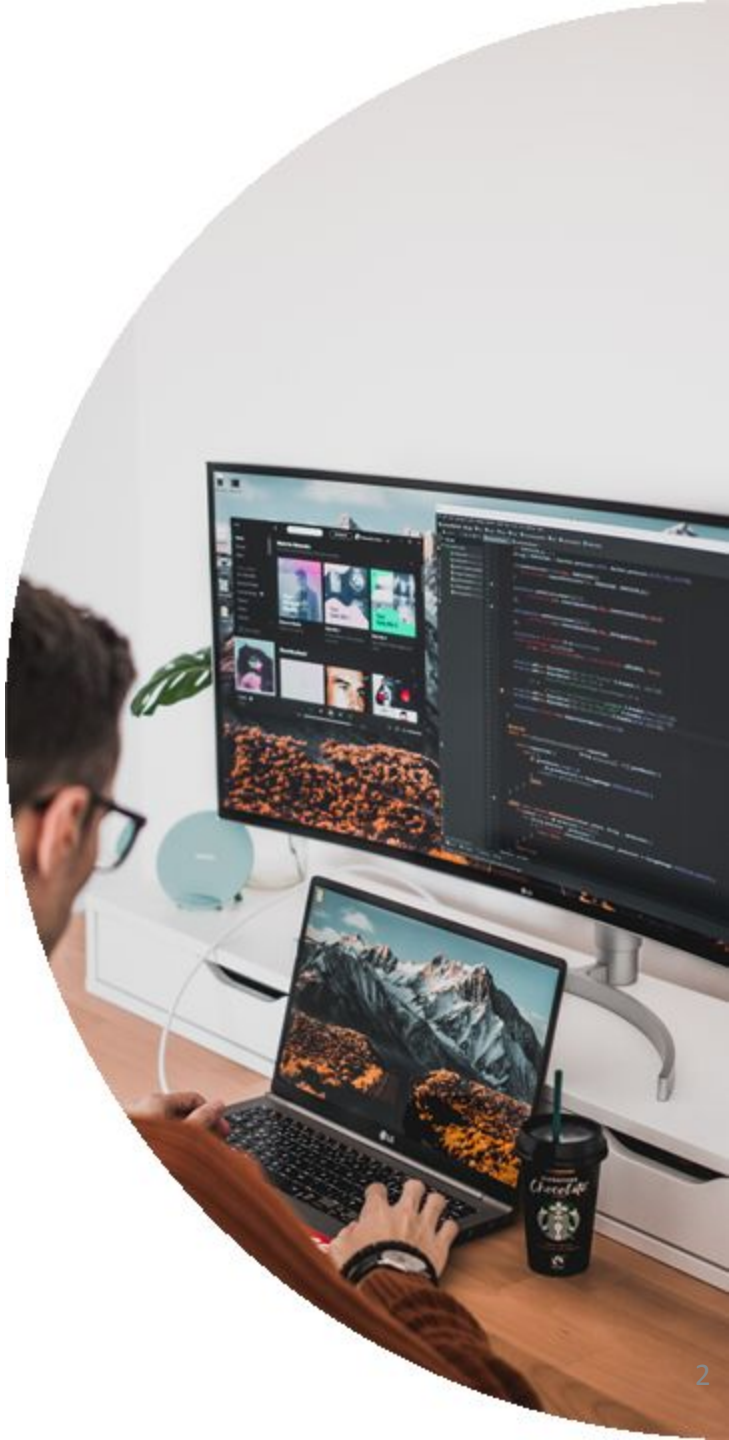
Arquitetura de software

Prof. Eduardo Cizeski Meneghel

Fundamentos para sistemas resilientes



Introdução

1. Conceitos de resiliência de sistemas;
2. Princípios de construção de sistemas tolerantes à falha;
3. Principais falhas e catástrofes;
4. Técnicas para criação de sistemas resilientes.

