MBA USP ESALQ

TCC em

ENGENHARIA DE SOFTWARE

áreas e linhas de pesquisa

Vitor Anfrizio Souza

MBA USP ESALQ

TCC em

ENGENHARIA DE SOFTWARE

áreas e linhas de pesquisa

Piracicaba/PECEGE
2024

Organizadores
Aline Bigaton
Daniela Flôres
Maria Júlia Xavier Belém

Projeto Gráfico
Lucas Moreira Rocha

Revisão
Helder Prado Santos
Maria Júlia Xavier Belém

SOUZA, VITOR ANFRIZIO
TCC EM ENGENHARIA DE SOFTWARE [LIVRO ELETRÔNICO] : ÁREAS E LINHAS DE PESQUISA / VITOR ANFRIZIO SOUZA ; ORGANIZADORES ALINE BIGATON, DANIELA FLÔRES, MARIA JÚLIA XAVIER BELÉM. -- PIRACICABA, SP : PECEGE EDITORA, 2024. PDF

BÍBLIOGRAFIA.
ISBN 978-85-92582-98-2

1. ENGENHARIA DE SOFTWARE - ESTUDO E ENSINO 2. PESQUISA - METODOLOGIA 3. TRABALHO DE CONCLUSÃO DE CURSO (TCC) I. BIGATON, ALINE. II. FLÔRES, DANIELA. III. BELÉM, MARIA JÚLIA XAVIER. IV. TÍTULO..

24-224785 CDD: 005.107

Apresentação

Depois de uma longa jornada no MBA em Engenharia de Software, chegou o momento de elaborar o seu trabalho de conclusão de curso (TCC). Desenvolver o TCC é desafiador. Por isso, este manual foi elaborado com o objetivo de auxiliar você neste processo. Neste guia inicial são indicadas as linhas de pesquisa, métodos e procedimentos que podem ser utilizados para a elaboração do TCC.

Este manual contempla as principais áreas e subáreas em cada linha de pesquisa em Engenharia de Software. O primeiro desafio na elaboração do TCC é a escolha do tema a ser pesquisado. Assim, este material busca ajudar nesta escolha e no planejamento de execução do TCC.

Tendo em vista que o início desta etapa final no MBA, muitas vezes, é cercado de dúvidas e incertezas, espera-se que a leitura deste manual seja útil na delimitação do seu tema de pesquisa nas seguintes linhas de temáticas: 1) Desenvolvimento Web Front-end; 2) Desenvolvimento Web Back-end; 3) Comunicação entre serviços; 4) Aplicativos e soluções para dispositivos móveis; 5) Infraestrutura e gerenciamento de ambientes de TI; 6) Estruturação de bancos de dados; 7) Segurança da informação; 8) Planejamento estratégico, gestão e entrega de projetos na engenharia de softwares; 9) Tecnologia e negócios. Para cada linha de pesquisa tem-se indicado os subtópicos que podem ser pesquisados, bem como a sugestão de referências bibliográficas a serem utilizadas no momento inicial da definição do tema de pesquisa.

Vale ressaltar que o objetivo deste manual é auxiliar nos primeiros passos da elaboração do trabalho de conclusão de curso. Portanto, ele não substitui o manual sobre as normas para a elaboração do TCC ou o tutorial com as instruções para a elaboração do TCC. Desta forma, esses três manuais são complementares e têm propósito distinto nesta etapa do MBA. Por fim, é esperado que o aluno do MBA seja capaz de identificar temas de pesquisa que sejam relevantes para a sua vida profissional e que ele seja capaz de aplicar os conhecimentos, técnicas e ferramentas aprendidas durante o MBA. Desejamos aos alunos e orientadores uma boa leitura e sucesso nesta etapa final.

SUMÁRIO

1. Desenvolvimento Web Front-end

A interface da web, ou seja, a parte visível e interativa com a qual o usuário se envolve, é construída por meio do desenvolvimento web front-end. Tecnologias como HyperText Markup Language [HTML] (estrutura), Cascading Style Sheets [CSS] (estilo) e JavaScript (interatividade) são utilizadas para criar a experiência do usuário, determinando a aparência e funcionalidades de sites e aplicações.

Essa área oferece diversas oportunidades para a realização de pesquisas destinadas ao desenvolvimento do Trabalho de Conclusão de Curso (TCC), incluindo analisar paradigmas de desenvolvimento específicos para o front-end, padrões de projeto (design patterns) que otimizam a estrutura do código, a experiência do usuário (UX), micro front-ends (interfaces modulares), testes no desenvolvimento front-end (TDD) e a aplicação do domain-driven design (DDD) na construção da interface. Cada um desses tópicos apresenta desafios e possibilidades de pesquisa para aprimorar a qualidade e a eficiência do desenvolvimento web front-end.

1.1. Paradigmas de desenvolvimento de software para front-end

O desenvolvimento front-end, responsável pela interface com a qual o usuário interage, tem evoluído rapidamente, impulsionado por novas tecnologias e demandas de mercado. Nesse contexto, os paradigmas de desenvolvimento de software para front-end desempenham um papel crucial na forma como os desenvolvedores estruturam, organizam e implementam o código, impactando diretamente a qualidade, a manutenibilidade e a escalabilidade das aplicações web. A escolha do paradigma de desenvolvimento adequado pode influenciar significativamente o sucesso de um projeto front-end. Paradigmas como o modelo-visão-controlador (MVC), o modelo-visão-apresentação (MVP) e o modelo-visão-viewmodel (MVVM) oferecem diferentes abordagens para separar as responsabilidades do código, facilitando a manutenção e a evolução da interface. Além disso, a crescente popularidade de frameworks JavaScript, como React, Angular e Vue.js, tem impulsionado a adoção de paradigmas baseados em componentes, que promovem a reutilização de código e a modularização da interface.

A área de paradigmas de desenvolvimento de software para front-end oferece diversas oportunidades para realizar estudos. No caso do desenvolvimento de um TCC, o aluno pode, por exemplo, comparar e avaliar as vantagens e desvantagens de paradigmas como MVC, MVP, MVVM e o desenvolvimento baseado em componentes, considerando fatores como complexidade do projeto, escalabilidade, curva de aprendizado e performance. Outra possibilidade é investigar como frameworks populares, como React, Angular e Vue.js, influenciam a escolha e a implementação de paradigmas de desenvolvimento front-end. Explorar paradigmas emergentes, como o desenvolvimento baseado em funções (functional programming) e a programação reativa (reactive programming), e avaliar seu potencial para o desenvolvimento front-end é também uma possibilidade. Por fim, identificar os principais desafios enfrentados pelos desenvolvedores na adoção de diferentes paradigmas e propor soluções e boas práticas para superá-los é um tema relevante e com grande potencial de impacto na comunidade de desenvolvimento.

Referências

Bagliotti, I.R.; Gibertoni, D. 2020. Reusabilidade no desenvolvimento de um sistema web utilizando o framework angular. Revista Interface Tecnológica. Revista Interface Tecnológica 17(1): 192-204.

Ballamudi, V.K.R.; Lal, K.; Desamsetti, H.; Dekkati, S. 2021. Getting Started Modern Web Development with Next. js: An Indispensable React Framework. Digitalization & Sustainability Review 1(1): 1-11.

Farroco, L.O.; Stefano, E.; Ebecken, N.F.F.; Paulo, F.S.; Nosoline, S.M. 2020. O paradigma funcional no desenvolvimento Front-End: oportunidades e desafios. Brazilian Journal of Development 6(2): 6464-6475.

Hutagikar, V.; Hegde, V. 2020. Analysis of front-end frameworks for web applications. Int. Research J. of Engineering and Tech 7(4): 3317-3320.

1.2. Padrões de projeto no desenvolvimento front-end

Os padrões de projeto (design patterns) no desenvolvimento front-end são ferramentas essenciais que ajudam a criar aplicações mais organizadas, manuteníveis e escaláveis. Fornecem soluções reutilizáveis para problemas recorrentes no desenvolvimento de software, possibilitando que os desenvolvedores sigam práticas recomendadas e reconhecidas pela comunidade. Fazendo uma analogia com uma receita de culinária, tem-se que os padrões de projeto atuam como receitas que guiam os desenvolvedores na criação de funcionalidades específicas. Existem três categorias principais de padrões de projeto: criacionais, estruturais e comportamentais. Os padrões criacionais, como Factory, Builder e Singleton, ajudam na criação de objetos de forma controlada. Os padrões estruturais, como Adapter, Decorator e Facade, facilitam a composição de classes e objetos para formar estruturas maiores. Os padrões comportamentais, como Observer, Strategy e Command, se concentram na comunicação e na atribuição de responsabilidades entre objetos.

Além dos padrões tradicionais, o desenvolvimento front-end também faz uso intensivo dos padrões de componentização. Estes padrões são fundamentais para organizar e reutilizar componentes de interface, como menus, botões e formulários, criando uma aplicação modular e eficiente. A componentização permite que os desenvolvedores construam interfaces complexas a partir de pequenos blocos reutilizáveis, facilitando a manutenção e a escalabilidade do código. Outro aspecto crucial no desenvolvimento front-end é o gerenciamento de estado. Ferramentas como Redux, Context API (para React), VueX (para Vue.js) e MobX ajudam a gerenciar o estado da aplicação, permitindo que as mudanças de estado sejam refletidas de maneira consistente na interface do usuário. Esses frameworks fornecem estruturas para armazenar e atualizar o estado da aplicação de forma previsível, melhorando a confiabilidade e a manutenibilidade do código. Os padrões arquiteturais, como MVC (Model-View-Controller) e Clean Architecture, também desempenham um papel importante no desenvolvimento front-end. O MVC separa a aplicação em três componentes principais: o modelo (dados), a visualização (interface) e o controlador (lógica de negócios). E a Clean Architecture promove a separação de responsabilidades em camadas bem definidas, facilitando a manutenção e a evolução do código.

O roteamento e a navegação são outros aspectos essenciais no desenvolvimento de aplicações web. Padrões de roteamento definem como as diferentes partes de uma aplicação são acessadas e exibidas, permitindo uma navegação fluida e intuitiva para o usuário. Ferramentas como React Router e Vue Router são exemplos de bibliotecas que implementam esses padrões, facilitando a criação de rotas e a gestão da navegação entre diferentes vistas da aplicação. Na área de pesquisa, para desenvolvimento dos TCCs, diversos tópicos podem ser explorados, como a comparação de diferentes

arquiteturas front-end, a implementação de uma mesma aplicação utilizando diversos frameworks, a aplicação de padrões de design responsivo e a integração de testes TDD (Test-Driven Development) no desenvolvimento front-end. Outras áreas de pesquisa incluem a comparação de estratégias de gerenciamento de estado e a análise de diferentes padrões de componentização em frameworks como React, Vue.js e Svelte. Em resumo, os padrões de projeto no desenvolvimento front-end são fundamentais para criar aplicações eficientes, manuteníveis e escaláveis. Fornecem uma base sólida para enfrentar desafios comuns, promovendo a adoção de práticas recomendadas e a criação de código de alta qualidade.

Referências

Ballamudi, V.K.R.; Lal, K.; Desamsetti, H.; Dekkati, S. 2021. Getting Started Modern Web Development with Next. js: An Indispensable React Framework. Digitalization & Sustainability Review 1(1):1-11.

Chen, S.; Thaduri, U.R.; Ballamudi, V.K.R. 2019. Front-End Development in React: An Overview. Engineering International 7(2): 117–126.

Kolomoets, M.; Kynash, Y. 2023. Front-End web development project architecture design. In18th International Conference on Computer Science and Information Technologies (CSIT), 2023, Lviv, Ukraine. Proceedings... p. 1-5.

Next.js. 2024. Building Your Application: Routing Fundamentals. Disponível em: https://nextjs.org/docs/app/building-your-application/routing. Acesso em: 12 jul. 2024.
React Native. 2022. Introduction. Disponível em: https://reactnative.dev/docs/getting-started. Acesso em: 25 jun. 2024.

1.3. User experience (UX)

No desenvolvimento front-end, a experiência do usuário (UX) é uma área fundamental que se concentra em criar interfaces intuitivas, agradáveis e eficientes. Ao contrário da interface do usuário (UI), que se preocupa com a aparência visual, o UX se concentra na experiência geral do usuário, incluindo a pesquisa, o design e a usabilidade. O processo de design da experiência do usuário envolve diversas etapas, desde a pesquisa e a empatia com o usuário até a criação, o teste e o desenvolvimento da interface. O objetivo é entender as necessidades, expectativas e desejos do usuário para criar uma experiência que seja relevante, significativa e agradável.

A área de UX oferece inúmeras oportunidades para desenvolver pesquisas e trabalhos acadêmicos. No desenvolvimento de um TCC, o aluno pode investigar, por exemplo, métodos para avaliar a usabilidade de aplicações, explorar a aplicação da teoria do design centrado no usuário, analisar a experiência do usuário em design de jogos, estudar a relação entre UX e ética de design, investigar o uso da neurociência e psicologia na experiência do usuário, ou ainda, explorar a importância da acessibilidade na criação de interfaces inclusivas. Além desses temas, existem outras áreas de pesquisa relacionadas ao UX, como a análise de métricas de engajamento, o estudo de padrões de comportamento do usuário, a aplicação de inteligência artificial para personalizar a experiência do usuário e a investigação de novas tecnologias e ferramentas para o design e desenvolvimento de interfaces. Ao explorar essas áreas, é possível contribuir para o desenvolvimento de interfaces mais intuitivas, eficientes e agradáveis, proporcionando uma melhor experiência para os usuários e impulsionando o sucesso de produtos e serviços digitais

Referências

Alomari, H.W.; Ramasamy, V.; Kiper, J.D.; Potvin, G. 2020. A User Interface (UI) and User eXperience (UX) evaluation framework for cyberlearning environments in computer science and software engineering education. Heliyon 6(5): e03917.

Entenberg, G.A.; Dosovitsky, G.; Aghakhani, S.; Mostovoy, K.; Carre, N.; Marshall, Z.; Benfica, D.; Mizrahi, S.; Testerman, A.; Rousseau, A.; Lin, G.; Bunge, E.L. 2023. User experience with a parenting chatbot micro intervention. Frontiers in Digital Health 4: 989022.

Hartson, R.; Pyla, P.S. 2018. The UX book: Agile UX design for a quality user experience. Morgan Kaufmann, Cambridge, MA, USA.

