico, bem como na verificação da autenticidade de projetos de código

aberto e do código desenvolvido por equipes de programadores. Instituições de ensino e empresas em todo o mundo utilizam o JPlag como uma solução eficaz para verificar a autoria e detectar possíveis casos de plágio.

2.2.6 MKDocs

A ferramenta MkDocs (Christie, 2014) é um gerador de sites HTML estáticos voltado para a construção de documentação de projetos. Os arquivos de origem da documentação são gravados em *Markdown* e configurados com um único arquivo de configuração YAML. Permite adicionar formatação simples, como títulos, listas, links e código embutido.

Além disso, o MkDocs tem ótima capacidade de personalização. Ele oferece a possibilidade de construção de temas que podem ser facilmente personalizados para se adequar a identidade visual do projeto. Ele também suporta a integração de *plugins*, o que permite estender ainda mais suas funcionalidades e adaptá-lo às necessidades específicas do projeto.

Durante o desenvolvimento é possível aos desenvolvedores visualizar as alterações feitas nos arquivos de origem. Isso facilita o processo de revisão e iteração, garantindo que a documentação esteja sempre atualizada e precisa. A página no GitHub (Github, 2008) possui uma documentação que explica tudo sobre como utilizar a partir de um tutorial.

2.2.7 Git

O Git (Torvalds, 2005) é um sistema de controle de versão amplamente utilizado para gerenciar, armazenar e conciliar o histórico de alterações em projetos de software. Ele permite que os desenvolvedores compartilhem código de uma forma gerenciada, facilita a colaboração.

Por sua vez, o GitHub (Github, 2008) é uma plataforma de hospedagem e colaboração de código-fonte baseada em nuvem. Ele fornece um ambiente para desenvolvedores compartilharem e colaborarem em projetos de software usando o sistema de controle de versão Git. Os desenvolvedores podem criar repositórios para seus projetos, que servem como armazenamento para o código-fonte, arquivos de documentação e recursos relacionados.

Além do armazenamento de código-fonte, o GitHub oferece recursos poderosos de colaboração, onde os usuários podem relatar problemas, sugerir melhorias ou discutir ideias relacionadas ao projeto. Também é possível realizar contribuições para propor alterações específicas ao código-fonte e revisá-las antes de serem incorporadas ao projeto principal. A plataforma é conhecida por sua vasta comunidade, onde desenvolvedores podem descobrir projetos, contribuir com projetos de código aberto e aprender com outros desenvolvedores.

2.3 IHC

A Interação Humano-Computador (IHC) (Oliveira, 2015) é um campo de estudo que se preocupa com o design, a avaliação e a implementação de sistemas computacionais interativos para uso humano. O objetivo principal da IHC é criar interfaces eficazes e eficientes entre os seres humanos e os computadores, buscando melhorar a experiência do usuário, a usabilidade e a interação entre as pessoas e as máquinas.

Seus principais aspectos incluem o design eficaz de interfaces, usabilidade, acessibilidade, avaliação de interfaces, busca por interações naturais e foco na experiência do usuário (UX). Essencial para diversos contextos, a IHC visa facilitar a interação entre humanos e computadores, tornando os sistemas mais intuitivos, eficientes e satisfatórios.

A importância da IHC no desenvolvimento de sistemas começa ao entendermos os benefícios que ela pode trazer, como:

a) aumento da produtividade das pessoas;

b) redução do custo de treinamento;

c) redução do número de erros cometidos durante o uso e a gravidade deles;

d) aumento da fidelidade do usuário;

e) redução do custo e de suporte técnico;

f) redução do custo do desenvolvimento.

2.3.1 Personas

Entender os usuários é fundamental para entregar uma boa documentação por diversas razões que impactam diretamente no design, na usabilidade e na eficácia. Compreender os usuários permite que você identifique suas reais necessidades, problemas e expectativas. O conhecimento sobre as preferências, habilidades e limitações dos usuários ajuda na criação de interfaces mais intuitivas e fáceis de usar. É possível antecipar potenciais pontos de confusão e criar um design que minimize erros. Os usuários são mais propensos a aceitar e adotar uma plataforma quando ela é projetada levando em consideração suas necessidades e preferências. Ao focar no que realmente importa, você evita o desperdício de recursos no desenvolvimento de funcionalidades desnecessárias.

A IHC entende que antes de tomar qualquer decisão a respeito do desenvolvimento, é preciso entender as pessoas, o que elas querem, o que elas não querem, quais as suas habilidades e quais são as suas limitações. Trata-se de projetar interações fáceis, agradáveis, úteis, independente de tecnologia ou plataforma.

Personas são arquétipos, personagens ficcionais, concebidos a partir da síntese de comportamentos observados entre o público alvo com perfis extremos. Elas representam as motivações, desejos, expectativas e necessidades, reunindo características significativas de

um grupo mais abrangente, possibilitando que sejam traçados perfis de usuário. (Cooper, 2002)

Para a construção da persona foi utilizado o método PATHY (Ferreira Simone Barbosa, 2016). Os campos que compõem o modelo são descritos a seguir:

Tabela 2.4 – Método PATHY

Rotina	Características da rotina da persona, incluindo seus *hobbies*, aspectos relevantes do ambiente em que vive e uma descrição das pessoas com quem convive.
Sentir, Pensar, Acreditar	Características subjetivas da persona, incluindo suas ideias, aspectos de sua personalidade, medos e frustrações.
Experiência com Tecnologia	As experiências que a persona teve com outras tecnologias, bem como características de aplicativos que agradam e desagradam à persona. O objetivo desta categoria é obter uma compreensão melhor das preferências do usuário.
Problemas	Os problemas que a persona enfrenta, os quais podem ser resolvidos pela aplicação que está sendo projetada. O objetivo desta categoria é obter uma compreensão melhor dos problemas do usuário.
Necessidades	O que é necessário para resolver os problemas descritos na categoria anterior.
Soluções Existentes	Soluções existentes para resolver os problemas, bem como ideias para melhorá-las ou incluí-las na aplicação que está sendo projetada.

Fonte: (Ferreira Simone Barbosa, 2016)

2.3.2 Estilos de Interação

No âmbito da Interação Humano-Computador, é fundamental compreender os diversos modos pelos quais os usuários interagem com sistemas computacionais. A comunicação entre humanos e computadores ocorre por meio de diferentes formas de linguagem, como mostra a tabela abaixo (C. Leite J., 1999):

Tabela 2.5 – Estilos de Interação

Linguagem natural	Linguagem com a qual o usuário se comunica com outros indivíduos (português, inglês, espanhol e etc.).
Linguagens de comando	São comandos específicos – seja por um único caractere ou por abreviações curtas, ou palavras inteiras ou, ainda, uma combinação de teclas e caracteres – que possibilitam ao usuário enviar instruções diretamente ao sistema.
Menu	Conjunto de opções apresentadas na tela, cuja seleção de uma ou mais opções resulta em uma mudança no estado da interface.
WIMP (Windows, Icons, Menus e Pointers)	Junção de tecnologia de hardware e software, associada aos conceitos de janelas e de componentes de interação virtuais denominados widgets que permitem a implementação de vários estilos. Pode ser também considerado um framework de interface apoiado pela tecnologia GUI (Graphical User Interfaces).
Preenchimento de formulário	Conjunto de entradas de dados do sistema que devem ser preenchidas pelo usuário através de campos.
Manipulação direta	O usuário atua diretamente sobre os objetos da aplicação através do mouse.

Fonte: (C. Leite J., 1999)

2.3.3 Usabilidade

A usabilidade, que também é outro critério de qualidade, refere-se à relação que se forma entre tarefa, usuário, equipamento, interface e outros aspectos do ambiente no qual o usuário utiliza. Ela é a qualidade que caracteriza o uso de um sistema interativo. Um conjunto de atributos relacionados com o esforço necessário para o uso de um sistema interativo, e relacionados com a avaliação individual de tal uso, por um conjunto específico de usuários. (Systems. . . , 2011)

De maneira informal e do ponto de vista do usuário, a usabilidade é o fator que assegura que os produtos são fáceis de usar, eficientes e agradáveis e, segundo Preece (Preece, 2011), a usabilidade é dividida nas seguintes metas:

Tabela 2.6 – Fatores de Usabilidade

Eficácia	O sistema deve ajudar os usuários a alcançarem seus objetivos com precisão e de forma completa. Envolve a precisão e completude das tarefas realizadas pelos usuários.
Eficiência	Tempo necessário para a conclusão de uma atividade com apoio computacional. Este tempo é determinado pela maneira como o usuário interage com a interface do sistema. Diz respeito à realização de objetivos de maneira oportuna.
Segurança	Grau de proteção de um sistema contra condições para desfavoráveis ou até mesmo perigosas para os usuários, e existem duas formas para alcançarmos a segurança, são elas: evitar problemas e ajudar o usuário a se recuperar de uma situação problemática.
Utilidade	O sistema deve fornecer o conjunto certo de funções para o usuário e apoiar as tarefas que desejam realizar. A utilidade está relacionada à utilidade geral do sistema.
Aprendizado	Tempo e esforço necessários para que o suário aprenda a utilizar o sistema com determinado nível de competência e desempenho. Essa meta enfatiza a facilidade com que novos usuários podem se tornar proficientes.
Memorização	Esforço cognitivo do usuário necessário para lembrar como interagir com a interface do sistema interativo. A memorização está relacionada à facilidade de lembrar como usar o sistema após um período de não utilização.

Fonte: (Preece, 2011)

2.3.4 Critérios Ergonômicos

Para que o usuário possa aproveitar ao máximo o sistema computacional que ele usa, é preciso que a interação e a interface sejam adequadas, e, em função disso, existem alguns critérios de qualidade que precisam ser conhecidos.

Dentre estes critérios de qualidade, está a ergonomia que se preocupa com o desenvolvimento de máquinas para o homem, assim como o desenvolvimento de tarefas para operar estas máquinas, buscando maior eficiência, qualidade e satisfação para o seu usuário.

Segundo a ISO 9241-11 (Requisitos..., 1998), ergonomia é um conjunto de atributos relacionados com o esforço necessário para o uso de um sistema interativo, e relacionados com a avaliação individual de tal uso, por um conjunto específico de usuários. O grau em que um produto é usado por usuários específicos para atingir objetivos específicos com eficácia, eficiência e satisfação em um contexto de uso específico.

Dominique Scapin e Christian Bastien propuseram um conjunto de oito critérios ergonômicos com o objetivo de minimizar a ambiguidade na identificação e na classificação

das qualidades e problemas ergonômicos do software interativo (Bastien; Scapin, 1992). São estes critérios e seus respectivos subcritérios:

Tabela 2.7 – Critérios Ergonômicos

Condução	Visa beneficiar principalmente o aprendizado e a utilização do sistema por usuários novatos, por uso de convites, agrupamento e distinção entre itens, legibilidade e feedback imediato. Neste contexto, a interface deve aconselhar, orientar, informar e conduzir o usuário na interação com o sistema;
Carga de Trabalho	Se aplica a um contexto de trabalho intenso e repetitivo, no qual os profissionais que operam o sistema precisarão de interfaces econômicas sob o ponto de vista cognitivo, motor, brevidade (concisão e ações mínimas) e densidade informacional. O critério diz respeito a todos os elementos da interface que têm um papel importante na redução da carga cognitiva e perceptiva do usuário e no aumento da eficiência do diálogo;
Controle Explícito	Aplica-se, em particular, às tarefas longas sequenciais e nas quais os processamentos sejam demorados e de controle do usuário. Quando os usuários definem explicitamente suas entradas, e quando estas estão sob controle, os erros e as ambiguidades são limitados. Cada possível ação do usuário deve ser antecipada, e as opções apropriadas devem ser oferecidas;
Adaptabilidade	É uma qualidade particularmente esperada em sistemas em que o público-alvo é vasto e variado. Ela deve também permitir que o usuário adapte as apresentações e estilos de diálogo a suas necessidades;
Gestão de Erros	Se aplica em todas as situações, em particular, quando as ações dos usuários forem sujeitas a erros de grande responsabilidade. De maneira geral, pode-se dizer que as interrupções provocadas pelos erros têm consequências negativas sobre a atividade do usuário, prolongando as transações e perturbando o planejamento;
Consistência	Aplica-se de forma geral, mas, principalmente, quando os usuários são novatos ou intermitentes. O critério homogeneidade/-coerência refere-se à forma na qual as escolhas no projeto da interface são conservadas idênticas em contextos idênticos e diferentes para contextos diferentes;

Significado de Códigos e Denominações	Aplica-se de forma geral, mas são os usuários novatos ou intermitentes que mais tirarão proveito de códigos e denominações bem escolhidos. O critério significado dos códigos e denominações diz respeito à adequação entre o objeto ou a informação apresentada, ou pedida e sua referência na interface;
Compatibilidade	Favorece tanto o aprendizado como a utilização eficiente do sistema por usuários experientes em suas tarefas. O critério compatibilidade diz respeito ao grau de similaridade entre diferentes sistemas que são executados em um mesmo ambiente.

Fonte: (Cybis Adriana Betiol, 2017)

2.3.5 Técnicas de Avaliação de Usabilidade

As técnicas podem se dividir em três, são elas: Técnicas prospectivas, Técnicas preditivas ou Técnicas diagnósticas e Técnicas objetivas ou Técnicas empíricas. Neste trabalho focaremos nas técnicas Preditivas.

