Tópicos Especiais em Sistemas de Informação

Aula 07

Da Monolítica ao Mundo dos Microserviços: Arquiteturas Modernas e Comunicação RESTful

Prof. Msc. Francisco de Assis de Souza Rodrigues

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Introdução

Você já usou um app que parecia travar tudo quando uma parte não funcionava?

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Introdução

“Pensem naquele app de delivery ou banco que vocês usam. Já aconteceu de tentar acessar uma funcionalidade, como o histórico de pedidos ou a transferência bancária, e o app inteiro travar ou nem abrir direito? Por que será que isso acontece?”

Introdução

“Alguém aqui já ficou preso esperando uma funcionalidade carregar, e acabou fechando o app de raiva?”

Introdução

“Esse tipo de comportamento geralmente está relacionado com a forma como o sistema foi arquitetado. Hoje vamos entender como arquiteturas diferentes podem afetar diretamente a experiência do usuário — e até o sucesso de uma aplicação.”

Introdução

“A arquitetura de software é como o esqueleto de um prédio. Ela define como as partes se organizam, se comunicam e como o sistema vai reagir quando algo dá errado. Com a popularização de sistemas cada vez maiores e mais distribuídos, pensar bem na arquitetura se tornou essencial.”

Introdução

“Hoje, empresas como Netflix, iFood, Amazon, e muitas outras operam com milhares de pequenos serviços independentes, chamados microserviços. Mas nem sempre foi assim — e nem sempre precisa ser assim.”

Objetivo

Compreender os fundamentos das arquiteturas modernas de software, com ênfase na transição de sistemas monolíticos para microserviços, e explorar como os serviços se comunicam utilizando padrões RESTful.

Visão Geral

- Conceitos Fundamentais (Arquitetura Monolítica e Microserviços)
- Comparativo: Monolito x Microserviços
- Comunicação entre Microserviços
- RESTful

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Conceitos Fundamentais

Arquitetura Monolítica

O que é?

Um sistema monolítico é aquele em que todas as funcionalidades estão reunidas em uma única aplicação. Tudo roda junto: autenticação, sistema de pagamentos, gerenciamento de usuários, relatórios, etc.

“É como um restaurante que faz tudo numa única cozinha — desde o sushi até a feijoada. Tudo junto, tudo misturado.”

- Interface do usuário (frontend)
- Lógica de negócio
- Acesso a dados

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Conceitos Fundamentais

Arquitetura Monolítica

Tudo dentro de um **único aplicativo** e um **único** processo de **deploy**



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Conceitos Fundamentais
Arquitetura Monolítica

Vantagens

- **Fácil de começar**: bom para projetos pequenos.
- **Desenvolvimento rápido no início**: menos complexidade.
- **Desenvolvedores têm visão do todo**: tudo está num lugar só.
- **Implantação simples**: um único artefato (ex: WAR, JAR, EXE).



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Conceitos Fundamentais
Arquitetura Monolítica

Desvantagens

- **Dificuldade para escalar partes específicas**: você escala o sistema inteiro.
- **Implantações lentas e arriscadas**: qualquer alteração exige um novo deploy completo.
- **Baixa resiliência**: se um módulo falha, o sistema inteiro pode parar.
- **Código se torna difícil de manter com o tempo**: acoplamento alto, difícil modularização.



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Conceitos Fundamentais
Arquitetura Monolítica

Exemplos típicos

- Aplicações legadas em grandes empresas.
- Sistemas feitos em frameworks como Laravel, Spring Boot, Django (usados de forma monolítica).
- Sistemas acadêmicos antigos ou ERPs.



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Conceitos Fundamentais
Arquitetura de Microserviços

Definição e principais características

- Definição simples:

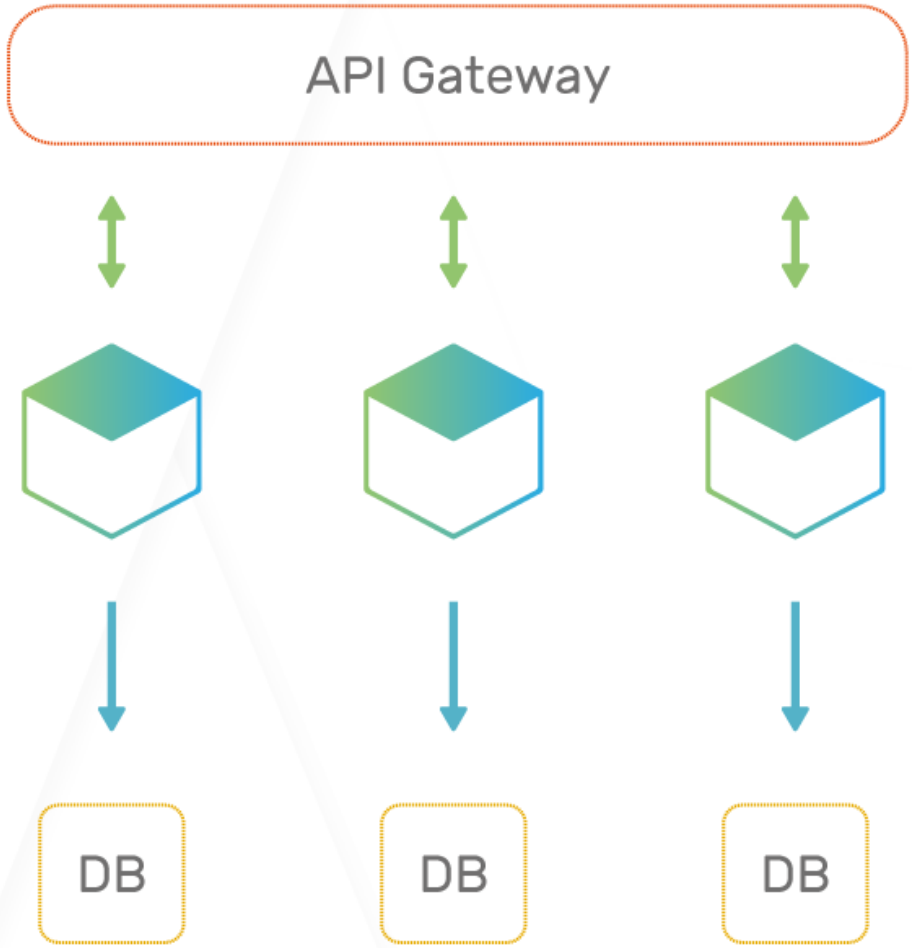
> “Microserviços são uma abordagem onde o sistema é dividido em **vários serviços pequenos, independentes e especializados**, que se comunicam entre si.”