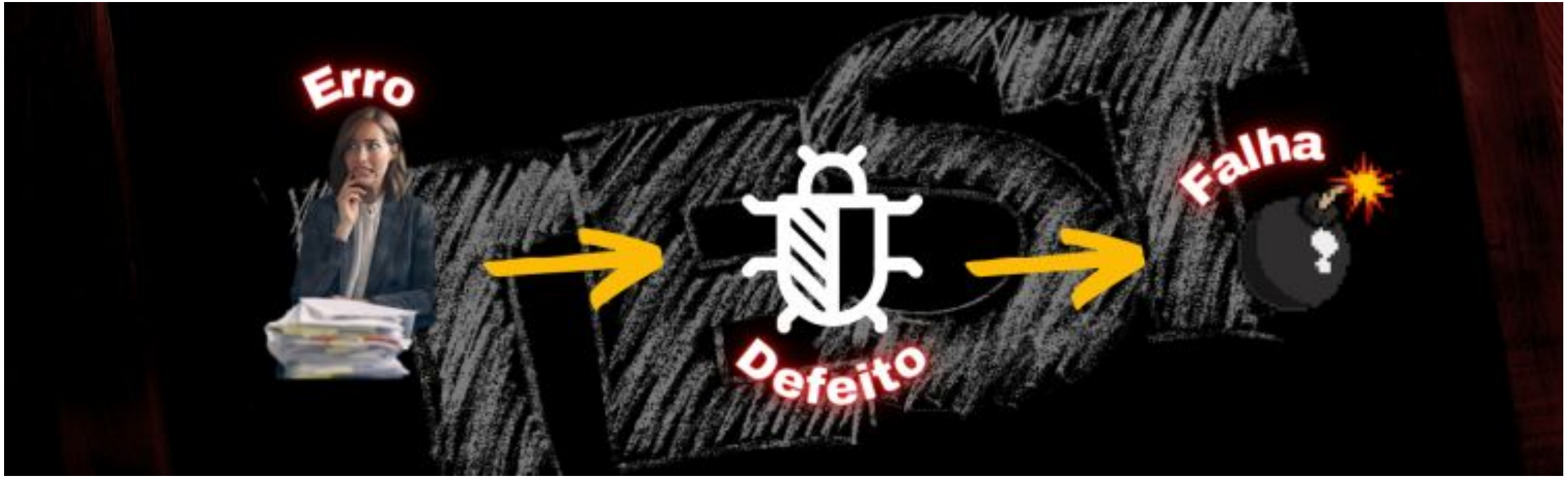
Conceitos

A realidade de construir sistemas de grande escala é que os componentes individuais podem e irão falhar e com frequência.

Conceitos

- Qualquer possível falha tende a acontecer;
- Falhas podem causar efeitos em cadeia.