Avaliações Preditivas ou diagnósticas dispensam a participação de usuários finais nas avaliações, porque são feitas pelos especialistas em usabilidade ou por projetistas. Ela se baseia em inspeções e verificações em versões intermediárias ou acabadas de software interativo e podem ser classificas em:

2.3.5.1 Avaliações Analíticas

Envolvem a decomposição hierárquica da estrutura da tarefa para verificar as interações propostas. Esta técnica costuma ser empregada no início da elaboração de interface humano-computador e pode não passar de um detalhamento das tarefas interativas realizadas. As especificações do que será desenvolvido pode ser realizado dentro de termos de um formalismo apropriado como GOMS (*Goals, Operators, Methods and Selections rules*).

2.3.5.2 Avaliações heurísticas

Baseiam-se nos conhecimentos ergonômicos e na experiência dos avaliadores que percorrem a interface ou o projeto para identificar possíveis problemas de interação humano-computador.

Jacob Nielsen propôs as "10 Heurísticas de Usabilidade"como princípios fundamentais para avaliação de interfaces de usuário. São elas:

Tabela 2.8 – Heurísticas de Usabilidade

Visibilidade do Estado do Sistema	Mantenha os usuários informados sobre o que está acontecendo por meio de *feedback* apropriado dentro de um tempo razoável, ou seja, informar o usuário sobre qual ambiente ele estava, em qual ele está e para quais outros ambientes ele poderá se dirigir a partir de sua localização.
Correspondência entre o Sistema e o Mundo Real	Utilize linguagem, palavras e conceitos familiares aos usuários, correspondendo à forma como eles pensam sobre o mundo, isso é, um sistema precisa falar a mesma linguagem do usuário.
Controle e Liberdade do Usuário	Forneça aos usuários a capacidade de desfazer ações e navegar facilmente, permitindo um senso de controle sobre o sistema, pois essa possibilidade de reverter ações remove a insegurança do usuário ao utilizar a aplicação.
Consistência e Padronização	Mantenha consistência em todo o sistema, seja em termos de terminologia, design ou interações, para evitar confusão. Manter consistência entre as telas de uma aplicação é essencial para não ser necessário o entendimento de vários padrões e formas de interações diferentes para cada tela, uma vez aprendido será algo replicável em outros contextos.
Prevenção de Erros	Projete o sistema de maneira a prevenir erros sempre que possível, ou, se ocorrerem, forneça mensagens claras e simples para orientar os usuários sobre como corrigi-los. Essa heurística propõe que a interface esteja apta a prevenir qualquer tipo de ação descuidada do usuário.
Reconhecimento em Vez de Lembrança	Minimize a carga cognitiva dos usuários, tornando as informações necessárias para a interação visíveis e disponíveis, em vez de exigir que elas se lembrem de detalhes.
Flexibilidade e Eficiência de Uso	Ofereça opções e atalhos para usuários experientes, permitindo que eles realizem tarefas de maneira eficiente, sem comprometer a usabilidade para usuários menos experientes.
Design Estético e Minimalista	Busque por um design estético e limpo, removendo informações desnecessárias e focando no essencial para melhorar a compreensão e a eficiência.
Ajuda e Documentação	Forneça suporte quando necessário, por meio de uma documentação clara e acessível, mas, sempre que possível, projete o sistema de forma que os usuários possam entender e usar sem depender de instruções.

Reconhecimento, Diagnóstico e Recuperação de Erros	Projete mensagens de erro que ajudem os usuários a entender o problema, sugiram soluções e facilitem a recuperação sem perder dados importantes.

Fonte: (Nielsen, 1994)

Essas heurísticas são diretrizes valiosas para avaliação e design de interfaces, proporcionando um caminho sólido para a criação de sistemas mais intuitivos e eficazes.

2.4 Requisitos

Os requisitos de software são fundamentais para a definição das propriedades e comportamentos do sistema, sendo essenciais para atender às necessidades dos usuários e clientes. Eles podem ser classificados em dois grupos principais: requisitos funcionais e requisitos não funcionais.

Os requisitos funcionais descrevem o comportamento do sistema, ou seja, as ações que ele deve realizar em resposta a determinadas entradas. Eles definem o que o sistema deve fazer e quais resultados deve produzir para o usuário.

Por outro lado, os requisitos não funcionais expressam como o sistema deve realizar suas funções, relacionando-se geralmente a padrões de qualidade como confiabilidade, desempenho e robustez. Eles são igualmente importantes, pois garantem que o sistema seja eficiente para a tarefa que se propõe a realizar.

Além disso, os requisitos não funcionais também incluem restrições e especificações de uso que complementam os requisitos funcionais, garantindo que o sistema seja desenvolvido de acordo com as necessidades e expectativas dos usuários.

É fundamental que a equipe de desenvolvimento compreenda completamente o contexto do negócio, suas operações, informações e expectativas em relação ao aplicativo a ser construído (MACHADO, 2018). Essa compreensão é essencial para elaborar e desenvolver todas as atividades de forma criteriosa, garantindo que o sistema atenda adequadamente às necessidades dos usuários e do negócio.

2.5 História de Usuário

Documentos de requisitos tradicionais, frequentemente se estendem por centenas de páginas e levam, por vezes, grande parte do tempo do projeto para serem finalizados. No entanto, esses documentos enfrentam desafios significativos:

Durante o desenvolvimento, os requisitos tendem a mudar, tornando os documentos rapidamente obsoletos. Descrições em linguagem natural são ambíguas e incompletas, o que

leva os desenvolvedores a necessitarem de esclarecimentos adicionais o desenvolvimento. A falta de comunicação eficaz pode resultar em desacordo entre o que foi especificado e o que o cliente desejava.

Diante desses desafios os profissionais da indústria que reconheceram esses problemas propuseram uma abordagem mais pragmática, conhecida como Histórias de Usuários. Características de boas histórias, encapsuladas no acrônimo *"INVEST"*:

Tabela 2.9 – Características História de Usuário

Independência	Histórias devem ser independentes entre si, permitindo que sejam implementadas em qualquer ordem.
Negociabilidade	Devem ser abertas para negociação entre clientes e desenvolvedores, permitindo ajustes durante o processo.
Valor	Devem agregar valor ao negócio do cliente, priorizadas de acordo com sua importância.
Estimáveis	Deve ser possível estimar o tempo necessário para implementar uma história.
Sucintas	As histórias do topo do *backlog*, a serem implementadas em breve, devem ser curtas para facilitar o entendimento e a estimativa.
Testáveis	Devem possuir critérios de aceitação objetivos para garantir testabilidade e clareza de entrega.

Fonte: (Valente, 2020)

Adotar a abordagem das Histórias de Usuários proporciona uma maneira mais flexível e adaptável de lidar com os requisitos do projeto, mitigando os problemas associados aos métodos tradicionais (Valente, 2020).

3 Trabalhos relacionados

Nesta seção, apresentamos uma revisão dos trabalhos relacionados à temática abordada visando explicitar o contexto que a discussão está inserida. Os trabalhos selecionados foram agrupados em duas categorias principais:

3.1 Trabalhos em juízes online

A revisão dos trabalhos contribui para situar o Contest-Driven Meta Online Judge (CD-MOJ) no contexto atual e destacar as contribuições e avanços proporcionados. Alguns desses trabalhos são:

Um estudo de caso na utilização de juízes eletrônicos e problemas oriundos da Maratona de Programação no ensino de programação da Faculdade UnB Gama (Pereira, 2015). O trabalho teve como objetivo analisar uma metodologia de avaliação baseada em competições de programação, buscando analisar a eficiência desta metodologia observando o impacto no desempenho dos alunos nas disciplinas de programação. Para isso foi feita a análise das métricas extraídas com base no desempenho dos alunos monitorados para obter conclusões sobre o impacto da metodologia em relação ao desempenho individual do aluno e em comparação com os demais alunos.

Além disso, uma pesquisa dedicada ao ambiente CD-MOJ focou na atualização dos métodos de inserção de problemas e modalidades (Souza, 2018). Um novo formulário foi desenvolvido para incorporar os métodos de inserção de problemas e modalidades no ambiente de correção do CD-MOJ, mantendo as exigências do modelo anterior e adicionando de algumas variáveis. Agora o sistema conta com um método de peso e penalidade por problema adicionado.

Por fim, outro trabalho de contribuição para melhorias no sistema CD-MOJ foram a construção de uma área para esclarecimentos, imagem *docker* para instalação de instância do CD-MOJ como ambiente de desenvolvimento, implementação do verificador de plágio JPlag e contribuições para a documentação CD-MOJ (Silva, 2022). No trabalho também é possível encontrar explicações de toda a estrutura do CD-MOJ, diagramas de componentes e descrição de *scripts*.

3.2 Trabalhos sobre IHC

A revisão e utilização de conceitos e convenções acerca da interação humano computador enriquece a argumentação e defende as soluções propostas ao longo do projeto. Dentre

os trabalhos consultados para embasar essa abordagem, destacam-se algumas contribuições importantes.

"HCI Remixed - Essays on Works That Have Influenced the HCI Community"(Erickson, 2008) oferece uma coletânea de ensaios sobre trabalhos no campo da Interação Humano-Computador. Os ensaios abordam projetos com pelo menos dez anos de idade, escolhidas por diversos contribuidores, cada um refletindo sobre o impacto dessas obras em suas visões sobre IHC. Os organizadores procuram resgatar trabalhos mais antigos da memória da disciplina, destacando a diversidade de influências que moldaram o campo. Os ensaios têm uma abordagem pessoal, buscando preservar as vozes individuais dos autores. Além de proporcionar uma visão mais pessoal do IHC, a coletânea visa oferecer uma entrada acessível aos recém-chegados à disciplina e promover uma abordagem mais positiva e apreciativa para a avaliação do trabalho em IHC.

Por sua vez, *"Interaction Design"*(Preece, 2011) é destinado a estudantes de graduação, mestrado e doutorado, assim como profissionais. O livro aborda disciplinas como interação humano-computador, design de interação, design de web, engenharia de software, mídia digital e sistemas de informação. Explora uma variedade de interfaces, desde as tradicionais até as mais recentes, como interfaces cerebrais, móveis, robóticas e de realidade mista. Abrange temas como questões cognitivas, sociais e afetivas no design de interação, coleta e análise de dados, e fornece atividades práticas para o desenvolvimento.

Além desses, "Interação Humano Computador"(Oliveira, 2015), feito por professores brasileiros, contempla pontos-chaves na interação humano computador. Tais como: teorias, métodos, gerência entre os processos do projeto, manipulação direta e ambientes virtuais. O texto aborda a importância do design de interface para sistemas interativos de computador, destacando que toda interação do usuário com o sistema ocorre por meio da interface. Uma interface bem projetada pode resultar na satisfação do usuário, aumento da produtividade, melhoria na qualidade do trabalho e redução de erros.

4 Contribuições

O objetivo deste trabalho é promover, facilitar a utilização, configuração, manutenção e compreensão do CD-MOJ. Fornecendo documentação com informações claras e precisas sobre seus recursos, funcionalidades e práticas adotadas para usuários, desenvolvedores e administradores.

Adicionalmente, a implementação de um módulo de treinamento visa incentivar a resolução independente dos problemas disponíveis no banco de questões do CD-MOJ. Simultaneamente, assegura o funcionamento adequado do Jplag (JPlag, 1996) em *contests*.

Dessa forma, podemos listar as contribuições realizadas nos seguintes tópicos:

4.1 Documentação

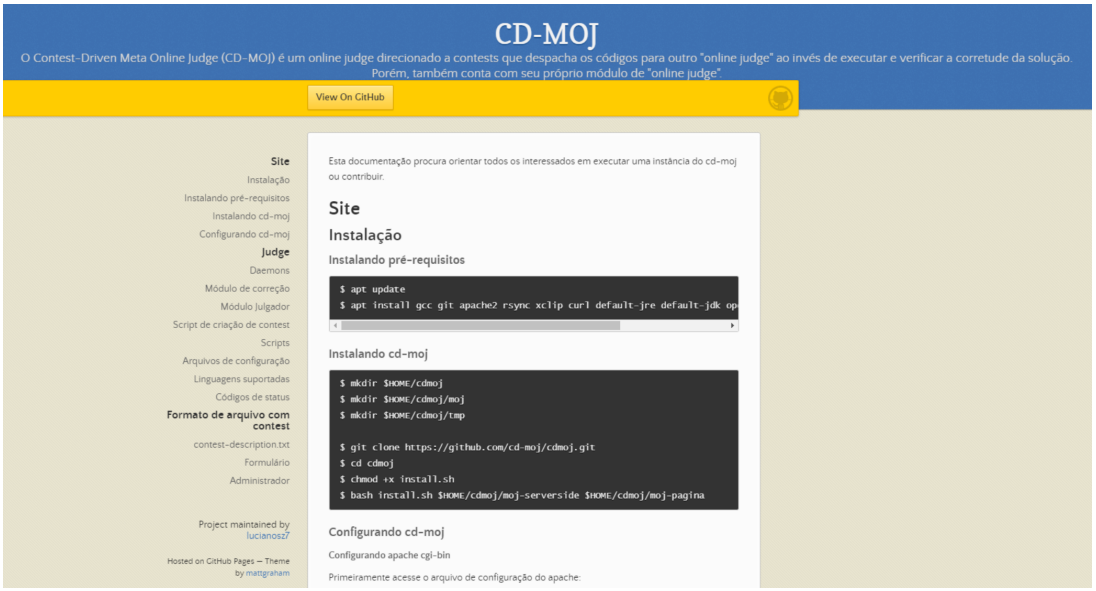
A documentação realizada no TCC "CD-MOJ: Contribuições para melhorias no sistema"(Silva, 2022), documenta informações para o entendimento do funcionamento interno do sistema do CD-MOJ. Também auxilia o usuário que deseja instalar e executar uma instância do CD-MOJ em uma máquina local. Para isso, foi utilizada a ferramenta MkDocs (Christie, 2014) com o objetivo de ajudar na organização, formatação e disponibilidade do serviço GitHub Pages (Github, 2008). A documentação estava separada em três seções em uma pagina monolítica: *Site*, *Judge* e Formato de arquivo com contest.

