UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LUIZ ANTONIO SCHONS

ANÁLISE COMPARATIVA DE ARQUITETURAS DE SOFTWARE PARA APLICAÇÕES WEB: DESEMPENHO E ESCALABILIDADE E EXPERIÊNCIA DE DESENVOLVIMENTO

GUARAPUAVA

2024

LUIZ ANTONIO SCHONS

ANÁLISE COMPARATIVA DE ARQUITETURAS DE SOFTWARE PARA APLICAÇÕES WEB: DESEMPENHO E ESCALABILIDADE E EXPERIÊNCIA DE DESENVOLVIMENTO

Comparative Analysis of Software Architectures for Web Applications: Performance, Scalability, and Development Experience

Projeto de Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Sistemas para Internet do Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador: Andres Jesse Porfirio

GUARAPUAVA

2024

SUMÁRIO

1 INTRODUÇÃO

1.1 Considerações iniciais

Desenvolver software é uma tarefa complexa e desafiadora e, nos dias atuais, com a evolução da tecnologia e a crescente demanda por aplicações web, a escolha da arquitetura de software é um fator crítico para o sucesso de um projeto. A arquitetura de software é um conjunto de padrões e práticas que definem a estrutura de um sistema de software, incluindo os componentes, as relações entre eles e as diretrizes para o seu desenvolvimento e evolução. Existem várias arquiteturas de software disponíveis para aplicações web, cada uma com suas próprias características, vantagens e desvantagens e muitas vezes é responsabilidade do arquiteto de software escolher a arquitetura mais adequada para um determinado projeto.

Fazer a escolha certa da arquitetura de software pode ter um impacto significativo no desempenho e escalabilidade de uma aplicação web, bem como na experiência de desenvolvimento dos desenvolvedores. Uma arquitetura bem projetada pode facilitar a manutenção, a evolução e a escalabilidade de um sistema de software, enquanto uma arquitetura mal projetada pode resultar em problemas de desempenho, escalabilidade e manutenção (FORD *et al.*, 2021).

A experiência de desenvolvimento é medida pela facilidade com que os desenvolvedores podem entender, modificar e estender o código de um sistema de software. Isso está atrelado a várias pontos que são mensuráveis tempo de build (tempo que o sistema leva para compilar), velocidade de execução da pipeline de CI/CD [1], tempo e facilidade de deploy [2].

Outro fator importante a ser considerado é a escalabilidade de uma aplicação web, que se refere à capacidade de um sistema de software lidar com um aumento na carga de trabalho, como o número de usuários, transações ou dados. Uma arquitetura escalável é aquela que pode crescer e se adaptar às demandas de um sistema de software em evolução, sem comprometer o desempenho ou a disponibilidade.

O mais crucial é que a escolha de uma arquitetura de software deve ser baseada em performance, que é medida de diversas formas, como tempo de resposta, tempo de carregamento de páginas, uso de recursos do sistema, entre outros. A performance de uma aplicação web é um fator crítico para a satisfação do usuário e o sucesso de um projeto, e uma arquitetura de software bem projetada pode contribuir significativamente para a performance de uma aplicação web.

---

[1] CI/CD cria um pipeline automatizado, desde o código-fonte até a produção, permitindo que as equipes entreguem software com mais rapidez, frequência e confiabilidade, saiba mais em https://martinfowler.com/articles/continuousIntegration.html e https://martinfowler.com/books/continuousDelivery.html

[2] Deploy é o processo de disponibilizar um software ou aplicação para uso, movendo-o de um ambiente de desenvolvimento para um ambiente de produção acessível aos usuários., saiba mais em https://www.ibm.com/docs/en/zos/3.1.0?topic=task-deploying-software

No entanto, a escolha da arquitetura de software certa para uma aplicação web não é uma tarefa fácil, pois envolve a consideração de vários fatores, como os requisitos do projeto, as restrições de tempo e orçamento, as habilidades da equipe de desenvolvimento e as tendências tecnológicas (RICHARDS; FORD, 2020). Além disso, as arquiteturas de software estão em constante evolução, com novas abordagens e tecnologias sendo desenvolvidas regularmente, o que torna ainda mais desafiador escolher a arquitetura certa para um projeto.

Neste trabalho é proposto analisar e comparar algumas das arquiteturas de software mais populares para aplicações web, incluindo a arquitetura monolítica, a arquitetura cliente-servidor e a arquitetura serverless, com foco no desempenho e escalabilidade e na experiência de desenvolvimento dos desenvolvedores, para ajudar os arquitetos de software a tomar decisões informadas sobre a escolha da arquitetura de software mais adequada para seus projetos.