Conceitos

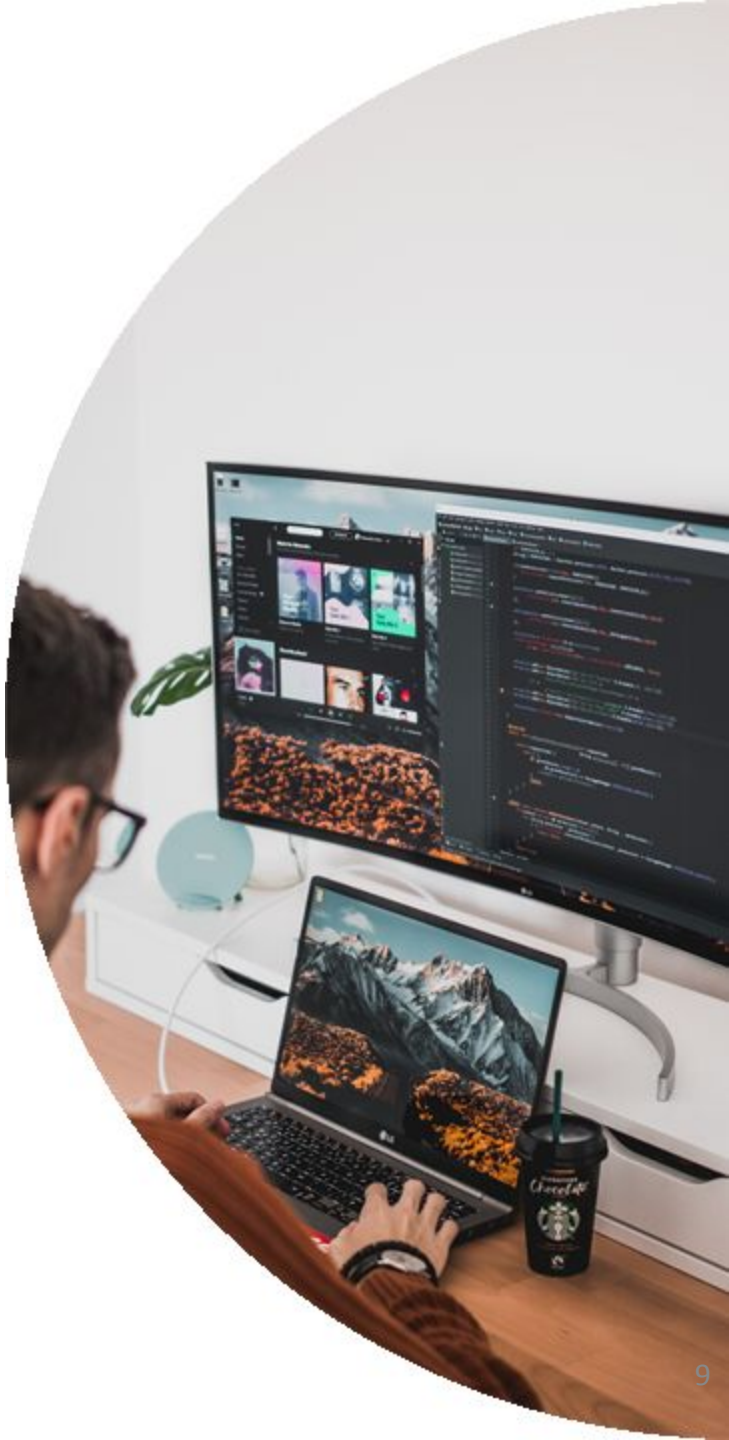


Conceitos

- Defeito: É qualquer imperfeição ou inconsistência no produto do software;
- Erros: Em termos mais técnicos, falamos que o erro é a diferença entre o valor obtido e o valor esperado (é resultante de uma falha);
- Falha: Quando o software apresenta comportamento e respostas diferentes do esperado e especificado em seu desenvolvimento.

Conceitos

- Defeito inconsistente;
- Defeito consistente.



Princípios de construção de sistemas resilientes

Princípios

- Eliminar pontos únicos de falhas;
  - É uma parte do software que pode derrubá-lo em caso de falha;
  - Não deveriam existir.

Princípios

- Eliminar pontos únicos de falhas:
  - Exemplo: Sistema de comunicação / Mensageria;

Princípios

- Eliminar pontos únicos de falhas;
  - Pode ser mitigado por avaliações e revisões arquiteturais ou após a falha ocorrer.

Princípios

- Identificar cenários de falha:
  - Pouco conhecimento da arquitetura do sistema;
  - Defeitos liberados pelo desenvolvimento;
  - Procedimentos instáveis de atualização;

Princípios

- Testes de resiliência:
  - Consistem em identificar os pontos de falha e submeter o sistema de forma ativa, repetida e aleatória;

Princípios

- Teste de carga:
  - Impor uma carga de tráfego sobre um recurso e avaliar o seu comportamento;

Princípios

- Teste de carga:
  - Identificar o quanto de tráfego o sistema suporta;
  - Quais gargalos existem na aplicação;
  - Quanto de recurso é consumido.

Princípios

- Teste de carga:
  - Executar em cada estágio da esteira de desenvolvimento;
  - Executar em ambientes idênticos ao de produção;

Princípios

- Teste do caos:
  - Forçar componentes a falhar em produção.

Princípios

- Teste do caos:
  - Existem ferramentas com pacote de testes de caos como a Simian Army;

Princípios

- Teste do caos:
  - Desativar os endpoints de uma dependência;
  - Introduzir latência aleatoriamente, para simular tráfego de rede;
  - Desligar um servidor aleatoriamente.