A primeira seção era destinada a auxiliar a instalação de uma instância do CD-MOJ e configurá-la de forma que consiga executar as funcionalidades desejadas.

A segunda seção, por sua vez, tinha o objetivo de explicar o funcionamento relacionado aos *daemons*, módulos de correção e julgador, *scripts* responsáveis por criar as listas, tratar as requisições com os outros juízes online, arquivos de configuração, as linguagens suportadas e códigos de status.

Por fim, a última seção era responsável por expor o padrão aceito para a criação das listas e seus formas de criação, assim como a criação de usuário administrador para contar com o privilégio de enviar arquivos ou acessar o formulário de criação.

Figura 4.1 – Página de documentação anterior



Após a análise dos tópicos já existentes na documentação uma nova proposta de organização foi elaborada, uma vez que todo o conteúdo encontrava-se em uma mesma página monolítica sem divisão coerente. Além disso, não existia uma forma prática e rápida de se pesquisar algum assunto pontual. Outro ponto de melhoria identificado foi a adição de etapas importantes para a correta configuração de uma instância local do CD-MOJ.

4.1.1 Personas da documentação

Para tal proposta o uso de personas(Cooper, 2002) na construção de documentação foi adotada para garantir que o conteúdo seja direcionado e adaptado às necessidades, interesses e níveis de experiência dos diferentes tipos de usuários. Para a construção das personas foi utilizado o método PATHY (Ferreira Simone Barbosa, 2016). Seguindo esse modelo foi definido um conjunto de perfis de usuários potenciais do CD-MOJ, sendo eles: usuários, administradores e desenvolvedores.

a) O usuário: Lucas, é um estudante universitário de 19 anos, calouro de Engenharia de Software. Ele tem uma rotina intensa, dedica grande parte do seu tempo a atividades relacionadas ao curso. Ele acredita na importância da prática e da aplicação dos conhecimentos teóricos. Possui pouca experiência com plataformas de juízes online, muitas vezes sente dificuldades na navegação e compreensão de algumas funcionalidades. Ele precisa de uma documentação clara e passo a passo para entender como navegar e utilizar as funcionalidades da plataforma. Uma documentação confusa ou incompleta pode fazer com que Lucas se sinta perdido, resultando em dificuldades na navegação e compreensão das funcionalidades.

b) O administrador: Bruno Ribas, além de lecionar, dedicar tempo à pesquisa e orienta-

ção de projetos de alunos, gosta de manter-se atualizado com as últimas tendências em tecnologia. Acredita que a prática é essencial para o aprendizado em Engenharia de Software e valoriza a eficiência nas ferramentas que utiliza. Fica frustrado com plataformas complicadas que dificultam a interação professor-aluno. Prefere sistemas que permitam uma fácil submissão e correção de tarefas. O professor Bruno precisa de uma documentação clara para orientar seus alunos e fornecer suporte eficiente. Uma documentação mal elaborada pode aumentar a carga de trabalho do professor, exigindo mais tempo para procedimentos automatizáveis e prejudicando a eficiência na submissão e correção de tarefas.

c) O desenvolvedor: Caio está imerso na cultura da comunidade open source, participando ativamente de projetos e colaborando em repositórios no GitHub e interação em fóruns de desenvolvedores. Ele valoriza a liberdade de código aberto, acredita na colaboração global e sente-se satisfeito ao superar desafios técnicos, mas fica frustrado com sistemas complicados e mal documentados. Ele busca informações detalhadas sobre como interagir com o projeto e envolver-se na comunidade. Uma documentação inadequada pode resultar em menor participação na comunidade e afetar negativamente a qualidade e evolução dos projetos.

Entendido o perfil dos usuários, podemos iniciar a fase de concepção das interfaces de usuário. A interação é o ponto central desse processo, e representa o processo de comunicação do usuário com o sistema através de interfaces. Ela engloba todas as ações e reações que ocorrem quando o usuário utiliza a interface para executar tarefas dentro do sistema computacional (Barbosa, 2010).

4.1.2 Interface da documentação

Para projetar interfaces eficazes, é fundamental considerar não apenas a funcionalidade, mas também a usabilidade, a acessibilidade e a experiência do usuário. Vamos considerar a interface como o meio de interação do usuário com o sistema, e a interação como a troca de informações que ocorre entre usuário e sistema.

A interface então pode ser entendida como o programa ou a imagem do sistema, ou ainda uma ferramenta que oferece os instrumentos para um processo comunicativo. Ela é responsável por promover estímulos de interação para que o usuário obtenha respostas relacionadas às suas atividades. De um lado, ela funciona como dispositivo de entrada de dados, e, de outro, ela é responsável por enviar respostas aos usuários. Desta forma, a interface é um sistema de comunicação.

Dentre todos os estilos de interação: linguagem natural, linguagem de comando, Menus, WIMP, preenchimento de formulários e manipulação direta, a escolhida foi a utilização de menus, um conjunto de opções apresentado na tela no qual a seleção de uma ou mais

opções resulta em uma mudança de interface. Neste estilo de interação, os usuários não precisam se lembrar, ou conhecer os itens que desejam, basta reconhecê-los.

Os sistemas de menu são considerados mais amigáveis para usuários iniciantes por várias razões:

a) Visibilidade das opções: Em um sistema de menu, as opções estão visíveis na tela, o que facilita a descoberta para os usuários iniciantes. Eles podem explorar as diferentes opções disponíveis sem a necessidade de memorizar comandos ou entender a estrutura interna do sistema.

b) Guiamento passo a passo: Os menus geralmente apresentam uma estrutura hierárquica, organizando as opções de forma lógica. Isso permite que os usuários naveguem de maneira passo a passo, seguindo uma lógica intuitiva. Para usuários iniciantes, essa abordagem guiada é mais fácil de entender do que a necessidade de saber antecipadamente quais comandos usar.

c) Redução de erros: Os menus podem ajudar a reduzir erros, já que as opções são apresentadas de maneira clara e não deixam espaço para interpretações ambíguas. Em sistemas baseados em linguagem de comando, por exemplo, um erro de digitação pode resultar em comandos inválidos, enquanto os menus minimizam esse tipo de problema.

d) Baixa curva de aprendizado: A curva de aprendizado é geralmente mais suave em sistemas de menus. Os usuários iniciantes não precisam aprender comandos específicos ou memorizar códigos. Eles podem começar a utilizar o sistema imediatamente, explorando as opções disponíveis nos menus.

e) *Feedback* visual: Os menus geralmente fornecem *feedback* visual imediato. Quando um usuário seleciona uma opção, há frequentemente uma resposta visual na tela, indicando que a ação foi reconhecida e está em andamento. Isso é valioso para usuários iniciantes, pois fornece confirmação de que estão interagindo corretamente com o sistema.

f) Acessibilidade: Os menus são mais acessíveis para usuários com menos experiência em tecnologia. Eles eliminam a necessidade de lembrar comandos específicos, tornando a interação mais natural e menos intimidadora.

4.1.3 Critérios ergonômicos aplicados a documentação

As mudanças feitas na documentação do CD-MOJ podem ser justificadas considerando os oito critérios ergonômicos propostos por Dominique Scapin e Christian Bastien:

a) Condução: Uma vez que todo o conteúdo da documentação encontrava-se na mesma página sem divisão coerente, a substituição do tema anterior pelo *Material Theme* e

uso de menus no *MkDocs* proporciona uma navegação mais intuitiva e direcionada. A adição de uma caixa de busca avançada facilita ainda mais a condução, permitindo que os usuários encontrem rapidamente as informações desejadas;

b) Carga de Trabalho: A nova organização da documentação, com a divisão em diferentes grupos de interesse (início, usuário, administrador e desenvolvedor), reduz a carga cognitiva dos usuários. Cada grupo é direcionado a informações específicas, evitando sobrecarga de informações irrelevantes;

c) Controle Explícito: A introdução da caixa de busca oferece aos usuários um controle explícito sobre a localização de informações. Além disso, a marcação semântica e a organização dos tópicos permitem uma navegação mais direta e controle sobre o conteúdo apresentado;

d) Adaptabilidade: A nova estrutura da documentação, a divisão em diferentes grupos de interesse, juntamente com o *Material Theme*, oferece adaptabilidade ao suportar não apenas usuários diferentes, mas também dispositivos, como telas sensíveis ao toque, teclados e leitores de tela. Isso garante que a documentação seja acessível a usuários com diferentes necessidades e capacidades;

e) Gestão de Erros: A clareza na documentação, e a consistência na apresentação das informações, contribui para a redução de erros de interpretação por parte dos usuários. A estrutura organizada e a acessibilidade também minimizam a probabilidade de cometer erros durante a navegação;

f) Consistência: A adoção do *Material Theme* proporciona uma experiência consistente em toda a documentação. A padronização na apresentação visual e na estrutura dos tópicos contribui para a compreensão coesa das informações, promovendo a consistência na experiência do usuário;

g) Significados de Códigos: A marcação semântica e a consistência de posicionamento de elementos utilizada na documentação garante que as informações sejam interpretadas corretamente pelos dispositivos e pelos usuários. Isso contribui para a clareza e a compreensão dos códigos, evitando ambiguidades.

h) Compatibilidade: A compatibilidade foi considerada ao escolher as personas, seus pontos de interesse e o *Material Theme*, sendo conhecido por sua flexibilidade e adaptabilidade a diferentes ambientes e dispositivos. A acessibilidade incorporada também garante que a documentação seja compatível com uma ampla gama de tecnologias assistivas.

4.1.4 Heurísticas de Nielsen aplicadas a documentação

Vamos também analisar as mudanças feitas na documentação utilizando as 10 heurísticas de Nielsen, que são princípios gerais de usabilidade(Nielsen, 1994):

a) Visibilidade do Estado do Sistema: A substituição do tema foi feita para o *Material Theme*, proporcionando um visual mais moderno e organizado. A nova organização da documentação, com a adição dos menus, divisão do conteúdo em personas e diferenciação de cores torna o estado do sistema mais visível e compreensível para os usuários.

b) Relacionamento Entre o Sistema e o Mundo Real: A nova organização da documentação facilita a localização e acesso às informações relevantes sobre o CD-MOJ, melhorando a relação entre o sistema documentado e as necessidades dos usuários do mundo real.

c) Controle e Liberdade do Usuário: A adição da caixa de busca proporciona controle aos usuários, permitindo que eles busquem informações de forma rápida e eficiente. A divisão em tópicos também oferece liberdade para os usuários escolherem os caminhos de navegação que mais atendem às suas necessidades.

d) Consistência e Padrões: A escolha do *Material Theme* traz consigo padrões de design conhecidos e uma interface consistente. O menu superior separa o conteúdo em personas, o menu esquerdo os seus pontos de interesse e o direito os tópicos trabalhados na página selecionada.

e) Prevenção de Erros: A melhoria na organização e na acessibilidade pode contribuir para a redução de erros por parte dos usuários.

f) Reconhecimento ao Invés de Lembrança: A representatividade em personas busca o reconhecimento. A inclusão de uma caixa de busca facilita o reconhecimento ao permitir que os usuários encontrem informações sem precisar lembrar onde elas estão localizadas na documentação.

g) Flexibilidade e Eficiência de Uso: A adição da caixa de busca e a organização em diferentes tópicos oferecem flexibilidade e eficiência aos usuários, permitindo que eles encontrem informações relevantes de maneira rápida e direta.

h) Estética e Design Minimalista: A escolha do *Material Theme* sugere um design minimalista e esteticamente agradável, o que contribui para uma experiência de usuário positiva.

i) Ajuda os Usuários a Reconhecerem, Diagnosticarem e Recuperarem-se de Erros: a estrutura organizada e a busca podem ajudar os usuários a se recuperarem de possíveis dificuldades de encontrarem o tema procurado.

j) Ajuda e Documentação: A nova organização da documentação torna as informações mais acessíveis e compreensíveis para os usuários.

4.1.5 MKdocs da documentação

Com isso posto, a primeira alteração feita foi a substituição do tema utilizado anteriormente pelo *MkDocs* (Christie, 2014). Existem vários temas que podem ser utilizados de forma bem intuitiva e prática. Entretanto foi escolhido o *Material Theme*(Donath, 2016), construído com tecnologia *Open Source* licenciada pelo padrão MIT.

