

# Criando Microserviços – 2ª Edição PDF

Sam Newman



Mais livros gratuitos no Bookey



Escanear para baixar

# Criando Microserviços – 2ª Edição

Domine as complexidades da arquitetura moderna de microserviços e seu provisionamento.

Escrito por Bookey

[Saiba mais sobre o resumo de Criando Microserviços – 2ª Edição](#)

[Ouvir Criando Microserviços – 2ª Edição Audiobook](#)

Mais livros gratuitos no Bookey



Escanear para baixar

## Sobre o livro

No cenário em rápida evolução dos sistemas distribuídos, a transição de aplicações monolíticas para microsserviços ágeis apresenta desafios únicos para desenvolvedores e arquitetos. Este guia abrangente de Sam Newman fornece os conceitos essenciais e insights práticos necessários para criar, gerenciar e evoluir arquiteturas de microsserviços. Através de exemplos ilustrativos e uma narrativa que acompanha uma empresa fictícia, você aprenderá a alinhar o design do sistema com os objetivos organizacionais, integrar serviços de forma eficaz, adotar estratégias incrementais para desconstruir monólitos e implantar microsserviços de maneira fluida por meio da integração contínua. Além disso, o livro aborda tópicos críticos como testes, monitoramento, modelos de segurança e complexidades de escalabilidade, disponibilizando as ferramentas necessárias para navegar pelas complexidades da arquitetura de sistemas modernos.

**Mais livros gratuitos no Bookey**



Escanear para baixar

## Sobre o autor

Sam Newman é um experiente especialista em tecnologia e autor renomado por suas contribuições nas áreas de desenvolvimento de software, construção de comunidades e gerenciamento de fóruns online. Com formação em ciência da computação e ampla experiência na indústria de tecnologia, Newman uniu sua paixão pelo engajamento comunitário com sua habilidade técnica para ajudar organizações e indivíduos a criar espaços online vibrantes. Seu trabalho, incluindo o livro "Criando Comunidades Online com Phpbb 2," reflete sua profunda compreensão das tecnologias colaborativas e das dinâmicas das interações online. Através de seus escritos e insights, Newman continua a inspirar e orientar aqueles que buscam aproveitar o poder da comunidade no reino digital.

Mais livros gratuitos no Bookey



Escanear para baixar



Ad



Escanear para baixar



# Experimente o aplicativo Bookey para ler mais de 1000 resumos dos melhores livros do mundo

Desbloqueie **1000+** títulos, **80+** tópicos

Novos títulos adicionados toda semana

Product & Brand

 Liderança & Colaboração

 Gerenciamento de Tempo

 Relacionamento & Comunicação

 Estratégia de Negócios

 Criatividade

 Memórias

 Conheça a Si Mesmo

 Psicologia

Empreendedorismo

 História Mundial

 Comunicação entre Pais e Filhos

 Autocuidado

 Mente

## Visões dos melhores livros do mundo

Desenvolvimento

Os 7 Hábitos das Pessoas Altamente Eficazes



Mini Hábitos



Hábitos Atômicos



O Clube das 5 da Manhã



Como Fazer Amigos e Influenciar Pessoas



Como Não



Teste gratuito com Bookey



# Lista de conteúdo do resumo

Capítulo 1 : 1. Microserviços

Capítulo 2 : 2. O Arquiteto Evolutivo

Capítulo 3 : 3. Como Modelar Serviços

Capítulo 4 : 4. Integração

Capítulo 5 : 5. Dividindo o Monólito

Capítulo 6 : 6. Implantação

Capítulo 7 : 7. Testes

Capítulo 8 : 8. Monitoramento

Capítulo 9 : 9. Segurança

Capítulo 10 : 10. A Lei de Conway e o Design de Sistema

Capítulo 11 : 11. Microserviços em Larga Escala

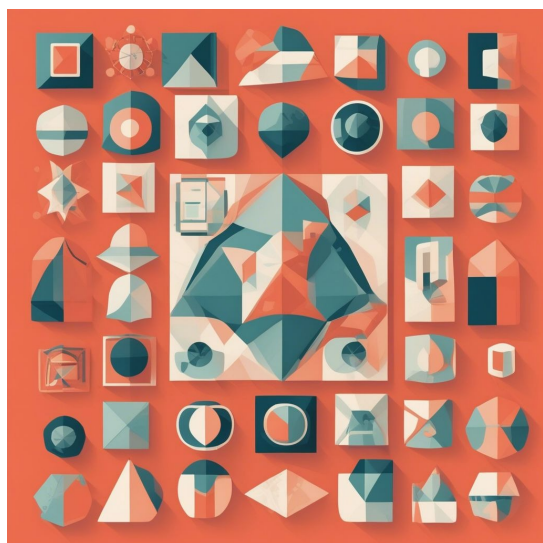
Capítulo 12 : 12. Unindo Tudo

Mais livros gratuitos no Bookey



Escanear para baixar

# Capítulo 1 Resumo : 1. Microserviços



## Capítulo 1: Microserviços

### Introdução aos Microserviços

Ao longo dos anos, os avanços tecnológicos levaram a melhores métodos para construir sistemas de TI, inspirados por empresas como Amazon, Google e Netflix. Conceitos como Design Orientado a Domínio, entrega contínua e arquitetura de microserviços surgiram, permitindo que as organizações entreguem software mais rapidamente e se adaptem rapidamente às mudanças.

### O Que São Microserviços?

Mais livros gratuitos no Bookey



Escanear para baixar

Microserviços são definidos como pequenos serviços autônomos que trabalham juntos, com foco em uma única responsabilidade, enquanto mantêm limites claros. Isso permite uma manutenção de código mais fácil, escalabilidade e implantação independente sem afetar outros serviços.

## **Principais Características dos Microserviços**

-

### **Pequenos e Focados**

: Os microserviços têm como objetivo atender a funcionalidades específicas, facilitando correções de bugs e implementações de recursos.

-

### **Autônomos**

: Eles operam de forma independente, permitindo que as equipes gerenciem e implantem serviços sem interdependências que possam complicar os processos.

## **Principais Benefícios dos Microserviços**

-

### **Heterogeneidade Tecnológica**

Mais livros gratuitos no Bookey



Escanear para baixar



: Diferentes tecnologias podem ser utilizadas para diferentes serviços, melhorando o desempenho e a adaptabilidade.

-

## **Resiliência**

: Os serviços, como componentes independentes, isolam falhas, evitando efeitos cascata no sistema como um todo.

-

## **Escalabilidade**

: Serviços individuais podem ser escalados de forma independente, otimizando o uso de recursos e gerenciando custos de forma eficaz.

-

## **Facilidade de Implantação**

: Alterações em um microsserviço podem ser implantadas de forma independente, reduzindo os riscos associados a atualizações em larga escala.

-

## **Alinhamento Organizacional**

: Equipes menores podem se concentrar em bases de código específicas, melhorando a produtividade e a propriedade.

-

## **Composabilidade**

: Os microsserviços permitem a reutilização de funcionalidades em vários contextos, adaptando-se às



necessidades dos usuários em diferentes plataformas.

-

### **Otimização para Substituição**

: Bases de código menores facilitam substituições ou exclusões mais simples, reduzindo os fardos dos sistemas legados.

### **Microserviços vs. Arquitetura Orientada a Serviços (SOA)**

Os microserviços evoluíram a partir dos princípios de SOA, focando nos desafios de aplicação do mundo real. Ao contrário da SOA convencional, os microserviços promovem melhores práticas arquitetônicas, evitando armadilhas, como acoplamento excessivamente rígido e falta de limites de serviços claros.

### **Outras Técnicas Decompositivas**

Embora bibliotecas compartilhadas e modularização possam fornecer alguns dos benefícios dos microserviços, muitas vezes elas não oferecem a mesma autonomia e resiliência devido ao acoplamento mais próximo dentro de processos compartilhados.



## Desafios dos Microserviços

A adoção de uma arquitetura de microserviços introduz complexidades associadas a sistemas distribuídos, incluindo implantação, monitoramento e manutenção da resiliência do sistema. As organizações devem estar preparadas para esses desafios.

## Conclusão

Os microserviços apresentam vantagens significativas, representando uma mudança de paradigma na arquitetura de software. Os capítulos seguintes explorarão como aproveitar esses benefícios enquanto evitam armadilhas comuns. O papel dos arquitetos evoluirá em resposta a este novo cenário arquitetônico, exigindo estratégias inovadoras para garantir o sucesso.



## Exemplo

**Ponto chave:** Serviços Pequenos e Autônomos Aumentam a Flexibilidade

**Exemplo:** Imagine que você é um desenvolvedor encarregado de aprimorar uma plataforma de e-commerce. Com uma arquitetura de microsserviços, você pode atualizar de forma independente o serviço de processamento de pagamentos sem interromper os serviços de catálogo ou de contas de usuário. Essa independência não só acelera a entrega, mas também minimiza riscos, permitindo que você experimente novos métodos de pagamento de forma tranquila. Em contraste, uma abordagem monolítica exigiria que você coordenasse mudanças em toda a aplicação, o que poderia levar a períodos de inatividade. Os microsserviços permitem que você inove e itere rapidamente, alinhando-se de perto com as necessidades dos usuários e as dinâmicas do mercado.



# Capítulo 2 Resumo : 2. O Arquiteto Evolutivo



## Capítulo 2: O Arquiteto Evolutivo

### Responsabilidades do Arquiteto

Os arquitetos desempenham um papel fundamental em garantir uma visão técnica unificada para atender às necessidades dos clientes. Suas funções podem variar desde trabalhar com uma única equipe até coordenar esforços em várias equipes ou organizações inteiras. Apesar de serem essenciais, o papel do arquiteto frequentemente enfrenta críticas, principalmente devido à jovem história da indústria e





à natureza em evolução da tecnologia.

## **Comparação com Outras Profissões**

A arquitetura em software carece da rigorosidade e responsabilidade encontradas em campos de engenharia tradicionais. Emprestar terminologia de outras profissões pode representar de forma errônea o processo de desenvolvimento de software e levar a expectativas irrealistas. Uma analogia mais precisa é a dos planejadores urbanos, que facilitam a adaptabilidade em vez de ditar especificidades.

## **O Arquiteto como Planejador Urbano**

A metáfora do arquiteto como planejador urbano enfatiza a criação de estruturas que permitam que os sistemas evoluam conforme as necessidades dos usuários mudam. Os arquitetos devem se concentrar nas fronteiras de serviço e na comunicação entre serviços, promovendo um ambiente saudável para os desenvolvedores.

## **Zoneamento de Serviços**

**Mais livros gratuitos no Bookey**



Escanear para baixar

Os arquitetos devem definir as fronteiras de serviço (zonas) e gerenciar cuidadosamente as interações entre os serviços. Cada equipe pode escolher tecnologias dentro de sua zona, mas deve se conformar com os padrões de comunicação entre zonas.

## **Colaboração com Desenvolvedores**

Arquitetos eficazes se envolvem com as equipes de desenvolvimento, dedicando tempo à codificação e compreendendo o impacto de suas decisões. A interação regular promove uma comunicação e colaboração melhores.

## **Tomada de Decisão Baseada em Princípios**

Os arquitetos devem estabelecer metas estratégicas que alinhem a tecnologia com os objetivos de negócios e criar princípios orientadores que reflitam essas metas. As práticas são os métodos táticos que garantem a implementação dos princípios, variando de acordo com o contexto organizacional.

## **Governança Técnica**



A governança envolve garantir a adesão à visão técnica e se adaptar conforme necessário. O arquiteto lidera um grupo que molda e governa coletivamente os padrões técnicos para facilitar a tomada de decisões eficazes e minimizar a fricção entre as equipes.

## **Construindo a Equipe**

Promover a propriedade da equipe na arquitetura de microsserviços fomenta o crescimento profissional individual, ao mesmo tempo em que aprimora as capacidades do sistema. Capacitar os desenvolvedores a assumir o controle dos serviços pode aliviar a carga do arquiteto.

## **Resumo das Principais Responsabilidades do Arquiteto**

O arquiteto evolutivo deve manter uma visão técnica clara, compreender as implicações das decisões, colaborar com os colegas, adaptar-se às mudanças, permitir autonomia e garantir a adesão aos padrões de governança. Este ato de equilibrar é crucial na gestão das complexidades das arquiteturas de microsserviços, preparando o terreno para o próximo capítulo sobre a definição de fronteiras apropriadas de microsserviços.



## Pensamento crítico

**Ponto chave:** O papel do arquiteto deve evoluir para promover a colaboração e a adaptabilidade entre as equipes.

**Interpretação crítica:** Enquanto Newman enfatiza o papel de liderança do arquiteto na orientação da tecnologia e dos padrões dentro das organizações, essa perspectiva pode simplificar demais a natureza dinâmica e contextual do desenvolvimento de software. O ideal do arquiteto como um planejador urbano sugere uma mudança em direção a sistemas mais flexíveis; no entanto, pode desconsiderar a realidade de que a colaboração eficaz muitas vezes surge de forma orgânica, em vez de ser estritamente imposta pela visão arquitetônica. Críticos argumentam que a natureza prescritiva da governança centralizada pode sufocar a inovação e a criatividade entre os desenvolvedores, sugerindo que uma abordagem mais descentralizada pode trazer melhores resultados na promoção de arquiteturas responsivas (Larman & Vodde, 2016). Compreender essa crítica pode incentivar os leitores a questionar o modelo de Newman e considerar abordagens variadas que priorizem a autonomia da



equipe ao lado da supervisão técnica.



# Capítulo 3 Resumo : 3. Como Modelar Serviços

## Resumo do Capítulo 3: Como Modelar Serviços

### Introdução aos Microserviços

Este capítulo explora os fundamentos da modelagem de microserviços, enfatizando a importância de definir os limites dos serviços para otimizar benefícios enquanto minimiza desvantagens. O capítulo apresenta uma empresa fictícia, MusicCorp, para ilustrar conceitos-chave.

### Definindo um Bom Serviço

-

#### Desacoplamento:

Os serviços devem ser projetados de forma que mudanças em um não necessitem de mudanças em outros. Isso requer a minimização das dependências de integração entre os serviços.



-

## **Alta Coesão:**

Funcionalidades relacionadas devem ser agrupadas dentro de um serviço para simplificar modificações. Isso reduz a complexidade e o risco associados ao lançamento de múltiplos serviços simultaneamente.

## **Entendendo os Contextos Delimitados**

- Introduzido por Eric Evans, o conceito de contextos delimitados identifica áreas distintas de um domínio onde os elementos interagem com seu próprio conjunto de regras e interfaces. Cada contexto delimitado tem um limite explícito, separando modelos internos de interações externas.
- No exemplo da MusicCorp, os departamentos de armazém e finanças ilustram dois contextos delimitados separados, cada um com responsabilidades específicas e modelos compartilhados.

## **Instalar o aplicativo Bookey para desbloquear texto completo e áudio**

Mais livros gratuitos no Bookey



Escanear para baixar



Escanear para baixar



# Por que o Bookey é um aplicativo indispensável para amantes de livros



## Conteúdo de 30min

Quanto mais profunda e clara for a interpretação que fornecemos, melhor será sua compreensão de cada título.



## Clipes de Ideias de 3min

Impulsione seu progresso.



## Questionário

Verifique se você dominou o que acabou de aprender.



## E mais

Várias fontes, Caminhos em andamento, Coleções...

Teste gratuito com Bookey



# Capítulo 4 Resumo : 4. Integração

Seção	Resumo
Importância da Integração	A integração é essencial para microsserviços para garantir autonomia e permitir alterações e lançamentos independentes, enquanto explora opções tecnológicas para evitar armadilhas de SOA.
Escolhendo a Tecnologia de Integração Certa	Considerações principais incluem evitar mudanças disruptivas, manter agnosticismo tecnológico, simplificar experiências dos consumidores e ocultar detalhes internos.
Estilos de Integração	1. Integração de Banco de Dados: aumenta o acoplamento e os riscos de mudanças disruptivas. 2. Comunicação Síncrona vs. Assíncrona: impacta os estilos de colaboração.
Orquestração vs. Coreografia	A orquestração oferece controle centralizado, mas pode criar fragilidade; a coreografia promove um acoplamento solto com respostas a eventos descentralizadas.
Chamada de Procedimento Remoto (RPC) vs. REST	RPC é amigável ao usuário, mas pode causar acoplamento tecnológico; REST encapsula recursos, proporcionando independência do armazenamento interno, sendo preferível para microsserviços.
Comunicação Assíncrona	Arquiteturas orientadas a eventos melhoram o desacoplamento, mas complicam a gestão de mensagens e o tratamento de erros; padrões como extensões reativas ajudam a agilizar as interações.
Versionamento e Gestão de Mudanças	Inclui estratégias como adiar mudanças, usar versionamento semântico e manter endpoints coexistentes para facilitar transições durante mudanças disruptivas.
Interfaces de Usuário e Integração	Foco nas UIs como camadas composicionais com estratégias como Composição de API, Composição de Fragmentos de UI, e Backend para Frontend (BFF) para integrar serviços conforme diferentes necessidades dos usuários.
Integrando Softwares de Terceiros	Uma gestão cuidadosa é necessária para integração com produtos COTS e SaaS usando padrões como serviços fachada e o padrão estrangulador para lidar com transições de sistemas legados.
Conclusão	Enfatiza a importância de estratégias de integração apropriadas para serviços desacoplados e uma gestão eficaz de mudanças, estabelecendo uma base para futuros aprofundamentos.