Conceitos Fundamentais
Arquitetura de Microserviços

“Lembra do restaurante monolítico? Aqui é como um shopping com vários quiosques — um só faz pizza, outro só faz sobremesa. Cada um pode abrir e fechar independente dos outros.”

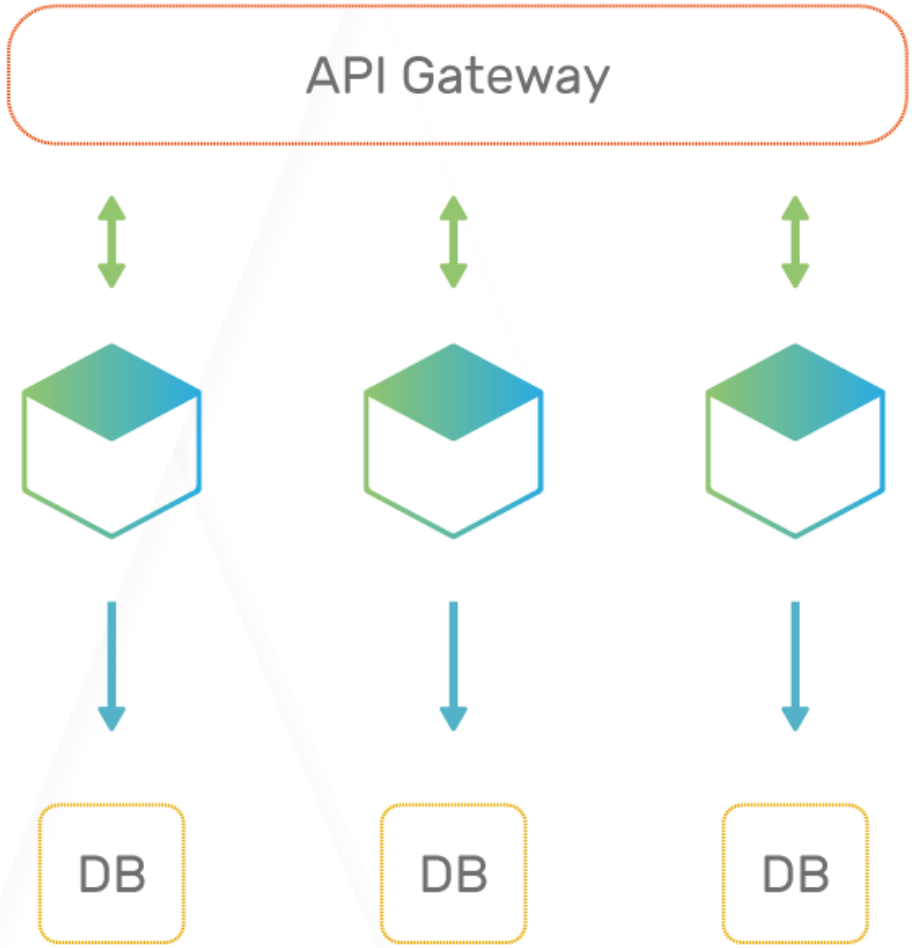
Conceitos Fundamentais
Arquitetura de Microserviç

- **Características:**
  - Cada microserviço tem **seu próprio código, banco de dados e deploy**.
  - São **autônomos**: desenvolvidos, testados e implantados separadamente.