O *Material Theme* foi selecionado por sua maturidade, robustez e flexibilidade, já que permite a fácil modificação e personalização da aparência, cores, fontes, idioma, ícones, logotipo e comportamento da documentação com uma variedade de opções disponíveis. Dentre as quais é importante evidenciar:

a) Divisão por abas - ele permite que se separe todo o conteúdo em documentos e os indexe de acordo com o conteúdo dentro de uma lógica;

b) Pesquisa - é possível realizar pesquisas por palavras ou frases, evitando assim ter que ficar procurando dentro de todos os documentos criados;

c) Manutenibilidade - a alteração da documentação criada e adição de posteriores informações é fácil e rápida, além de manter o histórico de quem fez as alterações.

A nova organização da documentação, combinada com o visual aprimorado, proporciona aos usuários uma experiência de navegação mais agradável e facilita a localização e o acesso às informações relevantes sobre o CD-MOJ.

Além disso, a acessibilidade para pessoas com qualquer tipo de incapacidade foi trabalhada e entregue. A marcação semântica garante que a documentação funcione para todos os tipos de usuários que venham a utilizar com dispositivos de toque, teclados e leitores de tela, adaptando-se automaticamente ao dispositivo.

Com a nova organização da documentação, duas melhorias importantes foram implementadas para aprimorar a experiência dos usuários. A primeira foi a adição de uma caixa de busca, que permite uma filtragem avançada de palavras-chave em todos os tópicos e subtópicos da documentação. Essa funcionalidade facilita a localização rápida e precisa das informações desejadas, tornando a navegação mais eficiente e eficaz.

O segundo ponto de melhoria foi a divisão mais clara dos tópicos em diferentes grupos de interesse. O tema permite que o conteúdo seja paginado e indexado de acordo com o desejo do desenvolvedor, facilitando a organização e a navegação do usuário final.

Assim foram criados os tópicos: início, usuário, administrador e desenvolvedor. Essa divisão coerente e estruturada permite aos usuários encontrar as informações relevantes de acordo com suas necessidades específicas. Abaixo é explicado a divisão:

4.1.5.1 Página de Início

A seção de início apresenta uma carta aberta dirigida a todas as partes interessadas, fornecendo explicações básicas sobre o CD-MOJ. Essa carta tem como objetivo oferecer um

resumo conciso da natureza do projeto, os motivos que levaram sua criação e seus principais objetivos. Além disso, são fornecidos dados de contato para esclarecimento de dúvidas, indicações e manifestações de interesse em futuras contribuições.

Ela tem como objetivo estabelecer uma base comum de entendimento e engajamento com todos os envolvidos no CD-MOJ, servindo como uma introdução e fornecendo informações essenciais sobre o sistema de forma a convidar as pessoas a explorarem a documentação e se envolverem na comunidade.

4.1.5.2 Página de Usuário

Para a sessão de usuário foram abordados dois tópicos, o primeiro citando as linguagens suportadas, sendo elas *Assembly (SPIM), C, C++, Golang, Java, JavaScrip, OCaml, Pascal, Python 3, RISC-V, Rust e Shell*. O segundo por sua vez discorre a respeito dos códigos de status composto por uma sigla recebidos ao se submeter um arquivo para avaliação na plataforma, são eles:

Tabela 4.1 – códigos de status

Código	**Definição**
AC - Accepted	Indica que a solução do problema passou nos testes e foi aceita.
WA- Wrong Answer	Indica que a solução do problema foi executada sem erros, porém a saída produzida não estava correta de acordo com os testes.
RE - Runtime Exception	Indica que a solução do problema causou uma exceção durante a execução. O código pode também ser emitido em erros de sintaxe em linguagens interpretadas como *Python*.
SIGSEGV (signal 11)	Violação de segmento.
SIGXFSZ (signal 25)	Limite de saída excedido.
SIGFPE (signal 8)	Erro de ponto flutuante.
TLE - Time Limit Exceeded	Indica que a solução do problema demorou muito tempo para ser executada e excedeu o limite estabelecido.
CE - Compilation Error	Indica que a solução não pôde ser compilada devido a erros no código-fonte.

Fonte: (Ribas, 2013)

4.1.5.3 Página de Administrador

A sessão dedicada ao administrador foi construída pensando em abranger todo o escopo de uso esperado. Para isso ela foi subdividida em seis subtópicos:

a) Daemons: No tópico foi discutido os requisitos mínimos que precisam ser cumpridos para se ter uma instância funcional da plataforma sendo executada, foi juntamente mencionado os comandos a serem executados.

b) Exercícios: Um breve tópico esclarecendo os exercícios que estão disponíveis e funcionais para que um administrador desfrute de todas as funcionalidades em uma instalação local do sistema.

c) Contest: O tópico *contest* é destinado a tratar sobre o arquivo que deve ser enviado ao sistema para se criar uma competição no CD-MOJ. Nele podemos encontrar todos os atributos necessários a serem passados, formulário de exemplo e um formulário modelo.

d) Passo a passo para criar um *contest*: É mostrado as etapas para se criar um *contest* no CD-MOJ em dois métodos, o primeiro enviando o arquivo com a formatação descrito acima e o segundo por meio de um formulário dentro da plataforma. (Souza, 2018)

e) Passo a passo para entrar no *contest*: É mostrado as etapas para se entrar em um *contest* no CD-MOJ.

4.1.5.4 Página de Desenvolvedor

A seção do desenvolvedor contém informações organizadas em vários tópicos para facilitar o acesso e a compreensão do conteúdo. Essa seção é especialmente destinada aos desenvolvedores interessados em contribuir para o CD-MOJ e envolver-se em seu desenvolvimento. Ela é dividida em:

a) Instalação : Esta seção contém comandos para instalar os requisitos e o CD-MOJ em um ambiente de desenvolvimento.

b) Configuração : Nessa etapa, estão as configurações necessárias para garantir o funcionamento completo do sistema. Começando pelo servidor Apache e a configuração local a ser feita no CD-MOJ. Além disso, são fornecidas soluções para erros comuns que ocorram durante o desenvolvimento do presente trabalho.

c) Possíveis erros: É mostrado as etapas para se verificar as mensagens de *log* disparadas pelo sistema.

d) MkDocs : Nesta seção, é apresentado um guia sobre a ferramenta MkDocs (Christie, 2014), utilizada para criar a documentação. É explicado como fazer o download do repositório de documentação, os comandos necessários para executá-lo localmente, como fazer o *deploy* e algumas informações relevantes para o tópico.

e) Contribuição : Na seção de contribuição foram implementadas três diretrizes principais:

- Política de branchs - Pelo numero reduzido de contribuidores, não há criação de *branchs*, todas as mudanças são realizadas na *main*, para evitar *overhead* em *merges*. Apostamos na boa comunicação entre os participantes;
- Política de issues - Há quatro padrões de issue pré definidos: Bug, Fix, Doc, Feat. Além disso, é possível criar uma issue em branco;
- Política de commit - Os *commits* devem seguir o padrão utilizado atualmente:

```
git commit -m "<ditetorio do arquivo>: <Descricao do commit>"
```

- Pareamento - Caso um commit seja realizado por duas ou mais pessoas, deve seguir o padrão:

```
git commit -m "<ditetorio do arquivo>: <Descricao do commit>


Signed-off-by: <Nome do responsavel>
   <email-responsavel.example.unb>
Co-authored-by: <Nome do auxiliar>
   <email-auxiliar.example.unb>"
```

f) Judge: Esta seção explica o funcionamento dos juízes online, a principal parte do CD-MOJ. São detalhados dois *daemons* que funcionam em paralelo: o módulo julgador, responsável por receber e endereçar todos os pedidos feitos ao sistema. E o módulo de correção, responsável por enviar as submissões para o corretor correspondente.

g) Contest: Aqui são mostrados vários *scripts* criados para facilitar a comunicação com os juízes online e o próprio sistema do CD-MOJ. Também é fornecida uma explicação sobre os arquivos de configuração e variáveis locais.

h) Simulação: Na página, é apresentado um passo a passo detalhado para que os desenvolvedores com uma instalação local possam realizar simulações a fim de testar e estudar o comportamento do sistema em vários cenários. Os principais passos incluem o login de um usuário, a realização de submissões em *contests* e a simulação de respostas pelo corretor.

4.1.5.5 Página de Contribuidores

Nessa sessão se encontra informações sobre todos os membros que participaram ativamente do projeto CD-MOJ. Cada integrante é apresentado com sua foto, nome, trabalho desenvolvido e os *commits* que realizaram durante o projeto.

4.1.6 Repositório do Projeto

A contribuição deste trabalho está integralmente presente no repositório de documentos do projeto CD-MOJ na plataforma GitHub (Github, 2008). Ele inclui todos os documentos em formato Markdown necessários para futuras contribuições, as imagens utilizadas e os arquivos de configuração utilizados pelo MkDocs (Christie, 2014). Além disso, a descrição do repositório foi atualizada, fornecendo explicações básicas sobre o CD-MOJ, suas motivações, informações de contato e um link para a página de documentação.

Além das atualizações no repositório, foram feitos ajustes em sua estrutura, a *branch "main"* foi definida como a raiz do projeto. Isso foi necessário para evitar interferências com a ferramenta "gh-deploy"fornecida pelo MkDocs (Christie, 2014), que estava substituindo a raiz do projeto para disponibilizar um *build* da documentação para ser utilizada no *deploy*.

Por fim, foram criados quatro *templates* para as *issues*, com o objetivo de facilitar o registro, melhorar a rastreabilidade e o acompanhamento de resolução e identificação de diferentes tipos de problemas e ideias relacionadas ao CD-MOJ. Esses *templates* são o relatório de bugs, utilizado para reportar problemas encontrados, o relatório de consertos, para sugerir a manutenção de alguma funcionalidade, o relatório de documentação, para sugerir novas ideias para a documentação, e o relatório de funcionalidade, para propor novas funcionalidades ao projeto.

Tendo em vista que os contribuidores ao sistema CD-MOJ são majoritariamente formados por estudantes e o professor Bruno Ribas, é comum que haja uma rotação muito grande e confusões sejam geradas ao longo do tempo. Logo, essa padronização é muito útil para organizar a divisão de tarefas entre os contribuintes e mantém um rastro de fácil entendimento para quem queira realizar melhorias ao sistema.

4.2 Soluções no Jplag

Com a implementação anterior referente ao uso do Jplag (JPlag, 1996) era feito o download da ferramenta toda vez que um contest fosse criado, assim teríamos diversos arquivos executáveis, algo que se mostrou desnecessário. Ademais, as acusações de plágio eram inconsistentes, não representavam a realidade observada.

Assim foi realizada uma manutenção corretiva no código para que o executável do Jplag (JPlag, 1996) não apresentasse redundâncias, mas que seja feito o download automático apenas uma vez. Além disso, foi feita a inserção de uma *flag* global de sensibilidade para a ferramenta.

4.3 Makefile

A escolha entre um Makefile e um *script Bash* para a instalação depende do contexto e dos requisitos específicos do projeto. Ambas as abordagens têm vantagens e desvantagens. A decisão de construir um Makefile sobre o *script bash* se da por ele ser especialmente útil quando há dependências entre diferentes partes do código e especificar regras claras para a construção do projeto e permissões de sistema. Além disso, são frequentemente usados em projetos de código-fonte aberto e são bem conhecidos pela comunidade de desenvolvedores, já que possuem uma sintaxe relativamente simples e são legíveis. O Makefile foi pensado seguindo a estrutura abaixo:

a) Definição de Variáveis: SERVERDIR e HTMLDIR são variáveis que representam os diretórios dos aquivos de servidor e de cliente, respectivamente. Se não forem especificadas, caminhos padrões estão disponíveis e seriam mostrados ao usuário.

b) Exportação de Variáveis: Exporta algumas variáveis para que elas possam ser usadas em comandos shell.

c) Alvo "packages": Instala pacotes necessários (como gcc, git, apache2, rsync, etc.) usando o comando "apt install". Essa é uma maneira de garantir que as dependências do sistema estejam instaladas. O prefixo -"indica para o sistema ignorar o status de saída do comando que é executado, normalmente, um status de saída diferente de zero interromperia essa parte da construção. Entretanto, queremos dar a escolha para o usuário de instalar ou não as dependências.

d) Alvo "change_token": Usa o comando find para percorrer diretórios e executar comandos. Cria diretórios temporários em /tmp e substitui variáveis em arquivos dentro dos diretórios server e html usando o comando "envsubst"pelos valores exportados anteriormente, sem interferir com os arquivos originais. Essa substituição serve para configurar o ambiente.

e) Alvo "install_html": Cria o diretório especificado em HTMLDIR. Instala arquivos HTML e diretórios relacionados nos locais apropriados e ajusta as permissões dos diretórios.

f) Alvo "install_server": Instala arquivos relacionados ao servidor nas localizações apropriadas e ajusta as permissões dos diretórios.

g) Alvo "apache_conf": Usa o comando envsubst para substituir variáveis em arquivos de configuração do Apache. Adiciona configurações ao arquivo principal de configuração do Apache e ao arquivo de configuração específico para o projeto (moj.conf). Habilita e desabilita alguns sites e módulos do Apache e por fim recarrega o serviço Apache.

h) Alvo "clear_tmp": Remove os diretórios temporários e arquivos temporários criados durante o processo.

i) Alvo "message": Exibe uma mensagem informativa, pedindo para garantir que um arquivo de configuração específico contenha credenciais pessoais.

j) Alvo "all": Limpa os diretórios temporários, instala pacotes, realiza alterações nos tokens, instala HTML, instala o servidor, configura o Apache e exibe uma mensagem informativa.

k) Alvo padrão "DEFAULT_GOAL": Define o alvo padrão como "all".