## Capítulo 4: Integração

### Importância da Integração

A integração é crucial para microsserviços, garantindo sua



autonomia e permitindo mudanças e lançamentos independentes. Este capítulo explora as escolhas de tecnologia de integração para evitar armadilhas comuns associadas à Arquitetura Orientada a Serviços (SOA).

## **Escolhendo a Tecnologia de Integração Certa**

Existem várias opções para a comunicação entre microserviços, incluindo SOAP, XML-RPC, REST e protocol buffers. Considerações importantes incluem:

-

### **Evitar Mudanças Quebradas**

: Selecione uma tecnologia que minimize a necessidade de os consumidores realizarem alterações quando os microserviços evoluem.

-

### **Neutralidade Tecnológica**

: Mantenha APIs independentes de tecnologia para se adaptar a futuras mudanças nas pilhas tecnológicas.

-

### **Facilitar a Experiência do Consumidor**

: Torne fácil para os consumidores utilizarem os serviços, possivelmente fornecendo bibliotecas de cliente, enquanto evita acoplamentos excessivos.





-

## **Ocultar Detalhes Internos**

: Evite expor detalhes de implementação interna para garantir independência em relação a mudanças.

## **Estilos de Integração**

1.

### **Integração de Banco de Dados**

: Comum, mas problemática devido à exposição de representações de dados internas, aumentando o acoplamento e arriscando mudanças quebradas.

2.

### **Comunicação Síncrona vs. Assíncrona**

: Decidir entre comunicação síncrona (chamadas bloqueantes) e comunicação assíncrona (chamadas não bloqueantes) influencia os estilos de colaboração.

## **Orquestração vs. Coreografia**

-

### **Orquestração**

: Controle centralizado (como um maestro), mas leva à fragilidade e aumento do acoplamento.



-

## **Coreografia**

: Abordagem descentralizada onde os serviços respondem a eventos, promovendo acoplamento solto e flexibilidade.

## **Chamada de Procedimento Remoto (RPC) vs. REST**

-

### **RPC**

: Oferece facilidade de uso, mas pode levar ao acoplamento tecnológico e fragilidade.

-

### **REST**

: Encapsula recursos e permite representações independentes do armazenamento interno, tornando-se uma escolha favorável para microsserviços.

## **Comunicação Assíncrona**

Arquiteturas orientadas a eventos melhoram o desacoplamento, mas introduzem complexidade relacionada ao gerenciamento de mensagens e tratamento de erros. Padrões como extensões reativas podem agilizar ainda mais as interações.



## **Versionamento e Gerenciamento de Mudanças**

Estratégias para gerenciar mudanças quebradas incluem:

-

### **Adiamento de Mudanças**

: Evite mudanças quebradas imediatas sempre que possível.

-

### **Versionamento Semântico**

: Use estratégias de versionamento claras para comunicar mudanças esperadas.

-

### **Pontos de Extremidade Cohabitantes**

: Mantenha versões antigas e novas funcionando simultaneamente para facilitar transições.

## **Interfaces de Usuário e Integração**

Concentre-se nas interfaces de usuário como camadas compostas que integram vários serviços enquanto acomodam diferentes restrições dos usuários. Várias estratégias de composição de UI são exploradas:

-

### **Composição de API**



: Chamadas diretas de UIs para microsserviços, permitindo flexibilidade, mas podendo ser potencialmente ineficientes.

-

### **Composição de Fragmentos de UI**

: Serviços fornecem fragmentos de UI para uma integração mais rápida.

-

### **Backend para Frontend (BFF)**

: Serviços dedicados para UIs específicas que lidam com requisitos diversos sem acoplamento excessivo.

## **Integrando Software de Terceiros**

A integração de produtos COTS e SaaS requer gerenciamento cuidadoso para evitar problemas de dependência. Padrões como serviços de fachada e o padrão estrangulador ajudam a gerenciar a integração e transições de sistemas legados.

## **Conclusão**

O capítulo enfatiza a escolha de estratégias de integração apropriadas para manter serviços desacoplados e gerenciar mudanças de forma eficaz, fornecendo uma base para a exploração posterior em capítulos subsequentes.



## Pensamento crítico

**Ponto chave:** O Ponto Chave sobre Estratégias de Integração

**Interpretação crítica:** Sam Newman defende a priorização de métodos de integração independentes de tecnologia que minimizam mudanças drásticas; no entanto, as tendências tecnológicas mudam rapidamente, e suas preferências podem não se aplicar universalmente a todos os contextos ou indústrias.

Mais livros gratuitos no Bookey



Escanear para baixar



# Capítulo 5 Resumo : 5. Dividindo o Monólito

Seção	Resumo
Introdução	Este capítulo discute como decompor aplicações monolíticas em serviços menores e gerenciáveis, sem reescritas completas, e destaca estratégias de transformação eficazes.
Identificação de Limites	Limites são áreas na base de código que podem ser isoladas para modificação. Alta coesão e baixo acoplamento são importantes, sendo os contextos delimitados limites eficazes que alinham a organização do código com a estrutura organizacional.
Estudo de Caso: MusicCorp	MusicCorp é utilizado como estudo de caso para delinear contextos delimitados de alto nível—Catálogo, Finanças, Armazém, Recomendação—defendendo a refatoração do código em pacotes organizados utilizando IDEs modernas.
Razões para Dividir o Monólito	Uma abordagem incremental é sugerida, priorizando áreas com base na velocidade de mudança, estrutura da equipe, necessidades de segurança e dependências tecnológicas.
Desafios de Banco de Dados	Separar serviços complica a gestão do banco de dados, exigindo análise das dependências das tabelas, utilização de APIs para gerenciar relacionamentos e lidar com dados mutáveis compartilhados por meio de duplicação ou serviços separados.
Estágio da Separação	Uma separação gradual de esquemas, mantendo o código da aplicação, é recomendada para gerenciar riscos e garantir melhor controle sobre as mudanças.
Limites Transacionais	A transição para serviços pode interromper a consistência transacional, exigindo estratégias como reexecução de operações, transações compensatórias e cautela ao gerenciar transações distribuídas.
Soluções de Relatórios	A arquitetura de microsserviços afeta relatórios, explorando estratégias como chamadas de serviço para recuperação de dados, sistemas de dados centrais via bombas de dados, e atualizações acionadas por eventos para relatórios em tempo real.
Custo da Mudança	Incentivar pequenas mudanças incrementais ajuda na mitigação de riscos, utilizando técnicas de design e validação iterativa para lidar com a complexidade durante a transição.
Conclusão	Ao identificar limites para as fronteiras de serviço e gerenciar o custo de dividir serviços, as organizações podem evoluir de forma sustentável enquanto se preparam para a implantação e aumentam a complexidade de múltiplos serviços.

## Capítulo 5. Dividindo o Monólito

### Introdução

Mais livros gratuitos no Bookey



Escanear para baixar

Este capítulo aborda o desafio de decompor aplicações monolíticas em serviços menores e mais gerenciáveis sem a necessidade de uma reescrita completa. Os monólitos costumam se tornar complexos e intimidadoras com o tempo, levando as equipes a evitarem fazer alterações. No entanto, com as estratégias certas, essas aplicações podem ser efetivamente transformadas.

## **Identificando Juntas**

Juntas são definidas como partes do código que podem ser isoladas para trabalho sem afetar a base de código mais ampla. O objetivo é identificar juntas que possam evoluir para limites de serviços, focando em alta coesão e baixo acoplamento. Contextos delimitados dentro da organização servem como excelentes juntas, permitindo um melhor alinhamento entre a organização do código e a estrutura organizacional.

## **Estudo de Caso: MusicCorp**

Usando a MusicCorp como exemplo, o capítulo identifica contextos delimitados de alto nível dentro de um monólito



existente, incluindo Catálogo, Finanças, Armazém e Recomendações. O primeiro passo envolve criar pacotes para esses contextos e mover gradualmente o código existente para eles, utilizando IDEs modernas para refatoração.

## **Razões para Dividir o Monólito**

Uma abordagem incremental para a divisão do monólito é recomendada, focando em áreas de alto impacto para a separação. Fatores a serem considerados incluem:

-

### **Ritmo de Mudança:**

Serviços que experimentam uma alta frequência de atualizações devem ser priorizados.

-

### **Estrutura da Equipe:**

Alinhar a separação do código com equipes distribuídas geograficamente pode aumentar a responsabilidade e a gerenciabilidade.

-

### **Segurança:**

Serviços que gerenciam informações sensíveis podem se beneficiar de proteções aumentadas quando separados.

-



## **Tecnologia e Dependências:**

Juntas com menos código interdependente são mais fáceis de desentrelaçar.

## **Desafios de Banco de Dados**

Os bancos de dados apresentam complicações significativas ao separar serviços. As técnicas para lidar com problemas incluem:

- Identificação de dependências entre tabelas e análise de restrições que impactam os limites dos serviços.
- Tratamento de relacionamentos de chave estrangeira por meio de chamadas de API em vez de acesso direto ao banco de dados para manter a independência entre os serviços.
- Abordar o uso de dados estáticos e mutáveis compartilhados por meio de duplicação, arquivos de configuração ou serviços separados.

## **Estágio da Separação**

Em vez de uma abordagem de Big Bang, é aconselhável primeiro separar os esquemas enquanto mantém o código da aplicação intacto. Isso permite um melhor controle sobre as mudanças no banco de dados, enquanto mitiga os riscos



associados à integridade transacional entre os serviços.

## **Limites Transacionais**

A mudança de um esquema monolítico para serviços separados complica a consistência transacional. As estratégias incluem:

- Tentar novamente operações que falharam e aceitar consistência eventual.
- Empregar transações compensatórias para reverter alterações, se necessário.
- Explorar transações distribuídas para operações entre serviços com gestão cuidadosa da complexidade.

## **Soluções de Relatório**

A transição para microsserviços afeta as capacidades de relatório e levanta questões sobre o acesso a dados centralizados. Várias estratégias exploradas incluem:

- Recuperação de dados por meio de chamadas de serviço, embora isso possa desacelerar processos maiores.
- Bombas de dados que enviam dados para um sistema de report central enquanto mantêm as responsabilidades da equipe de serviço.



- Bombas de dados de eventos que aproveitam eventos de emissão para atualizações de dados em tempo real.

## **Custo da Mudança**

Enfatizar a importância de fazer pequenas mudanças incrementais permite uma melhor mitigação de erros. O uso eficaz de técnicas de design e validação iterativa ajuda a gerenciar a complexidade na transição para uma arquitetura de microsserviços.

## **Conclusão**

Ao identificar juntas para limites de serviços e reduzir o custo de divisão de serviços, as organizações podem evoluir seus sistemas de forma sustentável e atender às necessidades futuras. A preparação para a implantação e o enfrentamento da complexidade aumentada devido a múltiplos serviços serão passos essenciais a seguir.





# Capítulo 6 Resumo : 6. Implantação

## Capítulo 6: Implantação

### Visão Geral

Implantar aplicações monolíticas é mais simples do que implantar microsserviços devido à sua natureza interconectada. Este capítulo explora técnicas e tecnologias para implantar microsserviços, enfatizando a integração contínua (CI) e a entrega contínua (CD).

### Integração Contínua (CI)

-

#### Definição

: A CI tem como objetivo manter o trabalho dos desenvolvedores sincronizado e garantir que o novo código se integre suavemente com o código existente. Um servidor de CI automatiza checagens de código, cria artefatos e assegura que os testes sejam aprovados.

-



## **Benefícios**

: Feedback rápido sobre a qualidade do código, automação na criação de artefatos e rastreabilidade que liga artefatos ao código correspondente.

-

## **Práticas-Chave**

: Commits regulares na linha principal, manutenção de uma suíte de testes e priorização da correção de builds quebradas.

## **Mapeando CI para Microserviços**

- A CI deve ser mapeada para microserviços individuais, permitindo a implantação independente. Duas abordagens são:

1. Um único repositório com um build de CI para todos os microserviços.
2. Repositórios individuais e builds de CI por microserviço para validação independente.

**Instalar o aplicativo Bookey para desbloquear  
texto completo e áudio**

Mais livros gratuitos no Bookey



Escanear para baixar

Ad



Escanear para baixar



App Store  
Escolha dos Editores



22k avaliações de 5 estrelas

## Feedback Positivo

Afonso Silva

...cada resumo de livro não só  
...o, mas também tornam o  
...n divertido e envolvente. O  
...tizou a leitura para mim.

**Fantástico!**



Estou maravilhado com a variedade de livros e idiomas  
que o Bookey suporta. Não é apenas um aplicativo, é  
um portal para o conhecimento global. Além disso,  
ganhar pontos para caridade é um grande bônus!

Brígida Santos

FI



O  
só  
o  
O

na Oliveira

...correr as  
...ém me dá  
...omprar a  
...ar!

**Adoro!**



Usar o Bookey ajudou-me a cultivar um hábito de  
leitura sem sobrecarregar minha agenda. O design do  
aplicativo e suas funcionalidades são amigáveis,  
tornando o crescimento intelectual acessível a todos.

Duarte Costa

**Economiza tempo!**



O Bookey é o meu apli  
crescimento intelectual  
perspicazes e lindame  
um mundo de conheci

**Aplicativo incrível!**



Eu amo audiolivros, mas nem sempre tenho tempo para  
ouvir o livro inteiro! O Bookey permite-me obter um resumo  
dos destaques do livro que me interessa!!! Que ótimo  
conceito!!! Altamente recomendado!

Estevão Pereira

**Aplicativo lindo**



Este aplicativo é um salva-vidas para  
de livros com agendas lotadas. Os re  
precisos, e os mapas mentais ajudar  
o que aprendi. Altamente recomend

Teste gratuito com Bookey



# Capítulo 7 Resumo : 7. Testes

Seção	Conteúdo
Título do Capítulo	Capítulo 7: Testes
Importância dos Testes	Crucial para o desenvolvimento de software, especialmente em sistemas distribuídos como microserviços; garante a qualidade da funcionalidade e a rapidez na produção.
Tipos de Testes	<p>Testes Voltados à Tecnologia: Ajudam os desenvolvedores (por exemplo, testes de unidade, testes de desempenho).</p> <p>Testes Voltados ao Negócio: Assistem as partes interessadas a entender a funcionalidade do sistema (por exemplo, testes ponta a ponta, testes de usuário).</p>
Escopo dos Testes	<p>Pirâmide de Testes de Mike Cohn:</p> <p>Testes de Unidade: Rápidos, testam funções únicas com alta granularidade.</p> <p>Testes de Serviço: Avaliam as capacidades de serviços individuais, ignorando a interface do usuário.</p> <p>Testes Ponta a Ponta: Validam o sistema inteiro; maior confiança, mas feedback mais lento.</p>
Compensações	Maior escopo de testes leva a feedback mais lento e diagnóstico mais difícil de falhas. Recomenda-se mais testes de escopo reduzido para facilitar o rastreamento de erros.
Implementando Testes de Serviço	Testes de unidade são fáceis de criar; testes de serviço exigem stubs para isolar funcionalidades de dependências.
Mocks vs. Stubs	<p>Stubs: Retornam respostas predefinidas, sem rastreamento de chamadas.</p> <p>Mocks: Validam interações, rastreiam frequências de chamadas; podem se tornar frágeis se usados em excesso.</p>
Testes Ponta a Ponta	Testes de alto escopo; lentos, podem falhar por razões não relacionadas (por exemplo, problemas de rede); exigem cuidado no manuseio.
Pertencimento e Gestão dos Testes	Equipes de serviço escrevem testes relacionados aos seus serviços; propriedade e manutenção conjuntas entre equipes são fundamentais para evitar o efeito “cone de teste”.
Estratégias de Implantação	<p>Implantação Azul/Verde: Testar novas versões sem direcionar tráfego até que estejam operacionais.</p> <p>Lançamentos Canary: Introduzir gradualmente novas versões com monitoramento de desempenho.</p>
Testes Funcionais Cruzados (CFR)	Foca em requisitos não funcionais como desempenho e segurança; testes de desempenho são realizados juntamente com testes funcionais.



