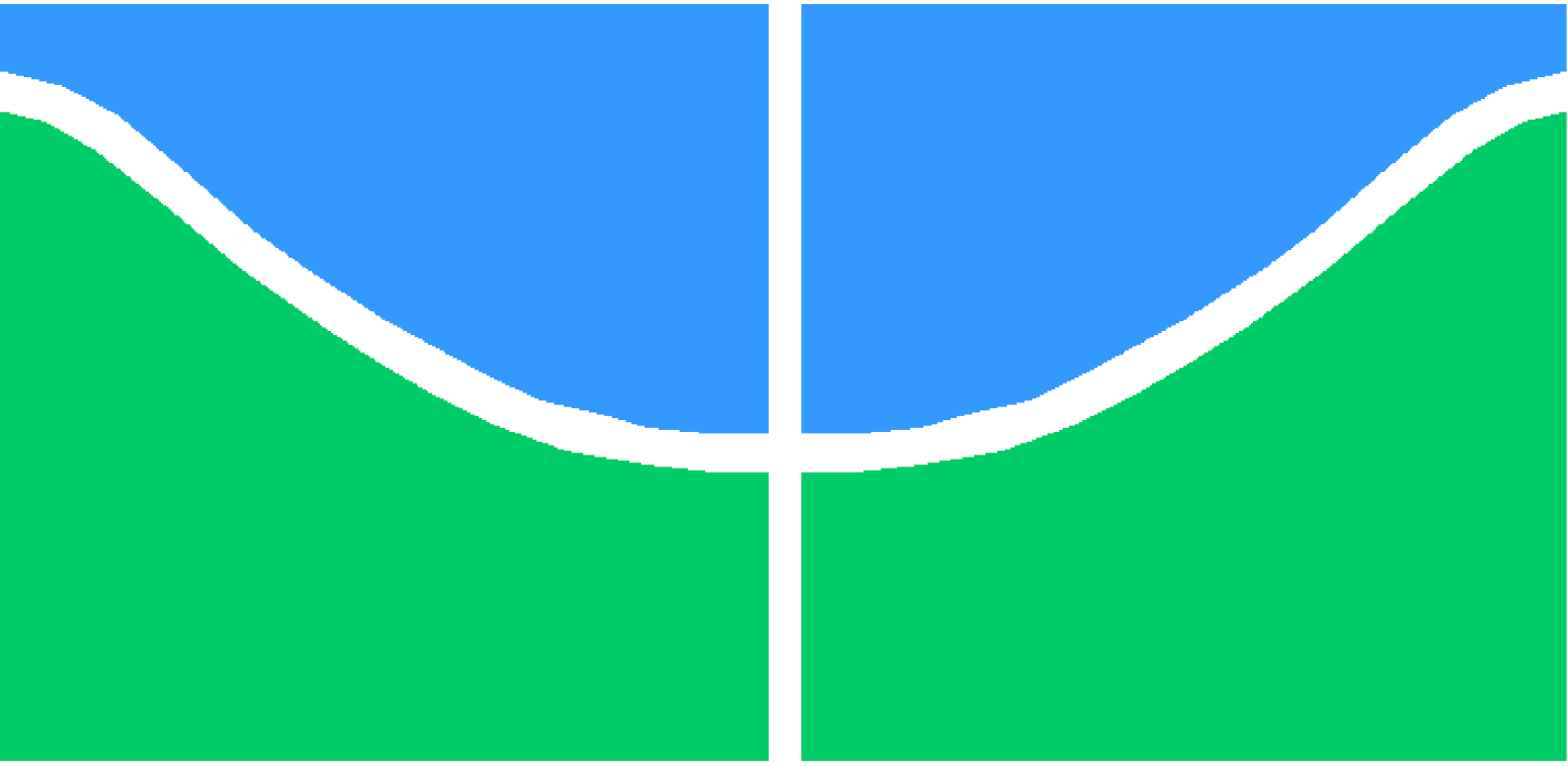

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Implementação de um Juiz Eletrônico para Aplicações Educacionais

Autores: Caio Nardelli Maranhão, Simião Correia Lima de Carvalho

Orientador: Dr. Edson Alves da Costa Junior

Brasília, DF

2016

Caio Nardelli Maranhão, Simião Correia Lima de Carvalho

Implementação de um Juiz Eletrônico para Aplicações Educacionais

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

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dr. Edson Alves da Costa Junior

Brasília, DF

2016

Caio Nardelli Maranhão, Simião Correia Lima de Carvalho
Implementação de um Juiz Eletrônico para Aplicações Educacionais/ Caio Nardelli Maranhão, Simião Correia Lima de Carvalho. – Brasília, DF, 2016-
40 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Edson Alves da Costa Junior

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2016.