Conceitos Fundamentais
Arquitetura de Microserviç

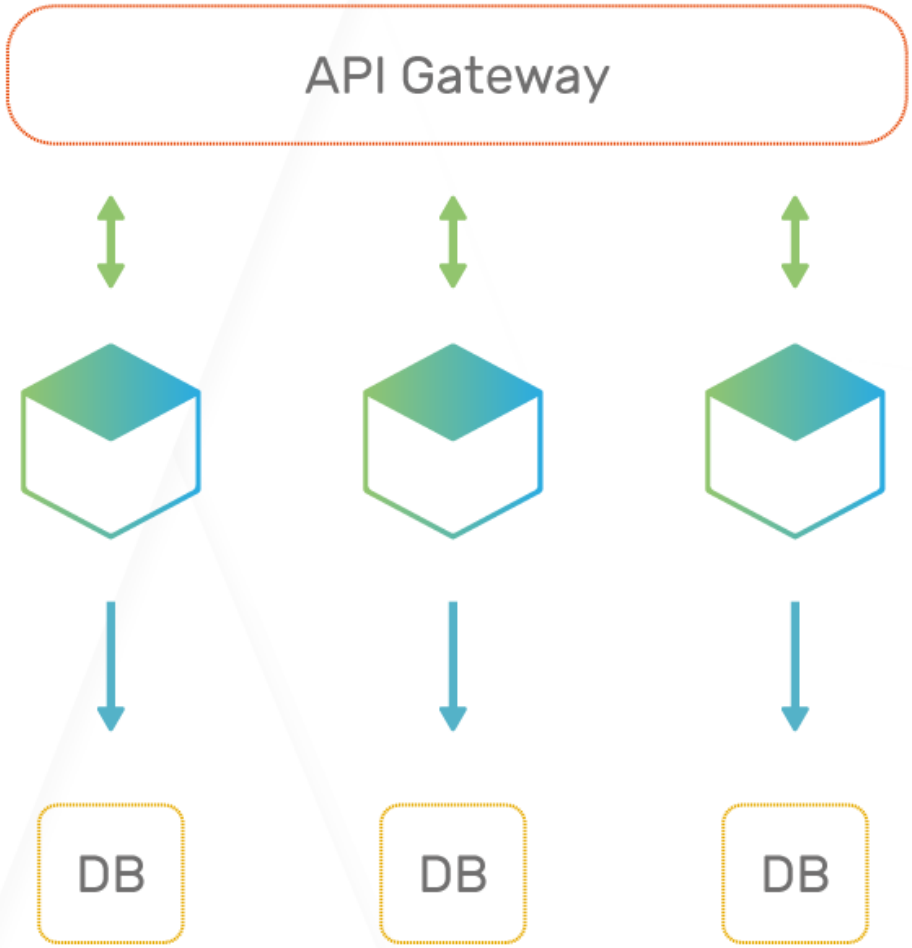
- **Características:**
  - Se comunicam via **protocolos leves**, como HTTP (REST), gRPC ou mensagens.
  - Facilitam o uso de **tecnologias diferentes** para problemas diferentes.



Conceitos Fundamentais
Arquitetura de Microserviç

Vantagens

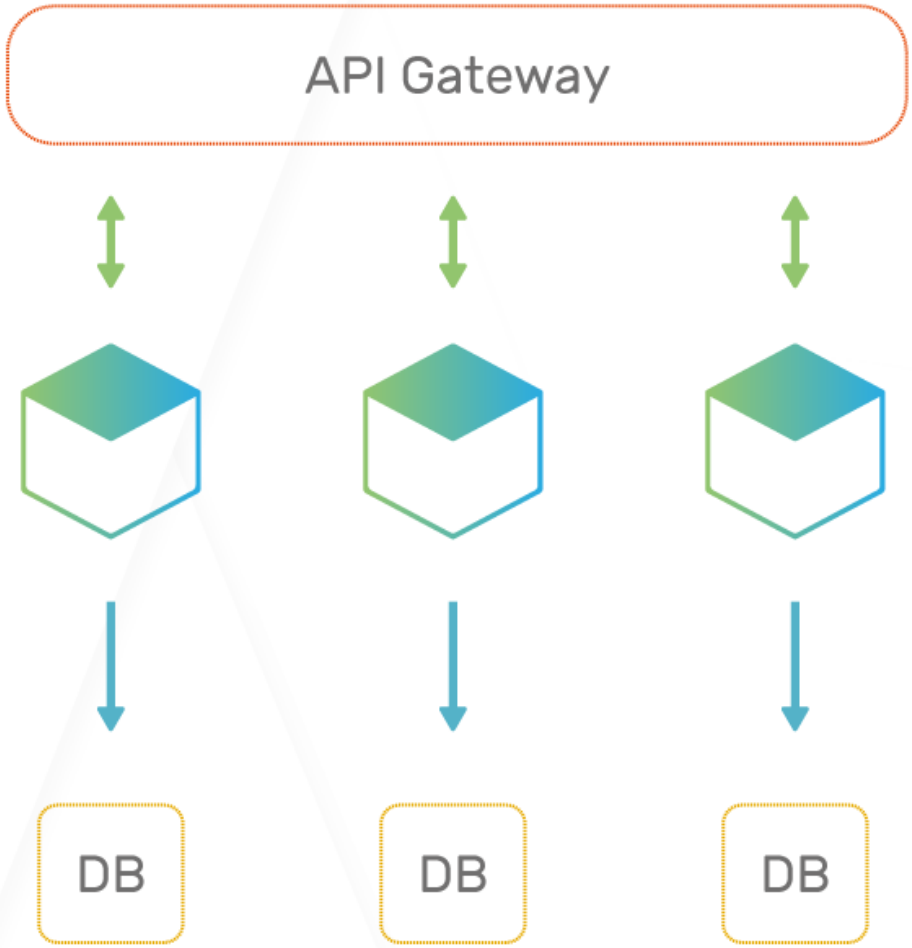
- **Escalabilidade independente**: só escala o que precisa.
- **Maior resiliência**: falha de um serviço não derruba tudo.
- **Deploys mais frequentes e seguros**: mudanças localizadas.
- **Facilidade de manutenção por times pequenos e focados**.