Seção	Conteúdo
Contratos Orientados ao Consumidor	Os consumidores especificam expectativas em relação ao serviço, garantindo conformidade sem testes extensos ponta a ponta.
Resumo	Otimize para feedback rápido e contratos orientados ao consumidor; equilibre os testes com a detecção de problemas pós-implantação; monitoramento contínuo e testes eficientes são críticos.

## Capítulo 7: Testes

Os testes são cruciais no desenvolvimento de software, especialmente em sistemas distribuídos como microsserviços. Este capítulo descreve várias estratégias de teste e sua importância para garantir a qualidade da funcionalidade enquanto se mantém a velocidade de produção.

### Tipos de Testes

-

#### Testes Focados em Tecnologia

: Focam em ajudar os desenvolvedores (por exemplo, testes unitários, testes de desempenho).

-

#### Testes Focados nos Negócios

: Ajudam as partes interessadas a entender a funcionalidade do sistema (por exemplo, testes de ponta a ponta, testes com



usuários).

A tendência é a automação, minimizando os testes manuais devido à ineficiência em uma arquitetura de microsserviços.

## **Escopo do Teste**

A

### **Piramide de Testes**

de Mike Cohn categoriza os testes automatizados em três tipos:

1.

#### **Testes Unitários**

: Rápidos, testam funções únicas com alta granularidade.

2.

#### **Testes de Serviço**

: Ignoram a interface do usuário para avaliar as capacidades de serviços individuais.

3.

#### **Testes de Ponta a Ponta**

: Validam o sistema completo, proporcionando maior confiança, mas com ciclos de feedback mais lentos.

## **Compensações**

Mais livros gratuitos no Bookey



Escanear para baixar



- Um escopo de teste maior resulta em feedback mais lento e diagnóstico mais difícil de falhas.
- Recomenda-se manter um número maior de testes de escopo menor, facilitando o rastreamento de erros.

## **Implementação de Testes de Serviço**

- Testes unitários são simples de criar.
- Testes de serviço requerem serviços stub para isolar funcionalidades de dependências externas.

## **Mocks vs. Stubs**

-

### **Stubs**

: Retornam respostas pré-definidas sem rastrear chamadas.

-

### **Mocks**

: Validam interações, rastreando frequências de chamadas, o que pode exacerbar a fragilidade se forem usados em excesso.

## **Testes de Ponta a Ponta**



- Testes de alto escopo que rodem pelo sistema completo.
- Eles são mais lentos, podem falhar por razões não relacionadas (por exemplo, problemas de rede) e requerem manuseio cuidadoso.

## **Propriedade e Gestão dos Testes**

- Testes relacionados a serviços específicos devem ser escritos pelas equipes de serviço, mas a propriedade compartilhada e a manutenção consistente entre as equipes são essenciais para evitar o efeito “cone de neve de testes”.

## **Estratégias de Implantação**

1.

### **Implantação Blue/Green**

: Testar novas versões sem direcionar tráfego até ser confirmada a operação.

2.

### **Lançamentos Canary**

: Introduzir gradualmente novas versões junto à versão atual, monitorando o desempenho antes da implantação completa.

## **Testes Funcionais Cruzados (CFR)**



- Enfatiza requisitos não funcionais como desempenho e segurança.
- Testes de desempenho devem ser conduzidos regularmente junto com testes funcionais.

## **Contratos Orientados a Consumidores**

- Usar cenários onde os consumidores especificam expectativas para os serviços, garantindo conformidade do produtor sem extensos testes de ponta a ponta.

## **Resumo**

- Focar na otimização para feedback rápido e contratos orientados a consumidores, enquanto entende o equilíbrio entre testar e detectar problemas após a implantação.
- Monitoramento contínuo e práticas de testes eficientes são críticos para manter a integridade do sistema.



# Capítulo 8 Resumo : 8. Monitoramento

## Capítulo 8: Monitoramento

### Introdução

Dividir um sistema em microsserviços aumenta a funcionalidade, mas complica o monitoramento em produção. Este capítulo discute os desafios e soluções para monitorar sistemas com granularidade fina.

### Os Desafios de Monitoramento dos Microsserviços

Em aplicações monolíticas, os problemas são mais fáceis de diagnosticar devido a um único ponto de falha. No entanto, as arquiteturas de microsserviços complicam o monitoramento, pois os problemas podem abranger múltiplos serviços e servidores. Para monitorar eficazmente os microsserviços, precisamos nos concentrar em métricas menores e usar agregação para entender a saúde geral do sistema.



## **Um Serviço, Um Servidor**

Comece monitorando o host, incluindo o desempenho da CPU e da memória, usando ferramentas como Nagios ou New Relic. O acesso aos logs da aplicação é crucial para identificar erros, e métricas básicas como tempos de resposta devem ser monitoradas.

## **Um Serviço, Vários Servidores**

Quando um serviço é executado em vários hosts, o rastreamento de métricas precisa ser tanto agregado quanto detalhado em níveis individuais. Ferramentas como Nagios podem ajudar a agrupar dados, enquanto a análise de logs pode se beneficiar de multiplexadores SSH. Balanceadores de carga podem fornecer insights adicionais sobre tempos de resposta.

## **Vários Serviços, Vários Servidores**

À medida que os sistemas se tornam mais complexos, a coleta de logs centralizada se torna essencial. Ferramentas como Logstash apoiam isso agregando logs para uma melhor análise. Soluções como Kibana permitem consultar e



visualizar dados de logs, revelando padrões ao longo do tempo.

## **Rastreamento de Métricas entre Vários Serviços**

Compreender o comportamento do sistema por meio das métricas coletadas ajuda a identificar tendências e o planejamento de capacidade. Ferramentas como Graphite facilitam a coleta de métricas em tempo real e permitem agregação para análise em todo o sistema.

## **Métricas de Serviço**

Os microsserviços devem expor suas próprias métricas, como tempos de resposta e taxas de erro. Isso ajuda a entender os padrões de uso e melhorar a funcionalidade do sistema. Bibliotecas de métricas como as de Codahale oferecem mais capacidades.

## **Monitoramento Sintético**

Para garantir a saúde do sistema além das métricas, transações sintéticas podem simular interações do usuário e relatar problemas com base no comportamento esperado.



Esse monitoramento avançado oferece melhores insights do que as métricas tradicionais sozinhas.

## **Implementando Monitoramento Semântico**

Teste de sistema de ponta a ponta pode ser adaptado para monitoramento contínuo, sendo executado periodicamente para verificar a saúde do sistema, garantindo que não produzam efeitos colaterais.

## **IDs de Correlação**

Implementar IDs de correlação permite rastrear o fluxo das chamadas de serviço, facilitando o diagnóstico de problemas. Os serviços devem ser padronizados para passar esses IDs em suas interações.

## **A Cascata**

Falhas em cascata devido a dependências exigem o monitoramento dos pontos de integração. Cada serviço deve acompanhar a saúde de suas conexões com outros serviços para evitar problemas não detectados.





## **Padronização**

Padrões uniformes de registro e métricas são essenciais para um monitoramento holístico do sistema. Ferramentas devem apoiar esses padrões para simplificar a agregação e os relatórios.

## **Considere o Público**

Os dados devem ser estruturados de acordo com as necessidades de diferentes partes interessadas. Alertas imediatos são necessários para questões críticas, enquanto tendências menores podem ser revisadas posteriormente.

## **O Futuro do Monitoramento**

As organizações estão se movendo em direção a sistemas unificados para métricas operacionais e de negócios para simplificar o processamento de eventos. Ferramentas como Riemann e Suro estão pavimentando o caminho para essa mudança.

## **Resumo dos Pontos-Chave**

**Mais livros gratuitos no Bookey**



Escanear para baixar

- Rastrear tempos de resposta, taxas de erro e métricas ao nível da aplicação para cada serviço.
- Monitorar a saúde das respostas de downstream, incluindo tempos de resposta e taxas de erro.
- Padronizar como e onde as métricas são coletadas para facilitar a agregação.
- Habilitar IDs de correlação entre serviços para facilitar a resolução de problemas.
- Avaliar a possibilidade de sistemas unificados de agregação de métricas para uma visão mais abrangente.
- Concentre-se nas tendências emergentes em direção a sistemas genéricos de processamento de eventos para um monitoramento holístico.

No próximo capítulo, o foco mudará para explorar os desafios de segurança únicos e as vantagens das arquiteturas com granularidade fina.



# Capítulo 9 Resumo : 9. Segurança

## Capítulo 9: Segurança

### Introdução às Preocupações de Segurança

A segurança dos dados ganhou atenção devido a violações de alto perfil e revelações sobre a privacidade dos dados. Este capítulo discute aspectos essenciais de segurança a serem considerados ao projetar sistemas, focando no equilíbrio entre medidas de segurança e eficiência operacional.

### Autenticação e Autorização

A autenticação verifica a identidade de um usuário ou sistema, tipicamente através de nomes de usuário e senhas. A autorização determina quais ações o usuário autenticado pode realizar. Em aplicações monolíticas, essas tarefas são geralmente tratadas internamente. Em sistemas distribuídos, soluções de Single Sign-On (SSO) como SAML e OpenID Connect são comuns, permitindo a gestão centralizada de identidades.



## Implementações Comuns de Single Sign-On

SSO simplifica a autenticação do usuário permitindo credenciais únicas para múltiplos sistemas. Ele pode utilizar provedores de identidade como Google, LDAP ou Active Directory, e frequentemente requer a ligação a serviços de diretório que gerenciam as informações dos usuários.

### Gateway de Single Sign-On

Em arquiteturas de microsserviços, um gateway pode centralizar o SSO, aliviando a redundância e possíveis problemas de acoplamento entre serviços. O gateway lida com a verificação de identidade, e a comunicação de serviço para serviço requer compartilhamento adequado de informações a respeito dos usuários.

**Instalar o aplicativo Bookey para desbloquear  
texto completo e áudio**

Mais livros gratuitos no Bookey



Escanear para baixar





# Ler, Compartilhar, Empoderar

Conclua Seu Desafio de Leitura, Doe Livros para Crianças Africanas.

## O Conceito



Esta atividade de doação de livros está sendo realizada em conjunto com a Books For Africa. Lançamos este projeto porque compartilhamos a mesma crença que a BFA: Para muitas crianças na África, o presente de livros é verdadeiramente um presente de esperança.

## A Regra



Ganhe 100 pontos



Resgate um livro



Doe para a África

Seu aprendizado não traz apenas conhecimento, mas também permite que você ganhe pontos para causas beneficentes! Para cada 100 pontos ganhos, um livro será doado para a África.

Teste gratuito com Bookee



# Capítulo 10 Resumo : 10. A Lei de Conway e o Design de Sistema

## Capítulo 10: A Lei de Conway e o Design de Sistema

### Introdução

Este capítulo explora os aspectos organizacionais do design de sistemas, enfatizando a importância de alinhar a arquitetura do sistema com a estrutura da empresa, de acordo com a Lei de Conway, que afirma que o design de um sistema refletirá a estrutura de comunicação da organização que o criou.

### Provas da Lei de Conway

- O trabalho original de Melvin Conway estabeleceu que os designs das organizações refletem suas dinâmicas de comunicação.
- Estudos mostram que organizações com pouca ligação produzem sistemas mais modulares em comparação com



aquelas com alta ligação.

## **Estudos de Caso**

1.

### **Microsoft Windows Vista**

: Pesquisas indicaram que a estrutura organizacional da Microsoft influenciou significativamente a qualidade do software do Vista.

2.

### **Amazon e Netflix**

: Ambas as empresas adotaram equipes pequenas e independentes responsáveis por todo o ciclo de vida do sistema, otimizando suas arquiteturas para velocidade e flexibilidade.

## **Implicações para o Design**

- A comunicação eficaz dentro de uma equipe resulta em um design e implementação de serviços mais eficientes.
- Equipes distribuídas geograficamente enfrentam custos de comunicação elevados, que podem levar a uma eficiência reduzida e a bases de código desatualizadas.
- A propriedade do serviço significa que as equipes são





responsáveis por todas as mudanças relacionadas ao seu serviço, promovendo autonomia e responsabilidade.

## **Serviços Compartilhados e Alternativas**

- A propriedade compartilhada de serviços pode ser problemática. Os fatores incluem a dificuldade em dividir serviços, gargalos de entrega e dinâmicas de equipes de recursos.
- Modelos de Código Aberto Interno permitem uma estrutura de contribuição colaborativa enquanto mantêm a qualidade através de controle das equipes centrais.

## **Alinhamento Organizacional**

- As equipes devem ser organizadas de acordo com os domínios de negócios para facilitar a comunicação e manter a propriedade sobre seus respectivos domínios.
- Serviços órfãos requerem uma clara definição de propriedade para garantir a manutenção e evolução adequadas.

## **Estudo de Caso: RealEstate.com.au**

**Mais livros gratuitos no Bookey**



Escanear para baixar

- A empresa exemplifica como alinhar a estrutura organizacional com as operações comerciais leva a uma maior autonomia e crescimento rápido. Cada equipe possui todo o ciclo de vida do serviço dentro de sua linha de negócios, facilitando a comunicação e a integração eficaz.

## **Reversão da Lei de Conway**

- Existe a possibilidade de o design do sistema influenciar a estrutura organizacional, como visto quando as empresas se adaptam a novas realidades operacionais.

## **Dinamismo das Pessoas**

- A transição para microsserviços requer entender as capacidades dos funcionários e seus níveis de conforto com mudanças.
- Uma articulação clara das responsabilidades em um ambiente de microsserviços é crucial para o sucesso.

## **Conclusão**

Alinhar o design do sistema com a estrutura organizacional, conforme ilustrado pela Lei de Conway, é essencial.



Reconhecer a relação entre esses elementos aumenta a sustentabilidade e a eficácia da arquitetura do sistema. O capítulo conclui com a nota de que, à medida que os sistemas escalam, considerações técnicas adicionais surgirão, que serão abordadas em seções subsequentes.

**Mais livros gratuitos no Bookey**



Escanear para baixar

## Pensamento crítico

**Ponto chave:** Alinhar a estrutura organizacional e a arquitetura do sistema pode aumentar a eficiência e a eficácia.

**Interpretação crítica:** Embora a Lei de Conway apresente argumentos convincentes para alinhar o design estrutural com a comunicação organizacional, é crucial reconhecer que esse ponto de vista pode simplificar excessivamente as complexidades do design de sistemas. As organizações são influenciadas por diversos fatores além das dinâmicas de comunicação, como demandas de mercado, avanços tecnológicos e capacidades da equipe. Críticos como Frederick P. Brooks argumentam em 'The Mythical Man-Month' que fatores humanos e a paralisia administrativa podem afetar significativamente os resultados dos projetos, independentemente do alinhamento organizacional. Consequentemente, enquanto caminhos de comunicação claros podem contribuir para um melhor design de sistema, eles não garantem o sucesso em todos os cenários, enfatizando a necessidade de uma compreensão mais ampla das dinâmicas do sistema.



# Capítulo 11 Resumo : 11. Microserviços em Larga Escala

## Capítulo 11: Microserviços em Larga Escala

### Introdução

As arquiteturas de microserviços podem rapidamente se transformar de sistemas simples em sistemas complexos. Este capítulo explora os desafios da escalabilidade dos microserviços, particularmente em relação a falhas e à gestão de numerosos serviços.

### Falhas Estão em Toda Parte

Falhas são inevitáveis em grandes sistemas devido a falhas de hardware e à natureza não confiável das redes. As organizações frequentemente se concentram em prevenir falhas em vez de se recuperar delas. Abraçar a ideia de que a falha pode acontecer permite um design de sistema melhor e estratégias de recuperação mais eficientes.



## **Compreendendo os Requisitos**

Identificar e compreender requisitos de múltiplas funções, como tempo de resposta, disponibilidade e durabilidade dos dados, é crítico. Diferentes serviços podem ter diferentes tolerâncias a falhas, influenciando como são construídos e gerenciados.

## **Degradação da Funcionalidade**

Criar um sistema resiliente exige a capacidade de degradar funcionalidade de forma segura durante interrupções de serviço. Compreender a criticidade de cada serviço permite que as equipes tomem decisões informadas sobre o que mostrar ou ocultar quando serviços dependentes estão fora do ar.

## **Medidas de Segurança Arquitetônica**

Implementar medidas de segurança arquitetônica, como timeouts, disjuntores e compartimentos, minimiza o impacto das falhas dentro dos microsserviços. Esses mecanismos ajudam a controlar as falhas em cascata que podem ocorrer



quando um serviço apresenta problemas.