Meftah, C.; Retbi, A.; Bennani, S.; Idrissi, M. K. 2019. Mobile serious game design using user experience: modeling of software product line variability. International Journal of Emerging Technologies in Learning (Online) 14(23): 55-66.

Shourmasti, E.S.; Colomo-Palacios, R.; Holone, H.; Demi, S. 2021. User experience in social robots. Sensors 21(15): 5052.

1.4. Micro Front-end

No desenvolvimento web front-end, a arquitetura de micro front-end surge como uma alternativa aos tradicionais aplicativos monolíticos, oferecendo maior flexibilidade, escalabilidade e facilidade de manutenção. Essa abordagem divide a interface do usuário em módulos menores e independentes, que podem ser desenvolvidos, testados e implantados separadamente, utilizando diferentes tecnologias e frameworks. A adoção de micro front-ends apresenta inúmeras vantagens, como a possibilidade de ter equipes menores e mais focadas trabalhando em cada módulo, a capacidade de atualizar partes da aplicação sem afetar o todo e a flexibilidade para utilizar diferentes tecnologias em cada micro front-end. No entanto, essa arquitetura também apresenta desafios, como a necessidade de gerenciar a comunicação entre os módulos, garantir a consistência da experiência do usuário e lidar com a complexidade de integrar diferentes tecnologias.

A análise da escalabilidade e desempenho, comparando diferentes abordagens e tecnologias, pode revelar insights valiosos para a construção de aplicações robustas e eficientes. A otimização do desenvolvimento e implantação contínua, visando a automação e a agilidade, é outro tema crucial para garantir a entrega rápida e confiável de software. A comparação entre micro front-ends e aplicações monolíticas permite avaliar os prós e contras de cada modelo em diferentes cenários, auxiliando na escolha da arquitetura mais adequada para cada projeto. O desafio do gerenciamento de estado, que envolve a sincronização e compartilhamento de dados entre componentes independentes, demanda soluções inovadoras para garantir a consistência e a fluidez da experiência do usuário. Por fim, a usabilidade e a coesão da interface do usuário são aspectos cruciais a serem investigados, buscando garantir que a modularidade dos micro front-ends não comprometa a experiência do usuário final.

Referências

Geers, M. 2020. Micro Frontends in Action. Simon and Schuster, Shelter Island, NY, USA.

Marco, V.; Farias, K. 2024. Exploring the technologies and architectures used to develop micro-frontend applications: A systematic mapping and emerging perspectives. Disponível em: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4750661. Acesso em: 12 jul. 2024.

Mezzalira, L. 2018. Front-End Reactive Architectures: Explore the Future of the Front-End using Reactive JavaScript Frameworks and Libraries. Apress Berkeley, CA, USA.

Mezzalira, L. 2021. Building Micro-Frontends. O'Reilly Media, Inc., Sebastopol, CA, USA.

Micro Front-ends. 2022. Extending the microservice idea to frontend development. Disponível em: https://micro-frontends.org/. Acesso em: 25 jun. 2024.

Pavlenko, A.; Askarbekuly, N.; Megha, S.; Mazzara, M. 2020. Micro-frontends: application of microservices to web front-ends. Journal of Internet Services and Information Security 10(2): 49-66.

Peltonen, S.; Mezzalira, L.; Taibi, D. 2021. Motivations, benefits, and issues for adopting micro-frontends: a multivocal literature review. Information and Software Technology 136: 106571.

Perera, Y. 2023. Enhancing the front end web applications performance using design patterns and microservices based architecture. Thesis in Science Honors in Computer Studies. Faculty of Science, University of Kelaniya, Kelaniya,Sri Lanka.

Yang, C.; Liu, C.; Su, Z. 2019. Research and application of micro frontends. IOP Conference Series: MaterialsScience and Engineering 490(1):1–8.

1.5. Micro-serviço Front-end

No desenvolvimento web front-end, a arquitetura de micro-serviços, ou micro front-ends, oferece uma alternativa aos tradicionais aplicativos monolíticos, dividindo a interface do usuário em módulos menores e independentes. Essa abordagem permite que cada módulo seja desenvolvido, testado e implantado separadamente, utilizando diferentes tecnologias e frameworks, proporcionando maior flexibilidade e escalabilidade. Uma das principais vantagens dos micro front-ends é a capacidade de isolar falhas. Se um módulo apresentar problemas, os demais continuam funcionando, ao contrário do que ocorre em um monólito, em que uma falha pode comprometer toda a aplicação. Além disso, os micro front-ends permitem a utilização de diferentes tecnologias e frameworks em cada módulo, o que pode ser útil para aproveitar as vantagens de cada um e atender a diferentes requisitos.

A arquitetura de micro-serviços front-end também facilita a escalabilidade da aplicação, pois cada módulo pode ser dimensionado de forma independente, de acordo com a demanda. Além disso, a organização em torno dos recursos de negócios permite

alocar recursos de forma mais eficiente, priorizando os módulos mais críticos. Dentro do campo de micro-serviços front-end, surgem inúmeras possibilidades de pesquisa e desenvolvimento dos TCCs. Entre os temas a serem explorados estão a escalabilidade e o desempenho dos micro front-ends, com comparações entre diferentes arquiteturas e frameworks. Além disso, o desenvolvimento e a implantação contínua em micro front-ends destacam-se como áreas de interesse, com o objetivo de encontrar soluções para automatizar a integração e o deploy. A análise comparativa entre aplicações monolíticas e micro front-ends é essencial para compreender os benefícios e desafios inerentes a cada abordagem em variados contextos. Outro aspecto crucial é o gerenciamento de estado em micro front-ends, que exige a sincronização de dados entre módulos independentes. Por fim, a experiência do usuário em micro front-ends não pode ser negligenciada, pois é fundamental que a divisão da interface em módulos mantenha a usabilidade e a coesão da aplicação.

Referências

Assunção, W. K.; Krüger, J.; Mosser, S.; Selaoui, S. 2023. How do microservices evolve? An empirical analysis of changes in open-source microservice repositories. Journal of Systems and Software 204: 111788.

Dragoni, N.; Giallorenzo, S.; Lluch-Lafuente, A.; Mazzara, M.; Montesi, F.; Mustafin, R.; Safina, L. 2017. Microservices: yesterday, today, and tomorrow. Present and ulterior software engineering 195-216.

Mezzalira, L. 2018. Front-End Reactive Architectures: Explore the Future of the Front-End using Reactive JavaScript Frameworks and Libraries. Apress Berkeley, CA, USA.

Nasab, A.R.; Shahin, M.; Raviz, S.A.H.; Liang, P.; Mashmool, A.; Lenarduzzi, V. 2023. An empirical study of security practices for microservices systems. Journal of Systems and Software 198: 111563.

Waseem, M.; Liang, P.; Shahin, M.; Ahmad, A.; Nassab, A.R. 2021. On the Nature of Issues in Five Open Source Microservices Systems: An Empirical Study. In International Conference on Evaluation and Assessment in Software Engineering (EASE), 2021, Trondheim Norway, Noruega. Proceedings... p. 201-210.

1.6. Gerenciamento de Testes

Dentro da área de desenvolvimento de software, o gerenciamento de testes é crucial para garantir a qualidade, a confiabilidade e a funcionalidade das aplicações. Uma metodologia importante nesse contexto é o Test Driven Development (TDD), que propõe a criação de testes automatizados antes mesmo do desenvolvimento do código. Esse ciclo iterativo de desenvolvimento, que envolve a escrita de um teste que falha, a implementação do código para que o teste passe e a refatoração do código, garante que a aplicação seja construída de forma sólida e livre de erros. A pirâmide de testes é um conceito que guia a organização dos testes em diferentes níveis, desde os testes unitários, que validam as menores unidades de código, até os testes de integração, que verificam a comunicação entre os componentes, e os testes end-to-end, que simulam o uso da aplicação pelo usuário final. Essa abordagem permite identificar e corrigir erros

em diferentes níveis, garantindo a qualidade e a funcionalidade da aplicação como um todo.

Ferramentas como Jest, Mocha, Jasmine e Cypress auxiliam na criação e execução de testes automatizados, facilitando o processo de gerenciamento de testes e permitindo a identificação de problemas antes que cheguem ao usuário final. A integração contínua (CI) e a entrega contínua (CD) são práticas que automatizam o processo de teste e implantação de novas versões do software, garantindo que as alterações não introduzam novos erros e que a aplicação esteja sempre atualizada e funcionando corretamente. A área de gerenciamento de testes oferece diversas oportunidades para a pesquisa acadêmica. No TCC, o aluno pode, por exemplo, investigar estratégias de implementação do TDD no desenvolvimento front-end, analisar a integração de testes automatizados no fluxo de trabalho, estudar o impacto do TDD na performance e na manutenção da aplicação, ou ainda, explorar o uso de testes de acessibilidade para garantir que a aplicação seja inclusiva e acessível a todos os usuários

Referências

Abushama, H.M.; Alassam, H.A.; Elhaj, F.A. 2021. The effect of test-driven development and behavior-driven development on project success factors: A systematic literature review based study. In International Conference on Computer, Control, Electrical, and Electronics Engineering, 2020, Khartoum, Sudan. Proceedings… p. 1-9.

Ghafari, M.; Gross, T.; Fucci, D.; Felderer, M. 2020.Why research on test-driven development is inconclusive? In 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2020, Bari, Itália. Proceedings… p. 1-10.

Romano, S.; Zampetti, F.; Baldassarre, M.T.; Di Penta, M.; Scanniello, G. 2022. Do static analysis tools affect software quality when using test-driven development? In 16th ACM/IEEE international symposium on empirical software engineering and measurement, 2022, Helsinki Finland. Proceedings… p. 80-91.

Santos, A.; Vegas, S.; Dieste, O.; Uyaguari, F.; Tosun, A.; Fucci, D.; Turhan, B.; Scanniello, G.; Romano, S.; Karac, I.; Kuhrmann, M. 2021. A family of experiments on test-driven development. Empirical Software Engineering 26(42):1-53.

Staegemann, D.; Volk, M.; Jamous, N.; Turowski, K. 2020. Exploring the applicability of test driven development in the big data domain. In Australasian Conference on Information Systems 2020, Wellingtonm, New Zealand. Proceedings... p. 1-12.

1.7. Domain Driven Design (DDD)

O Domain-Driven Design (DDD) é uma metodologia que visa alinhar o design de software com o domínio do negócio, ou seja, com o conjunto de conhecimentos e regras que a aplicação deve incorporar. Seu propósito é estabelecer uma linguagem comum, conhecida como linguagem ubíqua, que promove uma comunicação eficaz entre especialistas do negócio e desenvolvedores, facilitando a compreensão mútua e a criação de software que atenda às necessidades específicas do negócio. No DDD, o domínio do negócio representa o cerne da aplicação, a partir do qual são definidas entidades, agregados e regras de negócio

que compõem o sistema. As diferentes camadas da aplicação, como a camada de aplicação, a camada de infraestrutura e o próprio domínio, são projetadas para serem independentes umas das outras, permitindo o desenvolvimento e manutenção separados de cada uma delas.

A área de DDD no desenvolvimento front-end oferece diversas oportunidades para realizar estudos. O aluno no TCC pode, por exemplo, investigar a integração do domínio front-end e back-end, buscando soluções para garantir a consistência dos dados e a comunicação eficiente entre as camadas. Outro tema relevante é o uso de EventStorming, uma técnica de modelagem de eventos, para auxiliar na compreensão e no design do domínio da aplicação. A implementação de técnicas e padrões DDD em aplicações front-end é um campo fértil para pesquisa, permitindo avaliar o impacto do DDD na qualidade, manutenibilidade e escalabilidade do código. O gerenciamento de estado no front-end baseado em DDD também é considerado um tema interessante, explorando como as entidades e agregados do domínio podem ser utilizados para representar e manipular o estado da aplicação. Por fim, a migração de aplicações existentes para uma arquitetura baseada em DDD pode ser um desafio interessante, exigindo a refatoração do código e a adaptação dos processos de desenvolvimento.

Referências

Jordanov, J.; Petrov, P. 2023. Domain Driven Design Approaches in Cloud Native Service Architecture. TEM Journal 12(4): 1985.

Kapferer, S. 2020. A Modeling Framework for Strategic Domain-driven Design and Service Decomposition. Dissertação de Mestrado em Engenharia. University of Applied Sciences of Eastern Switzerland, Switzerland.

Khononov, V. 2021. Learning Domain-Driven Design: Aligning Software Architecture and Business Strategy. O'Reilly Media, Inc., Sebastropol, CA, USA.

Sangabriel-Alarcón, J.; Ocharán-Hernández, J.O.; Cortés-Verdín, K.; Limón, X. 2023. Domain-Driven Design for Microservices Architecture Systems Development: A Systematic Mapping Study. In 11th International Conference in Software Engineering Research and Innovation (CONISOFT), 2023, León, Guanajuato, Mexico. Proceedings… p. 25-34.

Schmidt, R.A.; Thiry, M. 2020. Microservices identification strategies: A review focused on Model-Driven Engineering and Domain Driven Design approaches. In 15th Iberian Conference on Information Systems and Technologies (CISTI), 2020, Seville, Spain. Proceedings… p. 1-6.

Zhong, C.; Li, S.; Huang, H.; Liu, X.; Chen, Z.; Zhang, Y.; Zhang, H. 2024. Domain-Driven Design for Microservices: An Evidence-Based Investigation. IEEE Transactions on Software Engineering 50(6): 1425-1449.

2. Desenvolvimento Web Back-end

No desenvolvimento web, o back-end gerencia dados, regras de negócio e a lógica da aplicação, comunicando-se com o front-end para fornecer informações ao usuário. É comparável à parte invisível de um iceberg, mas essencial para o funcionamento da aplicação. Diversas áreas temáticas, como paradigmas de desenvolvimento (modelo

em camadas, arquitetura orientada a serviços), padrões de projeto (Singleton, Factory), e estruturação de Application Programming Interface (APIs), são fundamentais para o back-end. A adoção de microserviços melhora a escalabilidade e flexibilidade, mas também apresenta desafios de gerenciamento e comunicação. O Test Driven Development (TDD) é crucial para a qualidade e confiabilidade do código, criando testes automatizados para validar o comportamento do código. O Domain Driven Design (DDD) alinha o desenvolvimento de software com o domínio do negócio, facilitando a comunicação entre desenvolvedores e especialistas. Esses tópicos oferecem oportunidades para pesquisa e desenvolvimento de novas soluções, permitindo que desenvolvedores aprimorem suas habilidades e contribuam para aplicações web mais robustas, escaláveis e eficientes.