Conceitos Fundamentais
Arquitetura de Microserviç

Desafios

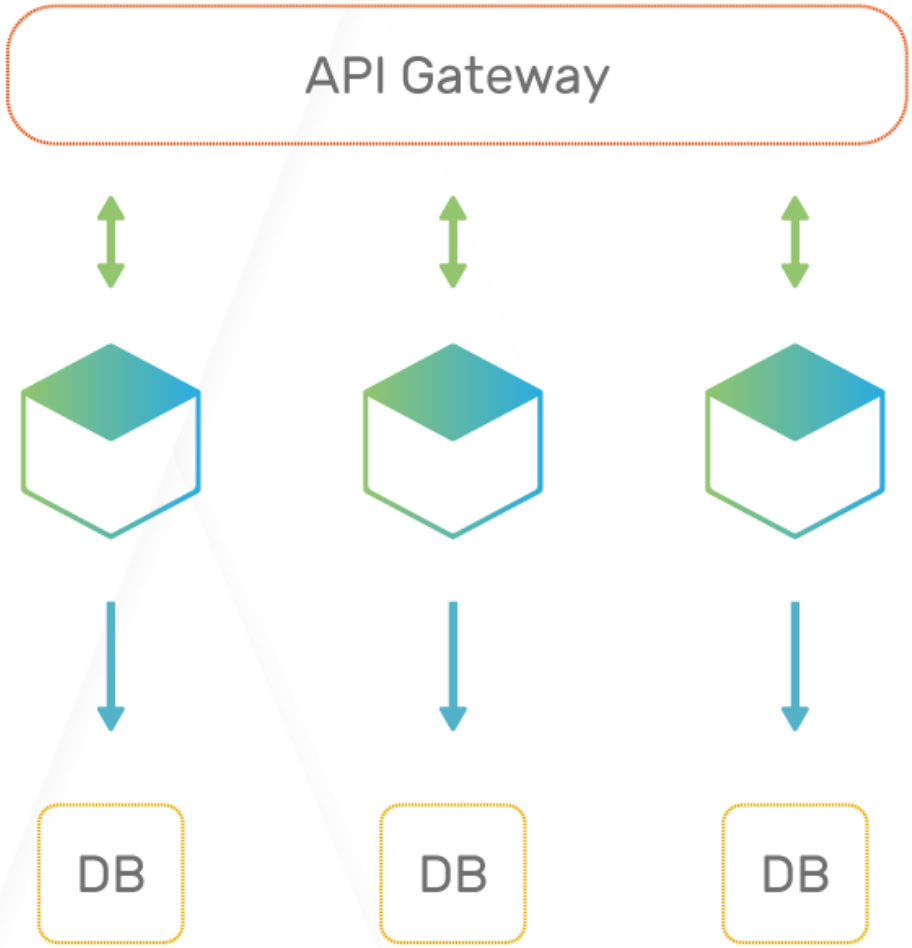
- **Complexidade de infraestrutura**: precisa de orquestração (ex: Docker, Kubernetes).
- **Monitoramento e logs mais complexos**.
- **Gerenciamento de comunicação e consistência** entre serviços.
- **Curva de aprendizado mais alta**.