## **A Organização Antifrágil**

Empresas como a Netflix abraçam as falhas, utilizando técnicas como engenharia do caos para preparar seus sistemas para interrupções. Desenvolver uma cultura que aprende com as falhas pode aumentar a resiliência do sistema.

## **Timeouts e Disjuntores**

Definir timeouts apropriados nas chamadas de serviço e implementar disjuntores ajuda os sistemas a falharem rapidamente, prevenindo desacelerações adicionais e permitindo uma recuperação rápida.

## **Compartimentos**

Implementar compartimentos ajuda a isolar falhas dentro dos serviços, garantindo que a falha de um serviço não impacte outros. Vários métodos, incluindo pools de conexões separados para diferentes serviços, criam resiliência contra interrupções.





## Estratégias de Escalabilidade

A escalabilidade pode ocorrer verticalmente (usando máquinas maiores) ou horizontalmente (distribuindo serviços entre várias instâncias). Introduzir redundância e distribuir carga entre múltiplos nós ou centros de dados pode melhorar significativamente a resiliência.

## Teorema CAP

O teorema CAP afirma que um sistema distribuído pode garantir apenas duas entre três: consistência, disponibilidade e tolerância a partições. As organizações devem avaliar suas necessidades para determinar quais compromissos adotar.

## Descoberta de Serviços

À medida que os sistemas crescem, torna-se essencial ter mecanismos de descoberta de serviços eficientes. As soluções incluem DNS tradicional, registros de serviços dinâmicos como Zookeeper e Consul, e ferramentas personalizadas que aproveitam as capacidades da nuvem.



## Documentação e Interação Humana

Uma documentação adequada é crítica para entender os serviços, levando a ferramentas como Swagger e HAL, que ajudam a documentar APIs e facilitam a exploração.

Construir um registro humano que reflita o estado do sistema melhora a compreensão e a gestão.

## Conclusão

Escalar microsserviços requer compreensão de suas complexidades e preparação para falhas inevitáveis. Este capítulo enfatiza estratégias para resiliência e gestão de serviços, sugerindo leituras adicionais para aplicar esses insights de forma eficaz.



# Capítulo 12 Resumo : 12. Unindo Tudo

## Capítulo 12: Unindo Tudo

Este capítulo resume os conceitos e princípios-chave relacionados a arquiteturas de microsserviços abordados nos capítulos anteriores.

### Princípios dos Microsserviços

- Defina seus próprios princípios com base nos conceitos principais descritos. Esses princípios ajudarão a criar pequenos serviços autônomos que possam trabalhar de forma eficaz juntos.

### Modelar em Torno de Conceitos de Negócio

- Estruture interfaces em torno de contextos de negócio limitados para garantir estabilidade e responsividade a mudanças organizacionais.

### Adote uma Cultura de Automação

Mais livros gratuitos no Bookey



Escanear para baixar

- Abrace a automação para lidar com a complexidade dos microsserviços, otimizando processos de implantação e garantindo entrega contínua.

## **Oculte Detalhes de Implementação Interna**

- Oculte detalhes de implementação para permitir a evolução independente dos serviços e evitar acoplamento forte entre eles.

## **Descentralize Tudo**

- Delegue decisões e controle para equipes responsáveis pelos serviços, promovendo autoatendimento e governança compartilhada para aumentar a autonomia.

## **Implantável de Forma Independente**

**Instalar o aplicativo Bookey para desbloquear  
texto completo e áudio**

**Mais livros gratuitos no Bookey**



Escanear para baixar





# As melhores ideias do mundo desbloqueiam seu potencial

Essai gratuit avec Bookey



Escanear para baixar



# Melhores frases do Criando Microserviços – 2ª Edição por Sam Newman com números de página

Ver no site do Bookey e gerar imagens de citações bonitas

## Capítulo 1 | Frases das páginas 15-34

1. Microserviços são pequenos serviços autônomos que trabalham juntos.
2. Muitas organizações descobriram que, ao adotar arquiteturas de microserviços mais refinadas, conseguem entregar software mais rapidamente e adotar novas tecnologias.
3. Quanto menor o serviço, mais você maximiza os benefícios e desvantagens da arquitetura de microserviços.
4. Um fator importante para nos ajudar a responder como pequeno é pequeno? é o quão bem o serviço se alinha às estruturas das equipes.
5. Se houver compartilhamento excessivo, nossos serviços consumidores se tornam acoplados às nossas representações internas.

Mais livros gratuitos no Bookey



Escanear para baixar

- 6.Com microserviços, podemos decidir usar diferentes tecnologias dentro de cada um. Isso nos permite escolher a ferramenta certa para cada tarefa, em vez de ter que selecionar uma abordagem mais padronizada e única.
- 7.Se um componente de um sistema falhar, mas essa falha não causar uma cascata, você pode isolar o problema e o restante do sistema pode continuar funcionando.
- 8.Com microserviços, podemos fazer uma alteração em um único serviço e implantá-lo independentemente do restante do sistema.
- 9.Empresas vencedoras adotaram o uso de equipes autônomas que possuem todo o ciclo de vida de seus serviços.
- 10.Os microserviços oferecem a possibilidade de as organizações transferirem a propriedade dos serviços entre as equipes para tentar manter as pessoas trabalhando em um único serviço co-localizadas.

## **Capítulo 2 | Frases das páginas 35-61**

- 1.Mais do que qualquer outro papel, os arquitetos





podem ter um impacto direto na qualidade dos sistemas construídos, nas condições de trabalho de seus colegas e na capacidade da organização de responder a mudanças, e ainda assim frequentemente parecemos errar nesse papel.

2. Se nosso software não é limitado pelas mesmas regras físicas com as quais arquitetos ou engenheiros reais têm que lidar, e o que criamos é projetado para ser flexível, adaptar-se e evoluir com os requisitos do usuário.
3. O equilíbrio entre parte artista, parte engenheiro, supervisionando a criação do que normalmente é uma visão singular, com todos os outros pontos de vista sendo subordinados...
4. Em vez de se preocupar demais com o que acontece em uma zona, o planejador urbano passará muito mais tempo trabalhando em como as pessoas e os serviços se movem de uma zona para outra.
5. O papel do arquiteto já é suficientemente assustador, então felizmente geralmente não temos que também definir



objetivos estratégicos!

6. Acredito em facilitar a realização do que é certo.

7. Eu tomaria a posição de que fiz o meu melhor para convencer as pessoas, mas, em última análise, não fui convincente o suficiente.

### **Capítulo 3 | Frases das páginas 62-79**

1. O que faz um bom serviço? Se você sobreviveu a uma implementação fracassada de SOA, pode ter uma ideia de onde estou querendo chegar. Mas, caso você não tenha essa (in)felicidade, quero que você se concentre em dois conceitos principais: desacoplamento e alta coesão.

2. Um serviço com baixo acoplamento sabe o mínimo necessário sobre os serviços com os quais colabora.

3. Queremos que comportamentos relacionados fiquem juntos, e comportamentos não relacionados fiquem em outro lugar.

4. A ideia é que um determinado domínio consiste em múltiplos contextos delimitados, e dentro de cada um



existem coisas que não precisam se comunicar externamente, bem como coisas que são compartilhadas externamente com outros contextos delimitados.

5.Quando modeladas como serviços, essas capacidades se tornam as operações chave que serão expostas através da rede para outros colaboradores.

6.Decompor prematuramente um sistema em microsserviços pode ser custoso, especialmente se você é novo no domínio.





Baixe o app Bookey para desfrutar

# Mais de 1000 resumos de livros com quizzes

**Teste grátis disponível!**

Escanear para baixar



## Capítulo 4 | Frases das páginas 80-144

1. Acredito que acertar na integração é o aspecto mais importante da tecnologia associada a microserviços. Faça isso bem, e seus microserviços manterão sua autonomia, permitindo que você os altere e libere independentemente do todo. Erre, e um desastre espera por você.
2. Queremos escolher tecnologias que garantam que isso aconteça o mais raramente possível. Por exemplo, se um microserviço adicionar novos campos a um conjunto de dados que envia, os consumidores existentes não devem ser afetados.
3. Mantenha suas APIs agnósticas em relação à tecnologia. Isso significa evitar tecnologias de integração que determinam quais pilhas de tecnologias podemos usar para implementar nossos microserviços.
4. Não queremos que nossos consumidores fiquem atados à nossa implementação interna. Isso leva a um acoplamento



maior.

5. De maneira geral, percebi que sistemas que tendem mais para a abordagem coreografada são mais fracamente acoplados, e são mais flexíveis e amenos a mudanças.
6. Falamos um pouco sobre algumas tecnologias que podem nos ajudar a implementar padrões de solicitação/resposta. E quanto à comunicação assíncrona baseada em eventos?
7. Entretanto, tenha cuidado com o mundo do middleware...  
Certifique-se de saber o que está adquirindo: mantenha seu middleware simples e concentre a inteligência nos pontos finais.
8. O conceito central do serviço como uma máquina de estados é poderoso.
9. Ao seguir os links, o cliente descobre progressivamente a API, o que pode ser uma capacidade realmente útil quando estamos implementando novos clientes.
10. Os males da otimização prematura já foram bem documentados antes, então não preciso me estender sobre isso aqui.



## Capítulo 5 | Frases das páginas 145-187

1. Mas nem tudo está perdido! Com as ferramentas certas à nossa disposição, podemos derrotar essa besta.
2. Uma abordagem incremental ajudará você a aprender sobre microserviços ao longo do caminho, e também limitará o impacto de cometer erros (e você vai cometer erros!).
3. Pense em nosso monólito como um bloco de mármore. Poderíamos destruí-lo completamente, mas isso raramente termina bem. Faz muito mais sentido trabalhar nele aos poucos.
4. Nosso código deve representar nossa organização, então nossos pacotes representando os contextos delimitados em nossa organização devem interagir da mesma forma que os grupos organizacionais no nosso domínio interagem.
5. Às vezes, fazer uma coisa mais devagar em troca de outras coisas é a decisão certa, especialmente se um ritmo mais lento ainda for perfeitamente aceitável.
6. Discutimos como é um bom serviço e por que servidores





menores podem ser melhores para nós.

7. Não há necessidade de que isso seja uma abordagem

explosiva. É algo que pode ser feito aos poucos, dia após dia, e temos muitas ferramentas à nossa disposição para acompanhar nosso progresso.

8. Se você puder ver as várias emendas que encontrou como

um grafo acíclico dirigido de dependências... isso pode ajudá-lo a identificar as emendas que provavelmente serão mais difíceis de desenredar.

9. Ao separar os esquemas, mas mantendo o código da

aplicação junto, damos a nós mesmos a capacidade de reverter nossas mudanças ou continuar ajustando as coisas sem impactar qualquer consumidor do nosso serviço.

10. Decompomos nosso sistema encontrando emendas ao

longo das quais os limites de serviço podem emergir, e isso pode ser uma abordagem incremental.

## **Capítulo 6 | Frases das páginas 188-230**

1. A integração contínua (CI) já existe há vários

anos. Vale a pena dedicar um tempo para revisar



os conceitos básicos, pois, especialmente quando pensamos sobre o mapeamento entre microserviços, builds e repositórios de controle de versão, há algumas opções diferentes a considerar.

- 2.Com a CI, o objetivo principal é manter todos sincronizados, o que conseguimos garantindo que o código recém-verificado se integre corretamente ao código existente.
- 3.Você realmente está fazendo isso? Suspeito que você provavelmente esteja usando integração contínua em sua própria organização. Se não, você deve começar. É uma prática fundamental que nos permite realizar mudanças de forma rápida e fácil, e sem a qual a jornada em direção aos microserviços será dolorosa.
- 4.A ferramenta é apenas algo que possibilita a abordagem.
- 5.Para permitir que esses artefatos sejam reutilizados, os colocamos em um repositório de algum tipo, seja fornecido pela própria ferramenta de CI ou em um sistema separado.
- 6.Um build verde que passa significa que nossas mudanças



foram integradas com segurança. Um build vermelho significa que a última mudança possivelmente não se integrou.

7. Se você deixar mais mudanças se acumularem, o tempo necessário para corrigir o build aumentará drasticamente.

8. Mapeando a Integração Contínua para Microserviços... queremos garantir que possamos fazer uma mudança em um único serviço e implantá-la independentemente do restante.

9. A abordagem que prefiro é ter um único build de CI por microserviço, para nos permitir fazer e validar rapidamente uma mudança antes da implantação em produção.





Baixe o app Bookey para desfrutar

# Mais de 1000 resumos de livros com quizzes

**Teste grátis disponível!**

Escanear para baixar



## Capítulo 7 | Frases das páginas 231-267

1. ‘A tendência recente tem sido abandonar os testes manuais em larga escala, em favor de automatizar o máximo possível.’
2. ‘A principal conclusão é que você vai querer testes de diferentes escopos para diferentes propósitos.’
3. ‘Se você atualmente realiza grandes volumes de testes manuais, eu sugeriria que abordasse isso antes de seguir muito adiante no caminho dos microsserviços, já que você não obterá muitos dos seus benefícios se não conseguir validar seu software rapidamente e eficientemente.’
4. ‘Uma boa regra geral é que você provavelmente vai querer uma ordem de magnitude a mais de testes à medida que desce a pirâmide.’
5. ‘Esses testes de ponta a ponta podem levar um tempo considerável. Já vi que eles levam até um dia para serem executados, se não mais... um conjunto de testes que ocupa o dia todo e, frequentemente, possui falhas que não têm nada a ver com funcionalidades quebradas é um desastre.’



6. ‘Quando detectamos testes instáveis, é essencial que façamos o nosso melhor para removê-los... essa tendência muito humana significa que precisamos encontrar e eliminar esses testes o mais rápido possível antes de começarmos a assumir que testes com falhas estão OK.’
7. ‘Se um desses CDCs quebrar durante a construção do serviço ao cliente, fica óbvio qual consumidor seria impactado.’
8. ‘À medida que você melhora essas outras áreas, pode começar a reduzir sua dependência de testes de ponta a ponta a tal ponto em que eles não são mais necessários.’

## **Capítulo 8 | Frases das páginas 268-285**

1. Para ver como, começaremos com o sistema mais simples que podemos: um único nó.
2. Precisamos dar sentido ao que de outra forma poderia ser um emaranhado caótico.
3. O segredo para saber quando entrar em pânico e quando relaxar é coletar métricas sobre como seu sistema se comporta ao longo de um período suficientemente longo





para que padrões claros emergjam.

4.Quanto mais complexas as interações entre os serviços, mais distantes estamos de realmente responder a essa pergunta.

5.Cada instância de serviço deve acompanhar e expor a saúde de suas dependências subordinadas.

6.Então, por que lidar com métricas operacionais e de negócios da mesma forma?

7.Esses sistemas conseguem fornecer muito mais flexibilidade, enquanto ao mesmo tempo realmente simplificam nossa arquitetura.

## **Capítulo 9 | Frases das páginas 286-322**

1.Eu vi algumas pessoas colocando todos os seus ovos em uma única cesta, dependendo do gateway para gerenciar cada passo por elas. E todos sabemos o que acontece quando temos um único ponto de falha...

2.Se você for uma empresa grande o suficiente, pode ter uma equipe de segurança da informação dedicada que pode



ajudá-lo. Caso contrário, encontre uma parte externa que possa.

3.A Regra de Ouro: Se há algo que você deve levar deste capítulo, que seja isto: não escreva sua própria criptografia.

4.Dados não protegidos são um passivo, especialmente se forem sensíveis.

5.Uma vez que você entenda os níveis de ameaça das diferentes partes do seu sistema, deve começar a ter uma noção de quando considerar a segurança durante a transmissão, em repouso ou não considerar de jeito nenhum.





Baixe o app Bookey para desfrutar

# Mais de 1000 resumos de livros com quizzes

**Teste grátis disponível!**

Escanear para baixar



## Capítulo 10 | Frases das páginas 323-346

1. Qualquer organização que projeta um sistema inevitavelmente produzirá um design cuja estrutura é uma cópia da estrutura de comunicação da organização.
2. Organizações com vínculos mais frouxos realmente criaram sistemas mais modulares e menos acoplados, enquanto o software de organizações mais focadas era menos modularizado.
3. A propriedade se estende a todos os aspectos do serviço, desde a definição de requisitos até a construção, implementação e manutenção da aplicação.
4. Aquelas organizações que são adaptáveis o suficiente para mudar não apenas sua arquitetura de sistema, mas também sua estrutura organizacional podem obter enormes benefícios em termos de maior autonomia das equipes e um tempo mais rápido para o mercado de novos recursos e funcionalidades.
5. Sem as pessoas a bordo, qualquer mudança que você queira



fazer pode estar condenada desde o início.