2.1. Paradigmas de desenvolvimento de software para back-end

O back-end é a principal camada da aplicação dentro do desenvolvimento web. É o responsável por gerenciar dados, regras de negócio e lógica, comunicando-se com o front-end para fornecer as informações exibidas ao usuário. Essa área envolve diversas decisões arquiteturais, desde a escolha da linguagem de programação até a estrutura do servidor e o tipo de banco de dados utilizado. A escolha da linguagem de programação é crucial, pois cada linguagem possui características e vantagens específicas, como performance, facilidade de uso e disponibilidade de bibliotecas. A arquitetura do servidor também é importante, podendo ser monolítica, baseada em microserviços ou serverless, cada uma com suas vantagens e desafios. A persistência de dados, por sua vez, envolve a escolha do tipo de banco de dados, como SQL, NoSQL ou séries temporais, que deve ser adequada às necessidades da aplicação. Outras preocupações importantes no desenvolvimento back-end incluem a autenticação e autorização de usuários, a proteção de dados, a escalabilidade e a capacidade de lidar com tráfego variável. Essas questões exigem atenção especial e podem ser abordadas de diferentes formas, dependendo das características da aplicação e dos requisitos do negócio. A área de paradigmas de desenvolvimento de software para back-end oferece diversas oportunidades para realizar a pesquisa acadêmica.

No desenvolvimento do TCC, o aluno pode, por exemplo, investigar o desenvolvimento de APIs RESTful, que são amplamente utilizadas para a comunicação entre aplicações. Outro tema relevante é o desenvolvimento de aplicações com foco em alto desempenho e escalabilidade, explorando técnicas de otimização e utilizando ferramentas de teste de estresse. A implementação de técnicas e padrões de Domain-Driven Design (DDD) em aplicações back-end também é uma área promissora, buscando alinhar o desenvolvimento com o domínio do negócio e facilitando a comunicação entre desenvolvedores e especialistas. A migração de um back-end monolítico para uma arquitetura baseada em microsserviços é outro tema possível de estudo, que pode trazer benefícios em termos de escalabilidade e flexibilidade, mas exige uma maior atenção no planejamento e na execução. Por fim, a utilização de técnicas DevOps no desenvolvimento back-end, como a automação de testes e a integração contínua, pode melhorar a qualidade e a eficiência do processo de desenvolvimento, garantindo que a aplicação esteja sempre atualizada e funcionando corretamente.

Referências

Arbabzadeh, M.; Sreenath, R.; Gençer, E. 2021. Back-End Design and Development of an Energy Systems Analysis Tool. Computer Aided Chemical Engineering 50: 1433-1438.

Farhat, H. 2021. Design and development of the back-end software architecture for a hybrid cyber range. Dissertação de Mestrado em Comunicações e Engenharia de Redes de Computadores, Politecnico di Torino, Itália.

Madurapperuma, I.H.; Shafana, M.S; Sabani, M.J.A.2022. State-of-art frameworks for front-end and back-end web development. In 2nd International Conference on Science and Technology, 2022, Sri Lanka. Proceeding… p. 62-67.

Zahroh, U. 2022. Back-End Design and Development on Rekaruang Application with Microservices Architecture. Jurnal Teknik Informatika dan Sistem Informasi (JATISI) 9(1): 86-96.

2.2. Padrões de Projetos no desenvolvimento back-end

A aplicação de padrões de projeto, dentro do processo de desenvolvimento back-end, é essencial para criar software escalável, manutenível e de alta qualidade. Além dos padrões de projeto tradicionais, como os criacionais, estruturais e comportamentais, existem padrões específicos para o back-end que abordam a arquitetura, a segurança, a injeção de dependência e o uso de middlewares. Os padrões arquiteturais, como o Domain-Driven Design (DDD), a arquitetura limpa e o Model-View-Controller (MVC), fornecem diretrizes para a estruturação e organização do código, facilitando a manutenção e a evolução da aplicação. Padrões de autenticação e autorização garantem a segurança da aplicação, controlando o acesso dos usuários e protegendo os dados sensíveis. A injeção de dependência é um padrão que permite desacoplar os componentes da aplicação, facilitando a reutilização de código e a realização de testes. Os middlewares são funções intermediárias que interceptam as requisições e respostas entre o cliente e o servidor, permitindo a implementação de funcionalidades como logging, autenticação, tratamento de erros e cache.

A área de padrões de projeto no desenvolvimento back-end oferece diversas oportunidades para a pesquisa. No desenvolvimento do TCC, o aluno pode, por exemplo, comparar diferentes padrões de projeto e avaliar seu impacto na performance e escalabilidade da aplicação. A implementação de diferentes arquiteturas, como o DDD e a arquitetura limpa, em um mesmo projeto, pode revelar as vantagens e desvantagens de cada abordagem. A integração de padrões de projeto com testes automatizados e a adoção de práticas de desenvolvimento ágil, como a integração contínua, podem ser investigadas para avaliar seu impacto na qualidade e na eficiência do desenvolvimento back-end. Além disso, a análise do papel dos padrões de projeto no desempenho e na manutenção de sistemas back-end ao longo do tempo pode fornecer insights valiosos para a criação de software mais robusto e escalável.

Referências

Alam, A.; Bush, Vicky. 2023. The Model View Controller as a Functional Reactive Program. Disponível em: <https://www.researchsquare.com/article/rs-3743640/v1>. Acesso em: 12 jul. 2024.

Dehbozorgi, N.; Norkham, A. 2021. An Architecture Model of Recommender System for Pedagogical Design Patterns. In IEEE Frontiers in Education Conference, 2021, Lincoln, NE, USA. Proceedings… p. 1-4.

Necula, S. 2024. Exploring The Model-View-Controller (MVC) Architecture: A Broad

Analysis of Market and Technological Applications. Disponível em: https://www.preprints.org/manuscript/202404.1860/v1. Acesso em: 12 jul. 2024.

React Native. 2022. Introdução. Disponível em: https://reactnative.dev/docs/getting-started. Acesso em: 25 jun. 2024.

Velepucha, V.; Flores, P. 2023. A Survey on Microservices Architecture: Principles, Patterns and Migration Challenges. IEEE Access 11: 88339- 88358.

2.3. Gerenciamento e estruturação de API´s

As APIs (Application Programming Interface) são a ponte que permite a comunicação entre diferentes sistemas, como o front-end e o back-end de uma aplicação. Elas definem como os dados são solicitados, formatados e entregues, permitindo que diferentes aplicações troquem informações de forma padronizada e eficiente. Para entender o funcionamento de uma API, pode-se utilizar uma analogia com o garçom, que recebe o pedido do cliente (front-end), o leva para a cozinha (back-end) e retorna com a comida (dados) pronta. Assim como o garçom, a API conhece os diferentes "pratos" (endpoints) disponíveis e sabe como obtê-los do back-end para entregar ao cliente. Existem diversos tipos de APIs, como REST, SOAP, GraphQL e gRPC, cada um com suas características e vantagens. A escolha do tipo de API depende dos requisitos da aplicação, como a necessidade de flexibilidade, performance e facilidade de uso.

A documentação de APIs é essencial para que os desenvolvedores possam utilizá-las corretamente, fornecendo informações sobre os endpoints, os métodos de requisição (GET, POST, PUT, DELETE) e os formatos de dados utilizados. A autenticação e autorização são aspectos importantes na segurança de APIs, garantindo que apenas usuários autorizados tenham acesso aos dados e funcionalidades. A área de gerenciamento e estruturação de APIs convida o aluno a explorar diferentes vertentes. Uma possibilidade é investigar a construção de APIs RESTful, que são amplamente utilizadas e oferecem flexibilidade e facilidade de uso. A comparação de diferentes padrões arquiteturais de APIs, como REST e SOAP, pode revelar as vantagens e desvantagens de cada abordagem em diferentes contextos.

O versionamento de APIs também é um tema importante para garantir a compatibilidade com aplicações existentes, permitindo a evolução da API sem quebrar o funcionamento de sistemas que a utilizam. O estudo do ciclo de vida de uma API, desde sua concepção até sua descontinuação, pode fornecer insights valiosos para o planejamento e gerenciamento de APIs. A implantação e escalabilidade de APIs também são temas relevantes, envolvendo a escolha de tecnologias e ferramentas adequadas para garantir o desempenho e a disponibilidade da API sob diferentes cargas de trabalho. O uso de APIs em ambientes de microserviços, que dividem a aplicação em módulos menores e independentes, apresenta desafios específicos de comunicação e gerenciamento, que podem ser explorados em pesquisas. Por fim, o estudo de ferramentas e técnicas de monitoramento de APIs permite acompanhar o desempenho, identificar gargalos e garantir a disponibilidade da API, proporcionando uma melhor experiência para os usuários.

Referências

Afonso, J.; Caffy, C.; Patrascoiu, M.; Leduc, J.; Davis, M.; Murray, S.; Cortes, P. 2024. An HTTP REST API for Tape-backed Storage. In 26th International Conference on Computing in High Energy and Nuclear Physics, 2023, Norfolk, VA, USA. Proceedings…

p.1-8.

Dehbozorgi, N.; Norkham, A. 2021. An Architecture Model of Recommender System for Pedagogical Design Patterns, 2021, Lincoln, NE, USA. Proceedings… p.1-4.

Gough, J.; Bryant, D.; Auburn, M. 2022. Mastering API Architecture: Design, Operate, and Evolve Api-Based Systems. O'Reilly Media, Incorporated, Sebastopol, CA, USA.

Muhammad, I.R.D.; Paputungan, I.V. 2024. Development of Backend Server Based on REST API Architecture in E-Wallet Transfer System. Jurnal Sains, Nalar, dan Aplikasi Teknologi Informasi 3(2): 79-87.

Rosidin, R.; Jacob, D.; Lubis, M.; Panduwiyasa, H. 2023. Development of REST API for Digital Advertising Application Using an Iterative Incremental Method (Case Study: Show-Up Apps). In International Conference on Advanced Information Scientific Development [ICAISD], 2023, Proceedings… p. 25-31.

Wolde, B.G.; Boltana, A.S. 2021. REST API composition for effectively testing the Cloud. Journal of applied research and technology 19(6): 676-693.

2.4. Micros serviços

A arquitetura de microsserviços oferece uma alternativa aos tradicionais sistemas monolíticos, dividindo a aplicação em módulos menores e independentes, cada um com sua própria responsabilidade e banco de dados. Essa abordagem traz benefícios como maior flexibilidade, escalabilidade e resiliência, mas também aumenta a complexidade do sistema, exigindo atenção especial à comunicação e ao gerenciamento dos diferentes serviços. Um dos principais desafios na arquitetura de microsserviços é a comunicação entre os serviços, que pode ser feita de forma síncrona ou assíncrona, utilizando diferentes protocolos e tecnologias. A escalabilidade também é um ponto crucial, pois cada microsserviço deve ser capaz de lidar com o aumento da demanda de forma independente. A segurança em ambientes de microsserviços exige atenção especial, pois a comunicação entre os serviços pode expor vulnerabilidades que não existiriam em um sistema monolítico.

A área de microsserviços oferece diversas oportunidades para investigação. O aluno poderia, por exemplo investigar padrões de comunicação em ambientes de microsserviços, comparando diferentes abordagens e avaliando seu impacto no desempenho e na escalabilidade do sistema. O estudo da escalabilidade de microsserviços, tanto em termos de capacidade de processamento quanto de armazenamento de dados, é fundamental para garantir que a aplicação possa crescer de forma sustentável. A segurança em ambientes de microsserviços também é um tema relevante, explorando técnicas de autenticação, autorização e proteção de dados que possam ser aplicadas em cada microsserviço e na comunicação entre eles. A migração de um sistema monolítico para uma arquitetura de microsserviços é um desafio complexo, que envolve a identificação dos módulos que podem ser separados, a definição das interfaces de comunicação e a adaptação do código existente. A análise de desempenho em ambientes de microsserviços permite identificar gargalos e otimizar o sistema para garantir uma boa experiência para o usuário. Por fim, o gerenciamento de estado e consistência de dados em microsserviços é um tema crucial, pois a divisão da aplicação em módulos independentes pode dificultar a garantia da consistência dos dados. A pesquisa nessa área pode explorar diferentes abordagens, como o uso de

bancos de dados distribuídos.

Referências

Gaol, F. L., Soeparno, H., Arifin, Y. 2023. Analyzing the Impact of Service Design on Maintainability Factor in Microservices Architecture. In 15th International Congress on Advanced Applied Informatics Winter (IIAI-AAI-Winter), 2023, Bali, Indonesia. Proceedings… p. 75-81.

Hannousse, A.; Yahiouche, S. 2021.Securing microservices and microservice architectures: A systematic mapping study. Computer Science Review 41: 100415.

Lu, Z.; Delaney, D. T.; Lillis, D. 2023. A survey on microservices trust models for open systems. IEEE Access 11: 28840-28855.

Lu, Z.; Delaney, D. T.; Lillis, D. 2024. Comparing the Similarity of OpenAPI-Based Microservices. In 39th ACM/SIGAPP Symposium on Applied Computing, 2024, Avila, Spain, Proceedings… p. 1201-1208.

Newman, S. 2020. Migrando Sistemas Monolíticos Para Microsserviços: Padrões Evolutivos Para Transformar seu Sistema Monolítico. Editora Novatec, São Paulo, SP, Brasil.

Newman, S. 2022. Criando Microsserviços: Projetando Sistemas com Componentes Menores e Mais Especializados. 2ed. Editora Novatec, São Paulo, SP, Brasil.

Waseem, M.; Liang, P.; Shahin, M. 2020. A systematic mapping study on microservices architecture in devops. Journal of Systems and Software 170: 110798.

Waseem, M.; Liang, P.; Shahin, M.; Di Salle, A.; Márquez, G. 2021. Design, monitoring, and testing of microservices systems: The practitioners' perspective. Journal of Systems and Software 182: 111061.

