UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Nathan Augusto Coelho dos Santos

Sistemas Web Backend: Uma Avaliação Comparativa entre Abordagens Síncronas e Assíncronas

Uberlândia, Brasil

2024

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Nathan Augusto Coelho dos Santos

Sistemas Web Backend: Uma Avaliação Comparativa entre Abordagens Síncronas e Assíncronas

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Orientador: Professor Thiago Henrique Pereira Silva

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2024

Nathan Augusto Coelho dos Santos

Sistemas Web Backend: Uma Avaliação Comparativa entre Abordagens Síncronas e Ássíncronas

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Professor Thiago Henrique Pereira Silva
Orientador

Professor

Professor

Uberlândia, Brasil
2024

Resumo

Palavras-chave:

Lista de ilustrações

Sumário

1 Introdução

Com o crescimento exponencial da digitalização no mundo moderno, nos mais diversos âmbitos da sociedade, cada vez mais é disseminada a criação de sistemas voltados para o contexto WEB, como por exemplo: criação de sistemas governamentais (e.g., Serviços e Informações do Brasil[1]), sistemas para instituições de ensino (MENDES, 2023), sistemas para a àrea de saúde (DOMINGUES et al., 2021) e até para divulgar e buscar vagas profissionais (SANTIAGO, 2021). O acesso massivo à internet faz com que esses sistemas tenham que lidar com cada vez maiores volumes de acessos e tenham necessidade de alta disponibilidade o tempo todo. Dessa forma, em alguns cenários para alcançar a resiliência, escalabilidade e desempenho, os sistemas passam a utilizar abordagens assíncronas (Stefan Potthast,Mike Rowe, 2007; HAMILTON et al., 2020). Entretanto, faz-se necessário entender o real volume de carga que cada sistema está propenso a receber para se decidir qual abordagem de arquitetura selecionar, assíncrona ou síncrona. Uma vez que estas abordagens possuem cenários que se adaptam melhor.

Na literatura e no mercado, conforme pode ser observado no guia de práticas recomendadas para o desempenho de microsserviços do Google Cloud (Google, 2023), há um consenso de que requisições assíncronas são recomendadas em casos de problemas de desempenho. Atualmente, com a necessidade de cada vez mais altas disponibilidades dos sistemas, problemas de performance que deixem uma aplicação irresponsiva por alguns minutos ou horas podem chegar a custar milhões de dólares para uma companhia (Atlassian, 2024). Compreender as técnicas para otimizar o desempenho de uma aplicação, a fim de torná-la altamente escalável, representa um diferencial para lidar com desafios modernos (NARDIN et al., 2020).

Existem alguns trabalhos já realizados que discutem sobre como as técnicas de programação assíncrona podem melhorar o desempenho de algumas aplicações em comparação com a programação síncrona convencional. O artigo *An Analysis of Approaches for Asynchronous Communication in Web Applications* (Loigen Sodian,Jaden Peterson Wen,Leonard Davidson,Pavel Loskot, 2022) investiga como o como o comportamento de algumas tecnologias se diferem utilizando a abordagem de comunicação assíncrona em sistemas *front-end*, além de comparar aspectos de complexidade e desempenho dessas tecnologias. Já o artigo *Concurrency and Parallelism in Speeding Up I/O and CPU-Bound Tasks in Python 3.10* (Stefan Potthast,Mike Rowe, 2007) traz resultados de como a utilização de bibliotecas assíncronas da linguagem de programação Python podem melhorar a velocidade e desempenho de aplicações utilizando um CPU multi-core.

[1] Governo Digital: <https://www.gov.br/governodigital>, acessado em 21/02/2024.

O objetivo deste trabalho é a implementação e análise de dois sistemas WEB com abordagens diferentes, síncrona e assíncrona. Pretendemos investigar e discutir a partir de quais cenários a implementação de um sistema assíncrono passa valer a pena. Para isso, criaremos ambos sistemas com a mesma funcionalidade, regra de negócio, mesma infraestrutura e variando a quantidade de carga direcionada aos sistemas. Por fim, analisaremos o comportamento de cada sistema diante as variações de carga e analisaremos o comportamento para tirarmos conclusões sobre os resultados.