## Capítulo 11 | Frases das páginas 347-410

1. Em larga escala, mesmo que você compre o melhor equipamento, o hardware mais caro, não pode evitar o fato de que as coisas podem e vão falhar.
2. Incorporar a suposição de que tudo pode e irá falhar leva você a pensar de forma diferente sobre como resolver problemas.
3. A necessidade de mudar nossos sistemas para lidar com a escala não é um sinal de falha. É um sinal de sucesso.
4. Se podemos lidar com a falha de um serviço de forma graciosa, então é lógico que também podemos fazer atualizações no local de um serviço, uma vez que uma interrupção planejada é muito mais fácil de lidar do que uma não planejada.
5. Projete para um crescimento de  $\sim 10\times$ , mas planeje reescrever antes de  $\sim 100\times$ .
6. Abraçar e incitar falhas por meio de software, e construir sistemas que possam lidar com isso, é apenas parte do que



a Netflix faz.

7. Os timeouts são algo que é fácil de passar por alto, mas em um sistema downstream são importantes para acertar.
8. Ao saber qual carga você precisa gerenciar, quanto atualizados seus dados precisam estar e o que seu sistema pode fazer agora, você pode utilizar o cache de forma eficaz.
9. Quanto de falha você pode tolerar, ou quanto rápido seu sistema precisa ser, é determinado pelos usuários do seu sistema.
10. Sistemas que apenas agem lentamente são muito mais difíceis de lidar do que sistemas que falham rapidamente.

## **Capítulo 12 | Frases das páginas 411-421**

1. O todo deve ser maior do que a soma das partes.
2. Modele em torno de conceitos de negócios.
3. Adote uma cultura de automação.
4. Oculte detalhes de implementação interna.
5. Descentralize todas as coisas.
6. Isolar falhas.



7.Mudança é inevitável. Aceite-a.

**Mais livros gratuitos no Bookey**



Escanear para baixar





Baixe o app Bookey para desfrutar

# Mais de 1000 resumos de livros com quizzes

**Teste grátis disponível!**

Escanear para baixar



# Criando Microserviços – 2ª Edição

## Perguntas

[Ver no site do Bookey](#)

### Capítulo 1 | 1. Microserviços| Perguntas e respostas

#### 1.Pergunta

**O que influenciou o surgimento de microserviços no desenvolvimento de software moderno?**

Resposta:Os microserviços surgiram como uma tendência moldada por vários conceitos e designs anteriores, incluindo Design Orientado a Domínio, entrega contínua, automação de infraestrutura e a adoção de equipes pequenas e autônomas, como as da Amazon e da Netflix. Essas tendências se concentraram coletivamente em melhorar a arquitetura do sistema, aumentar a satisfação dos desenvolvedores e aumentar a capacidade de resposta às mudanças.

#### 2.Pergunta

**Como o conceito de 'pequeno' se aplica aos**

Mais livros gratuitos no Bookey



Escanear para baixar

## **microserviços?**

Resposta: No contexto dos microserviços, 'pequeno' refere-se a serviços que são focados em uma única função, que podem ser reescritos em um curto período, como duas semanas.

Embora não haja uma linha rígida sobre quantas linhas de código tornam um serviço 'pequeno', a ideia é que o serviço deve ser gerenciável por uma equipe pequena e focado em um limite de negócio específico.

### **3.Pergunta**

#### **Por que a autonomia é importante para os microserviços?**

Resposta: A autonomia é crucial porque permite que cada microserviço seja implantado e atualizado de forma independente, sem afetar os outros. Esse desacoplamento reduz as complexidades das interações entre serviços, permitindo que as equipes façam alterações de forma rápida e eficiente, o que é essencial para manter a agilidade do sistema.

### **4.Pergunta**



## **Quais são os principais benefícios da adoção de microsserviços?**

Resposta: Os principais benefícios dos microsserviços incluem heterogeneidade tecnológica, resiliência, escalabilidade independente, facilidade de implantação, melhor alinhamento organizacional e componibilidade. Essas vantagens permitem que as equipes escolham as melhores tecnologias para seus serviços, isolem falhas sem afetar o sistema inteiro e se adaptem mais prontamente a requisitos em mudança.

### **5. Pergunta**

## **Quais desafios os microsserviços abordam em comparação com sistemas monolíticos?**

Resposta: Os microsserviços ajudam a enfrentar os desafios de escalabilidade, complexidade de implantação e produtividade da equipe. Em sistemas monolíticos, as implantações são mais arriscadas porque uma única mudança frequentemente requer a reimplantação de uma grande base de código, o que pode causar interrupções em todo o sistema.



Os microsserviços permitem que alterações sejam feitas e implantadas no nível do serviço, reduzindo o risco e aumentando a frequência de lançamento.

## 6.Pergunta

**Como os microsserviços melhoram o alinhamento organizacional?**

Resposta:Os microsserviços permitem um melhor alinhamento com as estruturas organizacionais ao permitir que equipes menores possuam microsserviços, aumentando a produtividade. Eles também podem facilitar mudanças de propriedade entre equipes para garantir que os mantenedores do serviço sejam as mesmas pessoas que desenvolvem novos recursos.

## 7.Pergunta

**Por que a heterogeneidade tecnológica deve ser considerada um benefício dos microsserviços?**

Resposta:A heterogeneidade tecnológica permite que diferentes microsserviços usem a pilha tecnológica mais adequada para suas necessidades específicas, em vez de se



conformar a um único padrão. Essa flexibilidade pode levar a um melhor desempenho, adoção mais rápida de novas tecnologias e capacidade de responder efetivamente a demandas em mudança.

## 8.Pergunta

**O que torna os microserviços mais resilientes do que arquiteturas monolíticas?**

Resposta:Os microserviços melhoram a resiliência por meio de sua arquitetura intrínseca que isola falhas dentro de limites de serviço específicos. Se um microserviço falhar, os outros podem continuar a funcionar, evitando interrupções completas do sistema, comuns em arquiteturas monolíticas. Esse design promove um sistema mais confiável e mantível.

## 9.Pergunta

**Como os microserviços facilitam a componibilidade?**

Resposta:Os microserviços promovem a componibilidade ao permitir que serviços individuais sejam reutilizados em vários contextos e aplicações, permitindo que as organizações criem uma experiência de software coesa em





várias plataformas, como web e mobile, sem ficar restritas a implementações específicas.

## 10.Pergunta

### **Qual é a relevância do Princípio da Responsabilidade Única nos microsserviços?**

Resposta:O Princípio da Responsabilidade Única reforça a ideia de que cada microsserviço deve ter um propósito focado e encapsular a funcionalidade que muda pela mesma razão. Este princípio ajuda a manter um limite claro de responsabilidade e minimiza a complexidade desnecessária dentro dos serviços.

## 11.Pergunta

### **Quais são os riscos envolvidos na adoção de microsserviços?**

Resposta:Os microsserviços introduzem as complexidades dos sistemas distribuídos, incluindo desafios aumentados de implantação, monitoramento e necessidades de testes. As organizações podem enfrentar dificuldades com transações distribuídas e precisam entender como gerenciar falhas e



problemas de rede de forma eficaz.

## **Capítulo 2 | 2. O Arquiteto Evolutivo| Perguntas e respostas**

### **1.Pergunta**

**Qual é o papel de um arquiteto em um ambiente de microserviços?**

Resposta:O papel de um arquiteto em um ambiente de microserviços é criar uma visão técnica unificada que ajude a entregar sistemas que atendam às necessidades dos clientes. Isso inclui coordenar entre várias equipes ou até mesmo através da organização, tomar decisões críticas sobre limites de serviço, pilhas de tecnologia e garantir capacidade de adaptação às mudanças. Os arquitetos devem facilitar um ambiente colaborativo e garantir que os sistemas permaneçam gerenciáveis e escaláveis à medida que evoluem.

### **2.Pergunta**

**Como o autor sugere redefinir o título de 'arquiteto'?**

Resposta:O autor sugere redefinir o título de 'arquiteto' para

**Mais livros gratuitos no Bookey**



Escanear para baixar

se alinhar mais com o papel de um planejador urbano, em vez de um arquiteto tradicional que dita designs específicos. Os planejadores urbanos se concentram no zoneamento e permitem o crescimento orgânico e a evolução de uma cidade, o que se assemelha a como o software deve evoluir por meio da interação do usuário. Essa redefinição enfatiza flexibilidade, mudanças guiadas pelos usuários e tomada de decisões estratégicas, sem ditar cada aspecto.

### 3.Pergunta

**Que metáfora o autor usa para comparar a arquitetura de software ao planejamento urbano?**

Resposta:O autor compara a arquitetura de software ao planejamento urbano, equiparando o papel de um arquiteto ao de um planejador urbano. Assim como os planejadores urbanos designam áreas para uso industrial, residencial e comercial sem especificar o que cada edifício individual deve ser, os arquitetos de software devem definir limites e princípios de design que permitam que os serviços se desenvolvam de forma dinâmica, enquanto se concentram em



como interagem entre si.

#### 4.Pergunta

**Quais responsabilidades o autor atribui ao arquiteto evolutivo?**

Resposta:O arquiteto evolutivo tem várias responsabilidades centrais: 1) Visão - assegurar uma visão técnica clara que atenda às necessidades dos clientes e da organização; 2) Empatia - entender o impacto de suas decisões sobre usuários e membros da equipe; 3) Colaboração - envolver-se com colegas para refinar e executar a visão; 4) Adaptabilidade - evoluir a visão à medida que as necessidades dos clientes mudam; 5) Autonomia - equilibrar padronização e autonomia da equipe; 6) Governança - garantir que a implementação esteja alinhada com a visão técnica.

#### 5.Pergunta

**Por que o autor acredita que os arquitetos devem passar tempo codificando com suas equipes?**

Resposta:O autor acredita que os arquitetos devem passar tempo codificando com suas equipes para compreender



melhor os desafios e experiências dos desenvolvedores.

Engajar-se diretamente na codificação permite que os arquitetos entendam o impacto real de suas decisões arquitetônicas, facilita uma melhor comunicação e assegura que os sistemas em construção sejam realmente habitáveis e eficientes para aqueles que os gerenciam e trabalham com eles.

## 6.Pergunta

**O que significa 'dívida técnica' no contexto das decisões arquitetônicas?**

Resposta:Dívida técnica refere-se às compensações e concessões feitas durante a tomada de decisões arquitetônicas que podem trazer benefícios de curto prazo, mas acarretar custos a longo prazo. Simboliza o empréstimo de eficiência à custa da futura manutenibilidade, sugerindo que tais dívidas precisam ser monitoradas e pagas para evitar a deterioração da qualidade do software ao longo do tempo.

## 7.Pergunta

**O que os arquitetos devem considerar ao definir os limites de serviço?**



Resposta: Ao definir os limites de serviço, os arquitetos devem considerar como os serviços se comunicam entre si, as tecnologias usadas para a integração e a saúde geral do sistema. É crucial evitar a divergência excessiva entre os serviços para garantir uma integração sem falhas e proporcionar autonomia suficiente para as equipes dentro de suas zonas designadas.

## 8. Pergunta

**Como o autor recomenda governar escolhas tecnológicas e práticas?**

Resposta: O autor recomenda um modelo de governança coletivo liderado pelo arquiteto, mas envolvendo líderes de equipe de diferentes times de entrega. Essa estrutura garante a responsabilidade compartilhada pelas diretrizes, fomenta uma melhor comunicação e inclui várias perspectivas na tomada de decisões, alinhando as práticas com a visão técnica mais ampla.

## 9. Pergunta

**O que o autor diz sobre o equilíbrio entre simplicidade e complexidade na arquitetura de microsserviços?**





Resposta:O autor enfatiza a necessidade de um equilíbrio entre simplicidade e complexidade na arquitetura de microsserviços, alertando contra a diversidade excessiva em pilhas de tecnologia e estilos de integração. Embora permita que as equipes tenham autonomia para tomar decisões locais, deve haver também uma padronização entre os serviços para evitar desafios de integração e manter um sistema coeso.

## 10.Pergunta

### **Como os princípios arquitetônicos podem guiar as práticas de desenvolvimento?**

Resposta:Os princípios arquitetônicos guiam as práticas de desenvolvimento ao fornecer um quadro para a tomada de decisões, ajudando as equipes a alinhar seu trabalho com os objetivos estratégicos e oferecendo um conjunto de critérios para avaliar as compensações nas escolhas de design. Ao estabelecer princípios claros e memoráveis, os arquitetos podem aumentar a consistência em todo o sistema, permitindo ao mesmo tempo inovação prática e flexibilidade.

## **Capítulo 3 | 3. Como Modelar Serviços| Perguntas e**



## respostas

### 1.Pergunta

**Quais são os dois conceitos-chave que definem um bom microserviço, como discutido no capítulo?**

Resposta:Desacoplamento e alta coesão.

### 2.Pergunta

**Por que o desacoplamento é importante na arquitetura de microserviços?**

Resposta:O desacoplamento garante que mudanças feitas em um microserviço não exigem alterações em outro, permitindo um deployment mais fácil e independente dos serviços.

### 3.Pergunta

**O que significa alta coesão no contexto de microserviços?**

Resposta:Alta coesão significa que comportamentos relacionados estão agrupados em um serviço, permitindo uma gestão e modificações mais fáceis em um só lugar sem afetar outros serviços.

### 4.Pergunta

**Como o conceito de contextos delimitados ajuda na**



## **modelagem de microsserviços?**

Resposta: Contextos delimitados definem limites explícitos dentro de um domínio, determinando quais dados ou funcionalidades são expostos externamente e encapsulando responsabilidades específicas, o que ajuda a reduzir o acoplamento e aumentar a coesão.

### **5.Pergunta**

#### **Você pode fornecer uma analogia para contextos delimitados mencionada no capítulo?**

Resposta: Eric Evans usa a analogia de células com membranas, onde as membranas definem o que é interno e externo, ilustrando como os contextos delimitados funcionam dentro de um sistema.

### **6.Pergunta**

#### **Qual é a principal conclusão sobre a relação entre estrutura organizacional e microsserviços?**

Resposta: A estrutura organizacional influencia como modelar microsserviços; se equipes gerenciam capacidades específicas, essas devem ser refletidas como microsserviços



separados.

### **7.Pergunta**

**Qual é um risco associado à decomposição prematura de um sistema em microsserviços?**

Resposta:A decomposição prematura pode levar a limites de serviço incorretos, resultando em altos custos de mudanças e riscos devido a dependências inesperadas.

### **8.Pergunta**

**Quais foram as lições aprendidas com o projeto SnapCI na ThoughtWorks?**

Resposta:A equipe aprendeu que a identificação inicial das fronteiras de serviço estava incorreta e resultou em altos custos de mudança, levando-os a reverter para um sistema monolítico para estabilizar as fronteiras antes de reintroduzir os microsserviços.

### **9.Pergunta**

**Por que você deve pensar em termos de capacidades em vez de dados compartilhados ao definir contextos delimitados?**

Resposta:Focar em capacidades incentiva o design de



serviços que oferecem funções específicas, em vez de criar serviços CRUD anêmicos que operam apenas em dados sem definir comportamentos claros.

### 10.Pergunta

**Como contextos delimitados aninhados podem ser benéficos em arquiteturas de microsserviços?**

Resposta:Contextos delimitados aninhados permitem serviços mais granulares que podem encapsular funcionalidades específicas enquanto ainda operam sob uma fronteira de serviço mais ampla, simplificando a comunicação e testes.

### 11.Pergunta

**O que a frase 'tartarugas até o fim' significa no contexto de microsserviços?**

Resposta:Significa que contextos delimitados podem ser quebrados em níveis mais finos de complexidade, onde cada um pode ter suas próprias capacidades e responsabilidades específicas dentro de um contexto maior.

### 12.Pergunta

**Como a comunicação entre microsserviços deve ser**



## **modelada de acordo com o capítulo?**

Resposta:A comunicação deve ser modelada em termos de conceitos de negócios, espelhando os termos e ideias usados dentro da organização, para garantir clareza e alinhamento entre os serviços.

### **13.Pergunta**

## **Qual é o aviso sobre modelar limites de serviço ao longo de costuras técnicas?**

Resposta:Modelar limites ao longo de costuras técnicas pode levar a sistemas frágeis e excessivamente complexos; fatias focadas nos negócios devem ter prioridade como o principal motor para definir limites de serviço.