Conceitos Fundamentais
Arquitetura de Microserviç

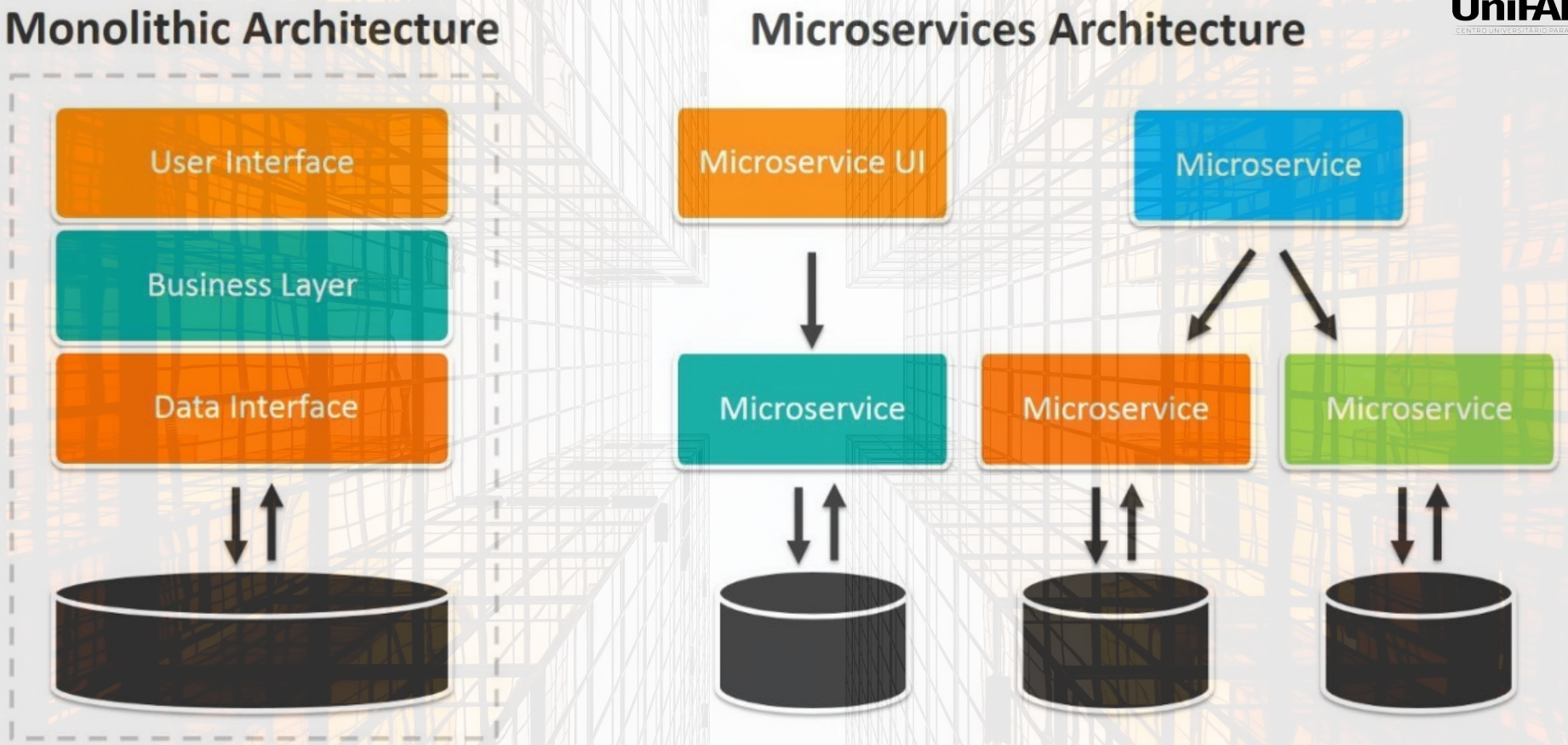
Exemplos de empresas que usam

- **Netflix**: milhares de microserviços para streaming, recomendações, pagamentos, etc.
- **Amazon**: cada parte do marketplace e serviços (como AWS) funciona de forma independente.
- **Spotify**: cada seção do app (player, playlists, recomendação) é gerenciada por um time com seus próprios microserviços.



Comparativo: Monolito x Microserviços

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO


Monolithic Architecture
User Interface
Business Layer
Data Interface
Microservices Architecture
Microservice UI
Microservice
Microservice
Microservice
Microservice

Aspecto	Arquitetura Monolítica	Arquitetura de Microserviços
Estrutura	Tudo em um único sistema	Vários serviços pequenos e independentes
Deploy	Um único artefato	Cada serviço pode ser implantado separadamente
Escalabilidade	Escala o sistema todo	Escala serviços individualmente
Resiliência	Uma falha pode derrubar tudo	Uma falha afeta apenas o serviço relacionado
Tempo de desenvolvimento	Mais rápido no início	Exige mais planejamento
Facilidade de manutenção	Dificulta à medida que cresce	Mais fácil se bem organizado
Comunicação interna	Local, direta (métodos internos)	Via rede (REST, gRPC, mensageria)
Tecnologias	Geralmente uma stack única	Pode variar entre os serviços
Testes e Debug	Simples de rodar localmente	Pode ser mais complexo

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Comparativo
Arquitetura de Microserviços

Quando usar cada abordagem?

Não existe uma arquitetura “melhor” em todos os casos. O ideal depende do contexto.

Use Monolito quando:

- O sistema é **pequeno ou em estágio inicial**.
- O time de desenvolvimento é **pequeno**.
- O foco está em **entregar rápido** com menos complexidade.
- Não há infraestrutura para orquestração (como Docker, CI/CD etc.).

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Comparativo
Arquitetura de Microserviços

Quando usar cada abordagem?

Não existe uma arquitetura “melhor” em todos os casos. O ideal depende do contexto.

Use Microserviços quando:

- O sistema é **grande, cresce rápido e tem muitas funcionalidades**.
- É necessário **escalar partes específicas** do sistema.
- O sistema precisa de **alta disponibilidade e resiliência**.
- A equipe é grande e pode ser dividida em **times especializados**.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Comparativo
Arquitetura de Microserviços

“Uma startup começando seu MVP talvez vá de monolito. Mas se o sistema crescer e ganhar milhares de usuários, migrar para microserviços pode ser o próximo passo.”

Comparativo
Transição e Trade-offs

Existem **complexidade de migrar** de monolito para microserviços.

- **Não se faz tudo de uma vez** — a transição costuma ser gradual.
- Prática comum: **extrair funcionalidades específicas** (ex: autenticação, pagamentos) e transformá-las em serviços independentes.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Comparativo
Transição e Trade-offs

- **Trade-offs (trocas):**
  - Ganho em escalabilidade **vs.** aumento na complexidade de monitoramento, comunicação e deploy.
  - Mais flexibilidade de tecnologia **vs.** necessidade de integração constante entre serviços.

“É como trocar as peças de um carro enquanto ele ainda está andando. Por isso, toda transição precisa ser muito bem planejada.”

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Comunicação entre Microserviços

Microserviços precisam conversar entre si, já que cada um cuida de uma parte do sistema. Essa comunicação pode acontecer de duas formas principais: síncrona e assíncrona.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Comunicação entre Microserviços

Comunicação Síncrona vs. Assíncrona

Síncrona

- O serviço A **faz uma requisição e espera a resposta** do serviço B.
- Exemplo mais comum: chamadas HTTP (REST).

> “É como ligar para um amigo e esperar ele atender para te dar uma resposta. Enquanto isso, você não pode fazer outra coisa.”

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Comunicação entre Microserviços

Comunicação Síncrona vs. Assíncrona

Síncrona

- **Prós:**
  - Fácil de implementar.
  - Ideal para dados que precisam de resposta imediata.
- **Contras:**
  - Um serviço depende da **disponibilidade imediata do outro**.
  - Pode gerar lentidão ou falhas em cascata.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Comunicação entre Microserviços

Comunicação Síncrona vs. Assíncrona

Assíncrona

- O serviço A **envia uma mensagem** e segue sua vida. O serviço B **processa quando puder**.

- Pode usar filas (ex: RabbitMQ, Kafka) ou eventos.

> “É como mandar uma mensagem de voz no WhatsApp. Você envia, mas não precisa que a pessoa ouça na hora.”

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Comunicação entre Microserviços

Comunicação Síncrona vs. Assíncrona

Assíncrona

- **Prós:**
  - Reduz acoplamento entre serviços.
  - Maior resiliência.
  - Permite escalabilidade mais eficiente.
- **Contras:**
  - Lógica mais complexa.
  - Difícil rastrear o fluxo completo.