A fim de desenvolver essa análise comparativa entre dois sistemas WEB iremos utilizar de algumas arquiteturas, tecnologias e ferramentas. Para o desenvolvimento do código iremos utilizar a linguagem Java [2] em sua versão 11, para o desenvolvimento da interface de comunicação REST (MASSE, 2011) e processamento de mensagens utilizaremos a biblioteca Spring-Boot [3]. Para configurar a infraestrutura em que as aplicações irão rodar utilizaremos o Docker [4]. Como banco de dados utilizaremos o Postgres [5]. Por fim, para gerenciamento de mensagens utilizaremos o RabbitMQ [6]. As aplicações irão ser desenvolvidas seguindo a arquitetura MVC.

[2] Java: <https://www.java.com/en/download/help/whatis_java.html>, acessado em 21/02/2024.
[3] Spring Boot: <https://spring.io/projects/spring-boot>, acessado em 21/02/2024.
[4] Docker: <https://www.docker.com/>, acessado em 21/02/2024.
[5] PostgreSQL: <https://www.postgresql.org/about/>, acessado em 21/02/2024.
[6] RabbitMQ: <https://www.rabbitmq.com/>, acessado em 21/02/2024.

2 Revisão Bibliográfica

2.1 Programação Assíncrona

A programação assíncrona é um paradigma de programação em que as tarefas são executadas independentemente do fluxo principal da aplicação, permitindo operações não bloqueantes e melhora de eficiência. É amplamente utilizadas em vários contextos, desde sistemas de baixo nível (e.g., Assembly) até sistemas de alto nível (e.g., Java). Neste tipo de programação, as tarefas podem ser executadas de forma concorrente sem ter que esperar que as outras completem sua execução, melhorando a perfomance e responsividade das aplicações (MAJUMDAR; THINNIYAM; ZETZSCHE, 2021).

Existem diversas formas de se implementar este tipo de programação, podemos destacar as principais sendo: *callback*, *async/await*, *Threads* e fila de mensagens. O *callback* é uma técnica de programação em que funções, conhecidas como callbacks, são passadas como argumentos para funções assíncronas, em vez de aguardar o processamento síncrono, onde a execução é bloqueada, a função assíncrona retorna imediatamente, permitindo que outras operações continuem, posteriormente, quando a operação assíncrona é concluída, os callbacks são invocados para execução, possibilitando o tratamento dos resultados (MOZILLA, 2024). A técnica *async/wait* visa simplificar a programação assíncrona evitando a inversão de controle inerente dos modelos baseados em *callback*, utilizando esse método é possível escrever códigos que se pareçam síncronos mas que se beneficiem das vantangens de performance das operações assíncronas (HALLER; MILLER, 2015). As *threads* 1 assíncronas se referem ao conceito de programação em que as *threads* operam de forma independente umas das outras, permitindo assim uma execução concorrente sem a necessidade de sincronização constante (PENG et al., 2016). Já as filas de mensagens 2 são essenciais na programação assíncrona, elas facilitam a disseminação de informação entre multiplas aplicações, otimizam a comunicação entre processos, habilita uma coordenação eficiente e troca de dados sem a necessidade de uma estrita sincronização (LI; CUI; MA, 2015).

Essa técnica de desenvolvimento está muito associada a sistemas que envolvem milhares ou até milhões de transações por minuto, uma vez que esta possibilita a execução de diversas tarefas ao mesmo tempo. Sistemas que buscam escalabilidade e resiliência certamente precisam ou precisarão em algum momento da programação assíncrona.