4.4 Módulo de Treinamento

Neste novo capítulo, exploramos a abordagem para aprimorar a experiência de aprendizado apresentando um módulo de treinamento dentro do CD-MOJ. A essência desse módulo consiste na criação de uma visualização clara das questões, possibilitando aos usuários uma navegação fluida pelo sistema. Com uma compreensão completa dos problemas apresentados, os participantes são livres para praticar livremente, buscando áreas específicas de estudo através de uma organização por meio de *tags*.

As tags serão usadas para direcionar os esforços de aprendizado dos alunos. Essa estrutura permite que eles se concentrem nas áreas de maior relevância para seu desenvolvimento. Além disso, os participantes têm a oportunidade de acompanhar suas submissões, identificar áreas de melhoria e monitorar seu próprio progresso.

Os professores por sua vez desempenham o papel na administração e na manutenção do banco de questões. A capacidade de etiquetar questões com *tags* que indicam conceitos abordados facilitam esse processo, ao mesmo tempo, sinaliza questões para bloqueio, garantindo a segurança e a integridade de conteúdo restrito.

A disponibilização automática de questões atualizadas para os alunos, conforme são criadas pelos professores, simplifica significativamente o processo de atualização e manutenção do banco de questões. Isso assegura que os desafios apresentados permaneçam relevantes e atualizados.

Neste contexto, este capítulo explora em detalhes como esse módulo de treinamento proporciona essa experiência tanto para os alunos quanto para os professores.

4.4.1 Personas do treinamento

A implementação do módulo de treinamento tem como propósito facilitar a resolução livre de problemas disponíveis no banco de questões do CD-MOJ. Para o seu desenvolvimento novamente foi usado o conceito de personas para melhor entender as necessidades dos usuários. São elas:

a) Persona 1: Lucas, um calouro de 19 anos na Engenharia de Software, está ansioso para aprimorar suas habilidades de resolução de problemas usando o módulo de

treinamento disponível no banco de questões do CD-MOJ. Ele procura recursos abrangentes que o auxiliem na organização, visualização, interação e motivação durante suas práticas. Além disso, deseja uma ferramenta que o ajude a identificar seus pontos fortes e áreas que necessitam de mais estudo para alcançar seus objetivos acadêmicos.

b) Persona 2: Professor Bruno Ribas, além de lecionar e dedicar-se à pesquisa, está interessado no módulo de treinamento para facilitar a interação aluno-professor em tarefas relacionadas à Engenharia de Software. Ele valoriza a eficiência nas ferramentas que utiliza tão quanto alunos proativos que buscam estudar além das aulas lecionadas. Como professor ocupado, ele precisa de procedimentos automatizados que não prejudicam a eficiência na submissão e correção de tarefas.

c) Persona 3: Desenvolvedor Caio, imerso na cultura da comunidade *open source*, está interessado no módulo de treinamento para colaborar e interagir com outros desenvolvedores. Como desenvolvedor recém formado, está em busca de bons empregos em multi-nacionais, com a crescente demanda de bons profissionais no mercado de trabalho ele busca meios de melhorar sua rotina de estudos em algoritmos e tópicos específicos para suas futuras entrevistas.

4.4.2 Histórias de usuário no treinamento

Com base nos perfis acima foram definidas as seguintes histórias de usuário:

a) Eu como Estudante Universitário e Desenvolvedor desejo uma visualização adequada das questões quanto ao seu enunciado e exemplos de input e output esperados, de forma clara e concisa, para compreender completamente os problemas.

b) Eu como Estudante Universitário e Desenvolvedor desejo ter a capacidade de submeter minhas soluções para as questões do banco de problemas do CD-MOJ, permitindo-me praticar resoluções livremente.

c) Eu como Estudante Universitário e Desenvolvedor desejo visualizar o resultado atualizado da minha submissão após enviar minha solução, para acompanhar meu progresso e identificar áreas de melhoria.

d) Eu como Estudante Universitário e Desenvolvedor desejo visualizar as tags associadas a cada questão, indicando como elas se encaixam dentro dos conceitos abordados, facilitando a seleção de problemas relacionados a temas específicos.

e) Eu como Estudante Universitário e Desenvolvedor desejo visualizar todas as questões que se encaixam dentro de um determinado conceito ou categoria, permitindo-me focar em áreas específicas de estudo de acordo com minhas necessidades e interesses.

f) Eu como Estudante Universitário e Desenvolvedor desejo visualizar todas as questões que já fiz uma submissão, permitindo-me acompanhar meu progresso e identificar áreas de melhoria.

g) Eu como professor desejo que as questões do banco de problemas sejam disponibilizados para os alunos quando eu fizer uma atualização no bando de questões.

h) Eu como professor desejo que eu possa etiquetar as questões com tags que indiquem os conceitos abordados, para facilitar a busca e seleção de problemas relacionados a temas específicos durante o planejamento de tarefas e atividades.

i) Eu como professor desejo poder sinalizar questões no banco para que elas sejam bloqueadas aos alunos para que possa manter os problemas com outras finalidades, como provas, seguros.

Sendo assim o sistema de treinamento busca oferecer recursos abrangentes para organização, visualização, interação e motivação tanto para usuários individuais quanto para administradores do sistema.

Para automatizar tarefas relacionadas à organização e atualização do sistema de treinamento, simplificando a manutenção do repositório de problemas e garantindo que as questões apresentadas estejam sempre atualizadas, o sistema gera arquivos HTML a partir do enunciado de cada problema. Esses arquivos facilitam o acesso e a navegação pelos problemas e os organiza por *tags*, tornando a busca mais eficiente.

Além disso, o sistema proporciona uma experiência mais amigável e informativa para os usuários, permitindo que eles enviem suas soluções através de um formulário. Ele também carrega e formata os dados das submissões dos usuários para o treinamento, oferecendo *feedback* sobre o desempenho deles.

Para facilitar a navegação e a busca por *tags* específicas, o sistema as organiza em uma lista com seções em ordem alfabética. Ele também oferece uma página com os problemas associados a uma única *tag*, proporcionando uma visão mais focada em um tópico específico. Além disso, inclui informações sobre outras *tags* associadas aos problemas, permitindo uma exploração mais ampla do conteúdo relacionado.

O sistema exibe as conquistas dos usuários no treinamento, oferecendo reconhecimento e motivação para o progresso alcançado. Ele calcula e exibe informações sobre o desempenho do usuário em questões específicas e consolidada de todas as submissões, permitindo uma avaliação objetiva do progresso ao longo do tempo.

De forma resumida, o modulo de treinamento se propõe em fazer:

a) Visualização - Visualização adequada das questões quanto seu enunciado e exemplos de input e output esperados;

b) Submissão - Submeter soluções para a questão;

c) Score - Visualizar o resultado atualizado da submissão;

d) Tags - Visualizar como a questão se encaixa dentro dos conceitos que são abordados;

e) Organização - Visualizar todas as questões que se encaixam dentro de um determinado conceito;

f) Análise - Visualização consolidada e individual do desempenho nas questões.

4.4.3 Usabilidade do treinamento

O sistema proporciona uma experiência amigável e informativa, seguindo as convenções de interface já familiares aos usuários do CD-MOJ. Ele organiza as informações de maneira lógica e intuitiva, facilitando a navegação e a busca por problemas em categorias específicas. Os usuários podem rapidamente aprender a enviar soluções, receber *feedback* e acompanhar seu progresso no treinamento.

Todos os dados relevantes para facilitar o entendimento, o *feedback* e a submissão de um problema foram posicionados a em uma única página. Com foco na eficiência, eficácia e utilidade (Systems. . . , 2011), bem como no critério ergonômico de carga de trabalho (Preece, 2011).

Em busca da facilidade de aprendizado e memorização (Preece, 2011), assim como dos critérios ergonômicos de consistência, significado de códigos e denominações (Cybis Adriana Betiol, 2017), a interface foi inspirada em outras plataformas conhecidas com recursos semelhantes, o SPOJ (SPOJ, 2013) e o BOCA (Campos; Ferreira, 2004), também atendendo a heurística de Correspondência (Nielsen, 1994).

Visando o Controle Explícito do usuário (Cybis Adriana Betiol, 2017), foram eliminadas as ambiguidades em relação aos componentes visuais, garantindo que ícones visuais de mesma aparência tenham sempre o mesmo funcionamento. Isso se aplica, por exemplo, às *tags*, apresentadas sempre com a mesma aparência, e quando clicadas direcionam o usuário para uma lista de problemas associados a elas. Trazendo também flexibilidade e eficiência de uso (Nielsen, 1994).

Sobre as Heurísticas de Usabilidade (Nielsen, 1994). Quanto à Visibilidade do Estado do Sistema, o menu de navegação lateral foi preservado em todas as páginas, assim como a funcionalidade das *tags* é indicada pela mudança do cursor. Para Consistência, Padronização e Reconhecimento se mantém a padronização de paginação e tabelamento de problemas, bem como a aparência e comportamento dos componentes de interface já existentes.

4.4.4 Implementação do treinamento

Para o funcionamento adequado do módulo foi necessário a implementação dos seguintes arquivos:

4.4.4.1 sync-training.sh

Este *script* é projetado para automatizar várias tarefas relacionadas à organização, manutenção e atualização do sistema de treinamento. Seu intuito é ser mantido na *crontable* (cron, 2009) para ser executado semanalmente. Uma explicação detalhada das principais seções do *script*:

a) Configuração de diretórios e variáveis: O *script* começa carregando configurações comuns do arquivo "`common.conf`" como a variável de ambiente `$CONTESTSDIR` que aponta para o diretório onde é armazenado todos os *contests* a fim de manter a estrutura de diretórios já implementada no CD-MOJ inalterada. Em seguida, ele cria a seguinte estrutura:

   - `$CONTESTSDIR/treino/controle:` Assim como em *contests* comuns, será criado uma pasta com o nome do usuário logado e arquivos com o nome dos problemas que ele já efetuou uma submissão. Esse arquivo se faz necessário para o cálculo das conquistas de usuário, pois nele podemos encontrar o número de tentativas efetuadas e se o usuário conseguiu solucionar a questão.
   - `$CONTESTSDIR/treino/data:` Assim como em *contests* comuns, sera criado dentro dessa pasta um arquivo com o nome do usuário logado que ira armazenar o status de suas submissões.
   - `$CONTESTSDIR/treino/enunciados:` Esse diretório ira armazenar os arquivos *htmls* gerados dos problemas elegíveis para o treinamento e recebidos pela *api* do CD-MOJ. Note que eles podem ser alterados por uma atualização
   - `$CONTESTSDIR/treino/submissions:` Assim como em *contests* comuns o diretório ira armazenar, para todas as submissões, o arquivo enviado pelos usuários.
   - `$CONTESTSDIR/treino/var:` Esse diretório contem os outros arquivos gerados a partir as informações buscadas da *api* e podem ser alterados por uma atualização. O arquivo "`all-tags`" armazena em texto plano todas as *tags* que foram detectadas. A pasta "`questoes`" contém diretórios nomeados a partir de cada questão disponibilizada, onde podemos encontrar três arquivos. (1) No arquivo "`li`" será encontrado os itens da tabela que irá listar os problemas na página de treinamento. (2) No arquivo "`tags`" estará todas as *tags* da questão. (3) Já no arquivo "`tl`" é armazenado o *time limit default* e específicos de cada linguagem de programação. Por fim, na pasta "`tags`" podemos encontrar arquivos nomeados por cada *tag*, seu conteúdo específica uma tabela de todos os problemas associados a *tag*.
   - `$CONTESTSDIR/treino/:` O arquivo "`conf`" é criado caso ele não exista, nele será armazenado as configurações do treinamento, como nome e ID. O arquivo "`passwd`" pode ser criado pelo *script* "`judge.sh`" quando houver uma requisição para se criar um novo usuário.

b) Atualização das questões disponíveis: Uma lista de problemas é definida através de uma requisição feita pelo *netcat* para um *host* e porta especificados. Os Nomes dos problemas são recuperados em uma lista no formato *JSON*.

   Primeiro, uma lista de todos os enunciados locais é obtida por um diretório especificado. Em seguida. Os nomes dos arquivos locais é comparado com os nomes dos problemas na lista do servidor. Se um nome de arquivo local não estiver presente na lista remota, isso indica que o enunciado local não existe no servidor, e então ele é removido.

   Após isso, para cada problema, é extraído o nome e a data de modificação do problema. Em seguida, ele transforma o nome do problema em um formato local, substituindo a barra por cerquilha, que será interpretado pelo *jq*, e então iterados um por um.