Ad



Escanear para baixar



# Experimente o aplicativo Bookey para ler mais de 1000 resumos dos melhores livros do mundo

Desbloqueie **1000+** títulos, **80+** tópicos

Novos títulos adicionados toda semana

Product & Brand

 Liderança & Colaboração

 Gerenciamento de Tempo

 Relacionamento & Comunicação

 Estratégia de Negócios

 Criatividade

 Memórias

 Conheça a Si Mesmo

 Psicologia

Empreendedorismo

 História Mundial

 Comunicação entre Pais e Filhos

 Autocuidado

 Mente

## Visões dos melhores livros do mundo

Desenvolvimento

Os 7 Hábitos das Pessoas Altamente Eficazes



Mini Hábitos



Hábitos Atômicos



O Clube das 5 da Manhã



Como Fazer Amigos e Influenciar Pessoas



Como Não



Teste gratuito com Bookey



## Capítulo 4 | 4. Integração| Perguntas e respostas

### 1.Pergunta

**Qual é o aspecto mais importante da tecnologia para microserviços?**

Resposta:Acertar a integração preserva a autonomia dos microserviços, permitindo mudanças e lançamentos independentes.

### 2.Pergunta

**O que deve ser evitado ao fazer alterações em um microserviço?**

Resposta:Alterações que quebrem a compatibilidade e exijam que os consumidores modifiquem seu consumo devem ser evitadas para manter uma experiência contínua.

### 3.Pergunta

**Por que é importante manter APIs indiferentes à tecnologia?**

Resposta:Manter APIs indiferentes à tecnologia permite flexibilidade na escolha de pilhas tecnológicas no futuro, evitando o bloqueio por fornecedor.

### 4.Pergunta



## **Qual é a principal desvantagem da integração direta com o banco de dados?**

Resposta: A integração com o banco de dados expõe detalhes da representação interna, levando a um acoplamento apertado e a possíveis mudanças que quebrem compatibilidade quando atualizações de esquema são feitas.

### **5.Pergunta**

## **Quais são as características da comunicação síncrona em comparação com a assíncrona?**

Resposta: A comunicação síncrona é mais fácil de entender e bloqueia até a conclusão. A comunicação assíncrona é não bloqueante e pode aumentar a responsividade.

### **6.Pergunta**

## **O que são orquestração e coreografia no contexto de processos de serviço?**

Resposta: A orquestração gerencia processos com uma autoridade central, enquanto a coreografia permite que os serviços reagem a eventos, promovendo o desacoplamento.

### **7.Pergunta**

## **Qual deve ser o foco ao lidar com serviços relacionados a**



## **clientes na MusicCorp?**

Resposta:O foco deve ser em forte coesão e baixo acoplamento, permitindo que os serviços gerenciem eventos do ciclo de vida do cliente de forma independente.

## **8.Pergunta**

### **O que é o padrão Tolerant Reader e por que é importante?**

Resposta:O padrão Tolerant Reader permite que os clientes ignorem campos desnecessários nas respostas, prevenindo mudanças que quebrem a compatibilidade quando a representação interna evolui.

## **9.Pergunta**

### **Como o versionamento pode ser gerenciado de forma eficaz em microsserviços?**

Resposta:O versionamento pode ser gerenciado usando versionamento semântico e coexistindo endpoints antigos e novos para facilitar a transição para os consumidores.

## **10.Pergunta**

### **O que é o Padrão de Aplicativo Estrangulador?**

Resposta:O Padrão de Aplicativo Estrangulador permite





substituir gradualmente sistemas legados interceptando chamadas e redirecionando-as para novas implementações.

### **11.Pergunta**

**Quais são os riscos envolvidos na criação de bibliotecas de clientes para microsserviços?**

Resposta: Bibliotecas de clientes podem levar a um acoplamento apertado se implementarem a lógica de negócios do lado do servidor, resultando em mudanças necessárias em múltiplos clientes para atualizações do servidor.

### **12.Pergunta**

**Como interfaces de usuário podem compor efetivamente múltiplos microsserviços?**

Resposta: Utilizando abordagens como composição de API, composição de fragmentos de UI ou backends para frontends para combinar capacidades em uma experiência de usuário coesa.

### **13.Pergunta**

**Por que é melhor adiar a personalização de software de terceiros (COTS)?**



Resposta:A personalização geralmente gera altos custos, e pode ser mais eficiente adaptar os processos organizacionais para se adequar às capacidades do software existente.

#### **14.Pergunta**

**Qual é uma consequência potencial de depender demais de middleware ou corretores de mensagens?**

Resposta:Sobrecarregar-se de middleware pode levar a configurações complexas, onde a lógica de negócios acaba sendo distribuída inadequadamente, tornando o sistema mais difícil de gerenciar.

#### **15.Pergunta**

**Por que o REST é frequentemente preferido ao RPC para comunicação entre microserviços?**

Resposta:O REST oferece melhor desacoplamento e flexibilidade, permitindo que os serviços evoluam de forma independente sem mudanças que quebrem a compatibilidade.

### **Capítulo 5 | 5. Dividindo o Monólito| Perguntas e respostas**

#### **1.Pergunta**

**Qual é a importância de identificar 'seams' em uma base**



## **de código monolítica?**

Resposta: Seams são partes do código que podem ser tratadas isoladamente, permitindo que mudanças sejam feitas independentemente do resto do sistema. Identificar seams ajuda a definir fronteiras de serviço que são coesas e fracamente acopladas, facilitando a eventual decomposição de uma aplicação monolítica em microserviços.

## **2.Pergunta**

### **Como alguém deve começar a dividir uma aplicação monolítica em microserviços?**

Resposta: Comece identificando contextos delimitados de alto nível dentro da aplicação, que correspondem a funcionalidades ou módulos distintos. Crie pacotes em torno desses contextos e refatore a base de código de forma incremental, progredindo gradualmente para dividi-la em serviços independentes.

## **3.Pergunta**

### **Por que uma abordagem incremental é preferida ao**





## **decompor um sistema monolítico?**

Resposta: Uma abordagem incremental permite que as equipes aprendam e se adaptem à medida que progridem, limitando os riscos associados a mudanças significativas. Isso também permite que a organização gerencie os custos e impactos de forma eficaz, garantindo que erros possam ser corrigidos sem grandes interrupções.

### **4.Pergunta**

#### **Como podemos lidar com o desafio das dependências de banco de dados ao dividir um monólito?**

Resposta: Para abordar as dependências de banco de dados, deve-se mapear o código que acessa o banco de dados aos seus respectivos contextos delimitados. Isso inclui refatorar o esquema para delinear fronteiras e, potencialmente, implementar APIs para acesso a dados, reduzindo o acoplamento forte e a complexidade das transações entre serviços.

### **5.Pergunta**

#### **Qual é o papel de um gerenciador de transações em**



## **transações distribuídas?**

Resposta:Um gerenciador de transações coordena a conclusão de múltiplas transações entre diferentes serviços, garantindo que todos os sistemas confirmem ou revertam as alterações coletivamente para manter a consistência dos dados.

## **6.Pergunta**

**Quais são as soluções potenciais para relatórios em uma arquitetura de microserviços que utiliza bancos de dados separados?**

Resposta:As soluções incluem bombas de dados para enviar periodicamente dados para um banco de dados central de relatórios, recuperar dados por meio de chamadas de serviço ou usar arquiteturas orientadas a eventos para atualizar um sistema de relatórios com base em eventos de serviços.

## **7.Pergunta**

**O que as organizações podem fazer para reduzir o custo e a complexidade de mudar a arquitetura de um sistema?**

Resposta:Investir em provisionamento autônomo,



frameworks de serviços leves e bibliotecas pode reduzir significativamente a barreira para a criação de novos serviços e facilitar modificações mais simples no sistema. Promover uma cultura de mudanças incrementais apoia ainda mais isso.

## 8.Pergunta

**Como o reconhecimento das causas raiz do crescimento monolítico ajuda a abordar desafios arquitetônicos?**

Resposta:Entender por que os serviços crescem muito permite que as equipes antecipem a necessidade de dividi-los antes que se tornem inadmissíveis. Isso também possibilita estratégias proativas para gerenciar o crescimento, facilitando transições mais suaves para uma arquitetura mais modular.

## 9.Pergunta

**Qual é a importância de monitorar o impacto das mudanças incrementais na arquitetura dos sistemas?**

Resposta:Monitorar ajuda a identificar as repercussões de cada mudança, permitindo que as equipes tomem decisões informadas sobre os próximos passos. Também aumenta a



agilidade na resolução de problemas que surgem a partir das mudanças e evolui efetivamente o sistema.

## 10.Pergunta

**De que maneiras as equipes podem documentar seu design de serviços e responsabilidades para garantir clareza durante uma transição?**

Resposta:Usar técnicas como cartões CRC

(Classe-Responsabilidade-Colaboração) pode ajudar as equipes a articular claramente as responsabilidades e interações de vários serviços, promovendo uma melhor colaboração e compreensão entre os membros da equipe.

## Capítulo 6 | 6. Implantação| Perguntas e respostas

### 1.Pergunta

**Qual é o principal objetivo da Integração Contínua (CI)?**

Resposta:O principal objetivo da Integração

Contínua (CI) é manter todos sincronizados entre si, garantindo que as novas alterações no código sejam integradas corretamente ao código existente. A CI automatiza o processo de verificação, checando



regularmente o código, verificando se ele compila e se os testes passam. Isso minimiza problemas de integração e permite lançamentos mais rápidos e confiáveis.

## 2.Pergunta

**Como a CI pode beneficiar a implantação de microserviços?**

Resposta:A CI pode beneficiar significativamente a implantação de microserviços, automatizando o processo de construção e teste de cada microserviço de forma independente. Isso permite que as equipes façam alterações com confiança em serviços individuais, sem o risco de afetar os outros, acelerando assim o ciclo de desenvolvimento e reduzindo conflitos de implantação.

## 3.Pergunta

**Por que é crucial ter um conjunto de testes para validar alterações na CI?**

Resposta:É crucial ter um conjunto de testes na CI porque, sem eles, você só pode confirmar que o código compila, sem



saber se quebra funcionalidades existentes. Os testes oferecem a segurança de que as alterações realizadas não afetaram negativamente o comportamento do sistema, mantendo a qualidade geral do software.

#### 4.Pergunta

**O que a prática eficaz de CI exige em termos de prioridades da equipe quando uma compilação falha?**

Resposta:A prática eficaz de CI exige que consertar uma compilação quebrada se torne a prioridade número um da equipe. Se a compilação falhar, nenhuma nova alteração no código deve ser integrada até que o problema seja resolvido. Isso ajuda a manter uma compilação estável e garante que os problemas de integração sejam tratados prontamente.

#### 5.Pergunta

**Quais são algumas desvantagens de usar uma única compilação de CI para múltiplos microsserviços?**

Resposta:Usar uma única compilação de CI para múltiplos microsserviços pode levar a tempos de compilação mais longos, uma vez que alterações em um serviço acionam



testes para todos os serviços. Isso complica a implantação de artefatos, já que pode ser confuso quais serviços precisam ser implantados após uma alteração. Muitas vezes resulta em equipes implantando tudo junto, aumentando os riscos e reduzindo os benefícios dos microsserviços.

## 6.Pergunta

**Por que um repositório de código-fonte único e uma compilação de CI por microsserviço são preferidos?**

Resposta:Um repositório de código-fonte único e uma compilação de CI por microsserviço são preferidos porque permitem alterações independentes, validações mais rápidas e uma propriedade mais clara. Cada microsserviço pode ser desenvolvido, testado e implantado separadamente, minimizando o impacto das alterações e facilitando operações mais suaves como um todo.

## 7.Pergunta

**Quais desafios podem surgir ao implantar vários serviços por host?**

Resposta:Implantar vários serviços por host pode complicar o





gerenciamento de recursos e a monitorização, uma vez que o impacto de desempenho de um serviço pode afetar os outros. Além disso, as implantações podem se tornar complexas devido a dependências conflitantes entre os serviços, tornando menos claro quando surgem problemas e dificultando o gerenciamento independente dos serviços.

## 8.Pergunta

**Qual é o papel da automação na gestão de um grande número de hosts e microsserviços?**

Resposta:A automação é crucial, pois reduz a sobrecarga manual na implantação e gerenciamento de serviços em numerosos hosts. Ela permite implantações, monitoramento e escalonamento consistentes e eficientes, garantindo que os desenvolvedores sejam produtivos e possam se concentrar na qualidade do código em vez de tarefas operacionais.

## 9.Pergunta

**Como o uso do Docker melhora a implantação de microsserviços?**

Resposta:O uso do Docker melhora a implantação de



microserviços ao permitir a contenção leve e rápida dos serviços. Isso simplifica a configuração do ambiente, permite escalonamento flexível e gerencia as dependências de forma eficaz, aumentando significativamente as velocidades de desenvolvimento e implantação.

### **10.Pergunta**

**Qual é a importância de manter uma interface de implantação uniforme?**

Resposta:Manter uma interface de implantação uniforme é vital, pois simplifica o processo de implantação em diferentes ambientes, reduzindo a probabilidade de erros durante a implantação. Tal interface permite comandos claros e consistentes para gerenciar implantações ao longo do desenvolvimento, teste e produção.

### **11.Pergunta**

**Qual é a estratégia sugerida quando as equipes estão começando com microserviços?**

Resposta:Ao começar com microserviços, geralmente é sugerido que os serviços sejam maiores e todos em uma



única compilação durante as fases iniciais para reduzir o custo de alterações entre serviços. No entanto, à medida que os limites dos serviços se estabilizam e a compreensão do domínio melhora, as equipes devem transitar para a separação em compilação individuais para melhor gerenciamento e eficiência de implantação.

**Mais livros gratuitos no Bookey**



Escanear para baixar



Escanear para baixar



# Por que o Bookey é um aplicativo indispensável para amantes de livros



## Conteúdo de 30min

Quanto mais profunda e clara for a interpretação que fornecemos, melhor será sua compreensão de cada título.



## Clipes de Ideias de 3min

Impulsione seu progresso.



## Questionário

Verifique se você dominou o que acabou de aprender.



## E mais

Várias fontes, Caminhos em andamento, Coleções...

Teste gratuito com Bookey



## Capítulo 7 | 7. Testes| Perguntas e respostas

### 1.Pergunta

**Quais são os principais tipos de testes automatizados abordados no Capítulo 7?**

Resposta:Os principais tipos de testes automatizados discutidos incluem Testes de Unidade, Testes de Serviço e Testes de Ponta a Ponta. Os Testes de Unidade se concentram em funções ou métodos individuais, os Testes de Serviço avaliam as capacidades de serviços individuais, e os Testes de Ponta a Ponta avaliam a funcionalidade do sistema como um todo.

### 2.Pergunta

**Por que é importante ter uma variedade de testes automatizados em uma arquitetura de microsserviços?**

Resposta:Ter uma variedade de testes automatizados é crucial em microsserviços para garantir que diferentes áreas de funcionalidade sejam cobertas, proporcionando um retorno rápido para os desenvolvedores enquanto se mantém



a confiança na confiabilidade geral do sistema. Testes diferentes abordam as complexidades distintas introduzidas pelos microsserviços.

### 3.Pergunta

**O que é a Pirâmide de Testes e o que ela representa?**

Resposta:A Pirâmide de Testes, introduzida por Mike Cohn, representa um modelo para categorizar testes automatizados de acordo com seu escopo: Testes de Unidade na base (grande quantidade, feedback rápido), Testes de Serviço no meio e Testes de Ponta a Ponta no topo (menos em número, feedback mais lento). A pirâmide enfatiza que a maioria dos testes deve ser de unidade para um feedback rápido.

### 4.Pergunta

**Quais são os trade-offs ao lidar com diferentes tipos de testes em arquiteturas de microsserviços?**

Resposta:À medida que se avança na Pirâmide de Testes, o escopo e a confiança de que a funcionalidade funciona aumentam, mas o tempo do ciclo de feedback também cresce. Testes de Ponta a Ponta fornecem uma visão mais





ampla, mas podem ser mais lentos e mais difíceis de diagnosticar quando falham, enquanto os Testes de Unidade oferecem resultados rápidos, mas cobertura menos abrangente.

## 5.Pergunta

**Como os Contratos Orientados ao Consumidor podem melhorar os testes em microsserviços?**

Resposta:Os Contratos Orientados ao Consumidor (CDCs) permitem que os consumidores de serviços definam suas expectativas em relação a um serviço, que são então testadas em relação ao provedor de serviços. Isso proporciona testes mais focados e confiáveis e ajuda a prevenir mudanças que quebram a funcionalidade, possibilitando um feedback e uma colaboração mais rápidos entre as equipes.