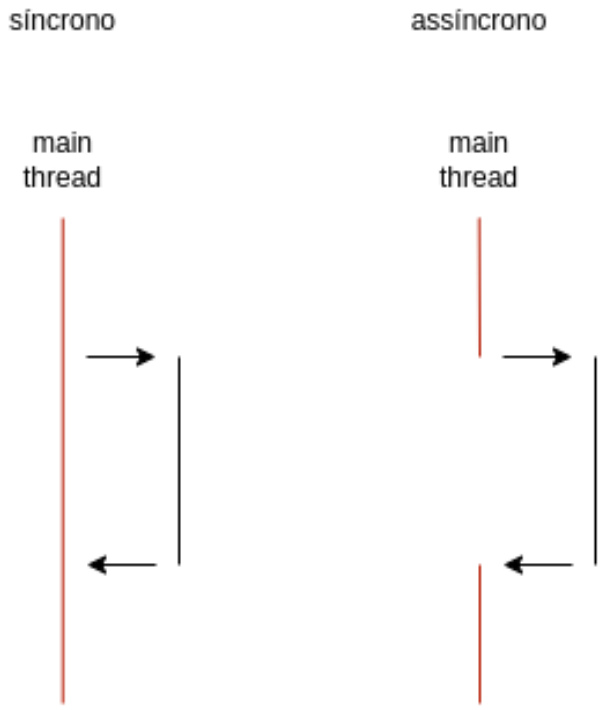

Figura 1 – Exemplo comparação entrer o processamento síncrono e assíncrono de *Threads*. Fonte: Autoria própria.

Figura 2 – Exemplo de processamento assíncrono utilizando filas de mensagens. Fonte: Autoria própria.

2.2 Performance de sistemas WEB

A performance das aplicações WEB é, atualmente, umas das principais preocupações das empresas que utilizam estes softwares em seus negócios, o monitoramento de *APM*[1] se tornou algo vital para essas organizações (SERVICES, 2023).

A performance de um sistema pode ser definida como um conjunto de métricas sendo as principais: uso de CPU, tempo de resposta, taxas de erros e número solicitações. Apartir dessa métrica que chamamos de performance é possível dizer se os serviços digitais fornecidos pelo software estão operantes, se os usuários e clientes estão enfrentando erros e identificar pontos em que a aplicação possui gargalos. Melhorar a performance de uma aplicação web pode envolver a otimização de vários elementos, como estrutura de código, tempo de resposta dos servidores, estratégias de cacheamento e latência de rede

[1] APM: Application Performance Monitoring

para garantir interações rápidas e ininterruptas dos usuários (ENGHARDT; ZINNER; FELDMANN, 2019).

O trabalho *High-Performance Web Site Design Techniques* (IYENGAR JIM CHALLENGER; DANTZIG, 2000) demonstra como que a preocupação com a perfomance de sistemas WEB é um assunto crítico quando se trata de sistemas que recebem um grande número de requisições. Este apresenta algumas técnicas que foram usadas no desenvolvimento e arquitetura para melhorar a performance do website dos jogos olímpicos de inverno de 1998 no Japão, o site mais visitado na época quebrando recordes de acessos do *Guiness Book*.

2.3 Testes de carga

Os testes de carga na programação envolvem a avaliação da performance e comportamente de um sistema sob condições específicas, como em casos de pico de acessos ou cenários de *stress* da aplicação. Esses testes são conduzidos para se ter a garantia de que um sistema pode lidar de forma eficiente com uma quantidade de carga inesperada. (ZHANG; ELBAUM; DWYER, 2011).

Esses testes possuem extrema importância no cenário de aplicações de grande empresas ou orgãos, isto porque eles possibilitam a identificação de possíveis problemas de performance antes mesmo desses problemas ocorrerem em ambientes produtivos. Além disso, descobrem problemas que outros testes como testes de unidade e de integração seriam incapazes de reproduzir, pois não se tratam de erros relacionados à regra de negócio e sim relacionados a infraestrutra ou arquitetura de código fonte. Portanto, desempenham um papel fundamental na garantia da qualidade e confiabilidade dos sistemas 3.

Segundo o artigo *A Survey on Load Testing of Large-Scale Software Systems* (JIANG; HASSAN, 2015) para garantir a qualidade de sistemas de larga escala o teste de carga é indespensável somados as técnicas convencionais de testes funcionais. Além disso, acrescenta que esses testes estão se tornando mais importantes proporcionalmente ao número de serviços sendo oferecidos na *cloud* para milhões de usuários.