   Verifica se existe um arquivo local correspondente ao enunciado do problema. Se não existir ou se a data de modificação diferir do repositório, o *script* atualiza o enunciado local. Se o enunciado do problema já estiver atualizado localmente e não houver necessidade de atualização, será informado que não há nada a fazer para aquele problema.

   Para a atualização é feito outra solicitação ao servidor, usando o nome do problema como parâmetro. A resposta é recebida em formato *JSON*, que inclui o *HTML* do enunciado e suas *tags* codificadas em *base64*.

   O *HTML* é decodificado e, se não estiver vazio, é salvo em um arquivo local no diretório especificado. É verificado as *tags* associadas ao problema, se existirem, elas são decodificadas e armazenadas em um arquivo local. Além disso, também e requisitado os limites de tempo de execução do problema. Se existirem, são armazenados em um arquivo local.

   O processo continua para todos os problemas na lista até que todos tenham sido verificados e atualizados conforme necessário.

c) Atualização dos diretórios locais: Para a lista de problemas, o *script* começa percorrendo cada diretório encontrado dentro do diretório de questões (`$CONTESTSDIR/treino/var/questoes/*/`), verifica se existe um arquivo *HTML* de enunciado correspondente. Se não existir, o arquivo `"li"`é removido do diretório da questão para que ele não seja exibido ao usuário. Em seguida verifica se existe um arquivo de *tags* (`$CONTESTSDIR/treino/var/questoes/$questao/tags`) e então constrói uma linha *HTML* para a lista de questões. O nome da questão é usado para construir um *link*, as *tags* associadas também são incluídas como *links* com auxílio do *awk*. E então o arquivo `"li"`é subscrevido no diretório da questão (`$CONTESTSDIR/treino/var/questoes/$questao/li`).

Para a lista de problemas por *tags* é verificado se o diretório de destino para os arquivos de *tags* existe. Se existir, ele é removido para garantir uma atualização limpa. Então cada diretório de questão (`CONTESTSDIR/treino/var/questoes/*/`), que contém informações sobre cada problema e iterado. Verifica se existe um arquivo de enunciado *HTML* correspondente ao problema. Se um arquivo de *tags* existir para a questão atual, entra em uma estrutura de repetição para processá-lo. Para cada *tag* no arquivo é criada uma entrada *HTML* que inclui o nome da questão e a *tag* associada. A entrada é então adicionada ao arquivo correspondente à *tag* no diretório de *tags* (`$CONTESTSDIR/treino/var/tags`). Cada *tag* tem um arquivo dedicado onde as questões associadas a ela serão listadas.

Por ultimo o *script* procura por arquivos chamados "`tags`"em cada diretório de problema (`$CONTESTSDIR/treino/var/questoes/$questao/tags`). Ele coleta todas as *tags* encontradas de maneira única e as armazena no arquivo "`all-tags`" (`$CONTESTSDIR/treino/var/all-tags`), como já mencionado.

4.4.4.2 treino.sh

É responsável por montar a estrutura de uma página *html* que exibe informações sobre os problemas, incluindo *tags* associadas a esses problemas e mensagens personalizadas para os usuários logados e não logados. Ele usa os dados armazenados nos arquivos citados para gerar o conteúdo da página.

Para isso é armazenado na variável "`SHOW_TAGS`", utilizando do comando "awk"(Aho Brian W. Kernighan, 1988), as 10 primeiras *tags* armazenadas no arquivo "`all-tags`", ele cria um *link* que aponta para o *script* "`tag.sh`", usando o nome de cada uma das *tag* como parte do *link*.

Se o usuário estiver com uma sessão valida aberta, o *script* cria a variável "`MENSAGEM`"e armazena o *link* para *logout*. Caso contrario, instruções de como obter uma conta.

O *script* então lê o conteúdo dos arquivos "`li`" no diretório de questões (`$CONTESTSDIR/treino/var/questoes/$questao/li`), que contém a lista de problemas. E coloca suas informações na variável "`RUNNING`".

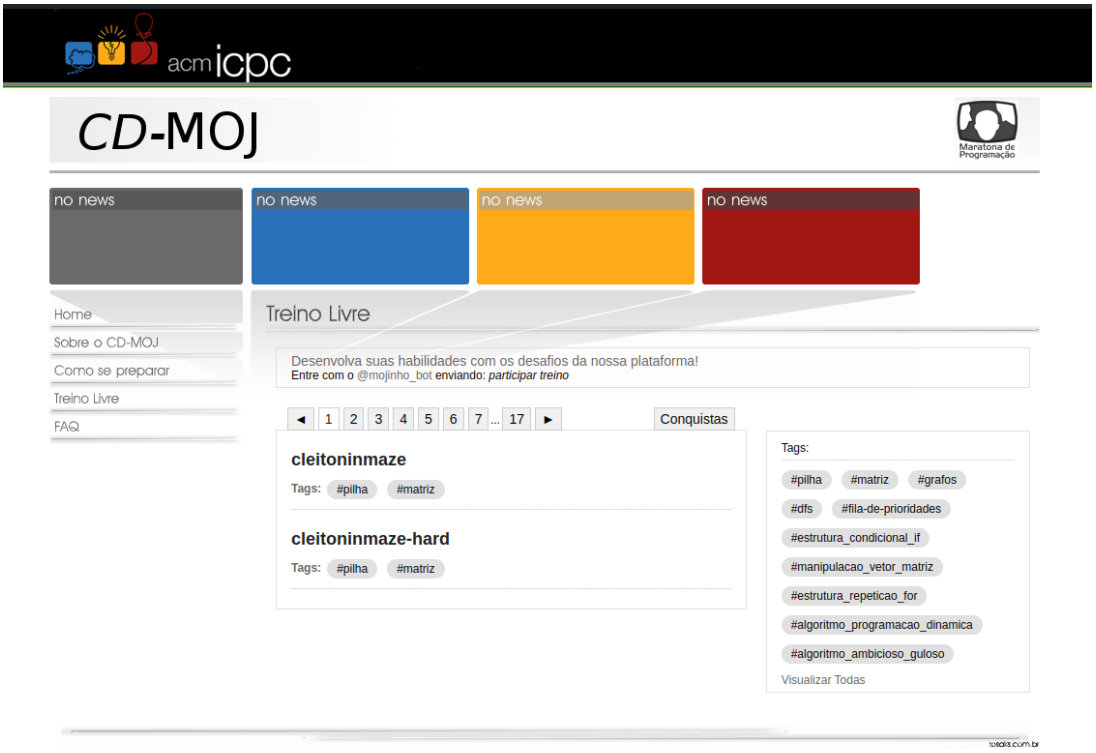
Por fim é gerado o cabeçalho da página e, em seguida, constrói o corpo da página usando as informações coletadas anteriormente, como a mensagem, a lista de problemas em execução e *links* para as páginas de *tags*.

O resultado da implementação pode ser visto na figura abaixo.

4.4.4.3 questao.sh

O *script* carrega e formata dados de submissões de usuários para uma determinada questão, exibindo-os em uma página junto com a descrição do problema. Também permite que os usuários enviem suas soluções através de um formulário.

Figura 4.2 – Página de treino



Primeiro, o *script* define a variável "`CAMINHO`"como o endereço completo para si mesmo, ou seja, o caminho para o próprio *script* chamado pelo Common Gateway Interface (CGI). Em seguida extrai o caminho relativo do *script*, removendo o caminho anterior até o arquivo "`index.sh`". Identifica a parte inicial do caminho, antes da primeira barra, que representa o nome da questão.

É feita uma verificação se o usuário está com uma sessão válida. De outra maneira é exibido a tela de login.

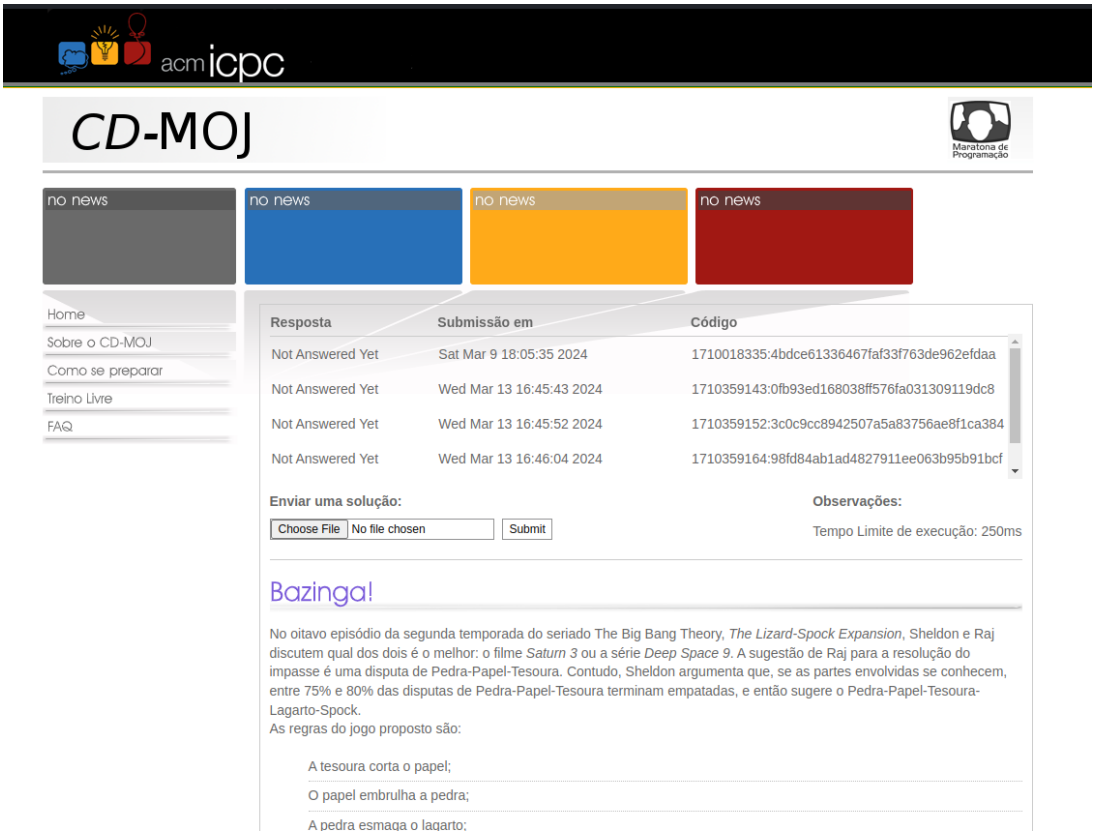
O arquivo html do problema contendo informações como enunciado, *inputs e outputs* esperados é resgatado do diretório de enunciados e é armazenado na variável "`CONTEST_HTML`".

A tabela contendo informações das submissões anteriores para o problema é gerada com o "awk"(Aho Brian W. Kernighan, 1988) utilizando os dados mantidos nos arquivos do usuário no diretório "`treino/data`". A tabela é armazenada na variável "`TABLE`". Caso não tenha sido encontrado registros do usuário para o problema, a tabela é substituída por uma mensagem.

Por fim é gerado o cabeçalho da página e, em seguida, se constrói o corpo da página usando as informações coletadas anteriormente, com a tabela, formulário e descrição do problema.

O resultado da implementação pode ser visto na figura abaixo.

Figura 4.3 – Página de questão



4.4.4.4 all-tags.sh

Tem como objetivo ler um arquivo de entrada contendo uma lista de *tags* e organizá-las em seções por ordem alfabética.

O *script* começa definindo um vetor vazio para armazenar as *tags* do arquivo de entrada. Em seguida adiciona as *tags* armazenadas no arquivo "`treino/var/all-tags`"ao vetor e são ordenadas alfabeticamente.

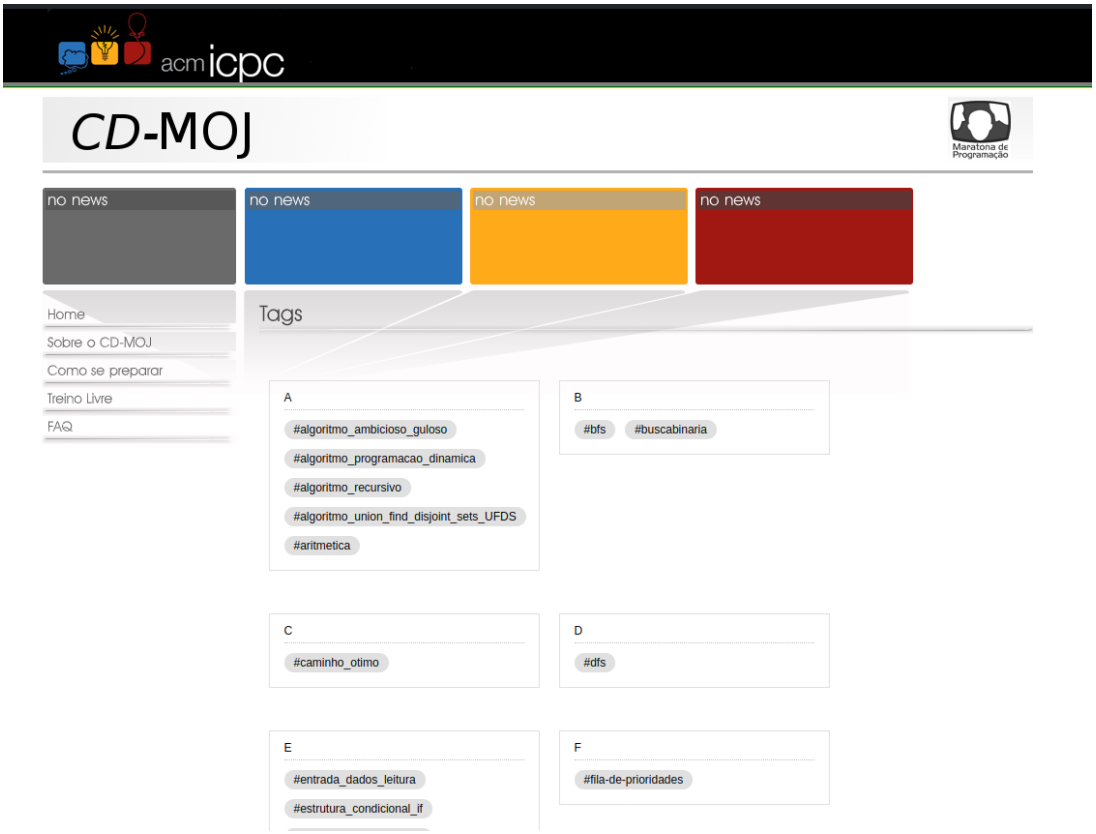
Uma *string* vazia chamada "`SECTIONS`"é definida para armazenar o *html* final das seções de *tags*.