2.5. Test Driven Development Back-end

No desenvolvimento de software back-end, a metodologia Test Driven Development (TDD) desempenha um papel fundamental na garantia da qualidade, confiabilidade e manutenibilidade do código. Essa abordagem, centrada na criação de testes automatizados antes da implementação do código, promove um ciclo de desenvolvimento iterativo e incremental, em que cada nova funcionalidade é validada por testes antes de ser integrada ao sistema. O processo de TDD se inicia com a escrita de um teste que falha, representando a nova funcionalidade desejada. Em seguida, o desenvolvedor implementa o código necessário para que o teste passe, garantindo que a funcionalidade esteja funcionando corretamente. Por fim, o código é refatorado, melhorando sua estrutura e legibilidade, sem alterar seu comportamento. Esse ciclo se repete para cada nova funcionalidade, garantindo que o código seja testado e validado a cada etapa do desenvolvimento.

A pirâmide de testes é um conceito importante no TDD, que orienta a organização dos testes em diferentes níveis, com foco nos testes unitários, que são mais rápidos e baratos de executar, e menor ênfase nos testes de integração e end-to-end, que são mais complexos e demorados. Essa abordagem permite identificar e corrigir

erros rapidamente, antes que se propaguem e causem problemas maiores. Diversas ferramentas e frameworks, como Jest, PyTest, JUnit e Mocha, facilitam a implementação do TDD no desenvolvimento back-end, oferecendo recursos para a criação e execução de testes automatizados. A integração contínua (CI) e a entrega contínua (CD) são práticas que complementam o TDD, automatizando o processo de teste e implantação do código, garantindo que as alterações sejam validadas e entregues de forma rápida e segura. O universo do TDD no back-end apresenta inúmeras possibilidades de estudo. Um caminho interessante para o aluno seria, por exemplo, investigar estratégias de implementação do TDD em diferentes linguagens e frameworks, analisar o impacto do TDD na qualidade e na produtividade do desenvolvimento, ou ainda, explorar o uso de testes de performance para garantir que a aplicação atenda aos requisitos de desempenho.

Referências

Agha, D.; Sohail, R.; Meghji, A. F.; Qaboolio, R.; Bhatti, S. 2023. Test Driven Development and Its Impact on Program Design and Software Quality: A Systematic Literature Review. VAWKUM Transactions on Computer Sciences 11(1): 268-280.

Bhadauria, V.S.; Mahapatra, R.K.; Nerur, S.P. 2020. Performance outcomes of test-driven development: an experimental investigation. Journal of the Association for Information Systems, 21(4): 1045-1071.

Calais, P.; Franzini, L. 2023. Test-Driven Development Benefits Beyond Design Quality: Flow State and Developer Experience. In 45th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), 2023, Melbourne, Austrália. Proceedings... p. 106-111.

Ramzan, H. A.; Ramzan, S.; Kalsum, T. 2024. Test-Driven Development (TDD) in Small Software Development Teams: Advantages and Challenges. In: International conference on advancements in computational sciences (ICACS), 2024, Lahore, Pakistan. Proceedings... p. 1-5.

Staegemann, D.; Volk, M.; Abdallah, M.; Turowvski, K. 2023.Towards the Application of Test Driven Development in Big Data Engineering. In: International conference on information technology (ICIT), 2023, Amman, Jordan. Proceedings... p. 163-167.

2.6. Domain-Driven Design Back-end

O Domain-Driven Design (DDD) é uma abordagem que coloca o domínio do negócio no centro do processo de desenvolvimento de software. O objetivo é criar uma linguagem universal, chamada de linguagem ubíqua, que permita a comunicação eficiente entre especialistas do negócio e desenvolvedores, facilitando o entendimento mútuo e a criação de um software que atenda às necessidades do negócio de forma precisa e eficaz. Essa linguagem ubíqua é fundamental para garantir que todos os envolvidos no projeto, desde os especialistas do negócio até os desenvolvedores, tenham uma compreensão clara e compartilhada do domínio da aplicação. Isso permite que o software seja modelado de forma mais precisa, refletindo as regras e processos de negócio de maneira fiel. No DDD, a aplicação é dividida em camadas distintas, cada uma com suas responsabilidades e interfaces bem definidas. A camada de domínio é o coração da aplicação, contendo as entidades, os agregados e as regras de negócio que

representam o conhecimento do domínio. A camada de aplicação, por sua vez, contém os casos de uso, que representam as interações do usuário com o sistema, e a camada de infraestrutura lida com a persistência de dados e a comunicação com sistemas externos.

Essa abordagem em camadas, juntamente com o uso da linguagem ubíqua, permite que o software seja mais modular, flexível e fácil de manter. As mudanças no domínio do negócio podem ser implementadas de forma mais rápida e eficiente, pois o código está organizado de forma clara e coesa, refletindo a estrutura do próprio negócio. A área de DDD no desenvolvimento back-end abre portas para diversas descobertas. O aluno pode, por exemplo, investigar a modelagem de um domínio back-end utilizando técnicas e padrões de DDD, como o EventStorming, que permite visualizar e modelar os eventos e processos do domínio de forma colaborativa. Outro tema interessante é o desenvolvimento de APIs RESTful com DDD, explorando como as entidades e regras de negócio do domínio podem ser mapeadas para os recursos e operações da API. O gerenciamento de estado no back-end baseado em DDD também é um tópico relevante, investigando como as entidades e agregados do domínio podem ser utilizados para representar e manipular o estado da aplicação de forma consistente e eficiente. A migração de uma arquitetura back-end existente para uma arquitetura DDD pode ser um desafio interessante, exigindo a análise e refatoração do código para adequá-lo aos princípios do DDD. A implementação de técnicas e padrões de repositório para persistência de dados com DDD também é um tema relevante, explorando como os repositórios podem ser utilizados para acessar e persistir os dados do domínio de forma consistente e eficiente.

Referências

Johnson, P. 2022. Is EventStorming effective in defining the bounded contexts used to break down monolithic software into microservices? Disponível em: <https://oro.open.ac.uk/94301/1/JOHNSON_T847_VOR2.pdf>. Acesso em: 12 jul. 2024.

Levezinho, M.; Kapferer, S.; Zimmermann, O.; Silva, A.R. 2024. Domain-Driven Design Representation of Monolith Candidate Decompositions Based on Entity Accesses. Disponível em: <https://arxiv.org/pdf/2407.02512>. Acesso em: 12 jul. 2024.

Oukes, P.; Van Andel, M.; Folmer, E.; Bennett, R.; Lemmen, C. 2021. Domain-Driven Design applied to land administration system development: Lessons from the Netherlands. Land use policy 104: 105379.

Özkan, O.; Babur, Ö.; van der Brand, M. 2023. Domain-Driven Design in Software Development: A Systematic Literature Review on Implementation, Challenges, and Effectiveness. Disponível em: <https://arxiv.org/pdf/2310.01905>. Acesso em: 12 jul. 2024.

Zhong, C.; Li, S.; Huang, H.; Liu, X.; Chen, Z.; Zhang, Y.; Zhang, H. 2024. Domain-Driven Design for Microservices: An Evidence-Based Investigation. IEEE Transactions on Software Engineering 50(6): 1425-1449.

3. Comunicação entre Serviços

No desenvolvimento de sistemas distribuídos, a comunicação entre serviços é um aspecto fundamental para garantir o funcionamento e a interoperabilidade dos diferentes componentes. Sejam microsserviços, aplicações monolíticas ou sistemas legados, a troca de informações de forma eficiente e segura é crucial para o sucesso do projeto. Existem diversas técnicas e tecnologias que podem ser utilizadas para estabelecer a comunicação entre serviços, como o gerenciamento e estruturação de filas, que permite o envio e recebimento de mensagens de forma assíncrona e desacoplada, e a comunicação entre microsserviços, que envolve diferentes protocolos e padrões, como REST, gRPC (gRPC Remote Procedure Call) e message brokers.

A escolha da técnica ou tecnologia de comunicação depende dos requisitos da aplicação, como a necessidade de escalabilidade, performance, confiabilidade e segurança. O gerenciamento e estruturação de filas, por exemplo, é uma boa opção para sistemas que precisam lidar com grandes volumes de mensagens e garantir a entrega mesmo em caso de falhas. Já a comunicação entre microsserviços pode ser mais adequada para sistemas que precisam de alta performance e baixa latência. A área de comunicação entre serviços é fértil para o desenvolvimento de projetos, em que o aluno pode contribuir para o desenvolvimento de sistemas mais robustos, escaláveis e eficientes, capazes de lidar com os desafios da comunicação em ambientes distribuídos. Ao explorar as diferentes técnicas e tecnologias disponíveis, pode-se desenvolver soluções inovadoras para problemas complexos e contribuir para o avanço da área de desenvolvimento de software.

3.1. Gerenciamento e estruturas de filas

Em um mundo cada vez mais orientado a dados e serviços em nuvem, a eficiência no processamento de tarefas é fundamental. As filas de tarefas distribuídas desempenham um papel crucial nesse cenário, permitindo o gerenciamento, agendamento e execução de tarefas de forma assíncrona, otimizando o fluxo de trabalho e garantindo a escalabilidade do sistema. As filas são como canais por onde as tarefas fluem, compostas por elementos como tarefas (unidades de trabalho), produtores (responsáveis por adicionar tarefas à fila), consumidores (responsáveis por executar as tarefas), retentativas (controle da quantidade de vezes que uma tarefa pode ser executada), políticas de retenção (definição do tempo de vida de uma tarefa na fila) e monitoramento e registro (acompanhamento do desempenho das tarefas). Plataformas de Cloud Computing, como Google Cloud Tasks, Microsoft Azure Queue Service, Amazon SQS e IBM Cloud Message Hub, oferecem soluções robustas e escaláveis para o gerenciamento de filas. Além disso, existem soluções para implantação manual, como Apache Kafka, RabbitMQ, Redis, ZeroMQ, NATS, Apache ActiveMQ e Apache Pulsar, que podem ser utilizadas em servidores locais, data centers privados ou ambientes de hospedagem.

As filas de tarefas distribuídas encontram aplicações em diversas áreas, como e-commerce, processamento em lote, notificações, processamento de imagens e jogos online. No e-commerce, por exemplo, as filas podem ser utilizadas para processar pedidos, validar pagamentos e enviar notificações aos clientes, melhorando a experiência do usuário e a eficiência do sistema. No processamento em lote, as filas permitem dividir grandes volumes de dados em tarefas menores, que podem ser processadas em paralelo, otimizando o tempo de execução e evitando a sobrecarga do sistema. Em jogos online, as filas podem ser utilizadas para gerenciar o matchmaking, permitindo que os jogadores aguardem em uma fila até que uma partida seja formada, sem comprometer o desempenho do jogo. O campo de gerenciamento e estruturas de

filas apresenta inúmeras possibilidades de estudo. Pode-se, por exemplo, investigar a eficácia de diferentes plataformas e tecnologias de filas em diferentes cenários, como sistemas de alta disponibilidade, sistemas em tempo real e sistemas com grande volume de dados. Outro tema relevante é o estudo de algoritmos e técnicas de otimização para o gerenciamento de filas, como o balanceamento de carga, a priorização de tarefas e a gestão de recursos.

Referências

Chen, S.; Li, Q.; Zhou, M.; Abusorrah, A. 2021. Recent advances in collaborative scheduling of computing tasks in an edge computing paradigm. Sensors 21(3): 779.

Ding, D.; Fan, X.; Zhao, Y.; Kang, K.; Yin, Q.; Zeng, J. 2020. Q-learning based dynamic task scheduling for energy-efficient cloud computing. Future Generation Computer Systems 108: 361-371.

Kumar, R.; Soodan, B. S.; Kuaban, G. S.; Czekalski, P.; Sharma, S. 2021. Performance analysis of a cloud computing system using queuing model with correlated task reneging. Journal of Physics: Conference Series 2091(1): 012003.

Ma, Y.; Gao, X.; Bhatti, S.S.; Chen, G. 2024. Clustering Based Priority Queue Algorithm for Spatial Task Assignment in Crowdsourcing. IEEE Transactions on Services Computing 17(2): 452-465.

Potluri, S.; Rao, K. S. 2020. Optimization model for QoS based task scheduling in cloud computing environment. Indonesian Journal of Electrical Engineering and Computer Science 18(2): 1081-1088.

Zhou, J.; Tian, D.; Sheng, Z.; Duan, X.; Shen, X. 2021. Distributed task offloading optimization with queueing dynamics in multiagent mobile-edge computing networks. IEEE Internet of Things Journal 8(15):12311-12328.

3.2. Comunicação de microsserviços

Os microsserviços representam uma abordagem revolucionária no desenvolvimento de software, proporcionando maior flexibilidade, escalabilidade e facilidade de manutenção em sistemas complexos. Essa arquitetura modular, que divide a aplicação em pequenos serviços independentes, permite o uso de diferentes tecnologias e linguagens de programação, além de facilitar a escalabilidade e a evolução do sistema. No entanto, a comunicação entre microsserviços se torna um desafio crucial nessa abordagem. Diferentemente dos sistemas monolíticos, em que a comunicação ocorre internamente, os microsserviços precisam se comunicar por meio da rede, o que exige atenção especial para garantir a eficiência, a confiabilidade e a segurança da comunicação. Existem duas principais formas de comunicação entre microsserviços: síncrona e assíncrona. A comunicação síncrona, que geralmente utiliza o protocolo HTTP, é mais direta e imediata, mas pode apresentar problemas de latência e dependência da rede. Já a comunicação assíncrona, que utiliza sistemas de mensagens, como filas, permite o desacoplamento dos serviços e oferece maior flexibilidade, mas exige mecanismos de controle e gerenciamento das mensagens.

A escolha do protocolo de comunicação também é importante. O REST, por exemplo, é um protocolo amplamente utilizado e fácil de implementar, mas pode não ser a

melhor opção para sistemas que exigem alta performance. O gRPC, por sua vez, oferece maior desempenho, mas pode ser mais complexo de implementar. Já o Event Driven Messaging, baseado em filas, é uma boa opção para sistemas assíncronos e escaláveis. A comunicação entre microsserviços apresenta desafios como o gerenciamento de erros, a segurança e o monitoramento. É importante implementar mecanismos para lidar com falhas de comunicação, garantir a segurança dos dados transmitidos e monitorar o desempenho da comunicação para identificar e resolver problemas rapidamente. A área de comunicação entre microsserviços permite, por exemplo, investigar a eficácia de diferentes protocolos e padrões de comunicação em diferentes cenários, como sistemas de alta disponibilidade, sistemas em tempo real e sistemas com grande volume de dados. Outro tema relevante é o estudo de técnicas e ferramentas para o monitoramento e rastreamento da comunicação entre microsserviços, visando identificar gargalos e otimizar o desempenho do sistema.