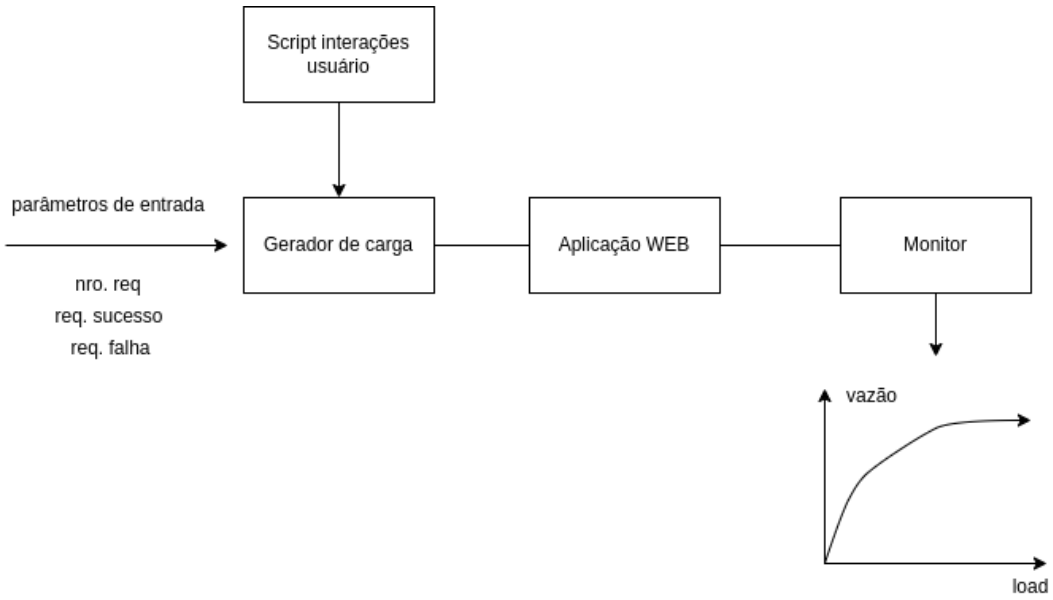

Figura 3 – Exemplo de um fluxo comum de um teste de carga. Fonte: Autoria própria.

3 Trabalhos Relacionados

É amplamente difundido na área da computação que sistemas que utilizam abordagem assíncronas podem trazer melhor manutenibilidade e impulsionar a performance quando comparados a sistemas síncronos (WANG et al., 2022). Diante disso, há diversos estudos que exploram os ganhos de desempenho ao adotar abordagens assíncronas.

Primeiramente, o trabalho *Concurrency and Parallelism in Speeding Up I/O and CPU-Bound Tasks in Python 3.10* (SODIAN et al., 2022), apresenta uma pesquisa sobre transferência e processamento de dados assincronamente visando melhorar substancialmente a eficiência e performance do sistema. A implementação utiliza a linguagem de programação *python* e explora o paralelismo e concorrência fornecidos pelos módulos da própria linguagem. O objetivo do trabalho se dá por conseguir superar a transferências e processamento de dados síncronos, os quais se mostram ineficientes para grandes volumes de dados. Utilizando de multithreading, assincronia e multiprocessing. Além disso, neste artigo as implementações foram validadas utilizando um grande número de requisições de I/O e grandes volumes de dados. O trabalho concluí que foi observado um significante melhoria de velocidade de execução quando comparado com uma execução de programação síncrona, teríamos entre 77% a 96% de redução de tempo de execução geral utilizando uma CPU multi-core. Este artigo se assemelha a abordagem do nosso trabalho, no qual iremos comparar, diferentemente do artigo, não trechos de código e sim sistemas como um todo que utilizam abordagens síncronas e assíncronas e analisar os impactos dessas abordagens na performance da aplicação web.