## 6.Pergunta

**Qual é um padrão anti-padrão comum em testes que envolvem testes de ponta a ponta, e como ele pode ser abordado?**

Resposta:Um padrão anti-padrão comum é o 'cone de neve de teste' ou pirâmide invertida, onde há testes de unidade muito





escassos e uma dependência excessiva de testes de grande escopo, tornando o feedback lento e difícil de gerenciar. Para abordar isso, as equipes devem garantir uma abordagem equilibrada, priorizando um número maior de testes de unidade pequenos e rápidos.

## 7.Pergunta

**Qual é o papel dos testes de desempenho no contexto de microsserviços?**

Resposta:Os testes de desempenho avaliam a escalabilidade e a latência do sistema, essenciais em uma arquitetura de microsserviços onde chamadas podem abranger vários serviços. Eles ajudam a identificar gargalos precocemente e garantem que o sistema possa lidar com cargas esperadas sem degradação no desempenho.

## 8.Pergunta

**Por que é crucial melhorar o Tempo Médio para Reparar (MTTR) em vez de se concentrar apenas no Tempo Médio Entre Falhas (MTBF)?**

Resposta:Focar na melhoria do MTTR permite que as equipes se recuperem rapidamente de falhas, reduzindo o



impacto nos usuários. Essa abordagem reconhece que, embora se possa esforçar para minimizar falhas, não se pode eliminá-las completamente, por isso é vital estar preparado para uma recuperação rápida.

## 9.Pergunta

**Como as equipes podem gerenciar efetivamente os desafios dos testes de ponta a ponta em um ambiente de microsserviços?**

Resposta:As equipes podem gerenciar os desafios dos testes de ponta a ponta limitando o escopo dos testes a jornadas principais, usando técnicas como implantações azuis/ verdes para lançamentos seguros, empregando lançamentos canário para avaliar novas funcionalidades e mantendo uma propriedade compartilhada do conjunto de testes de ponta a ponta entre as equipes.

## 10.Pergunta

**O que é monitoramento semântico e como ele se relaciona com testes de ponta a ponta?**

Resposta:O monitoramento semântico envolve o uso de testes de ponta a ponta para monitorar a funcionalidade do



sistema em produção após a implantação. Ele serve como uma rede de segurança para garantir que o sistema opere como esperado, especialmente após a liberação de mudanças, aproveitando execuções de testes de ponta a ponta anteriores para validar o ambiente ao vivo.

## **Capítulo 8 | 8. Monitoramento| Perguntas e respostas**

### **1.Pergunta**

**Qual é um dos principais desafios de usar microsserviços para monitoramento de sistemas?**

Resposta:O principal desafio com microsserviços é a complexidade aumentada de monitorar múltiplos serviços em vários servidores e sistemas, o que torna difícil identificar erros e problemas de desempenho em comparação com uma arquitetura monolítica.

### **2.Pergunta**

**Como deve-se abordar monitoramento de um sistema baseado em microsserviços de forma eficaz?**

Resposta:Para monitorar efetivamente um sistema baseado em microsserviços, deve-se focalizar o monitoramento dos



pequenos componentes e usar a agregação para criar uma visão unificada de todo o sistema, permitindo tanto insights detalhados quanto uma visão geral.

### 3.Pergunta

**O que deve ser monitorado na configuração mais simples de microsserviço?**

Resposta:Na configuração mais simples, deve-se monitorar o uso da CPU e da memória do host, acessar os logs do servidor em busca de erros e acompanhar os tempos de resposta da aplicação.

### 4.Pergunta

**Qual é o benefício de usar ferramentas de agregação de logs como Logstash e Kibana?**

Resposta:Ferramentas como Logstash e Kibana permitem a coleta e análise centralizada de logs, facilitando a filtragem de grandes volumes de logs de múltiplos serviços e a visualização de tendências e erros, ajudando assim na resolução de problemas.

### 5.Pergunta

**Como os IDs de correlação podem ajudar no**

Mais livros gratuitos no Bookey



Escanear para baixar

## **monitoramento de interações complexas entre microsserviços?**

Resposta:IDs de correlação permitem o rastreamento de solicitações através de múltiplos serviços ao vincular logs de vários componentes, facilitando o diagnóstico de problemas que surgem de chamadas entre serviços e proporcionando uma melhor compreensão da sequência de eventos que levaram a um erro.

### **6.Pergunta**

## **Por que a padronização é importante no monitoramento de microsserviços?**

Resposta:A padronização no monitoramento é crucial, pois garante consistência nos formatos de logs e nas métricas entre diferentes serviços, tornando a agregação e a análise mais simples e eficazes.

### **7.Pergunta**

## **O que são transações sintéticas e como elas ajudam no monitoramento?**

Resposta:Transações sintéticas simulam interações de



usuários com o sistema para testar sua funcionalidade e saúde. Essa forma de monitoramento pode revelar problemas sistêmicos que não seriam detectados apenas por métricas tradicionais.

## 8.Pergunta

**O que você deve considerar ao projetar alertas para seu sistema de monitoramento?**

Resposta:Ao projetar alertas, considere a urgência da informação, o público específico que receberá os alertas e como preferem consumir os dados, garantindo que questões críticas gerem atenção imediata.

## 9.Pergunta

**Como a unificação de métricas operacionais e de negócios pode melhorar o monitoramento do sistema?**

Resposta:A unificação de métricas operacionais e de negócios permite uma visão holística do desempenho do sistema e do comportamento do usuário, facilitando respostas oportunas a problemas e tornando a tomada de decisões mais informada.



## 10.Pergunta

**Qual é o papel de ferramentas como Graphite no rastreamento de métricas para microsserviços?**

Resposta:O Graphite oferece um meio para coletar, armazenar e visualizar métricas em tempo real, permitindo que equipes criem gráficos e analisem tendências de desempenho, o que é especialmente valioso em ambientes que escalam rapidamente.

## Capítulo 9 | 9. Segurança| Perguntas e respostas

### 1.Pergunta

**Quais são alguns dos principais aspectos de segurança que devemos considerar ao projetar sistemas?**

Resposta:Ao projetar sistemas, devemos considerar a proteção de dados tanto em trânsito quanto em repouso, a segurança dos sistemas operacionais subjacentes e a segurança da rede. Também precisamos pensar nos processos de autenticação e autorização, incluindo quem pode acessar os dados e seus respectivos papéis.





## 2.Pergunta

**Como podemos determinar quanto de segurança é suficiente?**

Resposta:Determinar quanto de segurança é suficiente envolve entender a sensibilidade dos dados manipulados, os possíveis vetores de ameaça e equilibrar as medidas de segurança com a usabilidade. É crucial avaliar o risco associado à exposição de dados e ajustar os protocolos de segurança de acordo.

## 3.Pergunta

**Qual é a diferença entre autenticação e autorização?**

Resposta:Autenticação é o processo de verificar a identidade de um usuário, tipicamente através de nomes de usuário e senhas. Autorização, por outro lado, determina quais ações um usuário autenticado está autorizado a realizar com base em sua identidade e papéis associados.

## 4.Pergunta

**O que é Single Sign-On (SSO) e por que é útil?**

Resposta:Single Sign-On (SSO) permite que os usuários se autentiquem uma vez e tenham acesso a múltiplos sistemas



sem precisar fazer login separadamente em cada um. Isso melhora a experiência do usuário ao reduzir a fadiga de senhas e melhora a gestão de segurança ao centralizar a autenticação.

## 5.Pergunta

**Por que devemos evitar criar nossos próprios protocolos criptográficos?**

Resposta: Evitar a criação de seus próprios protocolos criptográficos é vital, pois é fácil cometer erros que podem levar a vulnerabilidades. Em vez disso, confie em algoritmos criptográficos estabelecidos e bem revisados para garantir a segurança.

## 6.Pergunta

**Como podemos proteger dados em repouso?**

Resposta: Para proteger dados em repouso, utilize algoritmos de criptografia bem conhecidos, como AES-128 ou AES-256. Proteja adequadamente as chaves de criptografia, garantindo que não sejam armazenadas com os dados que protegem. Atualize e aplique correções em seu software de



criptografia regularmente à medida que vulnerabilidades são descobertas.

## 7.Pergunta

**Que papel o elemento humano desempenha na segurança?**

Resposta:O elemento humano é crítico na segurança, pois os funcionários podem, sem querer, criar vulnerabilidades através de engenharia social ou negligência. Implementar processos para gestão de credenciais, revogação de acessos e treinamento de usuários sobre práticas de segurança é essencial.

## 8.Pergunta

**Qual é o 'problema do deputado confuso' na comunicação entre serviços?**

Resposta:O 'problema do deputado confuso' ocorre quando um serviço realiza erradamente ações em nome de uma parte maliciosa, abusando de seus privilégios. Medidas eficazes devem estar em vigor para garantir que os serviços validem os pedidos com precisão para evitar acessos não autorizados.



## 9.Pergunta

**Quais são algumas estratégias para incorporar segurança em nossos processos de desenvolvimento?**

Resposta: Incorpore segurança no desenvolvimento educando as equipes sobre as melhores práticas de segurança, integrando testes de segurança automatizados em pipelines de CI/CD e realizando avaliações de segurança externas regulares, como testes de penetração.

## 10.Pergunta

**Qual é a importância da defesa em profundidade?**

Resposta: A defesa em profundidade é essencial, pois fornece múltiplas camadas de segurança, reduzindo a probabilidade de ataques bem-sucedidos. Mesmo que uma camada de segurança falhe, as outras ainda podem fornecer proteção, garantindo a integridade do sistema como um todo.



Ad



Escanear para baixar



App Store  
Escolha dos Editores



22k avaliações de 5 estrelas

## Feedback Positivo

Afonso Silva

...cada resumo de livro não só  
..., mas também tornam o  
...divertido e envolvente. O  
...tizou a leitura para mim.

**Fantástico!**



Estou maravilhado com a variedade de livros e idiomas  
que o Bookey suporta. Não é apenas um aplicativo, é  
um portal para o conhecimento global. Além disso,  
ganhar pontos para caridade é um grande bônus!

Brígida Santos

FI



O  
só  
o  
O

na Oliveira

...correr as  
...ém me dá  
...omprar a  
...ar!

**Adoro!**



Usar o Bookey ajudou-me a cultivar um hábito de  
leitura sem sobrecarregar minha agenda. O design do  
aplicativo e suas funcionalidades são amigáveis,  
tornando o crescimento intelectual acessível a todos.

Duarte Costa

**Economiza tempo!**



O Bookey é o meu apli  
crescimento intelectual  
perspicazes e lindame  
um mundo de conheci

**Aplicativo incrível!**



Eu amo audiolivros, mas nem sempre tenho tempo para  
ouvir o livro inteiro! O Bookey permite-me obter um resumo  
dos destaques do livro que me interessa!!! Que ótimo  
conceito!!! Altamente recomendado!

Estevão Pereira

**Aplicativo lindo**



Este aplicativo é um salva-vidas para  
de livros com agendas lotadas. Os re  
precisos, e os mapas mentais ajudar  
o que aprendi. Altamente recomend

Teste gratuito com Bookey





## Capítulo 10 | 10. A Lei de Conway e o Design de Sistema| Perguntas e respostas

### 1.Pergunta

**O que é a Lei de Conway e por que é significativa para o design de sistemas?**

Resposta:A Lei de Conway afirma que qualquer organização que projeta um sistema inevitavelmente produzirá um design cuja estrutura é uma cópia da estrutura de comunicação da organização. Sua importância reside na compreensão de que a forma como as equipes se comunicam e são estruturadas impactará diretamente a arquitetura dos sistemas que criam. Se as equipes estão organizadas em torno de funcionalidades específicas, mas têm uma comunicação deficiente, o sistema resultante provavelmente será fortemente acoplado e complexo. Por outro lado, equipes bem estruturadas e comunicativas podem produzir sistemas mais modulares e eficientes.

### 2.Pergunta

Mais livros gratuitos no Bookey



Escanear para baixar

## **Como a estrutura de uma organização impacta a qualidade do software?**

Resposta: Estudos, como os sobre o Windows Vista, mostram que a estrutura organizacional desempenha um papel crítico na qualidade do software. Métricas relacionadas à organização foram encontradas como as medidas estatisticamente mais relevantes dos componentes propensos a erros do software. Isso indica que, quando as equipes estão bem alinhadas e capazes de se comunicar eficientemente, o software resultante é tipicamente de maior qualidade.

### **3. Pergunta**

## **O que significa a propriedade do serviço no contexto do desenvolvimento de software?**

Resposta: A propriedade do serviço refere-se ao conceito de que uma única equipe é responsável por todos os aspectos de um serviço, desde o desenvolvimento e a implantação até a manutenção. Essa abordagem promove responsabilidade, autonomia e um senso de propriedade dentro da equipe, levando a entregas mais rápidas e resultados de maior





qualidade. Ela garante que a equipe possa fazer alterações com total compreensão e controle sobre o serviço, sem dependências excessivas.

#### **4.Pergunta**

##### **Quais desafios surgem a partir da propriedade compartilhada de serviços?**

Resposta:A propriedade compartilhada de serviços pode levar a problemas de coordenação e atrasos na entrega, pois várias equipes podem competir para fazer alterações ou implementar recursos. Isso frequentemente resulta em gargalos, ineficiências e pode diminuir o senso de autonomia entre as equipes. A falta de uma propriedade clara pode levar a uma má responsabilização e a uma tendência de adiar responsabilidades, o que complica os esforços de manutenção e desenvolvimento.

#### **5.Pergunta**

##### **Como as organizações podem garantir que sua estrutura esteja alinhada com a arquitetura do seu sistema?**

Resposta:As organizações devem buscar alinhar suas equipes



em torno de contextos delimitados que reflitam suas operações comerciais. Esse alinhamento facilita melhor comunicação dentro da equipe e interações mais suaves com partes interessadas externas. Além disso, ao elaborar estratégias organizacionais, os líderes devem considerar o ciclo de vida e as interdependências de seus serviços, visando estruturar as equipes de maneira a promover a independência enquanto garante o alinhamento com os objetivos gerais do negócio.

## 6.Pergunta

**Qual é o papel da cultura na adoção de microsserviços e mudanças de tecnologia?**

Resposta:A prontidão cultural para mudança é crucial ao adotar microsserviços. As equipes devem fazer a transição de uma mentalidade de 'jogar trabalho por cima do muro' para uma em que são totalmente responsáveis por seu trabalho. Um treinamento adequado e suporte devem ser fornecidos conforme os desenvolvedores se adaptam a novas responsabilidades e preocupações operacionais. Construir



uma cultura de apoio que incentive a experimentação e a colaboração é vital para uma transição suave para os microsserviços e para promover a adaptabilidade dentro da organização.

## 7.Pergunta

**Como as equipes podem lidar efetivamente com gargalos de entrega em ambientes de microsserviços?**

Resposta: Para gerenciar os gargalos de entrega, as equipes podem implementar várias estratégias: adicionar recursos adicionais às equipes sobrecarregadas, agilizar os processos de comunicação, padronizar as pilhas de tecnologia para facilitar a colaboração entre equipes, ou dividir serviços quando apropriado para permitir que grupos trabalhem de forma independente. Essas medidas podem ajudar a manter um fluxo de trabalho constante e garantir a entrega pontual de novos recursos ou funcionalidades.

## 8.Pergunta

**Qual é a importância de ferramentas na suporte ao modelo de código aberto interno?**



**Resposta:**As ferramentas são essenciais para gerenciar efetivamente um modelo de código aberto interno, pois fornecem a infraestrutura necessária para colaboração e contribuição. O uso de sistemas de controle de versão distribuído para gerenciar código, facilitar solicitações de pull e ter ferramentas amigáveis para construir e implantar software são todos cruciais para manter a qualidade do código e fomentar a participação da comunidade, ao mesmo tempo minimizando gargalos.

## **Capítulo 11 | 11. Microsserviços em Larga Escala| Perguntas e respostas**

### **1.Pergunta**

**O que devemos abraçar ao lidar com microsserviços em grande escala?**

**Resposta:**Falhas. Reconhecer que falhas podem e vão acontecer nos permite construir sistemas mais resilientes que conseguem lidar com interrupções de forma elegante, em vez de apenas tentar evitá-las.

### **2.Pergunta**

**Como as organizações podem se preparar melhor para**



## **falhas no sistema?**

Resposta: Ao projetar sistemas que podem se recuperar de maneira eficiente de falhas, como implementando limites de tempo, disjuntores e barreiras, as organizações podem evitar falhas em cascata.

## **3.Pergunta**

### **Qual é o teorema CAP e por que é importante?**

Resposta: O teorema CAP afirma que em um sistema distribuído, você não pode alcançar consistência, disponibilidade e tolerância a partições ao mesmo tempo. Compreender isso ajuda na tomada de decisões de design informadas, dependendo das necessidades do negócio.