1. juiz eletrônico. 2. aplicação educacional. I. Dr. Edson Alves da Costa Junior. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Implementação de um Juiz Eletrônico para Aplicações Educacionais

CDU 02:141:005.6

Caio Nardelli Maranhão, Simião Correia Lima de Carvalho

Implementação de um Juiz Eletrônico para Aplicações Educacionais

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, DF, 08 de dezembro de 2016:

Dr. Edson Alves da Costa Junior
Orientador

Dr. Fábio Macedo Mendes
Convidado 1

Dr. Fernando William Cruz
Convidado 2

Brasília, DF
2016

Dedicamos este trabalho aos que merecem.

Agradecimentos

Eu, Caio Nardelli, agradeço blablablabla.

Eu, Simião Carvalho, agradeço blablablabla.

Agradecemos aos nossos professores blablablabla.

"O que temos que ter sempre em mente é que a hegemonia do ambiente social acarreta um processo de transformação e modernização de todos os recursos funcionais envolvidos para a sociedade como um todo."

(Desconhecido)

Resumo

Texto resumo texto resumo.

Palavras-chaves: juiz eletrônico. aplicação educacional. programação competitiva.

Abstract

Text abstract text abstract.

Key-words: online judge. educacional application. competitive programming.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AC	Accepted
WA	Wrong Answer
TLE	Time Limit Exceeded
RE	Runtime Error
CE	Compilation Error
PE	Presentation Error
ACM	Association for Computing Machinery
ICPC	International Collegiate Programming Contest
SBC	Sociedade Brasileira de Computação
TLS	Transport Layer Security
SSL	Secure Sockets Layer

Sumário

1 Introdução

1.1 Contextualização

Para cada exercício dado à um estudante, existe um gabarito. Segundo Sadler (2005), "estudantes merecem ser classificados apenas com base na qualidade de seu trabalho, sem influência do desempenho de outros estudantes do curso nas mesmas tarefas ou não, e sem considerar o nível anterior de desempenho de cada estudante". A decisão de quando e como avaliar o desempenho de um estudante é extremamente relevante no que tange o desenvolvimento do conhecimento dele (CASE; SWANSON, 1998).

Conforme o número de exercícios e de estudantes cresce, a correção manual destes exercícios torna-se cada vez mais difícil, sendo propensa a erros, especialmente em um contexto de programação (CHEANG et al., 2003). Segundo Eckerdal, Thuné e Berglund (2005), "aprender programação é um jeito de pensar, que permite *problem solving*, e é experimentado como um "método" de pensamento". Esse aprendizado vem através de vários exercícios de programação, desenvolvendo raciocínio lógico e familiarizando conceitos de computação (NORONHA et al., 2015). Uma forma de automatizar a correção de exercícios seria benéfico tanto para quem os realiza quanto para quem os elabora e corrige.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é implementar um juiz eletrônico capaz de julgar vários tipos de exercícios (de múltipla escolha à exercícios de programação).

1.2.2 Objetivos Específicos

- Avaliar os diferentes tipos de exercícios possíveis.
- Levantar as aplicações de um juiz eletrônico em um contexto preferencialmente educacional.
- Analisar necessidades técnicas de um juiz eletrônico.
- Implementar um juiz eletrônico.

1.3 Organização do Trabalho

Este trabalho está organizado em Capítulos. No Capítulo 2 encontra-se o referencial teórico com os conceitos abordados neste trabalho como o juiz eletrônico e fundamentos da computação. No Capítulo 3 encontra-se o desenvolvimento deste trabalho, descrevendo os passos metodológicos utilizados. No Capítulo 4 encontra-se um relato do que foi feito até então, e considerações sobre nossos estudos e o futuro deste trabalho. No Capítulo 5 encontra-se o cronograma deste trabalho, englobando o que foi feito até agora e o que será feito.

2 Fundamentação Teórica

2.1 Juiz Eletrônico

Juízes Eletrônicos são sistemas automatizados, que compilam, executam e testam os códigos-fonte baseados em um conjunto de entradas e saídas pré-determinadas (ZHIGANG et al., 2001). Esse processo de correção automática acontece da seguinte maneira: o juiz executa o código recebido e alimenta-o com as entradas do problema, o programa processa esses dados e responde a saída da questão, por fim o juiz compara a saída do programa com a saída correta e responde com um veredito (CHAVES et al., 2013).