A análise será feita com base em um estudo de caso, onde será usado o Open Social Care [3] como aplicação web de exemplo e implementado cada uma das arquiteturas de software em um ambiente de teste e realizar testes em cada um dos setores mencionados acima.

1.2 Fundamentação Teórica

Esta seção descreve as arquiteturas de software analisadas neste trabalho, sendo: monolítica, cliente-servidor e serverless.

1.2.1 Monolítica

De acordo com (FOWLER, 2015), na arquitetura monolítica todo o código-fonte de uma aplicação web é desenvolvido e implantado como um único monólito, que consiste em vários módulos ou componentes que são interdependentes e compartilham o mesmo repositório/pasta no servidor. A arquitetura monolítica é caracterizada por ser tudo em um lugar só, tanto a parte viusal quanto a parte lógica. A Figura 1 ilustra essa arquitetura.

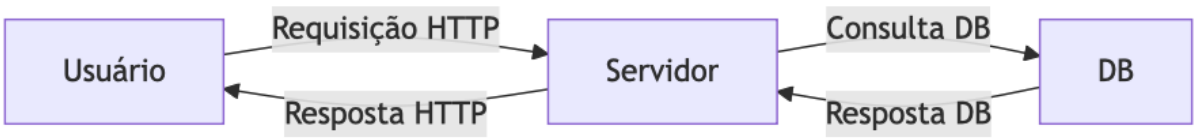

Figura 1 – Arquitetura Monolitica

A arquitetura monolítica é composta por três camadas principais: a camada de apresentação, a camada de lógica de negócios e a camada de acesso a dados. A camada de apresentação é responsável por interagir com os usuários, exibindo a interface do usuário e coletando entradas do usuário. A camada de lógica de negócios é responsável por implementar a lógica

[3] Sistema Open Social Care, disponível em https://github.com/open-social-care

de negócios da aplicação web, processando as entradas do usuário e gerando as saídas correspondentes. A camada de acesso a dados é responsável por interagir com o banco de dados, recuperando e armazenando dados para a aplicação web.

Essa divisão de camadas não é uma regra, mas sim uma prática comum na arquitetura monolítica, que pode variar dependendo das necessidades e requisitos de um projeto específico. A arquitetura monolítica é caracterizada por sua simplicidade, facilidade de desenvolvimento e implantação, e baixa complexidade, o que a torna uma escolha popular para pequenas e médias aplicações web.

1.2.2 Cliente-Servidor

Nesta arquitetura, o sistema de software é dividido em duas partes principais: o cliente, que é responsável por interagir com os usuários e exibir a interface do usuário, e o servidor, que é responsável por processar as solicitações dos clientes, executar a lógica de negócios e acessar os dados. A Figura 2 ilustra essa arquitetura.

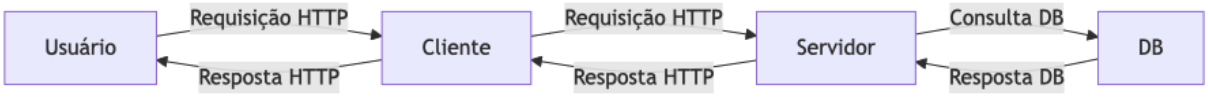

Figura 2 – Arquitetura Cliente Servidor

A arquitetura cliente-servidor é composta por duas camadas principais: a camada do cliente e a camada do servidor. A camada do cliente é responsável por interagir com os usuários, exibindo a interface do usuário e coletando entradas do usuário. A camada do servidor é responsável por processar as solicitações dos clientes, executar a lógica de negócios e acessar os dados.

1.2.3 Serverless

Segundo (FOWLER, 2018), na arquitetura serverless o sistema de software é dividido em várias funções independentes e autônomas, que são executadas em um ambiente de execução gerenciado por um provedor de serviços de nuvem. Cada função é responsável por executar uma tarefa específica, como processar uma solicitação HTTP, acessar um banco de dados ou enviar um e-mail. A Figura 3 ilustra essa arquitetura.

Uma API Gateway é uma camada intermediária que recebe as solicitações HTTP dos usuários e as encaminha para as funções apropriadas. Cada função é executada em um ambiente de execução isolado e é dimensionada automaticamente de acordo com a demanda.

Figura 3 – Arquitetura Serverless

A arquitetura Serverless está muito relacionada ao ambiente de Cloud [4] e por isso é algo mais recente, segundo (FOWLER, 2018).

1.3 Objetivos

O objetivo deste trabalho é analisar e comparar as arquiteturas de software monolítica, cliente-servidor e serverless para aplicações web, com foco no desempenho e escalabilidade e na experiência de desenvolvimento dos desenvolvedores.