Referências

Gote, A. 2024. E-Delivery Microservices Based Multi-Channel Communications Delivery Framework. International Journal of Computer Trends and Technology 72: 1-10.

Leines-Vite, L.; Pérez-Arriaga, J. C.; Limón, X. 2021. Confidentiality and Integrity Mechanisms for Microservices Communication. CS & IT Conference Proceedings 11(17): 1-16.

Loechel, L.; Akbayin, S. R.; Grünewald, E.; Kiesel, J.; Strelnikova, I.; Janke, T.; Pallas, F. 2024. Hook-in Privacy Techniques for gRPC-Based Microservice Communication. In International Conference on Web Engineering, 2024, Tampere, Finland. Proceeding… p. 215-229.

Niswar, M.; Safruddin, R. A.; Bustamin, A.; Aswad, I. 2024. Performance evaluation of microservices communication with REST, GraphQL, and gRPC. International Journal of Electronics and Telecommunication 70(2): 429-436.

Richardson, C. 2024. A pattern language for microservices microservices.io. Chris. Disponível em: https://microservices.io/patterns/. Acesso em: 26 jun. 2024.

Shehzadi, M.; Chaudhry, N. R.; Aslam, A.; Choudhary, R. 2024. Inter-Process Communication Amongst Microservices. Pakistan Journal of Scientific Research 4(1): 1-9.

4. Aplicativos e soluções para dispositivos móveis

O desenvolvimento de aplicativos e soluções para dispositivos móveis é uma área em constante crescimento e transformação, impulsionada pela crescente demanda por soluções inovadoras e personalizadas para smartphones e tablets. Essa área abrange desde a concepção e design da interface do usuário até a implementação da lógica de negócio e a integração com serviços externos. Para desenvolver aplicativos e soluções para dispositivos móveis, é fundamental conhecer as diferentes plataformas móveis, como Android e iOS, e as linguagens de programação mais utilizadas, como Java, Kotlin, Swift e Objective-C. Além disso, é importante entender os conceitos de arquitetura mobile, que envolvem a organização e estruturação do código, a gestão de dados, a comunicação com serviços externos e a otimização do desempenho da

aplicação. A área de aplicativos e soluções para dispositivos móveis permite investigar novas abordagens para o design de interfaces de usuário, explorar o uso de tecnologias emergentes, como inteligência artificial e realidade aumentada, ou ainda, desenvolver soluções inovadoras para problemas específicos, como a otimização do consumo de bateria ou a segurança de dados.

4.1. Arquitetura Mobile

A evolução constante dos dispositivos móveis exige que o desenvolvimento de aplicativos acompanhe as novas tecnologias, visando sempre a eficiência e a qualidade. Antes de iniciar o desenvolvimento, é fundamental conhecer as duas principais plataformas de sistemas operacionais para dispositivos móveis: iOS, exclusivo para dispositivos Apple, e Android, um sistema de código aberto que roda em diversos dispositivos. A escolha da linguagem de programação depende da plataforma alvo. Para iOS, as opções nativas são Swift e Objective-C, enquanto para Android, Java e Kotlin são as mais utilizadas. Alternativamente, o desenvolvimento multiplataforma com JavaScript, utilizando frameworks como React Native ou Flutter, permite criar aplicativos para ambas as plataformas com uma única base de código. A abordagem de desenvolvimento também é um fator importante. O desenvolvimento nativo permite utilizar todos os recursos do dispositivo e obter o máximo desempenho, enquanto os aplicativos web progressivos (PWA) oferecem uma experiência similar à de um aplicativo nativo, mas são executados no navegador. A abordagem multiplataforma, por sua vez, permite compartilhar código entre plataformas, reduzindo o tempo e o custo de desenvolvimento.

Uma vez definida a linguagem e a abordagem, é hora de pensar na arquitetura mobile, que envolve diversos aspectos. A interface do usuário (UI) é a primeira impressão que o usuário terá do aplicativo, e deve ser intuitiva e agradável. A lógica de negócios é responsável pela comunicação com serviços externos, como servidores e bancos de dados, e pela implementação das regras de negócio da aplicação. O armazenamento de dados é outro ponto crucial, podendo ser feito localmente no dispositivo, na nuvem ou em ambos, dependendo dos requisitos da aplicação. A segurança e privacidade são aspectos essenciais, que envolvem a criptografia de dados, a autenticação de usuários e a definição de políticas de acesso aos recursos do dispositivo. A área de arquitetura mobile oferece vários caminhos para o aluno. Pode-se, por exemplo, investigar novas abordagens para o design de interfaces de usuário, como o uso de design responsivo e adaptativo para diferentes tamanhos de tela e dispositivos. A otimização do desempenho da aplicação, tanto em termos de velocidade quanto de consumo de recursos, também é um tema relevante. A segurança e privacidade em aplicativos móveis são áreas de pesquisa em constante evolução, com novos desafios e soluções surgindo a cada dia. A investigação de técnicas de criptografia, autenticação e proteção de dados pode contribuir para a criação de aplicativos mais seguros e confiáveis.

Referências

Das, P. K.; Paul, H. S.; Tultul, M. H. 2023. Development of a Mobile Application: From University Website to Mobile App. Journal of Ubiquitous Computing and Communication Technologies 5(2): 203-219.

Fentaw, A.E. 2020. Cross platform mobile application development: a comparison study of React Native Vs Flutter. Disponível em: <https://jyx.jyu.fi/bitstream/handle/123456789/70969/1/URN%3ANBN%3Afi%3Ajyu-202006295155.pdf>. Acesso

em: 12 jul. 2024.

Mascetti, S.; Ducci, M.; Cantù, N.; Pecis, P.; Ahmetovic, D. 2021.Developing accessible mobile applications with cross-platform development frameworks. In 23rd International ACM SIGACCESS Conference on Computers and Accessibility, 2021, Virtual Event, New York, NY, USA. Proceedings… p. 1-5.

Zaręba, G.; Zarębski, M.; Smołka, J. 2024. C++ and Kotlin performance on Android–a comparative analysis. Journal of Computer Sciences Institute 30: 21-25.

5. Infraestrutura e gerenciamento de ambientes de TI

A infraestrutura e o gerenciamento de ambientes de TI são essenciais para o funcionamento eficiente e seguro de sistemas e aplicações, englobando desde a criação de servidores e redes até a segurança dos dados. A infraestrutura como código (IaC) automatiza a criação e o gerenciamento de ambientes de TI, enquanto o Cloud Computing oferece recursos sob demanda e flexíveis. O modelo Serverless permite que desenvolvedores foquem na lógica das aplicações sem se preocupar com a gestão de servidores. Containers e Kubernetes facilitam a implantação, escalabilidade e portabilidade do software. O campo oferece várias oportunidades de pesquisa, como a eficácia da IaC na redução de custos e aumento da agilidade, comparação de provedores de Cloud Computing em termos de serviços, preços e desempenho, estudo das arquiteturas serverless e suas vantagens e desvantagens, e a utilização de containers e Kubernetes para otimizar desempenho e escalabilidade em sistemas complexos.

5.1. Infra As a Code (IaC)

A Infraestrutura como Código (IaC) é uma abordagem que permite criar e gerenciar ambientes de TI utilizando código, em vez de processos manuais. Essa prática, impulsionada pelo Cloud Computing, possibilita a criação e o descarte de ambientes de forma rápida e automatizada, por meio de scripts e templates que definem a configuração desejada. Os princípios fundamentais da IaC incluem a automação, o versionamento, a consistência e a gestão de ambientes. A automação permite criar e destruir ambientes de forma rápida e eficiente, enquanto o versionamento garante o controle das alterações e a possibilidade de reverter para versões anteriores. A consistência garante que o ambiente seja sempre criado da mesma forma, evitando erros de configuração, e a gestão de ambientes permite criar e gerenciar diferentes ambientes, como desenvolvimento, teste e produção, de forma centralizada e organizada.

Ferramentas como Ansible, Terraform, Chef e Puppet são amplamente utilizadas para implementar a IaC, oferecendo recursos para provisionar servidores, configurar redes, gerenciar armazenamento de dados e aplicar políticas de segurança. Essas ferramentas se integram com as principais plataformas de cloud computing, facilitando a criação e o gerenciamento de ambientes em nuvem. A área de IaC oferece um terreno fértil para a pesquisa e o desenvolvimento de novas soluções. O aluno pode se aprofundar, por exemplo, em comparar diferentes ferramentas de IaC, avaliando seus recursos, desempenho e facilidade de uso. A investigação de práticas de IaC para provisionamento de nuvem, como a otimização de custos e a garantia da segurança, também é um tema relevante. A segurança em ambientes de IaC é um aspecto crucial, pois a automação da infraestrutura pode expor vulnerabilidades se não for implementada corretamente. A integração da IaC com práticas de DevOps, como a integração e entrega contínua, pode otimizar o fluxo de trabalho e garantir a qualidade do código e da infraestrutura. A

adoção de IaC em organizações pode ser estudada para avaliar os benefícios em termos de redução de custos, aumento da eficiência e melhoria da qualidade dos serviços de TI.

Referências

Batu, J. 2024. Implementing Infrastructure-as-Code with Cloud Disaster Recovery Strategies. International Journal of Computer Trends and Technology 72: 41-45.

Brikman, Y. 2019. Terraform: Up & Running: Writing Infrastructure as Code. 2ed. Editora: O'Reilly Media, Sebastopol, CA, USA.

Chittibala, D. 2024. Infrastructure as Code (IaC) and Its Role in Achieving DevOps Goals. International Journal of Science and Research (IJSR) 12(1): 1258-2262.

Hasan, M.M.; Bhuiyan, F.A.; Rahman, A. 2020. Testing practices for infrastructure as code. In 1st ACM SIGSOFT International Workshop on Languages and Tools for Next-Generation Testing, 2020, Virtual Event, New York, NY, USA. Proceedings… p. 7-12.

Morris, K. 2021. Infrastructure as Code: Dynamic Systems for the Cloud Age. 2ed. Editora: O'Reilly Media, Sebastopol, CA, USA.

Özdoğan, E.; Ceran, O.; Üstündağ, M. T. 2023. Systematic Analysis of Infrastructure as Code Technologies. Gazi University Journal of Science Part A: Engineering and Innovation 10(4): 452-471.

Sokolowski, D. 2022. Infrastructure as code for dynamic deployments. In 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2022, Singapore. Proceedings… p. 1775-1779.

5.2. Cloud Computing (IaaS, Paas, SaaS)

Cloud Computing, ou computação em nuvem, revolucionou a forma como utiliza-se a tecnologia, oferecendo serviços e recursos de TI por meio da internet. Essa abordagem permite que empresas e indivíduos acessem infraestrutura, plataformas e softwares sob demanda, sem a necessidade de investir em hardware e software próprios. Existem três principais modelos de serviços em Cloud Computing: IaaS (Infraestrutura como Serviço), PaaS (Plataforma como Serviço) e SaaS (Software como Serviço). O IaaS oferece a infraestrutura básica, como servidores virtuais, armazenamento e redes, enquanto o PaaS fornece uma plataforma completa para desenvolvimento e implantação de aplicativos. Já o SaaS oferece softwares prontos para uso, acessados diretamente pela internet. A escolha do modelo de serviço depende das necessidades e do nível de controle desejado. O IaaS oferece maior controle sobre a infraestrutura, mas exige mais conhecimento técnico para gerenciá-la. O PaaS facilita o desenvolvimento e a implantação de aplicativos, mas oferece menos flexibilidade na configuração da infraestrutura. O SaaS é a opção mais simples e acessível, mas oferece o menor nível de controle sobre o software.

O campo de Cloud Computing convida à exploração e aprofundamento em diversas frentes. O estudante pode, por exemplo, comparar os diferentes modelos de serviços, analisando suas vantagens e desvantagens em diferentes cenários. A segurança em cloud computing é um tema crucial, que envolve a proteção de dados, a gestão de identidades e o controle de acesso. A migração de sistemas legados para a nuvem

também é um desafio importante, que exige planejamento e cuidado para garantir a continuidade dos serviços. O custo e a otimização de recursos em cloud computing são temas relevantes para empresas que buscam reduzir seus gastos com TI e aproveitar ao máximo os recursos disponíveis na nuvem. A pesquisa nessa área pode envolver a análise de diferentes modelos de precificação, a otimização do uso de recursos e a identificação de oportunidades para reduzir custos. A estratégia de recuperação de desastres em nuvem é outro tema importante, pois garante a continuidade dos negócios em caso de falhas ou incidentes. A pesquisa nessa área pode envolver a análise de diferentes soluções de backup e recuperação, a criação de planos de contingência e a realização de testes para garantir a eficácia da estratégia.

Referências

Mohammed, C.M.; Zeebaree, S.R. 2021. Sufficient comparison among cloud computing services: IaaS, PaaS, and SaaS: A review. International Journal of Science and Business 5(2): 17-30.

Nadeem, F. 2022. Evaluating and ranking cloud IaaS, PaaS and SaaS models based on functional and non-functional key performance indicators. IEEE Access 10: 63245-63257.

Wulf, F.; Lindner, T.; Strahringer, S.; Westner, M. 2021. IaaS, PaaS, or SaaS? The why of cloud computing delivery model selection: Vignettes on the post-adoption of cloud computing. In 54th Hawaii International Conference on System Sciences, 2021, Kauai, Hawaii, USA. Proceedings… p. 6285-6294.

5.3. Serverless

A arquitetura serverless revoluciona o desenvolvimento de software ao abstrair a necessidade de gerenciar servidores, permitindo que os desenvolvedores se concentrem na lógica da aplicação e nas funcionalidades do front-end e back-end. Diferentemente da arquitetura tradicional, em que é preciso lidar com a configuração, manutenção e escalabilidade dos servidores, o serverless utiliza serviços de terceiros para executar funções sob demanda, simplificando o processo de desenvolvimento e implantação. Essa abordagem oferece diversos benefícios, como a redução de custos, pois os provedores geralmente oferecem uma faixa de requisições gratuitas, e a baixa manutenção, já que o provedor é responsável pela gestão da infraestrutura. Além disso, o serverless permite um menor tempo de lançamento no mercado, pois os desenvolvedores podem focar na criação da aplicação, e a escalabilidade é simplificada, pois a arquitetura pode se ajustar automaticamente à demanda.