Para o juiz uma questão pode ser vista como um conjunto de entradas, e o que essa entrada significa; como o programa processa as entradas e como deve ser o conjunto de saídas. A interpretação da entrada geralmente não é o foco das questões, já que se pode assumir que a entrada segue as especificações da questão corretamente (KURNIA; LIM; CHEANG, 2001).

Os juízes eletrônicos tem suas respostas divididas em duas categorias: o veredito de acerto geralmente representado por AC, YES ou OK, e os de erro, que tentam dar alguma informação do que o usuário está errando. Os vereditos mais comuns para um juiz eletrônico são: AC, WA, TLE, RE, PE e CE (SKIENA; REVILLA, 2006) vide Tabela 1.

Sigla	**Significado**
AC	Accepted
WA	Wrong Answer
TLE	Time Limit Exceed
RE	Runtime Error
PE	Presentation Error
CE	Compilation Error

Tabela 1 – Vereditos Comuns de um Juiz Eletrônico

2.2 Competição de Programação

Uma frase que define bem Competição de Programação é “Dado problemas conhecidos de Ciência da Computação, resolva-os o mais rápido possível” (HALIM et al., 2010).

A maior e mais velha competição de programação é a ACM-ICPC sendo dividida em várias fases eliminatórias até a Final Mundial da ACM-ICPC, a competição chegou

a envolver 2,736 universidades de 102 países diferentes de 6 continentes. A competição promove a criatividade, o trabalho em equipe e a inovação. Fazendo com que alunos de graduação testem suas habilidades trabalhando sobre pressão em um curto espaço de tempo (ACM-ICPC, 2016).

No Brasil a Maratona de Programação é um evento organizado pela SBC desde o ano de 1996, seguindo as regras da ACM-ICPC, é uma das competições regionais que classificam as melhores equipes brasileiras para a final mundial. No caso do Brasil antes da fase regional existem as sedes sub-regionais que são organizadas pelos estados para classificar os melhores times para a etapa regional.

A dificuldade dos problemas varia, não só de acordo com a fase da prova desde o sub-regional até a fase mundial, mas em cada prova existem problemas facilmente resolvíveis até problemas complexos. Onde pelas regras da ICPC, os times são classificados pela quantidade de problemas resolvidos, não tendo pesos diferentes. O desempate é feito através de penalidades de tempo por submissão (FERRASA; SOUZA, 2012).

2.3 Sistemas Distribuídos

Existem várias definições sobre sistemas distribuídos. Segundo Silberschatz et al. (1998), "um sistema distribuído é uma coleção de nós fracamente acoplados interconectados por uma rede de comunicação". Já Tanenbaum e Steen (2007) dizem que "um sistema distribuído é uma coleção de computadores independentes que, para o usuário, são um único sistema coerente". Assim como as definições, existem opiniões diferentes sobre quais são os pilares da computação distribuída. A seguir, serão elencados alguns dos mais relevantes.

2.3.1 Recursos

É possível compartilhar recursos entre os sistemas, facilitando a acessibilidade desses recursos para usuários e outros sistemas (TANENBAUM; STEEN, 2007).

2.3.2 Transparência

A transparência de um sistema distribuído é sua capacidade de se apresentar como sendo um único sistema. Algumas das formas de transparência estão na Tabela 2.

2.3.3 Escalabilidade

(**??**) definicao de escalabilidade

Transparência	Descrição
Acesso	Esconde diferenças entre representação de dados e como um recurso é acessado
Local	Esconde onde um recurso se encontra
Migração	Esconde que um recurso pode mudar de local
Relocação	Esconde que um recurso pode ser movido de local enquanto está sendo usado
Replicação	Esconde que um recurso é replicado
Concorrência	Esconde que um recurso pode ser compartilhado por vários usuários concorrentemente
Falha	Esconde a falha e a recuperação de um recurso

Tabela 2 – Formas de transparência em sistemas distribuídos (**??**)

A escalabilidade de um sistema tem três características pelas quais pode ser analisada (**??**).

(**??**) as tres caracteristicas

2.3.4 Confiabilidade

A confiabilidade depende da capacidade do sistema se manter em funcionamento sem falhar (TANENBAUM; STEEN, 2007). Com suficiente redundância de dados, mesmo com falhas, um sistema pode continuar funcionando normalmente (SILBERSCHATZ et al., 1998).