Já o trabalho, *An Analysis of Approaches for Asynchronous Communication in Web Applications* (POTTHAST; ROWE, 2007) analisou quatro diferentes técnicas e tecnologias para alcançar assincronicidade na comunicação web. O estudo incluiu a análise da implementação de funcionalidades, performance, estimativa de complexidade e desvantagens de cada tecnologia. A abordagem deste trabalho tinha como objetivo alcançar o seguinte resultado: o usuário ter a experiência de uma comunicação web assíncrona fluída. As tecnologias e técnicas estudadas foram: AJAX(XHR), HTML Inline Frames, Microsoft RDS e DOM3 Load and Save. Este artigo identificou que todas as abordagens, com a exceção do AJAX, não foram idealizadas e projetadas para o contexto de comunicação assíncrona. Na época as capacidades intrínsecas do AJAX para lidar com a comunicação assíncrona o destacaram em um cenários em que as demandas de comunicação assíncrona estavam em uma crescente. Diferentemente do nosso trabalho, este artigo aborda apenas a abordagem assíncrona e compara as tecnologias pesquisadas, não traz a comparação de desempenho entre síncrono e assíncrono.

Adicionalmente, o trabalho *Optimize along the way: An industrial case study on web performance* (RIET; MALAVOLTA; GHALEB, 2023) apresentou um estudo de caso em uma empresa de tecnologia agrícola, no qual visou estudar grande painel web da empresa, cujo desempenho foi melhorado por meio de 13 alterações no código fonte deste painel ao longo de um período de quatro meses. Além disso, realizou um estudo de usuário para analisar se as métricas de desempenho web se correlacionam com o tempo de carregamento percebido pelo usuário. A metodologia do trabalho foi desenvolver uma ferramenta de *benchmarking* que suportava 11 métricas de desempenho web e utilizaram uma ferramenta de *benchmarking* para avaliar o desempenho do aplicativo web e medir o efeito das 13 alterações tanto em dispositivos desktop quanto nos móveis. Foi observado nos resultados do estudo uma considerável melhoria de desempenho após as 13 intervenções, alcançaram reduções de tempo de 98,37% e 97,56% em desktop e móvel, respectivamente, para a métrica *First Contentful Paint* e de 48,25% e 19,85% para a métrica de *Speed Index*, SI. Este trabalho em comparação ao nosso, demonstra o estudo da análise de performance após intervenções no código fonte de uma aplicação, diferentemente da proposta do nosso trabalho, não utiliza de abordagens assíncronas nas sua intervenções para conseguir os ganhos de performance na aplicacação web.

Por fim, o artigo *An Analytical Approach to Performance Analysis of an Asynchronous Web Server* (PRAPHAMONTRIPONG et al., 2007) apresentou um modelo de performance da arquitetura de um Web Server Assíncrono que utiliza o padrão de design *Proactor*. Padrão, este, que encapsula os mecanismos de assíncronos disponibilizados pelo sistema operacional e pode ser usado para melhorar a performance de um Web Server. Além disso, este artigo traz que apesar de ser possível medirmos a performance do padrão após sua implementação, é possível analisar a performance do padrão ainda na sua fase de design, diante disso, ele aborda a implementação de um modelo de performance usando o paradigma de modelação chamado *Stochastic Reward Net*, SRN. Como resultado, o artigo menciona que foram representadas as características do padrão *Proactor* que são relevantes do ponto de vista de desempenho em um modelo de fila. Em seguida, apresentou uma estratégia de decomposição do modelo para permitir a aplicação do modelo em cenários práticos. O modelo de desempenho juntamente com a estratégia de decomposição do modelo foram implementados usando o paradigma de modelagem Stochastic Reward Net, SRN. Em comparação a nosso trabalho, este artigo aborda a implementação de um design pattern (*Proactor*) assíncrono para ganhos de performance em um Web Server. Entretanto, seu foco maior não está não na implementação e sim em utilizar um modelagem SRN para calcular o ganho de performance que este padrão trará para esse Web Server. Se assemelha a nosso trabalho ao trazer uma proposta implementação assíncrona para ganho de performance e se diferencia ao focar na modelagem de métricas de desempenho em tempo de design e não na implementação e resultados posteriores a ela.

4 Método

Neste capítulo abordaremos os detalhes do método que utilizado para desenvolver nossa análise comparativa entre as abordagens de programação síncrona e assíncrona. Para isso, dividimos o método em algumas etapas que nos guiarão ao longo do desenvolvimento do trabalho.