No entanto, a arquitetura serverless também apresenta desafios, como o custo elevado com altas cargas de trabalho, as limitações impostas pela falta de controle sobre o servidor e os desafios de teste, integração e deployment em um ambiente diferente. A área de serverless oferece diversas oportunidades com as quais o aluno pode trabalhar. É possível comparar diferentes plataformas serverless, como AWS Lambda, Google Cloud Functions e Azure Functions, avaliando seus recursos, desempenho e custos. O desenvolvimento de aplicações serverless, a migração de aplicações tradicionais para a arquitetura serverless e o monitoramento e depuração em ambientes serverless são outros temas relevantes para pesquisa. A integração de microsserviços com serverless também é um campo promissor, permitindo combinar as vantagens de ambas as abordagens para criar sistemas ainda mais flexíveis e escaláveis.

Referências

Hamza, M.; Akbar, M. A.; Capilla, R. 2023. Understanding Cost Dynamics of Serverless Computing: An Empirical Study. In International Conference on Software Business, 2023, Lahti, Finland. Proceedings… p. 456-470.

Katzer, J. 2020. Learning Serverless: Design, Develop, and Deploy with Confidence. 1ed. O'Reilly Media, Inc., Sebastopol, CA, USA.

Krishnamurthi, R.; Kumar, A.; Gill, S.S.; Buyya, R. (eds.). 2023. Serverless Computing: Principles and Paradigms. Springer International Publishing, USA.

Nastic, S. 2024. Self-Provisioning Infrastructures for the Next Generation Serverless Computing. SN Computer Science 5(6): 678.

Newman, S. 2022. Criando Microsserviços: Projetando Sistemas com Componentes Menores e Mais Especializados. 2ed. Editora Novatec, São Paulo, SP, Brasil.

Shafiei, H.; Khonsari, A.; Mousavi, P. 2022. Serverless computing: a survey of opportunities, challenges, and applications. ACM Computing Surveys 54(11): 1-32.

5.4. Containers (Kubernetes)

No cenário da infraestrutura de TI, containers e Kubernetes emergem como tecnologias-chave para o desenvolvimento e implantação de aplicações modernas. Ao contrário das máquinas virtuais, que emulam um sistema operacional completo, os containers isolam apenas as bibliotecas, arquivos binários e a aplicação em si, resultando em maior eficiência e agilidade no desenvolvimento e na implantação. Docker e Podman são duas das principais tecnologias de containers, permitindo a criação de ambientes de desenvolvimento e execução de aplicações de forma isolada e independente do sistema operacional hospedeiro. Por meio de um Dockerfile, um script que define as instruções para a criação do ambiente, é possível gerar imagens de container que podem ser executadas em qualquer sistema compatível com Docker. As vantagens do uso de containers incluem a portabilidade, o isolamento, a eficiência, a replicabilidade e a efemeridade. A portabilidade permite que a mesma imagem de container seja executada em diferentes sistemas operacionais, o isolamento garante que cada container seja um processo independente, a eficiência se dá pela proximidade do container com o hardware, a replicabilidade permite a criação de múltiplas instâncias do mesmo container e a efemeridade facilita a criação e o descarte de containers de forma rápida e eficiente.

Kubernetes, como orquestrador de containers, revoluciona a gestão de aplicações, automatizando implantação, escalabilidade e gerenciamento em diversos ambientes. Essa plataforma robusta permite a distribuição eficiente de containers em múltiplas máquinas, balanceamento de carga inteligente, automação de tarefas e recuperação rápida de falhas. É possível explorar a fundo as diferenças entre ferramentas como Docker Swarm e Kubernetes, comparando seus recursos, desempenho e segurança. O desenvolvimento de aplicações nativas para Kubernetes, a investigação de técnicas avançadas de escalabilidade e a garantia da resiliência em cenários de falha são outros temas cruciais. A segurança em ambientes de containers e a integração com práticas DevOps, como CI/CD, abrem novas perspectivas para a otimização do ciclo de vida

do software. Além disso, a análise de logs e o monitoramento contínuo de aplicações em Kubernetes permitem identificar gargalos e aprimorar o desempenho do sistema, garantindo uma experiência fluida e eficiente para os usuários.

Referências

Aqasizade, H.; Ataie, E.; Bastam, M. 2024. Kubernetes in Action: Exploring the Performance of Kubernetes Distributions in the Cloud. Disponível em: https://arxiv.org/abs/2403.01429. Acesso em: 12 jul. 2024.

Khan, H.H.; Zubair, S.; Nasim, F.; Akhter, S.; Ghazanfar, M.N.; Azeem, S. 2024. Role of Kubernetes in DevOps Technology for the Effective Software Product Management. Journal of Computing & Biomedical Informatics 7(01): 313-327.

Nadaf, S.R.; Krishnappa, H.K. 2023. Kubernetes in Microservices. International Journal of Advanced Science and Computer Applications 2(1): 7-18.

Poulton, N. 2023. The kubernetes book. Disponível em: <https://it-systems.su/wp-content/uploads/2022/03/nigel-poulton-the-kubernetes-book-2020.pdf>. Acesso em: 12 jul. 2024.

Vasireddy, I.; Ramya, G.; Kandi, P. 2023. Kubernetes and Docker Load Balancing: State-of-the-Art Techniques and Challenges. International Journal of Innovative Research in Engineering & Management 10(6): 49-54.

6. Estruturação de Banco de dados

A estruturação de bancos de dados é uma área fundamental na Engenharia de Software, responsável por organizar e gerenciar os dados de forma eficiente e segura. Essa área abrange diversas tecnologias e abordagens, desde os tradicionais bancos de dados relacionais (SQL) até os bancos de dados NoSQL, cada um com suas características e aplicações específicas. Dentro da temática de estruturação de bancos de dados, são destacados três tópicos principais: Engenharia de Dados, Data Warehouse e a comparação entre NoSQL e SQL. A Engenharia de Dados se concentra na coleta, transformação e armazenamento de dados, garantindo que eles estejam disponíveis e acessíveis para análise e tomada de decisões. O Data Warehouse, por sua vez, é um repositório centralizado de dados históricos, utilizado para análise e geração de relatórios. A comparação entre NoSQL e SQL permite avaliar as vantagens e desvantagens de cada modelo, auxiliando na escolha da tecnologia mais adequada para cada projeto.

O aluno pode explorar as fronteiras da Engenharia de Dados, investigando tecnologias emergentes como plataformas de Big Data para lidar com volumes massivos de dados e ferramentas de processamento em tempo real para obter insights instantâneos. A otimização de Data Warehouses, buscando aprimorar o desempenho e a escalabilidade, é outro desafio constante que demanda investigação. Além disso, a análise comparativa entre bancos de dados NoSQL e SQL, considerando seus pontos fortes e fracos em diferentes contextos, pode gerar conhecimento valioso para a escolha da melhor solução para cada problema.

6.1. Engenharia de Dados

A engenharia de dados é uma área fundamental na Engenharia de Software, responsável por todo o ciclo de vida dos dados, desde a coleta e transformação até o armazenamento e disponibilização para análise e tomada de decisões. Essa área abrange desde o planejamento e desenho da infraestrutura de dados até a implementação e manutenção dos sistemas que garantem a qualidade, a disponibilidade e a segurança dos dados. Dentro da engenharia de dados, diversos aspectos podem ser explorados em pesquisas, como a coleta de dados de diferentes fontes e tipos, a implementação de processos de ETL (Extração, Transformação e Carga), a organização e o processamento de dados em larga escala (Big Data), a escolha de ferramentas e tecnologias adequadas, como Hadoop, Spark, Airflow e NiFi, e a segurança dos dados em ambientes de nuvem.

Pode-se mergulhar em temas como a otimização de arquiteturas de dados para empresas de diferentes portes e setores, a análise comparativa de ferramentas e processos de ETL para maximizar a eficiência e reduzir custos, a avaliação do impacto de diferentes tecnologias no desempenho e escalabilidade em sistemas de Big Data, ou ainda, a investigação de novas soluções para garantir a segurança e a privacidade de dados em ambientes cada vez mais complexos e distribuídos.

Referências

Ahmed, N.; Barczak, A.L.; Susnjak, T.; Rashid, M.A. 2020. A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench. Journal of Big Data 7(1): 110.

Davoudian, A.; Liu, M. 2020. Big data systems: A software engineering perspective. ACM Computing Surveys (CSUR) 53(5): 1-39.

Mbata, A.; Sripada, Y.; Zhong, M. 2024. A Survey of Pipeline Tools for Data Engineering. Disponível em: https://arxiv.org/abs/2406.08335. Acesso em: 12 jul. 2024.

Mostafaeipour, A.; Jahangard Rafsanjani, A.; Ahmadi, M.; Arockia Dhanraj, J. 2021. Investigating the performance of Hadoop and Spark platforms on machine learning algorithms. The Journal of Supercomputing 77: 1273-1300.

Volk, M.; Staegemann, D.; Bosse, S.; Häusler, R; Turowski, K. 2020. Approaching the (Big) Data Science Engineering Process. Disponível em: <https://www.scitepress.org/Papers/2020/95698/95698.pdf>. Acesso em: 12 jul. 2024
.

6.2. Data Warehouse

No contexto da estruturação de bancos de dados, o Data Warehouse (DW) emerge como um repositório centralizado que consolida dados de diversas fontes, permitindo análises aprofundadas e auxiliando na tomada de decisões estratégicas. Essa ferramenta crucial para empresas e organizações armazena dados históricos, transformando-os em informações valiosas para o negócio. A implementação de um DW envolve diversas etapas, desde a coleta e integração de dados de diferentes fontes até a criação de modelos de dados que facilitem a análise e a geração de relatórios. O processo de ETL (Extração, Transformação e Carga) é fundamental para garantir a qualidade e a consistência dos dados armazenados no DW. Além disso, a escolha de ferramentas de Business Intelligence (BI) e de análise de dados é crucial para que os usuários possam

explorar e visualizar os dados de forma eficiente.

As arquiteturas de DW, adaptadas para diferentes volumes e tipos de dados, oferecem um ponto de partida para a investigação, buscando otimizar o desempenho e a escalabilidade. A eficiência e a qualidade do processo de ETL (Extração, Transformação e Carga) são cruciais para garantir a confiabilidade dos dados armazenados, abrindo espaço para pesquisas sobre as melhores ferramentas e técnicas. A integração do DW com tecnologias como Big Data e Machine Learning permite a criação de sistemas analíticos mais poderosos, expandindo as possibilidades de pesquisa em áreas como mineração de dados e inteligência artificial. Além disso, a segurança e a privacidade dos dados, especialmente em ambientes de nuvem, exigem atenção especial, tornando-se um tema relevante para pesquisas que busquem garantir a integridade e a confidencialidade das informações armazenadas.

Referências

Chhabra, R.; Kumar, P.; Pahwa, P. 2016. An approach to Design Object Oriented Data Warehouse. International Journal of Research and Engineering 3(3): 54-56.

Majid, M. E.; Marinova, D.; Hossain, A.; Chowdhury, M. E.; Rummani, F. 2024. Use of Conventional Business Intelligence (BI) Systems as the Future of Big Data Analysis. American Journal of Information Systems 9(1): 1-10.

Olaoye, F.; Potter, K. 2024. Business Intelligence (BI) and Analytics Software: Empowering Data-Driven Decision-Making. Disponível em: https://easychair.org/publications/preprint/cR8v. Acesso em: 12 jul. 2024.

Wang, B. 2022. Research on the Value Orientation of ELT Integration Based on Data Mining under the Background of Megadata. Mathematical Problems in Engineering 2022(1): 6774977.

6.3. NoSQL x SQL

A escolha do modelo de banco de dados é uma decisão crucial que impacta diretamente a estruturação, organização e acesso aos dados. Os dois principais modelos são o SQL (Structured Query Language) e o NoSQL (Not Only SQL), cada um com suas características, vantagens e desvantagens. O SQL é uma linguagem de consulta estruturada utilizada para bancos de dados relacionais, que armazenam dados em tabelas com linhas e colunas, relacionadas entre si por meio de chaves. Essa estrutura permite realizar consultas complexas, como junções, filtros e cálculos, tornando-o ideal para aplicações que exigem integridade e consistência dos dados.

O NoSQL, por sua vez, engloba diversos modelos de bancos de dados não relacionais, como chave-valor, documento, coluna e grafo. Esses modelos oferecem maior flexibilidade e escalabilidade, sendo adequados para lidar com grandes volumes de dados e aplicações com requisitos de alta performance e disponibilidade. A escolha entre SQL e NoSQL depende das necessidades específicas da aplicação. Enquanto o SQL, com sua estrutura rígida e garantias ACID (Atomicidade, Consistência, Isolamento e Durabilidade), é ideal para sistemas que demandam transações complexas e consistência de dados, o NoSQL, com sua flexibilidade e escalabilidade, destaca-se em cenários com grande volume de dados não estruturados e necessidade de adaptação rápida. O estudo aprofundado da estruturação de bancos de dados permite ao aluno explorar as nuances de cada modelo, investigando a adequação de cada um a diferentes

tipos de aplicações, avaliando o desempenho e a escalabilidade em diversos contextos, analisando a relação entre SQL e as propriedades ACID, e explorando o potencial de ambos os modelos em cenários de Big Data.

Referências

Chakraborty, S.; Paul, S.; Hasan, K. A. 2021. Performance comparison for data retrieval from nosql and sql databases: a case study for covid-19 genome sequence dataset. In 2nd International Conference on Robotics, electrical and signal processing techniques (ICREST), 2021, Dhaka, Bangladesh. Proceedings… p. 324-328.

Khan, M.Z.; Zaman, F.U.; Adnan, M., Imroz, A.; Rauf, M.A.; Phul, Z. 2023. Comparative Case Study: An Evaluation of Performance Computation Between SQL And NoSQL Database. Journal of Software Engineering 1(2): 14-23.

Khan, W.; Kumar, T.; Zhang, C.; Raj, K.; Roy, A.M.; Luo, B. 2023. SQL and NoSQL database software architecture performance analysis and assessments—a systematic literature review. Big Data and Cognitive Computing 7(2): 97.

Nield, T. 2016. Introdução à Linguagem SQL: Abordagem Prática Para Iniciantes. Editora Novatec, São Paulo, SP, Brasil.

Oliveira, V. F. D.; Pessoa, M. A. D. O.; Junqueira, F.; Miyagi, P. E. 2022. SQL and NoSQL databases in the context of industry 4.0. Machines 10(1): 1-26.