Comunicação entre Microserviços

Cite exemplos do cotidiano que lembram comunicação síncrona e assíncrona

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

APIs RESTful como padrão de comunicação

REST (Representational State Transfer) é um padrão muito usado para comunicação síncrona entre microserviços, especialmente via HTTP.

APIs RESTful como padrão de comunicação

Conceitos-chave:

- Baseado em **recursos** (ex: /users, /orders).
- Usa **verbo HTTP** para representar ações:
  - GET /users → buscar usuários
  - POST /users → criar usuário
  - PUT /users/1 → atualizar usuário
  - DELETE /users/1 → remover usuário

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

APIs RESTful como padrão de comunicação

Conceitos-chave:

- Usa **formatos leves**, como JSON.
- Cada recurso deve ser **acessado de forma clara e previsível**.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

APIs RESTful como padrão de comunicação

Boas práticas:

- URLs claras e descritivas.

- Evitar “verbos” na URL (usar verbos no HTTP!).

- Usar status HTTP adequadamente (
  a. 200 OK (requisição bem-sucedida),
  b. 201 Created (novo recurso criado),
  c. 400 Bad Request (erro do cliente),
  d. 404 Not Found (não encontrado),
  e. 500 Internal Server Error (erro no servidor)

).

Introdução ao conceito de mensagens/eventos

Mensagens

- Serviço A envia **uma mensagem com dados** para uma fila (como o RabbitMQ).

- Serviço B consome essa mensagem e executa alguma ação.

Introdução ao conceito de mensagens/eventos

Eventos

- Serviço A emite um **evento** como: “PedidoCriado”.
- Outros serviços interessados nesse evento **reagem** (ex: envia e-mail, atualiza estoque).

Introdução ao conceito de mensagens/eventos

"Quando você faz um pedido no iFood, o sistema pode emitir um evento PedidoRealizado. Vários serviços ouvem isso: o de notificações, o do restaurante, o do entregador."

Vantagens dos eventos:

- Estimula **baixo acoplamento**.
- Permite que novos serviços se conectem ao sistema sem mudar os antigos.
- Torna o sistema mais escalável e reativo.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Apps - APIs RESTful



POSTMAN



Insomnia

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Documentação - APIs RESTful

Swagger é uma especificação e um conjunto de ferramentas que permite definir, construir, documentar e consumir APIs REST. Ele é independente de linguagem e permite que humanos e computadores entendam os recursos de uma API sem acesso ao código-fonte.

https://swagger.io/

Swagger™
Supported by SMARTBEAR



Experimento - API RESTful

API:

https://colab.research.google.com/drive/1CbreOkqUyOwcmduJYJXLRGRNYTYYS3oD?usp=sharing



Client:

https://colab.research.google.com/drive/1GYEMQ0qukn_SY6d9f6LhsyyUmXucsr4U?usp=sharing

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Sugestões de Leitura

Qual é a diferença entre arquitetura monolítica e de microsserviços? < https://aws.amazon.com/pt/compare/the-difference-between-monolithic-and-microservices-architecture/ >

Monolito vs Microsserviços: Entendendo os prós e contras de cada abordagem < https://aquare.la/monolito-vs-microsservicos-entendendo-os-pros-e-contras-de-cada-abordagem/ >

Bibliografia

SAUDATE, Alexandre. APIs REST: seus serviços prontos para o mundo real. São Paulo, SP: Casa do Código, 2021. E-book. Disponível em: https://plataforma.bvirtual.com.br/Acervo/Publicacao/212784 . Acesso em: 11 abr. 2025.