1.3.1 Objetivo específicos

- Implementar cada uma das arquiteturas de software em um ambiente de teste;
- Realizar testes de desempenho e escalabilidade em cada uma das arquiteturas;
- Avaliar a experiência de desenvolvimento dos desenvolvedores em cada uma das arquiteturas;
- Comparar os resultados dos testes e avaliações para determinar as vantagens e desvantagens de cada arquitetura.

1.4 Justificativa

A escolha da arquitetura de software certa para uma aplicação web é um fator crítico para o sucesso de um projeto. Uma arquitetura bem projetada pode facilitar a manutenção, a evolução e a escalabilidade de um sistema de software, enquanto uma arquitetura mal projetada pode resultar em problemas de desempenho, escalabilidade e manutenção. Além disso, a experiência de desenvolvimento dos desenvolvedores é um fator importante a ser considerado, pois pode afetar a produtividade e a satisfação da equipe de desenvolvimento. Esse trabalho tem como objetivo ajudar os arquitetos de software a tomar decisões informadas sobre a escolha da

[4] A "nuvem"(cloud) refere-se à entrega sob demanda de recursos de computação—como servidores, armazenamento, bancos de dados, redes, software, análises e inteligência—pela Internet ("a nuvem"). As empresas que oferecem esses recursos de computação são chamadas de provedores de nuvem e normalmente cobram pelos serviços de nuvem com base no uso, semelhante à forma como você pagaria pela sua conta de água ou luz em casa.

arquitetura de software mais adequada para seus projetos, com base em testes de desempenho e escalabilidade e avaliações da experiência de desenvolvimento dos desenvolvedores.

2 TRABALHOS RELACIONADOS

2.1 Open Social Care

O Open Social Care é um sistema para gerenciamento de atendimentos sociais, ele foi criado para auxiliar o dia a dia de trabalho de assistentes sociais do CCG (Conselho da Comunidade de Guarapuava). O sistema surgiu como um projeto de disciplina de Sistemas Distribuídos da UTFPR, câmpus Guarapuava, quando uma assistente social do CCG entrou em contato com o professor da disciplina e sugeriu a ideia de informatização do processo até então utilizado no CCG. Através de reuniões e discussões, foi feita a coleta dos requisitos e planejado como o sistema teria que funcionar. Por fim, foi desenvolvido um protótipo como um objeto de estudos na disciplina. Com o passar do tempo, houve a necessidade de uma evolução no projeto, pois ele iniciou sendo Serverless usando o Firebase [1] e com o (SOUZA; PORFIRIO, 2024) e (BOEIRA; PORFIRIO, 2024) foi migrado para outras tecnologias e para a arquitetura Cliente-Servidor.

2.2 Comparativos

As discussões sobre arquitetura de software não são novas, diversos autores já fizeram algum tipo de comparativos como por exemplo (JIANG; PEI; ZHAO, 2020) que compara de maneira mais organizacional as arquiteturas Serverless e Micro Serviços.

Além disso, existem vários trabalhos que analisam e comparam diferentes arquiteturas de software para aplicações web, como (GOS; ZABIEROWSKI, 2020) e (MOSLEH; DALILI; HEYDARI, 2016), ou uma analise sobre Trade-offs [2] das arquiteturas monolito e distribuído como é o caso do (FELISBERTO, 2024).

[1] Firebase é uma plataforma de desenvolvimento de aplicativos móveis e web que oferece um conjunto de ferramentas e serviços para ajudar os desenvolvedores a criar aplicativos de alta qualidade, expandir seus negócios e gerenciar tudo com mais facilidade. Ele oferece recursos como banco de dados em tempo real, autenticação, armazenamento em nuvem, hospedagem, funções em nuvem (serverless), analytics, mensagens push, configurações remotas e muito mais, simplificando muitas tarefas complexas no desenvolvimento de apps. Essencialmente, ele atua como um back-end completo, permitindo que os desenvolvedores se concentrem na experiência do usuário.

[2] Trade-off é uma situação que envolve a perda de um aspecto ou qualidade de algo em troca de ganhos em outros aspectos ou qualidades. Em outras palavras, é uma troca, uma concessão, onde você abre mão de alguma coisa para obter outra. Frequentemente, envolve equilibrar prioridades conflitantes e escolher a melhor opção possível considerando as vantagens e desvantagens de cada alternativa.

3 CONCLUSÃO

A arquitetura de software é um fator crítico para o sucesso de um projeto de desenvolvimento de software, e a escolha da arquitetura certa pode ter um impacto significativo no desempenho e escalabilidade de uma aplicação web, bem como na experiência de desenvolvimento dos desenvolvedores.