Tanimura, C. 2022. SQL Para Análise de Dados: Técnicas Avançadas Para Transformar Dados em Insights. Editora Novatec, São Paulo, SP, Brasil.

7. Segurança da Informação

A proteção de dados e ativos digitais é fundamental para empresas e indivíduos. A área de segurança da informação abrange diversas técnicas e ferramentas para garantir a confidencialidade, integridade e disponibilidade da informação, protegendo-a contra ameaças como ataques cibernéticos, roubo de dados e fraudes. Pode-se destacar dois tópicos principais: os conceitos básicos de segurança da informação, que abordam os princípios, as ameaças e as melhores práticas para proteger os dados, e a Lei Geral de Proteção de Dados [LGPD], que estabelece regras e diretrizes para o tratamento de dados pessoais no Brasil.

São oferecidas diversas oportunidades para desenvolvimento acadêmico. Um exemplo seria investigar novas tecnologias e ferramentas para a proteção de dados, como criptografia, firewalls e sistemas de detecção de intrusão. A análise de riscos e vulnerabilidades em sistemas e aplicações também é um tema relevante, permitindo identificar pontos fracos e propor medidas de segurança adequadas. O estudo da LGPD e sua aplicação em diferentes contextos, como empresas, órgãos públicos e instituições de ensino, também é um tema importante, pois permite avaliar o impacto da lei na proteção de dados pessoais e propor soluções para garantir a conformidade. A pesquisa na área de segurança da informação pode contribuir para o desenvolvimento de sistemas e processos mais seguros, protegendo os dados e ativos digitais contra ameaças e garantindo a privacidade e a confiança dos usuários.

7.1. Fundamentos de Segurança da Informação

Para compreender a segurança da informação, é fundamental entender seus pilares e como se relacionam com as ameaças existentes. A confidencialidade garante que apenas pessoas autorizadas tenham acesso à informação, a integridade assegura que a informação não seja alterada ou corrompida durante o seu ciclo de vida e a disponibilidade garante que a informação esteja acessível quando necessário. Esses pilares são interdependentes e a quebra de um deles pode comprometer a segurança da informação como um todo. As ameaças à segurança da informação são diversas, desde softwares maliciosos (malware), como ransomware e spyware, até ataques de engenharia social, como phishing e quid pro quo, que exploram a vulnerabilidade humana.

Ataques cibernéticos como os de negação de serviço (DoS) representam uma ameaça crescente à disponibilidade de sistemas e causam perdas financeiras significativas. Pode-se explorar a segurança em dispositivos IoT, cada vez mais presentes em nossas vidas, ou desenvolver estratégias eficazes para mitigar ataques DoS. A análise de código estático em pipelines de CI/CD, a criação de plataformas de treinamento gamificadas e a aplicação de técnicas de criptografia e anonimização de dados são exemplos de temas relevantes que podem ser abordados em projetos de pesquisa, contribuindo para a proteção de sistemas e dados sensíveis.

Referências

Barbosa, J.S.; Borges, D.; Oliveira, D.C.; Jesus, D.C.; Miranda, W.F. 2021. A proteção de dados e segurança da informação na pandemia COVID-19: contexto nacional. Research, Society and Development 10(2): e40510212557-e40510212557.

Hintzbergen, J.; Hintzbergen, K.; Smulders, A.; Baars, H. 2018. Fundamentos de Segurança da Informação: Com base na ISO 27001 e na ISO 27002. Brasport, Rio de Janeiro, RJ, Brasil.

Korzhuk, V.M.; Arustamov, S.A. 2024. Foundation of Information Security. Disponível em: <https://books.ifmo.ru/file/pdf/3294.pdf>. Acesso em: 12 jul. 2024.

Neves, D.L.F.; Almeida Lopes, T.S.; Pavani, G.C.; Sales, R.M. 2021. A segurança da informação de encontro às conformidades da LGPD. Revista Processando o Saber 13: 186-198.

Somepalli, S. H.; Tangella, S. K. R.; Yalamanchili, S. 2020. Information security management. HOLISTICA–Journal of Business and Public Administration 11(2): 1-16.

Yesimov, S.; Borovikova, V. 2023. Methodological foundations of information security research. Social and Legal Studios 6(1): 49-55.

7.2. Lei Geral de Proteção de Dados

A LGPD é uma legislação brasileira que entrou em vigor em 2018, com o objetivo de proteger a privacidade e os dados pessoais dos cidadãos. A LGPD estabelece regras para o tratamento de dados, define os direitos dos titulares dos dados e as responsabilidades das empresas que coletam e utilizam esses dados. A LGPD abrange diversos aspectos do tratamento de dados, desde a coleta e o armazenamento até o compartilhamento e

a eliminação. Ainda, exige que as empresas obtenham o consentimento explícito dos titulares para coletar seus dados, informem sobre a finalidade do tratamento e garantam a segurança das informações.

O aluno pode, dentro do campo da Lei Geral de Proteção de Dados (LGPD), explorar a aplicação da LGPD em diferentes setores, como saúde, educação e finanças, avaliando seus impactos e desafios específicos. A análise das práticas de empresas e organizações em relação à lei pode revelar oportunidades de melhoria e inovação. A pesquisa pode também se aprofundar nos mecanismos de adequação à LGPD, como a criação de políticas de privacidade, treinamentos e auditorias de dados, além de investigar o papel da ANPD na aplicação da lei e na proteção dos direitos dos titulares de dados.

Referências

Alencar, L. 2023. Lei Geral de Proteção de Dados – LGPD e segurança na internet. Revista Judicial Brasileira 3: 429-447.

Araújo Neto, R.J.; Aguiar, J.J.B. 2024. The impacts of the General Data Protection Law (LGPD) on information security: a literature review. Revista de Gestão e Secretariado 15(2): e3442-e3442.

Brasil. 2012a. Lei nº 12.735, de 30 de novembro de 2012. Altera o Decreto de lei n° 2.848, de 7 de dezembro de 1940 - Código Penal, o Decreto de lei n° 1.001, de 21 de outubro de 1969 - Código Penal Militar, e a Lei n° 7.716, de 5 de janeiro de 1989, para tipificar condutas realizadas mediante uso de sistema eletrônico, digital ou similares, que sejam praticadas contra sistemas informatizados e similares e dá outras providências. Diário Oficial da União, Brasília, 30 nov. 2012. Seção 1, p. 1.

Brasil. 2012b. Lei nº 12.737, de 30 de novembro de 2012. Dispõe sobre a tipificação criminal de delitos informáticos; altera o Decreto Lei n° 2.848, de 7 de dezembro de 1940 - Código Penal; e dá outras providências. Também determina a instalação de delegacias especializadas para o combate de crimes digitais. Diário Oficial da União, Brasília, 30 nov. 2012. Seção 1, p. 1.

Brasil. 2014. Lei nº 12.965, de 23 de abril de 2014. Estabelece princípios, garantias, direitos e deveres para o uso da Internet no Brasil. Diário Oficial da União, Brasília, 23 abr. 2014. Seção 1, p. 1.

Brasil. 2018. Lei nº 13.709, de 17 de agosto de 2018. Lei Geral de Proteção de Dados Pessoais (LGPD). Diário Oficial da União, Brasília, 14 ago. 2018. Seção 1, p. 1.

Santos, L.; Faddoul, A.; Silva, O.; Silva, C.; Miguel, J. 2024. O impacto da lgpd nas políticas públicas de infraestrutura no Brasil. Revista Foco. Revista Foco (Interdisciplinary Studies Journal) 17(6).

Sousa, V.L. de; Morais, R. de 2024. Os impactos da lei geral de proteção de dados (LGPD) no sistema de saúde brasileiro. Revista JRG de Estudos Acadêmicos 7(14): e141129-e141129.

8. A Grande área de Planejamento Estratégico

A área de planejamento estratégico, gestão e entrega de projetos em engenharia de software abrange um conjunto de práticas e ferramentas essenciais para o sucesso de projetos de desenvolvimento de software. Essa área engloba desde a concepção e planejamento do projeto até a sua execução, entrega e manutenção, buscando garantir a qualidade, a eficiência e a satisfação do cliente. Dentro dessa grande área, é possível destacar cinco temas principais: DevOps, Design Thinking, metodologias ágeis, liderança e gestão de mudança. O Design Thinking é uma abordagem centrada no usuário que busca soluções inovadoras para problemas complexos, enquanto as metodologias ágeis, como Scrum e Kanban, promovem a colaboração, a flexibilidade e a entrega contínua de valor. O DevOps, por sua vez, integra as equipes de desenvolvimento e operações, buscando automatizar e otimizar o processo de entrega de software. A liderança e a gestão de mudança são fundamentais para garantir a adesão e o engajamento das equipes, além de lidar com os desafios e as resistências que surgem durante o processo de transformação.

O aluno pode explorar a aplicação de metodologias inovadoras, como o Design Thinking, em diferentes fases do projeto, analisar a eficácia de diferentes metodologias ágeis em contextos variados, investigar a adoção de práticas DevOps em empresas de diferentes portes e setores, ou ainda, aprofundar-se no papel crucial da liderança e da gestão de mudança na implementação de novas práticas e ferramentas.

8.1. Design Thinking

O Design Thinking é uma metodologia de resolução de problemas complexos que enfatiza a empatia, a colaboração e a experimentação. Originado na empresa de design IDEO e na Stanford Design School, o Design Thinking foi popularizado por autores como Richard Buchanan e Tim Brown, que exploraram seus conceitos e aplicações em diversas áreas. O processo de Design Thinking envolve cinco fases principais: empatia, definição do problema, ideação, prototipação e testes. A empatia é a base do processo, buscando compreender as necessidades e perspectivas dos usuários. A definição do problema envolve a análise das informações coletadas e a formulação de um problema claro e preciso. A ideação é a fase de geração de ideias, buscando soluções inovadoras e criativas. A prototipação envolve a criação de modelos e protótipos para testar as ideias e a implementação é a fase de colocar as soluções em prática e avaliar seus resultados.

O Design Thinking, com sua versatilidade, transcende fronteiras e se aplica a uma ampla gama de desafios, desde a criação de produtos e serviços inovadores até a resolução de problemas sociais complexos. Essa metodologia colaborativa e centrada no ser humano capacita equipes multidisciplinares a desenvolver soluções eficazes e disruptivas. O campo do Design Thinking possui inúmeras oportunidades para explorar sua aplicação em diversos contextos, como empresas, instituições de ensino e organizações do terceiro setor. A análise do impacto do Design Thinking na resolução de problemas complexos e a investigação de novas ferramentas e técnicas para aprimorar sua aplicação são temas de grande relevância para a área.

Referências

Rösch, N., Tiberius, V., & Kraus, S. 2023. Design thinking for innovation: context factors, process, and outcomes. European Journal of Innovation Management, 26(7), 160-176.

Bender-Salazar, R. 2023. Design thinking as an effective method for problem-setting and needfinding for entrepreneurial teams addressing wicked problems. Journal of Innovation and Entrepreneurship 12(1): 24.

Brown, T. 2020. Design Thinking. Uma Metodologia Poderosa Para Decretar o Fim das Velhas Ideias. Alta Books, Rio de Janeiro, RJ, Brasil.

Goel, P. 2024. Design Thinking. International Journal of Advanced Research 12(04): 290-294.

Muotka, S.; Togiani, A.; Varis, J. 2023. A design thinking approach: Applying 5S methodology effectively in an industrial work environment. Procedia CIRP 119: 363-370.

Poleac, D. 2024. Design Thinking with AI. In Proceedings of the International Conference on Business Excellence 18(1): 2891-2900.

Rösch, N.; Tiberius, V.; Kraus, S. 2023. Design thinking for innovation: context factors, process, and outcomes. European Journal of Innovation Management 26(7): 160-176.

8.2. DEVOPS

DevOps é uma abordagem que une desenvolvimento (Dev) e operações (Ops) em um fluxo de trabalho integrado, visando acelerar a entrega de software com qualidade e eficiência. Essa cultura de colaboração e automação busca eliminar os conflitos e gargalos que tradicionalmente separavam as equipes de desenvolvimento e operações, promovendo um ambiente de trabalho mais ágil e eficiente. O conceito de DevOps surgiu no início dos anos 2000, impulsionado pela necessidade de entregar software de forma mais rápida e confiável, sem comprometer a qualidade. A adoção de metodologias ágeis, como o framework Scrum, e a automação de processos de desenvolvimento e implantação foram fundamentais para o surgimento do DevOps. A cultura DevOps se baseia em princípios como colaboração, comunicação, automação, feedback contínuo e melhoria contínua. As equipes de desenvolvimento e operações trabalham em conjunto, compartilhando responsabilidades e objetivos, utilizando ferramentas de automação para agilizar o processo e buscando constantemente aprimorar suas práticas e processos.

O ciclo de vida DevOps é representado pelo símbolo do infinito, que ilustra a natureza contínua e iterativa do processo. Começa com o planejamento, seguido pela construção (build), teste, implantação (deploy), operação, monitoramento e feedback contínuo, que leva a novas descobertas e melhorias, reiniciando o ciclo. A área de DevOps oferece diversas oportunidades para desenvolvimento acadêmico. Um exemplo seria investigar a implementação de DevOps em diferentes tipos de projetos e organizações, analisando os desafios e benefícios da adoção dessa cultura. O estudo de ferramentas e práticas de DevOps, como integração contínua (CI), entrega contínua (CD) e infraestrutura como código (IaC), também é relevante para entender como a automação pode otimizar o processo de desenvolvimento e implantação de software. A pesquisa em DevOps pode abordar temas como a mensuração do impacto da adoção de DevOps na qualidade, velocidade e eficiência da entrega de software, a identificação de fatores críticos de sucesso para a implementação de DevOps em diferentes contextos e a análise de novas tendências e desafios na área, como o DevSecOps, que integra a segurança no ciclo de vida DevOps.

Referências

Almeida, F.; Simões, J.; Lopes, S. 2022. Exploring the benefits of combining DevOps and agile. Future Internet 14(2): 63.

Azad, N.; Hyrynsalmi, S. 2023. DevOps critical success factors: A systematic literature review. Information and Software Technology 157: 107150.

Faustino, J.; Adriano, D.; Amaro, R.; Pereira, R.; da Silva, M.M. 2022. DevOps benefits: A systematic literature review. Software: Practice and Experience 52(9): 1905-1926.