A função "add_section()"é definida para adicionar uma nova seção. O *script* itera sobre cada *tag* no vetor e para cada uma, ele extrai a primeira letra. Quando a primeira letra mudar, isso indica o início de uma nova seção. Então a seção anterior é adicionada à *string* "`SECTIONS`".

Após iterar sobre todas as *tags*, o *script* adiciona a última seção ao *html*. Por fim, ele gera o cabeçalho e imprime o conteúdo final que inclui as seções organizadas.

O resultado da implementação pode ser visto na figura abaixo.

Figura 4.4 – Página de lista de tags

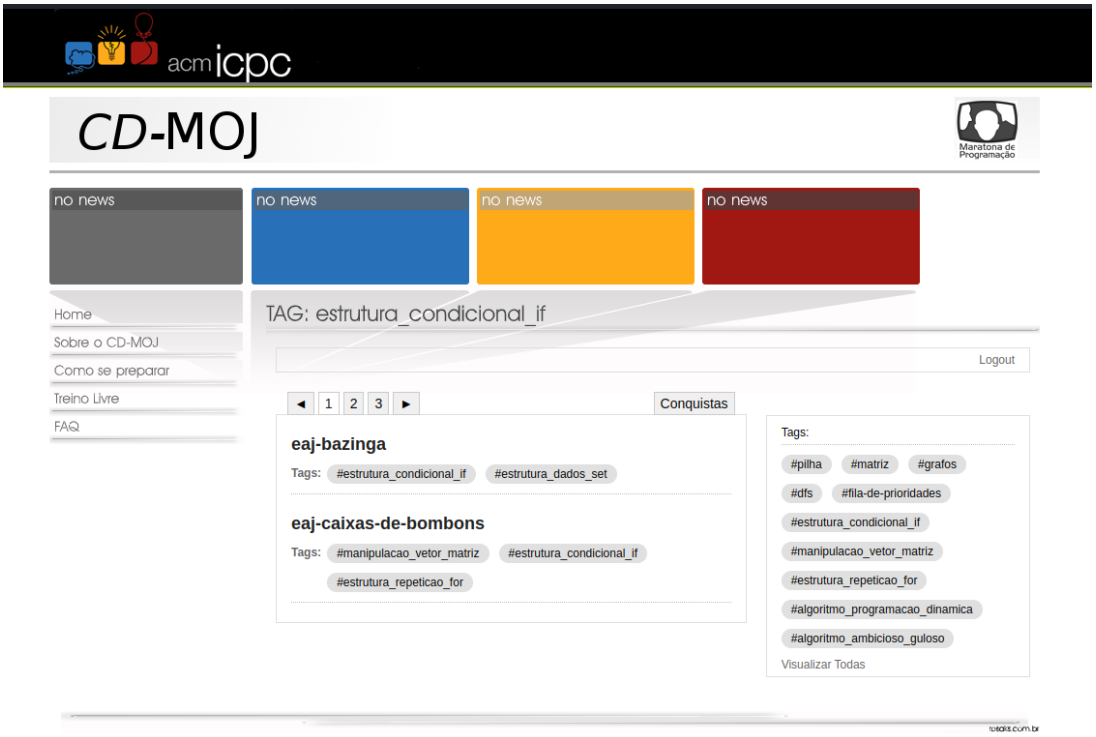


4.4.4.5 tags.sh

Possui um funcionamento análogo a página "`treino.sh`". Entretanto, é responsável por montar uma página que exibe informações sobre os problemas de uma única *tag*, incluindo as demais *tags* associadas aos problemas individuais. Ele usa os dados armazenados nos arquivos já mencionados para gerar dinamicamente o conteúdo da página.

O resultado da implementação pode ser visto na figura abaixo.

Figura 4.5 – Página de tags



4.4.4.6 conquistas.sh

É projetado para exibir as conquistas de um usuário no treinamento. Ele lê dados de conquistas de um usuário, calcula seu desempenho e exibe essas informações em uma página para visualização das conquistas divididas em uma aba de questões e de *tags*.

Inicialmente a variável "`CONQ_OPT`"armazena a segunda parte da URL fornecida, isso indica qual tela de conquista será exibida. Ela pode conter um *string* vazia ou "`tags`".

O *script* verifica se o login foi bem-sucedido para exibir a tela de conquistas do usuário. De outra forma a pagina de *login* é exibida.

O *script* verifica se existe um diretório para o usuário dentro do diretório de controle. Se esse diretório existir duas variáveis: "`ACERTOU`", para contar o número total de acertos, e "`SUBMISSOES`"para contar o número total de submissões, são inicializadas. Então sobre cada registro dentro do diretório é verificado se existe um arquivo *HTML* correspondente ao enunciado da questão no diretório de enunciados. Se existir o número de acertos (`ACERTOU`) e o número total de submissões (`SUBMISSOES`) são atualizados de acordo com as informações do registro.

Após iterar sobre todos os registros, o script calcula a relação de acertos por tentativas do usuário. O resultado calculado é então inserido na variável "`KD`", que contém *HTML*

formatado para exibir as estatísticas do usuário, incluindo seu nome de login, número de acertos, número de tentativas e a relação *"K/D"*.

A variável "TO_SHOW"é iniciada com uma *string* vazia e então é verificado se a variável "CONQ_OPT"está vazia. Caso sim, itera sobre os arquivos dentro do diretório de controle do usuário. Para cada arquivo seu nome e extraído e armazena na variável "questao". Verifica se existe um arquivo *HTML* relacionado à questão atual no diretório de enunciados. Se existirem as variáveis "JAACERTOU"e "TENTATIVAS"são extraídas do arquivo de controle. Assim, a cada iteração uma *string HTML* será gerada para a uma lista de itens, cada um representando uma questão, com links para as questões, suas *tags* associadas e estatísticas como número de acertos, número de tentativas e taxa de sucesso *"K/D"*.

Se a variável "CONQ_OPT"estiver definida como "*tags*", o *script* declara três *arrays* associativos: "tag_jaacertou_totals", "tag_tentativas_totals"e "tag_questions". Esses *arrays* são responsáveis por armazenar informações relacionadas às *tags* das questões, incluindo o número total de acertos, tentativas e as questões associadas a cada *tag* que o usuário já submeteu.

O *script* itera sobre todos os arquivos no diretório de controle do usuário logado. Para cada arquivo encontrado, verifica se há um arquivo *HTML* correspondente no diretório de enunciados. Caso exista, ele lê e atribui os valores de duas variáveis, "total_jaacertou"e "total_tentativas", que vêm do arquivo de dados do usuário.

Em seguida, o *script* verifica a existência de um arquivo de *tags* associado à questão no diretório de questões. Se o arquivo de *tags* existir, ele percorre cada linha do arquivo, que contém as *tags* da questão, e atualiza os *arrays* com os valores de acertos, tentativas e as questões associadas a cada *tag*.

Após processar todas as questões, o *script* percorre o *array* de *tags* ("tag_jaacertou _totals") e gera a saída *HTML*. Para cada *tag*, ele cria uma lista de questões associadas a ela, além de fornecer informações sobre o total de acertos, tentativas e a relação K/D ratio.

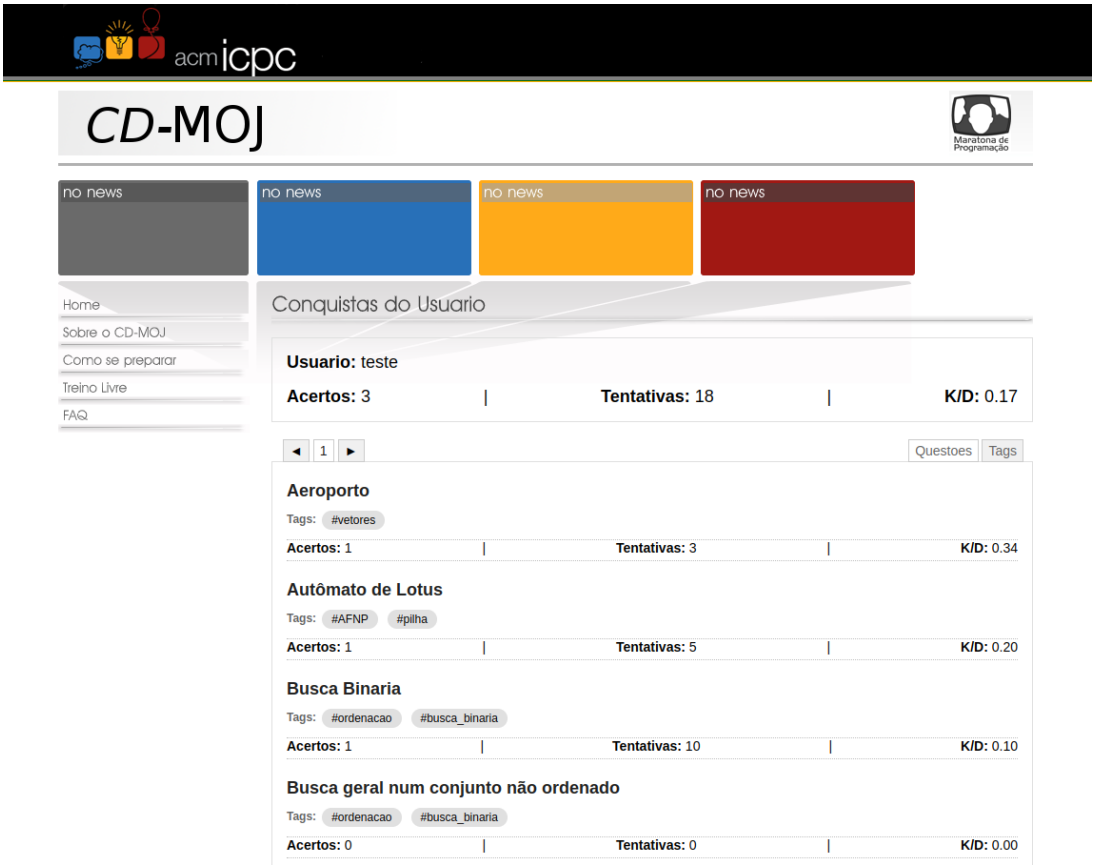
Ao término, se a variável "TO_SHOW"permanecer vazia, significa que o usuário ainda não possui conquistas registradas. Nesse caso, será informado ao usuário que ele ainda não possui conquistas.

Além disso, existe a possibilidade de um usuário *admin* ou monitor de personificar um usuário qualquer, para isso o *script* verifica se a variável "$LOGIN"contém a *substring* ".admin"ou ".mon". Se qualquer uma das condições for verdadeira, um formulário *HTML* é criado dinamicamente usando *JavaScript* e inserido no documento *HTML*. Esse formulário contém um campo de entrada para um nome de usuário.

Também é importante salientar que a página, diferente das demais, é gerada dinamicamente a cada vez que o usuário faz uma solicitação e a página que gerada anteriormente no sistema já existe a mais de cinco minutos.

O resultado da implementação pode ser visto na figura abaixo.

Figura 4.6 – Página de conquistas



4.4.4.7 Outras modificações

Nos *scripts* "`questao.sh`"e "`submete.sh`"a chamada da função "`tela-login`"declarada em "`common.sh`"esta sendo executada passando como parâmetro "`treino/$QUESTAO`"para que possamos redirecionar o usuário para a página da questão solicitada após o *login*, de outra forma ele seria redirecionado para a página "`$CONTEST`"que não existe.

Para isso, a função "`tela-login`"teve que ter seu fluxo alterado por uma condicional verificando se o *contest* possui o nome "treino", assim carregando as configurações do treino no diretório correto: "`$CONTESTSDIR/treino/conf`". A mesma alteração funciona para a página de "`conquistas.sh`", onde a função recebe o parâmetro "`treino/conquistas.usuario`".