## **4.Pergunta**

### **Por que é essencial entender os requisitos interfuncionais em uma arquitetura de microserviços?**

Resposta: Entender os requisitos interfuncionais ajuda a determinar os níveis aceitáveis de tempo de resposta, disponibilidade e durabilidade dos dados para cada serviço, com base nas necessidades dos usuários.



## 5.Pergunta

**O que é idempotência e por que é benéfica em microsserviços?**

Resposta:Idempotência garante que realizar a mesma operação várias vezes produza o mesmo resultado, o que é crucial para a confiabilidade em sistemas distribuídos onde falhas podem causar chamadas repetidas.

## 6.Pergunta

**Como os sistemas podem lidar com períodos de indisponibilidade de forma elegante?**

Resposta:Ao degradar a funcionalidade de maneira apropriada, como oferecendo serviço parcial ou exibindo mensagens informativas quando certos serviços estão fora do ar, os usuários ainda podem interagir com o sistema sem uma falha total.

## 7.Pergunta

**Qual técnica de escalonamento oferece resiliência contra interrupções de serviço?**

Resposta:Usar pools de conexão separados para diferentes microsserviços pode evitar que um serviço lento ou fora do



ar afete os outros, melhorando a resiliência geral do sistema.

### 8.Pergunta

**Qual foi a lição aprendida na exibição de servidores do Google com hard drives presos por Velcro?**

Resposta:Isso destaca a importância de se preparar para falhas de hardware e projetar sistemas que possam substituir componentes facilmente, em vez de temer falhas de equipamentos.

### 9.Pergunta

**Qual é a importância das medidas de segurança arquitetônicas no design de microsserviços?**

Resposta:Elas são cruciais para garantir que uma única falha não leve a interrupções generalizadas, permitindo que serviços individuais falhem discretamente sem derrubar todo o sistema.

### 10.Pergunta

**Como a abordagem antifrágil pode aumentar a robustez do sistema?**

Resposta:Ao incentivar testes regulares de falha por meio de práticas de engenharia do caos, as organizações se tornam





mais fortes e mais adaptáveis a interrupções inesperadas.

### **11.Pergunta**

**Qual é o papel da documentação em microserviços?**

Resposta:Ela garante que os usuários finais e os desenvolvedores entendam como usar e integrar vários serviços de forma eficiente, o que é crucial à medida que os sistemas se tornam maiores e mais complexos.

### **12.Pergunta**

**Por que é importante adotar uma abordagem iterativa para a arquitetura em microserviços?**

Resposta:Porque os designs iniciais podem não se sustentar ao serem escalados; adaptar a arquitetura à medida que o sistema cresce ajuda a manter o desempenho e a confiabilidade sem superdimensionar desde o início.

### **13.Pergunta**

**Quais são algumas estratégias práticas para gerenciar a descoberta de serviços em microserviços?**

Resposta:Usar registros dinâmicos de serviços, como Consul, ou implementar soluções DNS que resolvam em balanceadores de carga pode simplificar a gestão de



instâncias de serviços e facilitar a distribuição de carga.

#### **14.Pergunta**

**Como as organizações podem melhor utilizar o cache na arquitetura de microsserviços?**

Resposta:Implementar caching estrategicamente pode melhorar significativamente o desempenho, mas deve ser equilibrado para evitar a entrega de dados obsoletos, com atenção dada à complexidade do cache.

#### **15.Pergunta**

**Qual é a vantagem de usar sistemas baseados em trabalhadores para processar tarefas?**

Resposta:Eles ajudam a escalar cargas de trabalho de forma eficiente, utilizando capacidade excedente durante períodos de baixa demanda sem comprometer a confiabilidade do sistema, permitindo que as tarefas sejam tratadas de maneira resiliente.

#### **16.Pergunta**

**Quais são os benefícios de ter limites arquitetônicos claros entre os serviços?**

Resposta:Reduz as dependências e facilita a manutenção,



escalonamento e atualizações mais fáceis, permitindo que as equipes trabalhem de forma independente e tomem decisões rapidamente.

### **17.Pergunta**

**Como as organizações podem garantir que estão preparadas para falhas em seus microsserviços?**

Resposta:Ao praticar cenários de falha, ter mecanismos de fallback e garantir que os sistemas sejam resilientes e possam lidar com interrupções parciais, as organizações podem manter a continuidade do serviço.

### **18.Pergunta**

**Como a ideia de 'abraçar a falha' influencia o design do sistema?**

Resposta:Ela muda o foco de simplesmente prevenir falhas para antecipar e planejar para elas, promovendo designs que melhoram a recuperação e a continuidade apesar dos problemas.

### **19.Pergunta**

**Por que as empresas devem evitar acoplar fortemente seus microsserviços?**



Resposta:O acoplamento solto permite que as equipes operem de forma independente, reduz o risco de falhas em cascata e possibilita práticas de desenvolvimento mais ágeis e responsivas.

## 20.Pergunta

**Quais padrões-chave devem ser padronizados em uma arquitetura de microsserviços para garantir robustez?**

Resposta:Time-outs, disjuntores e barreiras são padrões essenciais que ajudam a manter a estabilidade do serviço e prevenir falhas generalizadas durante interrupções.

## 21.Pergunta

**Como as organizações podem testar efetivamente seus sistemas contra falhas?**

Resposta:Usando práticas controladas de engenharia do caos, simulações de falha e testes de arquitetura sob carga podem revelar vulnerabilidades e aumentar a resiliência do sistema.

## 22.Pergunta

**O que o design arquitetônico deve considerar para bancos de dados escaláveis em microsserviços?**

Resposta:Os bancos de dados devem acomodar



escalabilidade tanto em leituras quanto em gravações, estando cientes das implicações dos sistemas distribuídos em relação à consistência e disponibilidade.

### **23.Pergunta**

**Que lição a história do Chaos Monkey da Netflix ensina aos desenvolvedores?**

Resposta: Testar regularmente os sistemas ao induzir intencionalmente falhas garante que os desenvolvedores estejam preparados para interrupções reais e cria uma cultura de robustez dentro das equipes.

### **24.Pergunta**

**Por que é importante empregar uma mistura de estratégias de escalonamento em microsserviços?**

Resposta: Diferentes componentes podem ter requisitos de carga e disponibilidade variados, exigindo tanto escalonamento horizontal quanto vertical para otimizar desempenho e resiliência.

## **Capítulo 12 | 12. Unindo Tudo| Perguntas e respostas**

### **1.Pergunta**

**Quais são os princípios-chave dos microsserviços?**

Mais livros gratuitos no Bookey



Escanear para baixar

Resposta: Os princípios-chave dos microsserviços incluem:

1. Modelar em torno de conceitos de negócios:

Estruturar interfaces com base nas necessidades do negócio para estabilidade e adaptabilidade a mudanças.

2. Adotar uma cultura de automação: Utilizar automação em testes e implantação para lidar com a complexidade de forma eficaz.

3. Ocultar detalhes de implementação interna:

Focar em modelos compartilhados enquanto mantém detalhes de implementação privados para a independência do serviço.

4. Descentralizar tudo: Incentivar a autonomia da equipe na tomada de decisões e na implantação, minimizando a centralização.

5. Implantável de forma independente: Garantir que os microsserviços possam ser lançados separadamente, permitindo flexibilidade e



atualizações rápidas de recursos.

6. Isolar falhas: Planejar para falhas a fim de aumentar a resiliência, evitando quedas em cascata no sistema.

7. Altamente observável: Implementar monitoramento abrangente para entender o comportamento do sistema entre todos os serviços.

## 2.Pergunta

**Como as organizações devem abordar a transição para microsserviços?**

Resposta:As organizações devem adotar uma abordagem gradual para a transição para microsserviços. Isso envolve começar com uma compreensão sólida do domínio e identificar limites de módulos claros. Um sistema monolítico pode servir como um ponto de partida estável, permitindo uma decomposição gradual em microsserviços à medida que a organização se familiariza com a complexidade. É importante construir ferramentas e práticas automatizadas para gerenciar microsserviços de forma eficaz antes de





aumentar o número de serviços. As práticas de observabilidade e monitoramento devem evoluir juntamente com o sistema para enfrentar a complexidade crescente à medida que mais serviços são introduzidos.

### 3.Pergunta

**Por que abraçar a mudança é crucial na arquitetura de microserviços?**

Resposta: Abraçar a mudança é crucial porque as arquiteturas de microserviços introduzem numerosas decisões e complexidades que requerem adaptação constante. À medida que os sistemas evoluem, a capacidade de fazer mudanças incrementais ajuda a minimizar os riscos associados a reescritas em larga escala. Uma arquitetura evolucionária permite melhorias contínuas, facilitando o aprendizado e a adaptação a novos requisitos. As organizações devem ver essa evolução como uma jornada contínua, e não como um destino fixo, promovendo uma cultura que recebe e gerencia mudanças de forma eficaz.

### 4.Pergunta

Mais livros gratuitos no Bookey



Escanear para baixar

## **Que precauções devem ser tomadas ao usar microserviços?**

Resposta: Ao usar microserviços, as organizações devem ter cautela em relação à sua compreensão do domínio. Limites de serviço mal definidos podem levar a mudanças dispendiosas nas colaborações de serviço. Também é aconselhável começar com uma abordagem monolítica se o domínio não estiver bem compreendido e manter práticas de monitoramento adequadas para evitar que as complexidades se agravem à medida que o número de serviços cresce. Além disso, reconhecer os desafios associados à escala e aos processos manuais é essencial para evitar a supercomplicação da arquitetura.

### **5.Pergunta**

## **O que pode ser aprendido com as experiências do autor com microserviços?**

Resposta: A partir das experiências do autor, é evidente que construir as ferramentas e práticas necessárias para gerenciar microserviços leva tempo e adaptação gradual. As



organizações devem avaliar sua prontidão e capacidade de abraçar mudanças antes de se comprometer totalmente com microserviços. Aprender tanto com sucessos quanto com falhas em iterações pequenas e gerenciáveis pode fornecer insights valiosos que facilitam transições mais suaves e uma melhor gestão geral dos microserviços.

**Mais livros gratuitos no Bookey**



Escanear para baixar



# Ler, Compartilhar, Empoderar

Conclua Seu Desafio de Leitura, Doe Livros para Crianças Africanas.

## O Conceito



Esta atividade de doação de livros está sendo realizada em conjunto com a Books For Africa. Lançamos este projeto porque compartilhamos a mesma crença que a BFA: Para muitas crianças na África, o presente de livros é verdadeiramente um presente de esperança.

## A Regra



Ganhe 100 pontos



Resgate um livro



Doe para a África

Seu aprendizado não traz apenas conhecimento, mas também permite que você ganhe pontos para causas beneficentes! Para cada 100 pontos ganhos, um livro será doado para a África.

Teste gratuito com Bookee





# Criando Microserviços – 2ª Edição Quiz e teste

Ver a resposta correta no site do Bookey

## Capítulo 1 | 1. Microserviços| Quiz e teste

1. Microserviços são definidos como pequenos serviços autônomos que trabalham juntos, focando em uma única responsabilidade.
2. Microserviços promovem serviços fortemente acoplados para melhorar a comunicação entre serviços.
3. Adotar uma arquitetura de microserviços sempre simplifica os processos de implantação.

## Capítulo 2 | 2. O Arquiteto Evolutivo| Quiz e teste

1. Os arquitetos no desenvolvimento de software têm um papel crítico em fornecer uma visão técnica unificada para atender às necessidades dos clientes.
2. A arquitetura de software possui o mesmo nível de rigor e responsabilização das áreas de engenharia tradicional.
3. Os arquitetos devem se envolver diretamente com as

Mais livros gratuitos no Bookey



Escanear para baixar

equipes de desenvolvimento para promover uma melhor comunicação e colaboração.

## **Capítulo 3 | 3. Como Modelar Serviços| Quiz e teste**

- 1.O acoplamento frouxo é um princípio que enfatiza que mudanças em um serviço exigem mudanças em outros serviços.
- 2.Alta coesão refere-se a separar funcionalidades não relacionadas em diferentes serviços para simplificar modificações.
- 3.Os contextos delimitados ajudam a definir quais dados são internos e quais dados são compartilhados entre os serviços.





Baixe o app Bookey para desfrutar

# Mais de 1000 resumos de livros com quizzes

**Teste grátis disponível!**

Escanear para baixar





## Capítulo 4 | 4. Integração| Quiz e teste

- 1.A integração de microsserviços é importante para garantir sua autonomia e possibilitar mudanças e lançamentos independentes.
- 2.A integração de banco de dados é uma prática recomendada em microsserviços, pois minimiza a exposição de representações de dados internas.
- 3.A comunicação assíncrona aprimora o desacoplamento em microsserviços, mas adiciona complexidade relacionada à gestão de mensagens.

## Capítulo 5 | 5. Dividindo o Monólito| Quiz e teste

- 1.O capítulo sugere que isolar limites para as fronteiras dos serviços pode aproveitar a alta coesão e o baixo acoplamento.
- 2.O livro recomenda uma abordagem de big-bang para separar esquemas do monolito e agilizar a transição para microsserviços.
- 3.É aconselhável priorizar serviços que estão passando por uma baixa frequência de atualizações para separação.



## Capítulo 6 | 6. Implantação| Quiz e teste

1. A implantação de aplicações monolíticas é mais simples do que a implantação de microsserviços devido à sua natureza interconectada.
2. Integração Contínua (CI) e Entrega Contínua (CD) são essenciais apenas para aplicações monolíticas e não têm relevância para microsserviços.
3. Servidores imutáveis evitam a deriva de configuração ao exigir uma nova instância de servidor para cada mudança, em vez de alterar a existente.





Baixe o app Bookey para desfrutar

# Mais de 1000 resumos de livros com quizzes

**Teste grátis disponível!**

Escanear para baixar



## Capítulo 7 | 7. Testes| Quiz e teste

1. Testes são importantes apenas para o desenvolvimento de software com arquiteturas monolíticas, não para sistemas distribuídos como microsserviços.
2. Um escopo de teste maior resulta em feedback mais rápido e diagnósticos de falhas mais fáceis.
3. Contratos dirigidos pelo consumidor garantem que os produtores atendam às expectativas dos consumidores sem a necessidade de testes end-to-end extensos.

## Capítulo 8 | 8. Monitoramento| Quiz e teste

1. Em uma arquitetura de microsserviços, problemas de monitoramento podem surgir apenas de um único ponto de falha.
2. IDs de correlação são úteis para rastrear o fluxo de chamadas de serviço em uma arquitetura de microsserviços.
3. Registros padronizados e coleta de métricas são desnecessários para um monitoramento eficaz em



microserviços.

## **Capítulo 9 | 9. Segurança| Quiz e teste**

- 1.A autenticação é o processo de verificar a identidade de um usuário ou sistema.
- 2.O Single Sign-On (SSO) exige que cada aplicativo gerencie a autenticação de forma independente, sem o uso de credenciais compartilhadas.
- 3.A minimização de dados é uma prática essencial para reduzir os riscos associados a vazamentos de dados.





Baixe o app Bookey para desfrutar

# Mais de 1000 resumos de livros com quizzes

**Teste grátis disponível!**

Escanear para baixar



## Capítulo 10 | 10. A Lei de Conway e o Design de Sistema| Quiz e teste

1. A Lei de Conway afirma que o design de um sistema refletirá a estrutura de comunicação da organização que o cria.
2. Organizações com baixo acoplamento tendem a produzir sistemas menos modulares em comparação com aquelas com alto acoplamento.
3. Cada equipe da RealEstate.com.au é responsável por todo o ciclo de vida do serviço dentro de sua área de negócios.

## Capítulo 11 | 11. Microsserviços em Larga Escala| Quiz e teste

1. Falhas são inevitáveis em sistemas grandes devido a falhas de hardware e à natureza não confiável das redes.
2. De acordo com o teorema CAP, um sistema distribuído pode garantir consistência, disponibilidade e tolerância a partições simultaneamente.
3. A implementação de compartimentos ajuda a garantir que a falha de um serviço não impacte os outros.





## Capítulo 12 | 12. Unindo Tudo| Quiz e teste

1. Os microsserviços devem sempre ser implantáveis de forma independente para garantir atualizações mais suaves.
2. Você deve usar microsserviços mesmo que tenha uma compreensão fraca do domínio.
3. O monitoramento abrangente é importante para diagnosticar problemas em uma arquitetura de microsserviços.





Baixe o app Bookey para desfrutar

# Mais de 1000 resumos de livros com quizzes

**Teste grátis disponível!**

Escanear para baixar