2.3.5 Comunicação

A comunicação em sistemas distribuídos é baseada em mensagens de baixo nível trafegando na rede, devido a falta de memória compartilhada pelos 2 processos se comunicando. Para que a comunicação seja efetiva algumas coisas devem ser acordadas entre os processos como qual é o ultimo bit da mensagem, como detectar erros na mensagem e outros problemas fundamentais da comunicação (**??**).

2.4 Redes de Computadores

A comunicação de dados no contexto da computação é a troca de dados entre dois dispositivos através de algum meio de transmissão, e para que ela ocorra é necessário que os dispositivos sejam parte de um sistema de comunicação (FOROUZAN, 2006). O maior exemplo de uma rede de computadores é a Internet.

Nas seções seguintes serão descritos alguns tópicos relevantes sobre redes de computadores.

2.4.1 Camada de Transporte

Para organizar arquiteturalmente os protocolos de rede existe uma separação por camadas (KUROSE; ROSS, 2006), que pode ser visto na Tabela 3.

Aplicação
Transporte
Rede
Enlace
Física

Tabela 3 – A pilha de protocolos da Internet

Entre a camada de aplicação e a de rede, a camada de transporte tem um papel muito importante na arquitetura de redes, que é prover comunicação entre dois processos rodando em *hosts* diferentes (KUROSE; ROSS, 2006). Alguns dos protocolos da camada de transporte, como o TCP, tentam garantir que a informação vai ser trocada entre os processos de forma correta, já outros como o UDP são protocolos mais simples que abrem mão de controle para terem um desempenho maior (**??**).

2.4.2 Criptografia

Com a utilização generalizada do computador pelo cidadão comum, a criptografia assume um papel fundamental na nossa vida. Muitas das informações que trafegam na rede de computadores precisa estar disponível apenas para as pessoas envolvidas (FIARRESGA et al., 2010).

Neste trabalho, os objetivos da criptografia são:

- Confidencialidade: mantém o conteúdo da informação secreto para todos exceto para as pessoas que tenham acesso à mesma;
- Integridade da informação: assegura que não há alteração, intencional ou não, da informação por pessoas não autorizadas;
- Autenticação de informação: serve para identificar pessoas ou processos com quem se estabelece comunicação.

2.4.3 OpenSSL

OpenSSL é um projeto de código aberto que proporciona uma conjunto de ferramentas robusto para os protocolos TLS e SSL. É também uma biblioteca de criptografia para uso geral (OPENSSL, 2016). O conjunto de ferramentas OpenSSL incluem:

- libssl: provê implementações do lado do cliente e do servidor para os protocolos SSL e TLS;
- libcrypto: provê suporte à criptografias de modo geral e X.509, pré-requisitos do SSL/TLS, mas que não são parte deles.

(**??**)

3 Metodologia

3.1 Linguagem

A linguagem selecionada foi C++ pelo desempenho, pela abstração com nenhum *overhead* (STROUSTRUP, 2012) e pelo fato de ser compatível com bibliotecas e ferramentas da linguagem C (como o OpenSSL).

3.2 Maquinário

O desenvolvimento ocorre em distribuições *debian-based* do Linux, como o Linux Mint e o próprio Debian. As máquinas usadas têm suas especificações explicitadas na Tabela 4.

Máquina	**RAM**	**CPU**	**GPU**	**Disco**
I	8 GB	Intel Core i5-2500K @ 3.30 GHz	NVIDIA GeForce GTX 760	250 GB (SSD) & 1 TB (HDD)
II	8 GB	Intel Core i7-2617M @ 1.50 GHz	NVIDIA GeForce GT 540M	500 GB (HDD)
III	8 GB	Intel Core i7-3610QM @ 2.30 GHz	NVIDIA GeForce GT 630M	500 GB (HDD)

Tabela 4 – Especificação das Máquinas Utilizadas

3.3 Metodologia de Desenvolvimento

3.4 Testes

A organização do projeto está organizada de forma à integrar um *framework* de testes facilmente. O *framework* gtest (Google Test) foi usado como prova de conceito.

3.5 Arquitetura

O sistema do juiz eletrônico foi dividido em três: cliente, servidor de aplicação e servidor de correção.