Esta pesquisa busca comparar as arquiteturas monolítica, cliente-servidor e serverless para aplicações web, focando no desempenho, escalabilidade e experiência de desenvolvimento. A partir da análise de trabalhos relacionados, como o caso do Open Social Care, que migrou de uma arquitetura serverless com Firebase para uma cliente-servidor, e estudos comparativos como os de (JIANG; PEI; ZHAO, 2020), (GOS; ZABIEROWSKI, 2020), (MOSLEH; DALILI; HEYDARI, 2016) e (FELISBERTO, 2024) sobre trade-offs entre arquiteturas monolíticas e distribuídas, espera-se contribuir para a tomada de decisão na escolha da arquitetura mais adequada para diferentes projetos.

Além de fornecer métricas de desempenho e escalabilidade para cada arquitetura, este trabalho pretende analisar as implicações da migração entre elas, exemplificada pela experiência do Open Social Care. Espera-se identificar os principais benefícios e desafios de cada abordagem, considerando o contexto de desenvolvimento e as necessidades específicas de cada projeto. A análise dos resultados permitirá uma compreensão mais aprofundada das vantagens e desvantagens de cada arquitetura, fornecendo insights sobre como o mercado está se comportando em relação à adoção dessas arquiteturas e como as ferramentas de benchmarking [1] podem auxiliar na escolha da melhor solução. Finalmente, pretende-se oferecer um guia prático para arquitetos de software, auxiliando-os a navegar pelas complexidades da escolha arquitetural e a selecionar a opção que melhor atende aos requisitos de seus projetos.

[1] Benchmarking é o processo de comparar o desempenho de algo com um padrão ou ponto de referência. Ele pode ser usado para comparar produtos, serviços, processos e até mesmo empresas inteiras. O objetivo é identificar áreas de melhoria e melhores práticas para alcançar ou superar a concorrência. Em resumo, é medir e comparar para aprimorar.

REFERÊNCIAS

BOEIRA, S. L. dos S.; PORFIRIO, A. J. **Implementação e Atualização de Frontend para o Sistema Open Social Care**. 2024. 53 f. Monografia (Trabalho de Conclusão de Curso) — Universidade Tecnológica Federal do Paraná, Guarapuava, 2024. Disponível em: https://tcc.tsi.pro.br/uploads/academic_activity/pdf/267/GP_COINT_2024_1_CAMILA_EMANUELE_DE_SOUZA_MONOGRAFIA.pdf.

FELISBERTO, M. The trade-offs between monolithic vs. distributed architectures. **arXiv preprint arXiv:2405.03619**, 2024.

FORD, N. *et al.* **Software Architecture: The Hard Parts : Modern Trade-off Analysis for Distributed Architectures**. O'Reilly, 2021. ISBN 9781492086895. Disponível em: https://books.google.com.br/books?id=sWNozgEACAAJ.

FOWLER, M. **Bliki: Monolith First**. 2015. Disponível em: https://martinfowler.com/bliki/MonolithFirst.html.

FOWLER, M. **Articles: Serverless**. 2018. Disponível em: https://martinfowler.com/articles/serverless.html.

GOS, K.; ZABIEROWSKI, W. The comparison of microservice and monolithic architecture. *In*: IEEE. **2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)**. [*S.l.*], 2020. p. 150–153.

JIANG, L.; PEI, Y.; ZHAO, J. Overview of serverless architecture research. **Journal of Physics: Conference Series**, IOP Publishing, v. 1453, n. 1, p. 012119, jan 2020. Disponível em: https://dx.doi.org/10.1088/1742-6596/1453/1/012119.

MOSLEH, M.; DALILI, K.; HEYDARI, B. Distributed or monolithic? a computational architecture decision framework. **IEEE Systems journal**, IEEE, v. 12, n. 1, p. 125–136, 2016.

RICHARDS, M.; FORD, N. **Fundamentals of Software Architecture: An Engineering Approach**. O'Reilly Media, Incorporated, 2020. ISBN 9781492043454. Disponível em: https://books.google.com.br/books?id=_pNdwgEACAAJ.

SOUZA, C. E. D.; PORFIRIO, A. J. **Atualização do Backend do Sistema Open Social Care: Migrando da arquitetura serverless para uma API em Laravel**. 2024. 56 f. Monografia (Trabalho de Conclusão de Curso) — Universidade Tecnológica Federal do Paraná, Guarapuava, 2024. Disponível em: https://tcc.tsi.pro.br/uploads/academic_activity/pdf/267/GP_COINT_2024_1_CAMILA_EMANUELE_DE_SOUZA_MONOGRAFIA.pdf.