Princípios

- Teste de código:
  - Lint;
  - Unidade;
  - Integração;
  - Fim a fim.

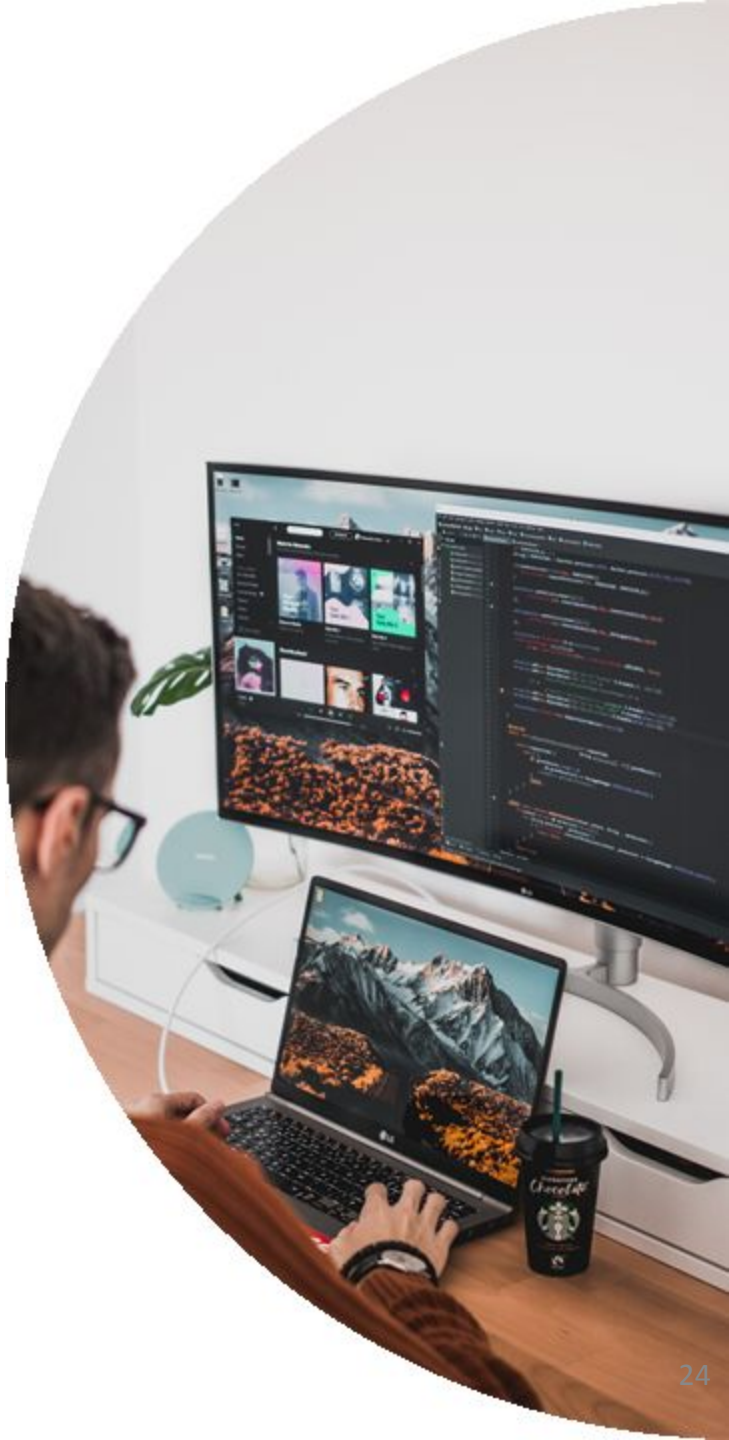
Princípios

- Detecção e mitigação de falhas;
  - O objetivo é reduzir o impacto para os usuários;

Princípios

- Detecção e mitigação de falhas;
  - Retornar para uma versão estável;
  - Comutar para alternativa estável;
  - Haver monitoramento do serviço;

Falhas comuns nos sistemas



Falhas comuns

- Falhas de hardware
  - Falha no datacerter;
  - Falha no provisionamento do serviço.