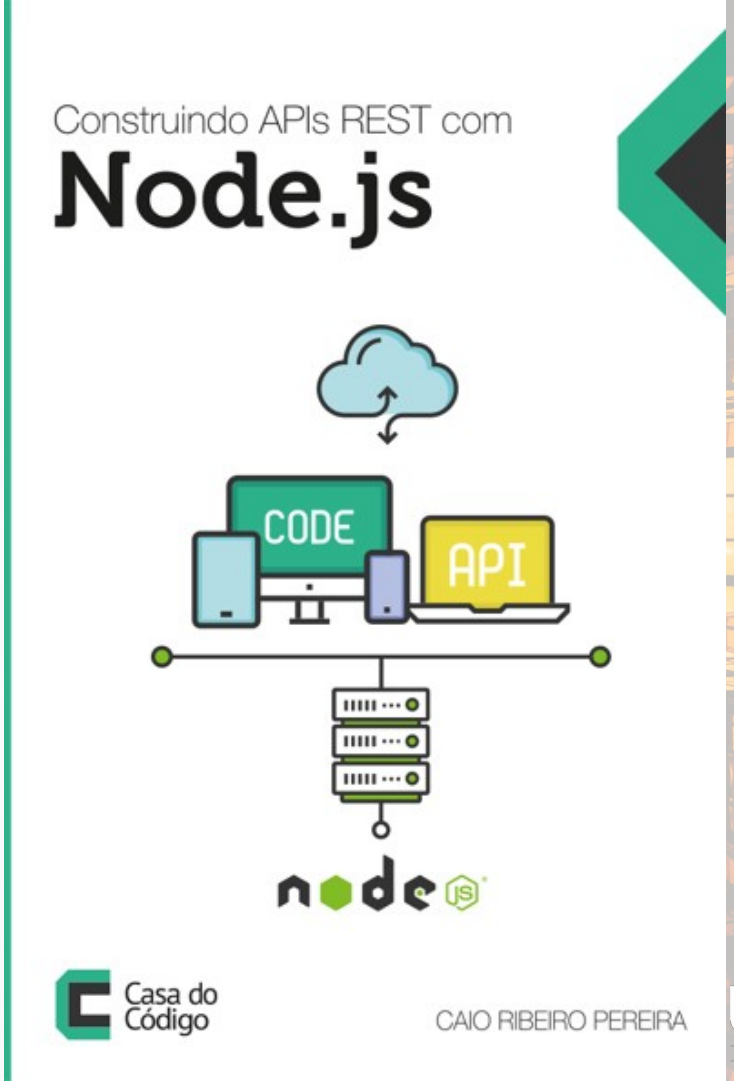
Bibliografia

SAUDATE, Alexandre. APIs REST em Kotlin: seus serviços prontos para o mundo real. São Paulo, SP: Casa do Código, 2021. E-book. Disponível em: https://plataforma.bvirtual.com.br/Acervo/Publicacao/212614 . Acesso em: 11 abr. 2025.



Bibliografia

PEREIRA, Caio Ribeiro. Construindo APIs REST com Node.js. São Paulo, SP: Casa do Código, 2016. E-book. Disponível em: https://plataforma.bvirtual.com.br/Acervo/Publicacao/212831. Acesso em: 11 abr. 2025.



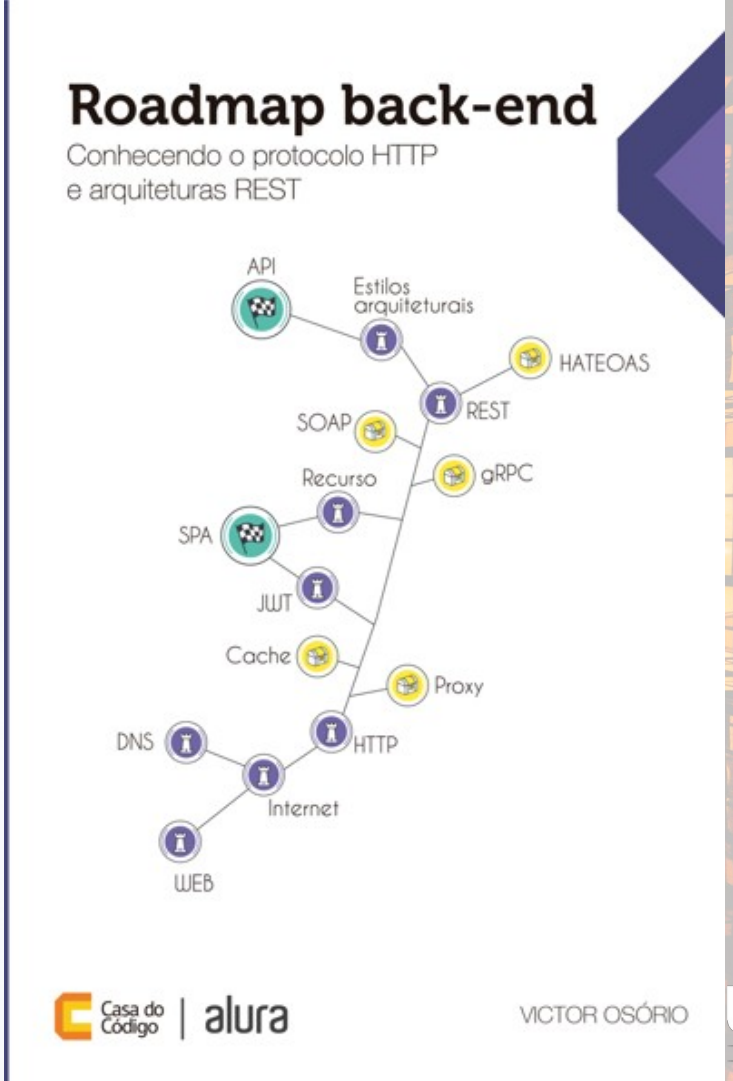
Bibliografia

SIÉCOLA, Paulo. Web services REST: com ASP .NET Web API e Windows Azure. São Paulo, SP: Casa do Código, 2016. E-book. Disponível em: https://plataforma.bvirtual.com.br/Acervo/Publicacao/212665 . Acesso em: 11 abr. 2025.