Sendo elas :

- Planejamento dos sistemas, como levantamento de requisitos e tecnologias a serem utilizadas.
- Implementação dos dois sistemas, utilizando as abordagens síncrona e assíncrona.
- Realização de testes de carga em ambos sistemas.
- Coleta e análise dos dados obtidos nos testes de carga.
- Comparação dos resultados obtidos em ambos cenários síncrono e assíncrono.
- Conclusão.

Na etapa de planejamento, iremos definir as regras de negócio, requisitos e tecnologias a serem utilizadas. Definiremos requistos funcionais que as aplicações devem atender, como por exemplo: cadastro de usuários, autenticação, cadastro de produtos, listagem de produtos, entre outros. Além disso, definiremos as tecnologias a serem utilizada para o desenvolvimento das aplicações e porque de cada escolha.

Na etapa de desenvolvimento, iremos implementar as duas aplicações, uma utilizando a abordagem síncrona e outra a abordagem assíncrona. Ambas aplicações terão a mesma regra de negócio, infraestrutura e tecnologias, diferindo apenas na abordagem de programação. Neste ponto explicaremos como implementamos a abordagem assíncrona e síncrona, como configuramos a infraestrutura e como foi a implementação da regra de negócio.

Na etapa de testes de carga, iremos realizar uma bateria de testes de carga em ambos sistemas, variando a quantidade de carga direcionada a eles. Iremos utilizar ferramentas de teste de carga para simular o comportamento de usuários acessando as aplicações e coletar métricas de desempenho.

Na etapa de coleta e análise dos dados, iremos coletar os dados obtidos nos testes de carga e analisar os resultados obtidos. Iremos analisar o comportamento de cada sistema diante as variações de carga e tirar conclusões sobre os resultados obtidos.

Por fim, na etapa de comparação dos resultados, iremos comparar os resultados obtidos em ambos cenários, síncrono e assíncrono. Iremos analisar os resultados obtidos e tirar conclusões sobre qual abordagem se saiu melhor em cada cenário.

5 Conclusão

Referências

Atlassian. **Calculating the cost of downtime**. 2024. <https://www.atlassian.com/incident-management/kpis/cost-of-downtime>. [Online; accessed 13-Fevereiro-2024]. Citado na página 6.

DOMINGUES, M. C. M.; MELO, K. M.; MIRANDA, A. F.; CAYRES, A. Z. de F.; ELIAS, R. Programa autoestima: Uma ferramenta web pública e amigável que integra formação de profissionais da saúde e acolhimento psicossocial da população. In: SBC. **Anais do IX Workshop de Computação Aplicada em Governo Eletrônico**. [S.l.], 2021. p. 251–258. Citado na página 6.

ENGHARDT, T.; ZINNER, T.; FELDMANN, A. Web performance pitfalls. **Passive and Active Measurement**, p. 286–303, 2019. Citado na página 10.

Google. **Práticas recomendadas para desempenho**. 2023. <https://cloud.google.com/appengine/docs/legacy/standard/java/microservice-performance?hl=pt-br#use_asynchronous_requests>. Citado na página 6.

HALLER, P.; MILLER, H. B. A formal model for direct-style asynchronous observables. 2015. Citado na página 8.

HAMILTON, M.; GONSALVES, N.; LEE, C.; RAMAN, A.; WALSH, B.; PRASAD, S.; BANDA, D.; ZHANG, L.; ZHANG, L.; FREEMAN, W. T. Large-Scale Intelligent Microservices. In: **2020 IEEE International Conference on Big Data (Big Data)**. [S.l.: s.n.], 2020. p. 298–309. Citado na página 6.

IYENGAR JIM CHALLENGER, D. D. A.; DANTZIG, P. High-performance web site design techniques. **IEEE Internet Computing**, IEEE, v. 4, n. 2, 2000. Citado na página 10.

JIANG, Z. M.; HASSAN, A. E. A survey on load testing of large-scale software systems. **Information Management & Computer Security**, IEEE, v. 18, n. 11, 2015. Citado na página 10.