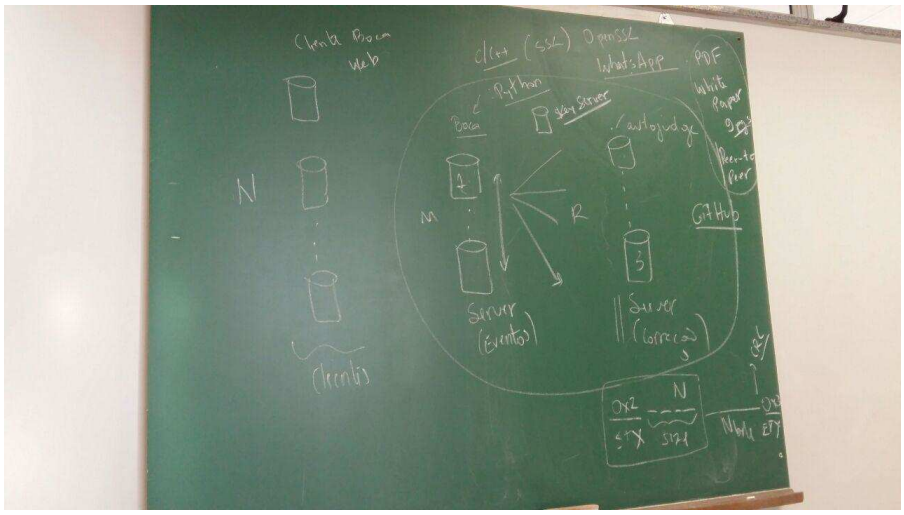

Figura 1 – Registro do Planejamento Inicial

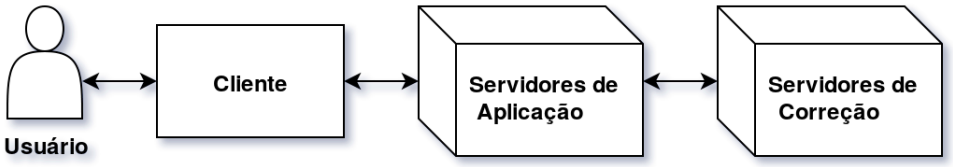

Figura 2 – Visão Lógica Arquitetural

Com a Figura 2 podemos ter uma visão lógica simplificada da arquitetura do sistema de modo geral. O usuário acessa um cliente que se comunica diretamente com os servidores de aplicação. Já estes servidores de aplicação se comunicam com os servidores de correção.

Tanto os servidores de aplicação quanto os servidores de correção são replicáveis e escaláveis, existindo N cópias distribuídas. Ao sofrerem modificações, sincronizam-nas com os outros servidores. Este comportamento é ilustrado na Figura 3

3.5.1 Cliente

O cliente é o subsistema que provê uma interface para o usuário para que o mesmo possa utilizar os serviços disponibilizados.

(**??**) parágrafo sobre como os clientes sabem com qual servidor de aplicação se comunicar

3.5.2 Servidor de Aplicação

Os servidores de aplicação são o ponto de controle do sistema do juiz eletrônico. Os comandos que podem ser executado através deles se encontram na Tabela 5.

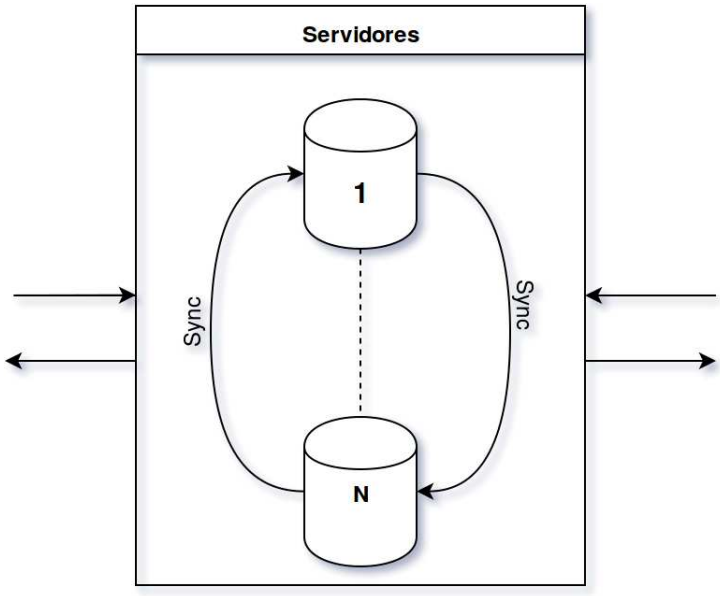

Figura 3 – Visão dos Servidores

Comando	Descrição
create	Cadastra um exercício
update	Atualiza um exercício
remove	Remove um exercício
search	Pesquisa por um exercício
submit	Submete uma resposta para um exercício
sync	Sincroniza o servidor com os outros
forward	Encaminha (**??**)
connect	Conecta (**??**)

Tabela 5 – Comandos no Servidor de Aplicação

3.5.3 Servidor de Correção

Os servidores de correção são onde os exercícios são de fato corrigidos, comparando a entrada recebida com a saída desejada.