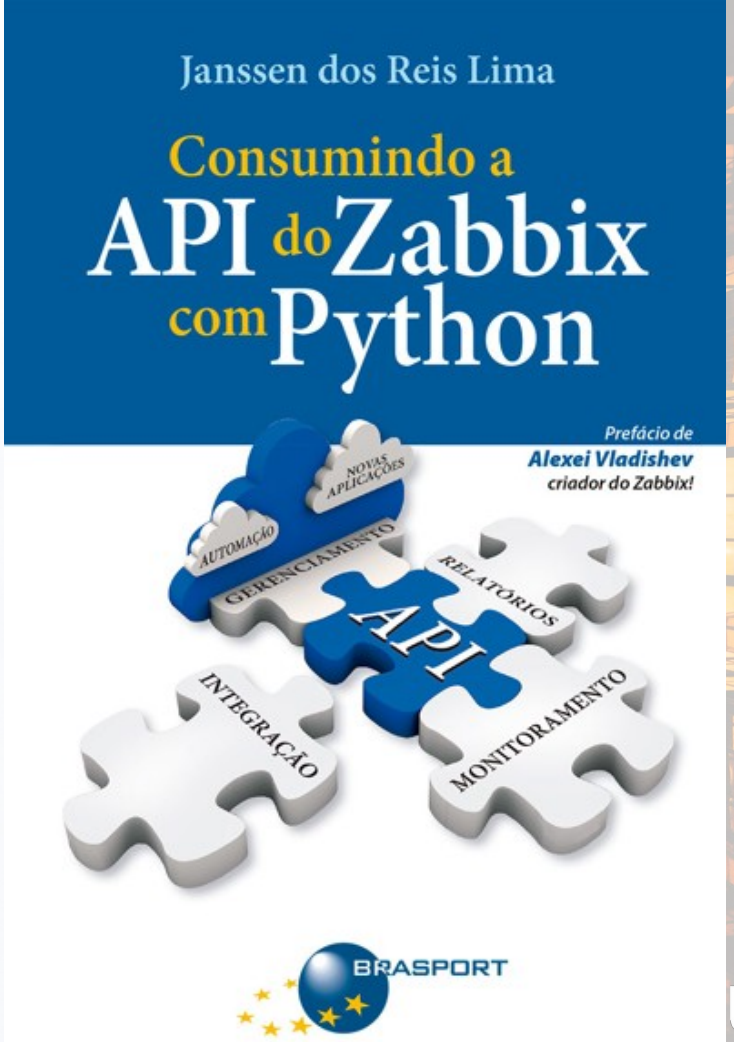
Bibliografia

OSÓRIO, Victor. Roadmap back-end: conhecendo o protocolo HTTP e arquiteturas REST. São Paulo, SP: Casa do Código, 2022. E-book. Disponível em: https://plataforma.bvirtual.com.br/Acervo/Publicacao/212863. Acesso em: 11 abr. 2025.



Bibliografia

LIMA, Janssen dos Reis. Consumindo a API do Zabbix com Python. 1. ed. Rio de Janeiro: Brasport, 2016. E-book. Disponível em: https://plataforma.bvirtual.com.br/Acervo/Publicacao/160738. Acesso em: 11 abr. 2025.



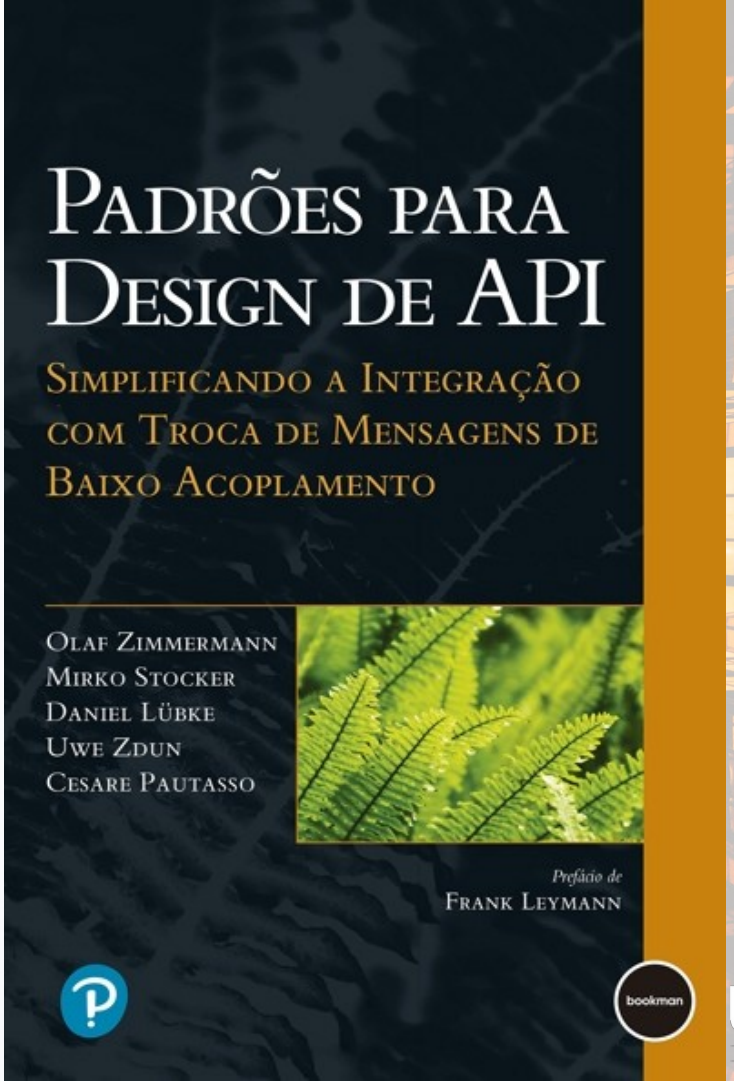
Bibliografia

BARROS, Ana Clara Gonzaga; MUNIZ, Antonio; CASTRO, Clara Érica Takayama de. Jornada API na prática: unindo conceitos e experiências do Brasil para acelerar negócios com a tecnologia. [S.l.]: Brasport, 2023. E-book. Disponível em: https://plataforma.bvirtual.com.br/Acervo/Publicacao/211453 . Acesso em: 11 abr. 2025.



Bibliografia

ZIMMERMANN, Olaf et al. Padrões para design de API: simplificando a integração com troca de mensagens de baixo acoplamento. 1. ed. Porto Alegre: Bookman, 2025. E-book. Disponível em: https://plataforma.bvirtual.com.br/Acervo/Publicacao/220897. Acesso em: 11 abr. 2025.



Referências

CHATGPT. **Fundamentos de Inteligência Artificial**. OpenAI, 2025. Disponível em: https://chat.openai.com. Acesso em: *20/03/2025*.


Perguntas?!

Eclesiastes 5:3 – "Da muita ocupação vêm os sonhos."

Sonhe alto, mas lembre-se de que o esforço constante é o que os realiza.