LI, J.; CUI, Y.; MA, Y. Modeling message queueing services with reliability guarantee in cloud computing environment using colored petri nets. **Mathematical Problems in Engineering**, v. 2015, p. 1–20, 2015. Citado na página 8.

Loigen Sodian,Jaden Peterson Wen,Leonard Davidson,Pavel Loskot. **Concurrency and Parallelism in Speeding Up I/O and CPU-Bound Tasks in Python 3.10**. 2022. 2nd International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI). Citado na página 6.

MAJUMDAR, R.; THINNIYAM, R. S.; ZETZSCHE, G. General decidability results for asynchronous shared-memory programs: higher-order and beyond. 2021. Citado na página 8.

MASSE, M. **REST API design rulebook: designing consistent RESTful web service interfaces**. [S.l.]: "O'Reilly Media, Inc.", 2011. Citado na página 7.

MENDES, F. C. **Sistema de acompanhamento de egressos UFU: levantamento e análise de requisitos**. 2023. Disponível em: <https://repositorio.ufu.br/bitstream/123456789/39896/1/SistemaAcompanhamentoEgressos.pdf>. Citado na página 6.

MOZILLA. **Introducing asynchronous JavaScript**. 2024. <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Introducing>. [Online; accessed 18-Março-2024]. Citado na página 8.

NARDIN, I. F. D.; LOPES, T. R. L.; COSTA, C. A. D.; OLIVEIRA, K. S. F. D.; BARBOSA, J. L. V.; RIGHI, R. D. R. Looking at performance metrics and scalability challenges in the context of microservices: a survey. **International Journal of High Performance Computing and Networking**, Inderscience Publishers (IEL), v. 16, n. 4, p. 221–251, 2020. Citado na página 6.

PENG, Z.; XU, Y.; YAN, M.; YIN, W. Arock: an algorithmic framework for asynchronous parallel coordinate updates. **SIAM Journal on Scientific Computing**, v. 38, p. A2851–A2879, 2016. Citado na página 8.

POTTHAST, S.; ROWE, M. An analysis of approaches for asynchronous communication in web applications. In: **40th Annual Midwest Instruction and Computing Symposium**. [S.l.: s.n.], 2007. Citado na página 12.

PRAPHAMONTRIPONG, U.; GOKHALE, S.; GOKHALE, A.; GRAY, J. An analytical approach to performance analysis of an asynchronous web server. **Simulation**, Sage Publications Sage UK: London, England, v. 83, n. 8, p. 571–586, 2007. Citado na página 13.

RIET, J. van; MALAVOLTA, I.; GHALEB, T. A. Optimize along the way: An industrial case study on web performance. **Journal of Systems and Software**, Elsevier, v. 198, p. 111593, 2023. Citado na página 13.

SANTIAGO, G. M. de S. **Severino: uma aplicação web para encontrar profissionais**. 2021. Disponível em: <https://repositorio.ufu.br/handle/123456789/32449>. Citado na página 6.

SERVICES, A. W. **O que é APM (monitoramento de desempenho de aplicativos)**. 2023. Disponível em: <https://aws.amazon.com/pt/what-is/application-performance-monitoring/>. Citado na página 9.

SODIAN, L.; WEN, J. P.; DAVIDSON, L.; LOSKOT, P. Concurrency and parallelism in speeding up i/o and cpu-bound tasks in python 3.10. In: IEEE. **2022 2nd International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI)**. [S.l.], 2022. p. 560–564. Citado na página 12.

Stefan Potthast,Mike Rowe. **An Analysis of Approaches for Asynchronous Communication in Web Applications**. 2007. The Midwest Instruction and Computing Symposium. Citado na página 6.

WANG, X.; ZHANG, J.; MAN, L.; HOU, D.; SONG, K. Performance analysis and evaluation of ternary optical computer based on asynchronous multiple vacations. **Soft Computing**, v. 27, p. 4107–4123, 2022. Citado na página 12.

ZHANG, P.; ELBAUM, S.; DWYER, M. B. Automatic generation of load tests. **2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)**, 2011. Citado na página 10.