(**??**) parágrafo detalhando a correção

4 Resultados Parciais

Até o momento estão implementados parcialmente o servidor de aplicação e o servidor de correção. Estes conversam entre si.

Usa-se o OpenSSL para realizar a comunicação, porém as mensagens não estão sendo encriptadas. O servidor de aplicação possui os comandos descritos na Tabela 5, ainda não todos implementados. É possível mandar uma mensagem qualquer para o servidor de correção e receber uma resposta de acordo.

5 Cronograma

(??)

Referências

ACM-ICPC. *About ACM-ICPC*. 2016. Disponível em: <https://icpc.baylor.edu/>. Citado na página 26.

CASE, S. M.; SWANSON, D. B. *Constructing written test questions for the basic and clinical sciences*. [S.l.]: National Board of Medical Examiners Philadelphia, PA, 1998. Citado na página 23.

CHAVES, J. O. M. et al. Integrando moodle e juízes online no apoio a atividades de programação. In: *Anais do Simpósio Brasileiro de Informática na Educação*. [S.l.: s.n.], 2013. v. 24, n. 1, p. 244. Citado na página 25.

CHEANG, B. et al. On automated grading of programming assignments in an academic institution. *Computers & Education*, Elsevier, v. 41, n. 2, p. 121–131, 2003. Citado na página 23.

ECKERDAL, A.; THUNÉ, M.; BERGLUND, A. What does it take to learn 'programming thinking'? In: ACM. *Proceedings of the first international workshop on Computing education research*. [S.l.], 2005. p. 135–142. Citado na página 23.

FERRASA, M.; SOUZA, M. Competições de raciocínio lógico e programação de computadores: um relato de experiência. *Anais do 10º CONEX–Conversando Sobre Extensão*, 2012. Citado na página 26.

FIARRESGA, V. M. C. et al. *Criptografia e matemática*. Tese (Doutorado), 2010. Citado na página 28.

FOROUZAN, A. B. *Data communications & networking*. [S.l.]: Tata McGraw-Hill Education, 2006. Citado na página 27.

HALIM, S. et al. *Competitive programming*. [S.l.]: Lulu, 2013, 2010. Citado na página 25.

KURNIA, A.; LIM, A.; CHEANG, B. Online judge. *Computers & Education*, Citeseer, v. 36, p. 299–315, 2001. Citado na página 25.

KUROSE, J. F.; ROSS, K. W. Redes de computadores e a internet. *São Paulo: Person*, 2006. Citado na página 28.

NORONHA, T. F. et al. Uma nova metodologia para o desenvolvimento de sistemas de correção automática de listas de exercıcios de programação baseada em testes de unidade. 2015. Citado na página 23.

OPENSSL. *OpenSSL*. 2016. Disponível em: <https://www.openssl.org/>. Citado na página 28.

SADLER, D. R. Interpretations of criteria-based assessment and grading in higher education. *Assessment & Evaluation in Higher Education*, Taylor & Francis, v. 30, n. 2, p. 175–194, 2005. Citado na página 23.

SILBERSCHATZ, A. et al. *Operating system concepts*. [S.l.]: Addison-Wesley Reading, 1998. v. 4. Citado 2 vezes nas páginas 26 e 27.

SKIENA, S. S.; REVILLA, M. A. *Programming challenges: The programming contest training manual*. [S.l.]: Springer Science & Business Media, 2006. Citado na página 25.

STROUSTRUP, B. Foundations of C++. In: SPRINGER. *European Symposium on Programming*. [S.l.], 2012. p. 1–25. Citado na página 31.

TANENBAUM, A. S.; STEEN, M. V. *Distributed systems*. [S.l.]: Prentice-Hall, 2007. Citado 2 vezes nas páginas 26 e 27.

ZHIGANG, S. et al. Moodle plugins for highly efficient programming courses. 2001. Citado na página 25.