Falhas comuns

- Falhas de comunicação
  - Falhas de rede;
  - Falha no sistema de troca de mensagens;
  - Falha no balanceamento de carga;

Falhas comuns

- Falhas de dependência
  - Interrupções de serviços externos;
  - Descontinuação de API;
  - Falhas em bibliotecas;

Falhas comuns

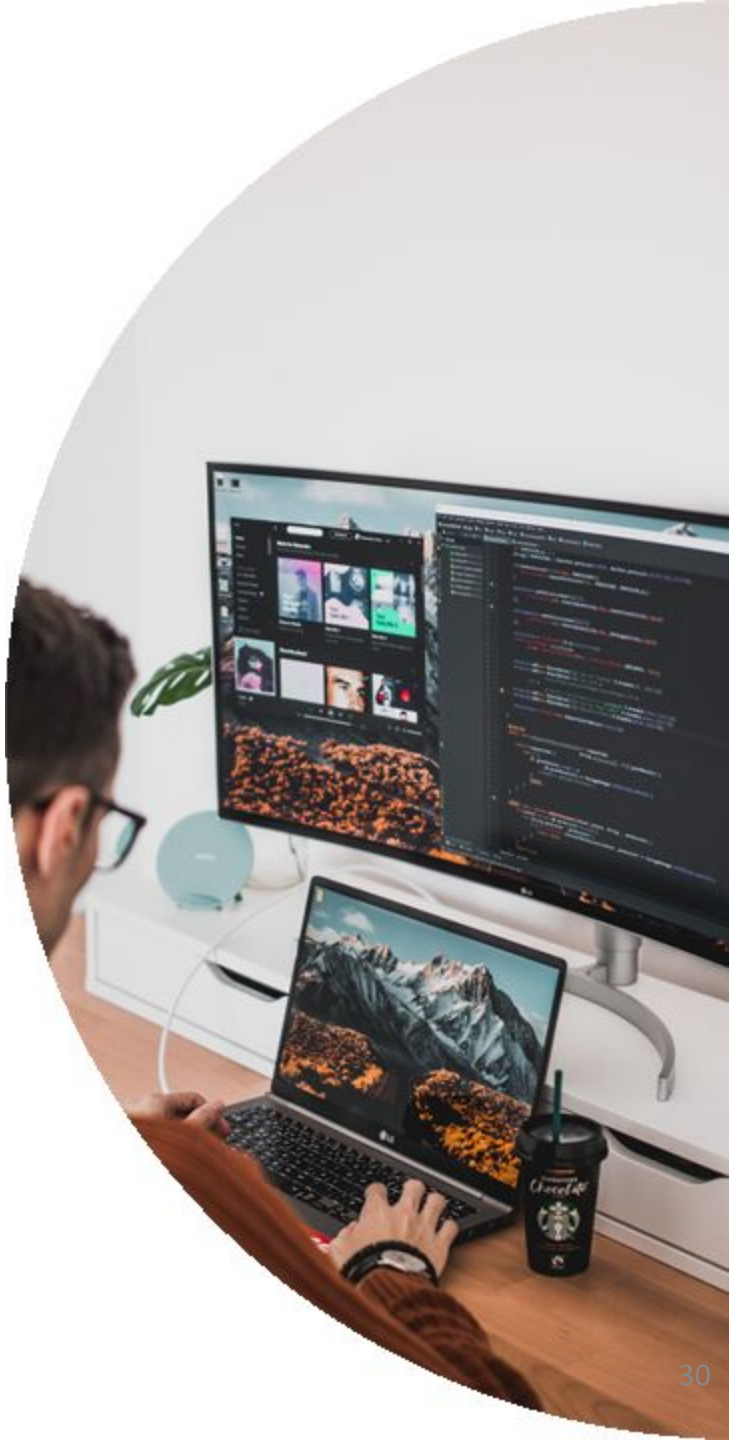
- Falhas internas
  - Falhas no desenvolvimento;
  - Atualizações incorretas;