Como consciência, outra mudança necessária foi no *script* "`login.sh`", o código primeiro ira verificar se a variável "`CONTEST`"é igual a "treino". Dentro do contexto de treinamento, o código verifica se a variável "`QUESTAO`"é igual a "`conquistas.usuario`". Isso determina se estamos lidando com a página de conquistas do usuário ou com uma questão do treinamento específica.

Com base nas condições anteriores, o código atribui um valor à variável "`HREF`", que será o destino para qual o usuário será redirecionado após um login bem-sucedido. Se estivermos na página de conquistas do usuário, o link aponta para um *script* chamado

"conquistas.sh", de outra forma, o link apontara para "`questao.sh`", com o identificador da questão na URL.

Se a variável "`CONTEST`"não for "treino", isso significa que estamos no contexto de *contests*, então o link aponta para um *script* chamado "`contest.sh`", com o identificador do *contest* na URL.

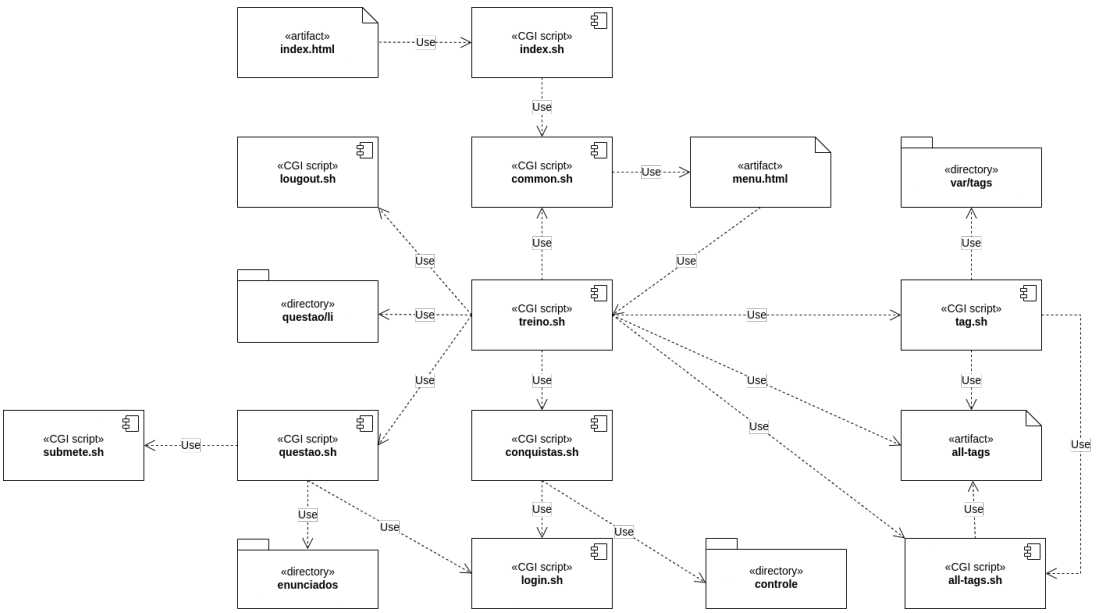
Além das mudanças acima, houve alterações nos em "`submete.sh`"e "`corrige.sh`". O *script* "`submete.sh` teve mudanças análogas a dos *scripts* de *login*, para que possamos redirecionar o usuário de volta a pagina da questão no treinamento ou do *contest* correto apos uma submissão.

No *script* "`corrige.sh`", quando o *contest* é definido como "treino", o site de correção é configurado para "cdmoj", e o ID do problema no site já está incorporado no nome do arquivo de submissão, eliminando a necessidade de percorrer as configurações do problema no arquivo "`conf`", como é feito em *contests* comuns.

Porém, ao utilizar a API do CD-MOJ, as questões de treino são recebidas no formato "repositório/questão". Para armazenar esses arquivos, foi necessário modificar essa formatação, já que os nomes de arquivos não podem conter o caractere "/". Por isso, foi substituído pelo caractere "". Ao passar esse nome por uma URL, o caractere de escape "%23"é utilizado no lugar da cerquilha. E no envio para correção, o caractere "/"é restaurado no nome.

Após as contribuições feitas através deste trabalho, foram adicionados novos recursos e também foram modificados alguns comportamentos dentro do sistema, sendo assim, para um melhor entendimento da interação entre componentes um diagrama pode ser visto na figura abaixo.

Figura 4.7 – Diagrama de Componentes



5 Conclusão

As modificações realizadas na documentação do CD-MOJ mostraram-se de extrema importância, pois organizaram o fluxo de informação em perfis de usuário. Essa alteração acaba filtrando o tempo e as informações necessárias para cada público-alvo. O aprimoramento é gradual e constante, e por esse motivo, adotou-se uma técnica de fácil manutenibilidade com a escolha da escrita em *markdown* e a ajuda do *MkDocs* (Christie, 2014) para o *deploy* e disponibilização por meio do *github pages* (Github, 2008).

O Jplag estava implementado, mas a sessão de *download* da ferramenta mostrou-se ineficaz já que era realizado diversas vezes, uma para cada *contest* criado. Agora não se faz mais os downloads excedentes da ferramenta.

A escolha entre *Makefile* e *script Bash* para instalar programas depende do contexto e dos requisitos do projeto. O *Makefile* é preferível em casos de dependências entre partes do código, regras claras de construção e permissões de sistema. O *Makefile* segue uma estrutura que inclui definição e exportação de variáveis, instalação de pacotes, alteração de *tokens* no código do repositório, instalação dos arquivos de cliente e servidor, configuração do Apache, entre outros. O objetivo foi oferecer uma abordagem clara e legível, especialmente em projetos de código-fonte aberto.

A implementação do módulo de treinamento traz uma série de benefícios significativos para estudantes e professores. Com uma visualização clara das questões, os usuários poderiam navegar pelo sistema com facilidade, entender completamente os problemas e praticar livremente. O acesso rápido e fácil a questões específicas, categorizadas por *tags* permite que os alunos foquem em áreas específicas de estudo. Os participantes podem acompanhar suas submissões e identificar áreas de melhoria, enquanto os professores podem monitorar o desempenho dos alunos e planejar atividades de acordo com as necessidades identificadas. A disponibilização automática de questões atualizadas para os alunos, conforme feitas pelo professor, simplificaria o processo de atualização e manutenção do banco de questões. Os professores tem a capacidade de etiquetar questões com *tags* que indicam conceitos abordados e sinalizar questões para bloqueio, mantendo a segurança de problemas destinados a outras finalidades.

Trabalhos futuros

Para trabalhos futuros cabe uma expansão complementar a página de *tags* e questões, permitindo que o administrador crie um novo *contest* a partir da seleção dos exercícios a partir desta página. Incluindo uma sessão de configurações com as datas de início e fim, permissões e configurações de entrada de usuário, entre outras.

Implementar um sistema de votação para *tags* na página de perguntas seria uma forma eficaz de melhorar a organização e relevância das categorizações. Permitindo que os participantes votem positivamente nas *tags* úteis e negativamente nas inadequadas, além de adicionar novas *tags*. Assim garantindo que as mais pertinentes fossem destacadas no topo da lista. Essa abordagem não só aprimoraria a navegação e busca por tópicos específicos, como também incentivaria maior engajamento dos usuários na melhoria contínua da plataforma.

Além disso, professores se beneficiariam com uma atualização e configuração adequada da ferramenta *Jplag*.

Referências

AHO BRIAN W. KERNIGHAN, P. J. W. A. V. **The AWK programming Language**. [*S.l.*]: Addison-Wesley, 1988. Citado nas pp. 16, 48 e 49.

BARBOSA, B. S. S. **Interação Humano-Computador**. [*S.l.*]: campus, 2010. Citado na p. 32.

BASTIEN, J.; SCAPIN, D. Ergonomic criteria for the evaluation of human-computer interfaces • critères ergonomiques pour l'Évaluation d'interfaces utilisateurs. **Int. J. Hum. Comput. Interaction**, v. 4, p. 183–196, 04 1992. Citado na p. 23.

BEECROWD. beecrowd. 2023. Disponível em: https://www.beecrowd.com.br/judge/en/login. Citado na p. 16.

C. LEITE J., P. R. e. B. S. S. Projeto de interfaces de usuário, perspectivas cognitivas e semióticas. 1999. Disponível em: https://www-di.inf.puc-rio.br/~clarisse/docs/JAI_Apostila1999.pdf. Citado nas pp. 20 e 21.

CAMPOS, C. P. de; FERREIRA, C. E. Boca: um sistema de apoio a competições de programação. *In*: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. **Workshop de Educação em Computação**. [*S.l.*], 2004. Citado nas pp. 14, 15 e 45.

CHRISTIE, T. Project documentation with markdown. 2014. Disponível em: https://www.mkdocs.org/. Citado nas pp. 12, 18, 30, 36, 38, 40 e 56.

COOPER, A. Persona – an overview. 2002. Disponível em: https://www.it.uu.se/edu/course/homepage/hcidist/vt05/Persona-overview.pdf. Citado nas pp. 20 e 31.

CRON. *In*: EXPERT Shell Scripting. Berkeley, CA: Apress, 2009. p. 81–85. ISBN 978-1-4302-1842-5. Disponível em: https://doi.org/10.1007/978-1-4302-1842-5_12. Citado na p. 46.

CYBIS ADRIANA BETIOL, R. F. W. **Ergonomia e Usabilidade. Conhecimentos, Métodos e Aplicações**. [*S.l.*]: Novatec, 2017. Citado nas pp. 24 e 45.

DONATH, M. mkdocs-material. 2016. Disponível em: https://github.com/squidfunk/mkdocs-material. Citado na p. 36.

ERICKSON, D. W. M. T. **HCI Remixed**. [*S.l.*]: Massachusetts Institute of Technology, 2008. Citado na p. 29.

FERREIRA SIMONE BARBOSA, T. C. B. Pathy: Using empathy with personas to design applications that meet the users' needs. 2016. Disponível em: https://link.springer.com/chapter/10.1007/978-3-319-39510-4_15. Citado nas pp. 20 e 31.

GITHUB. Sistema de controle de versão em nuvem. 2008. Disponível em: https://github.com/. Citado nas pp. 18, 30, 40 e 56.

IBM. Criando um makefile. 2021. Disponível em: https://www.ibm.com/docs/pt-br/developer-for-zos/9.1.1?topic=started-creating-makefile. Citado nas pp. 12 e 17.

JPLAG. Detector of software plagiarism and collusion. 1996. Disponível em: https://github.com/jplag. Citado nas pp. 17, 30 e 40.

MACHADO, F. **Análise e Gestão de Requisitos de Software – Onde nascem os sistemas**. Saraiva Educação S.A., 2018. ISBN 9788536509693. Disponível em: https://books.google.com.br/books?id=MYdiDwAAQBAJ. Citado na p. 26.

NIELSEN, J. **Heuristic Evaluation"**. [*S.l.*]: John Wiley Sons, 1994. Citado nas pp. 26, 34 e 45.

OLIVEIRA, F. C. **Interação Humano Computador**. [*S.l.*]: capes, 2015. Citado nas pp. 12, 19 e 29.

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. 2015. Disponível em: https://bdm.unb.br/handle/10483/11315. Citado nas pp. 11, 13 e 28.

PREECE, J. **Interaction Design: Beyond Human - Computer Interaction**. [*S.l.*]: John Wiley Sons Ltd, 2011. Citado nas pp. 21, 22, 29 e 45.

REQUISITOS Ergonômicos para Trabalho de Escritórios com Computadores Parte 11 – Orientações sobre Usabilidade. [*S.l.*], 1998. Citado na p. 22.

RIBAS, B. Contest driven meta online judge. 2013. Disponível em: https://moj.naquadah.com.br/about.shtml. Citado nas pp. 13, 14 e 37.

SILVA, L. dos S. Cd-moj: Contribuições para melhorias no sistema. 2022. Disponível em: https://bdm.unb.br/bitstream/10483/34023/1/2022_LucianoDosSantosSilva.pdf. Citado nas pp. 28 e 30.

SOUZA, G. M. de. Sistema de correção automatizada – ambiente cd-moj. 2018. Disponível em: https://eventos.utfpr.edu.br//sei/sei2018/paper/viewFile/3495/380. Citado nas pp. 28 e 38.

SPOJ. Sphere online judge. 2013. Disponível em: https://www.spoj.com/. Citado nas pp. 15 e 45.

STALLMAN, R. O projeto gnu. 2000. Disponível em: https://www.brapci.inf.br/_repositorio/2010/01/pdf_5ceeeb7f7e_0007418.pdf. Citado na p. 16.

SYSTEMS and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE). [*S.l.*], 2011. Citado nas pp. 21 e 45.

TORVALDS, L. Sistema de controle de versão distribuído. 2005. Disponível em: https://git-scm.com/. Citado na p. 18.

VALENTE, M. T. **Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade**. Editora: Independente, 2020. Disponível em: https://engsoftmoderna.info/. Citado na p. 27.

WASIK MACIEJ ANTCZAK, J. B. A. L. T. S. S. A survey on online judge systems and their applications. 2018. Disponível em: https://dl.acm.org/doi/10.1145/3143560. Citado na p. 13.