Maroukian, K.; Gulliver, S.R. 2020. Leading DevOps practice and principle adoption. Disponível em: <https://arxiv.org/pdf/2008.10515>. Acesso em: 12 jul. 2024.

Yarlagadda, R.T. 2021. DevOps and its practices. International Journal of Creative Research Thoughts (IJCRT) 9(3): 2320-2882.

8.3. Metodologias Ágeis, Liderança e Gestão da Mudança

As metodologias ágeis revolucionaram a gestão de projetos de software, oferecendo uma abordagem mais flexível e adaptável às mudanças, em contraste com o tradicional método cascata. Essa mudança de paradigma impactou não apenas a forma como os projetos são liderados, mas também como a mudança é gerenciada em ambientes de desenvolvimento de software. O Manifesto Ágil, lançado em 2001 por Jeff Sutherland e outros profissionais, estabeleceu os princípios fundamentais das metodologias ágeis, como colaboração, comunicação, adaptabilidade e entrega de valor contínua. O Scrum, um dos frameworks ágeis mais populares, oferece um conjunto de práticas e papéis que facilitam a implementação desses princípios, como sprints (ciclos de desenvolvimento de curta duração), product owner (responsável por definir e priorizar as funcionalidades do produto), scrum master (facilitador do processo) e equipe de desenvolvimento (responsável por construir o produto).

O campo das metodologias ágeis, liderança e gestão da mudança convida à exploração de múltiplas vertentes. O aluno pode, por exemplo, analisar a efetividade das metodologias ágeis em diferentes contextos organizacionais, investigando seus impactos na produtividade, qualidade e satisfação das equipes, além de permitir se aprofundar no papel crucial da liderança em projetos ágeis, examinando como diferentes estilos de liderança influenciam a dinâmica da equipe e os resultados do projeto. A gestão da mudança, elemento fundamental em ambientes ágeis, pode ser explorada com base em análise de estratégias e ferramentas que facilitam a adaptação a mudanças e a entrega contínua de valor. Adicionalmente, o aluno pode investigar como a identificação e o gerenciamento de riscos, a comunicação eficaz e o uso de métricas de desempenho contribuem para o sucesso de projetos ágeis.

Referências

Alotaibi, F.; Almudhi, R. 2023. Application of Agile Methodology in Managing the Healthcare Sector. iRASD Journal of Management 5(3): 147-160.

Al-Saqqa, S.; Sawalha, S.; AbdelNabi, H. 2020. Agile software development: Methodologies and trends. International Journal of Interactive Mobile Technologies 14(11).

Guerrero-Ulloa, G.; Rodríguez-Domínguez, C.; Hornos, M. J. 2023. Agile methodologies applied to the development of Internet of Things (IoT)-based systems: A review. Sensors 23(2): 790.

Hamza, M.; Basit, A. 2023. Effectiveness of Scrum Master in Agile Projects. Disponível em: https://www.preprints.org/manuscript/202312.0046/v1. Acesso em: 12 jul. 2024.

Rubin, K.S. 2017. Scrum Essencial: um Guia Prático Para o Mais Popular Processo ágil. Alta Books, Rio de Janeiro, RJ, Brasil.

9. Tecnologia e negócios

A área de tecnologia e negócios se concentra na aplicação e no impacto da tecnologia no ambiente empresarial, abrangendo desde a gestão e planejamento estratégico até a avaliação e investimento em empresas de tecnologia. Essa área multidisciplinar combina conhecimentos de tecnologia, gestão, finanças e marketing, oferecendo um amplo campo para pesquisa e desenvolvimento de soluções inovadoras. A intersecção entre tecnologia e negócios revela um campo fértil para pesquisa e desenvolvimento, com destaque para três pilares fundamentais: o mindset empreendedor e o planejamento estratégico, cruciais para o sucesso de empresas de base tecnológica; a avaliação e o financiamento de startups, que impulsionam a inovação e o crescimento do setor; e o canvas de modelo de negócios, uma ferramenta visual que auxilia na compreensão e análise de diferentes aspectos do empreendimento. O aluno interessado em aprofundar seus conhecimentos nessa área pode explorar o impacto da tecnologia em diversos setores da economia, analisar as tendências e os desafios do mercado tecnológico, desenvolver modelos de avaliação para empresas de tecnologia ou investigar o papel do mindset empreendedor e do plano de negócios no sucesso de startups.

9.1. Mindset Canvas e Plano de Negócio

No contexto da tecnologia e negócios, o planejamento estratégico é fundamental para o sucesso de qualquer empreendimento. Duas ferramentas importantes nesse processo são o Mindset Canvas e o Plano de Negócios, cada uma com suas características e aplicações específicas. O Mindset Canvas, baseado no Business Model Canvas, é uma ferramenta visual e objetiva que permite visualizar e planejar o modelo de negócios de uma empresa. Ele é dividido em nove blocos, que abordam aspectos como proposta de valor, parcerias, atividades chave, recursos chave, relacionamento com clientes, canais, segmentos de clientes, estrutura de custos e fontes de receita. Essa ferramenta é útil tanto para empresas em fase inicial quanto para aquelas que buscam se reestruturar ou expandir. O Plano de Negócios, por sua vez, é um documento mais formal e detalhado, que descreve todos os aspectos do negócio, incluindo análise de mercado, plano de marketing, plano operacional, plano financeiro e análise SWOT (Forças, Fraquezas, Oportunidades e Ameaças). Ele é útil para o planejamento inicial do negócio, para a busca de financiamento e para a gestão e expansão da empresa.

Ambas as ferramentas são importantes e complementares, e a escolha de qual utilizar depende do objetivo e do contexto do planejamento. O Mindset Canvas é mais visual e dinâmico, ideal para brainstorming e ideação, enquanto o Plano de Negócios é mais detalhado e formal, adequado para apresentar o negócio a investidores e parceiros. A área de Mindset Canvas e Plano de Negócios oferece diversas oportunidades para desenvolvimento acadêmico. Um exemplo seria investigar o impacto dessas ferramentas no sucesso ou fracasso de empresas, analisar seu uso em diferentes tipos de negócios,

como startups e pequenas e médias empresas, ou ainda, desenvolver novas ferramentas e metodologias para aprimorar o planejamento estratégico.

Referências

Abdullah, R. 2020. Importance and contents of business plan: A case-based approach. Jurnal Manajemen Indonesia 20(2): 161-173.

Jin, Y.; Ji, S.; Liu, L.; Wang, W. 2022. Business model innovation canvas: a visual business model innovation model. European Journal of Innovation Management 25(5): 1469-1493.

Mansoori, Y.; Lackéus, M. 2020. Comparing effectuation to discovery-driven planning, prescriptive entrepreneurship, business planning, lean startup, and design thinking. Small Business Economics 54: 791-818.

Nooh, M. 2024. Cooperative Business Model Canvas: A Conceptual Framework. International Journal of Academic Research in Business and Social Sciences 14(4): 837-855. 2024.

9.2. Investimento e Valuation

Dentro da grande área de tecnologia e negócios, o investimento e valuation em empresas de tecnologia são temas de grande relevância, especialmente devido à natureza dinâmica e inovadora desse setor. O valuation, que consiste em atribuir um valor a uma empresa, é essencial para determinar o preço justo de um investimento e garantir um retorno adequado. Diferentemente de empresas tradicionais, o valor das empresas de tecnologia, muitas vezes, reside em seus ativos intangíveis, como propriedade intelectual, marca e potencial de crescimento. Essa característica torna o processo de valuation mais desafiador, exigindo uma análise cuidadosa das perspectivas de mercado, das tecnologias inovadoras e das estratégias de crescimento da empresa.

O campo de investimento e avaliação de empresas de tecnologia permite, por exemplo, investigar as particularidades das metodologias de valuation aplicadas a startups, que, em sua maioria, não possuem histórico financeiro consolidado e dependem de projeções de crescimento. Analisar as premissas utilizadas nesses modelos, como taxas de crescimento, taxas de desconto e risco, permite avaliar a sensibilidade do valuation a diferentes cenários e incertezas. Além disso, o estudo das fontes de financiamento, custos e riscos envolvidos no investimento em empresas de tecnologia, bem como o impacto dessas empresas no mercado financeiro e na diversificação de carteiras, oferece um amplo leque de possibilidades de pesquisa. O papel das fintechs na inovação e transformação do setor financeiro também se apresenta como um tema relevante e atual para investigação.

Referências

AlSarah, J. 2024. Business Plan. International Journal of Entrepreneurship and Project Management 9(1): 18-31.

Assaf Neto, A. 2021. Valuation: Métricas de Valor e Avaliação de Empresas Editora Atlas, São Paulo, SP, Brasil.

Assaf Neto, A. 2022. Finanças Corporativas e Valor. Editora Atlas, São Paulo, SP, Brasil.

Damodaran, A. 2012. Valuation - Como Avaliar Empresas e Escolher as Melhores Ações. Editora LTC, Rio de Janeiro, RJ, Brasil.

Gurrib, I.; Kamalov, F.; Starkova, O.; Makki, A.; Mirchandani, A.; Gupta, N. 2023. Performance of equity investments in sustainable environmental markets. Sustainability 15(9): 7453.

Pinto, J.E.; Henry, E.; Robinson, T. R.; Stowe, J. D. 2020. Equity asset valuation. 3ed. John Wiley & Sons, New Jersey, NK, USA.

9.3. Blockchain e Criptomoedas

A tecnologia blockchain e as criptomoedas têm revolucionado o mundo das finanças e da tecnologia, oferecendo novas possibilidades e desafios para o futuro. A blockchain, um livro-razão digital descentralizado e imutável, registra transações de forma segura e transparente, enquanto as criptomoedas, como o Bitcoin, são moedas digitais que utilizam a criptografia para garantir a segurança das transações e controlar a criação de novas unidades. Embora a blockchain tenha surgido com o Bitcoin, sua aplicação vai muito além das criptomoedas. Essa tecnologia pode ser utilizada em diversas áreas, como contratos inteligentes, rastreamento de ativos, votação eletrônica e gestão de identidade, oferecendo soluções inovadoras para problemas complexos.

A área de blockchain e criptomoedas oferece diversas oportunidades para desenvolvimento acadêmico. Um exemplo seria investigar as perspectivas de uso da blockchain em diferentes setores, como saúde, educação e logística, analisar o impacto das criptomoedas no sistema financeiro e na economia global, ou ainda, estudar o desenvolvimento de novas aplicações e soluções baseadas em blockchain. Outros temas relevantes incluem a análise do mercado de criptomoedas, que envolve a avaliação de diferentes moedas, a análise de tendências e a identificação de oportunidades de investimento. A segurança em blockchain e criptomoedas também é um tema crucial, que abrange a proteção contra-ataques cibernéticos, a prevenção de fraudes e a garantia da privacidade dos usuários. A pesquisa em blockchain e criptomoedas pode contribuir para o desenvolvimento de novas tecnologias e soluções que possam transformar a forma como interagem com o mundo digital, promovendo a descentralização, a transparência e a segurança.

Referências

Bhutta, M.N.M.; Khwaja, A.A.; Nadeem, A.; Ahmad, H.F.; Khan, M.K.; Hanif, M.A.; Song, H.; Alshamari, M.; Cao, Y. 2021. A survey on blockchain technology: Evolution, architecture and security. Ieee Access 9: 61048-61073.

Gad, A. G.; Mosa, D. T.; Abualigah, L.; Abohany, A. A. 2022. Emerging trends in blockchain technology and applications: A review and outlook. Journal of King Saud University-Computer and Information Sciences 34(9): 6719-6742.

Guo, H.; Yu, X. 2022. A survey on blockchain technology and its security. Blockchain: research and applications 3(2): 100067.

Javaid, M.; Haleem, A.; Singh, R. P.; Khan, S.; Suman, R. 2021. Blockchain technology

applications for Industry 4.0: A literature-based review. Blockchain: Research and Applications 2(4): 100027.

Javaid, M.; Haleem, A.; Singh, R. P.; Suman, R.; Khan, S. 2022. A review of Blockchain Technology applications for financial services. BenchCouncil Transactions on Benchmarks, Standards and Evaluations 2(3): 100073.

Vopson, M.; Lepadatu, S.; Vopson, A.; Łukaszyk, S. 2024. Next generation blockchain technology: The Entropic Blockchain. Disponível em: <https://www.preprints.org/manuscript/202406.1301/v1>. Acesso em: 12 jul. 2024

9.4. Internet of Things

A Internet das Coisas (IoT) é uma rede de dispositivos físicos, veículos, eletrodomésticos e outros itens que possuem eletrônica, software, sensores, atuadores e conectividade, permitindo que esses objetos se conectem e troquem dados. Essa tecnologia revolucionária está presente em diversos aspectos do nosso cotidiano, desde dispositivos domésticos inteligentes, como geladeiras e termostatos, até aplicações industriais e cidades inteligentes. O IoT possibilita a coleta e análise de dados em tempo real, permitindo a automação de processos, a tomada de decisões baseadas em dados e a criação de novos serviços e produtos. Sensores desempenham um papel fundamental nesse ecossistema, coletando informações sobre o ambiente, o comportamento dos usuários e o funcionamento dos dispositivos.

O aluno interessado em explorar esse campo fértil pode investigar a evolução da internet e seu impacto na criação de novas tecnologias e aplicações de IoT. O desenvolvimento de softwares para IoT, como plataformas de gerenciamento de dispositivos e aplicativos de controle e monitoramento, oferece um promissor caminho de pesquisa. A aplicação de IoT em cidades inteligentes, na indústria, na saúde e em outros setores, com foco na otimização, eficiência e bem-estar, também se apresenta como um campo de estudo relevante. A segurança, privacidade, interoperabilidade, novos protocolos de comunicação e o impacto socioeconômico do IoT são temas adicionais que merecem atenção e aprofundamento.

Referências

Hassine, T. B. 2024. Internet of Intelligent Things (IoIT). World Journal of Advanced Research and Reviews 22(03): 1062-1066.

Laghari, A.A.; Wu, K.; Laghari, R.A.; Ali, M.; Khan, A.A. 2021. A review and state of art of Internet of Things (IoT). Archives of Computational Methods in Engineering 1395–1413.

Lombardi, M.; Pascale, F.; Santaniello, D. 2021. Internet of things: A general overview between architectures, protocols and applications. Information 12(2): 87.

Magrani, E. 2018. A internet das coisas. BOD GmbH DE. Editora FGV, Rio de Janeiro, RJ, Brasil.

MBA
USP
ESALQ