Princípios

Case Netflix:

https://hnz.com.br/chaos-monkey-e-a-importancia-do-aprendizado-organizacional/

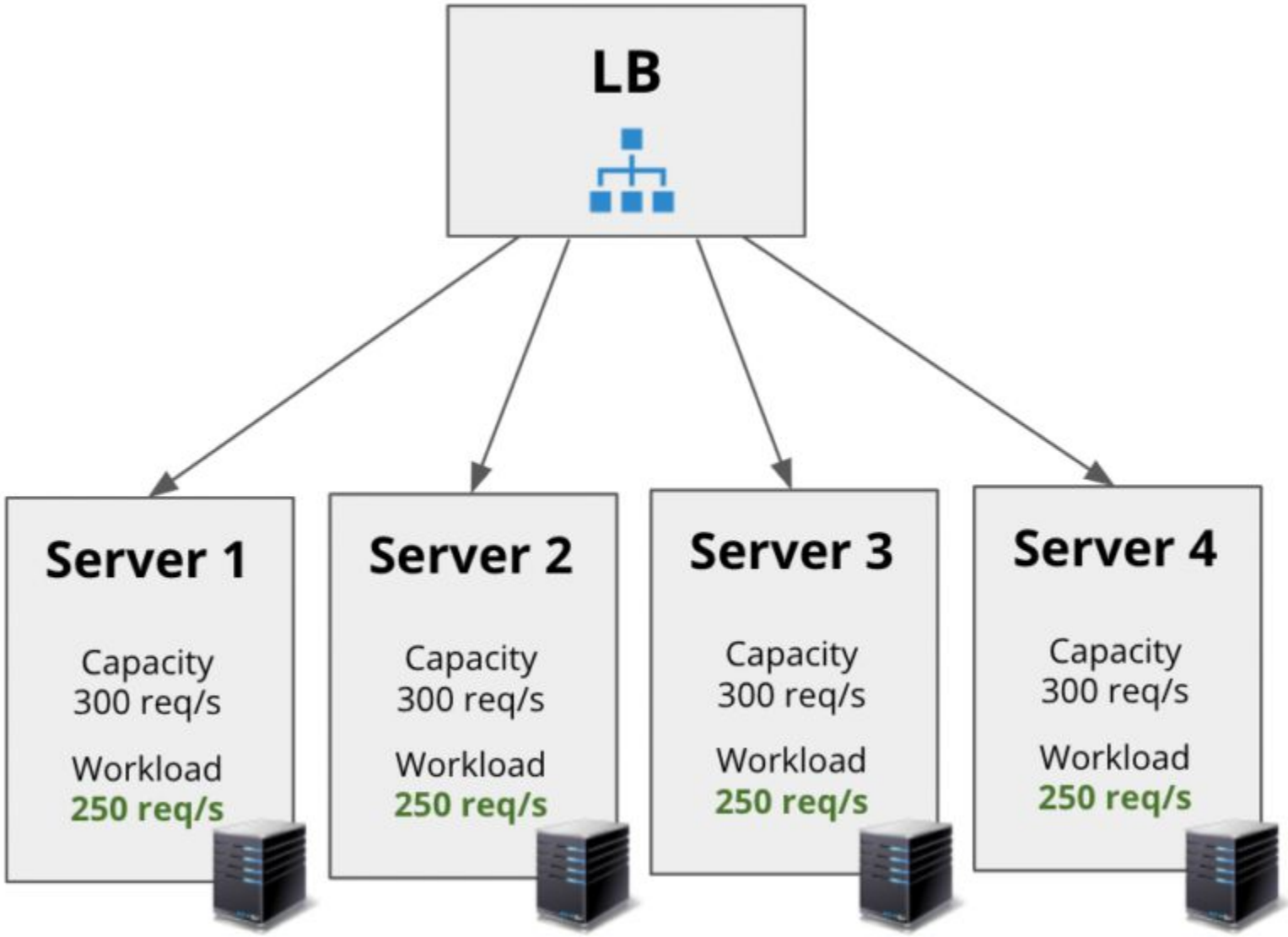
Técnicas para criação de sistemas resilientes



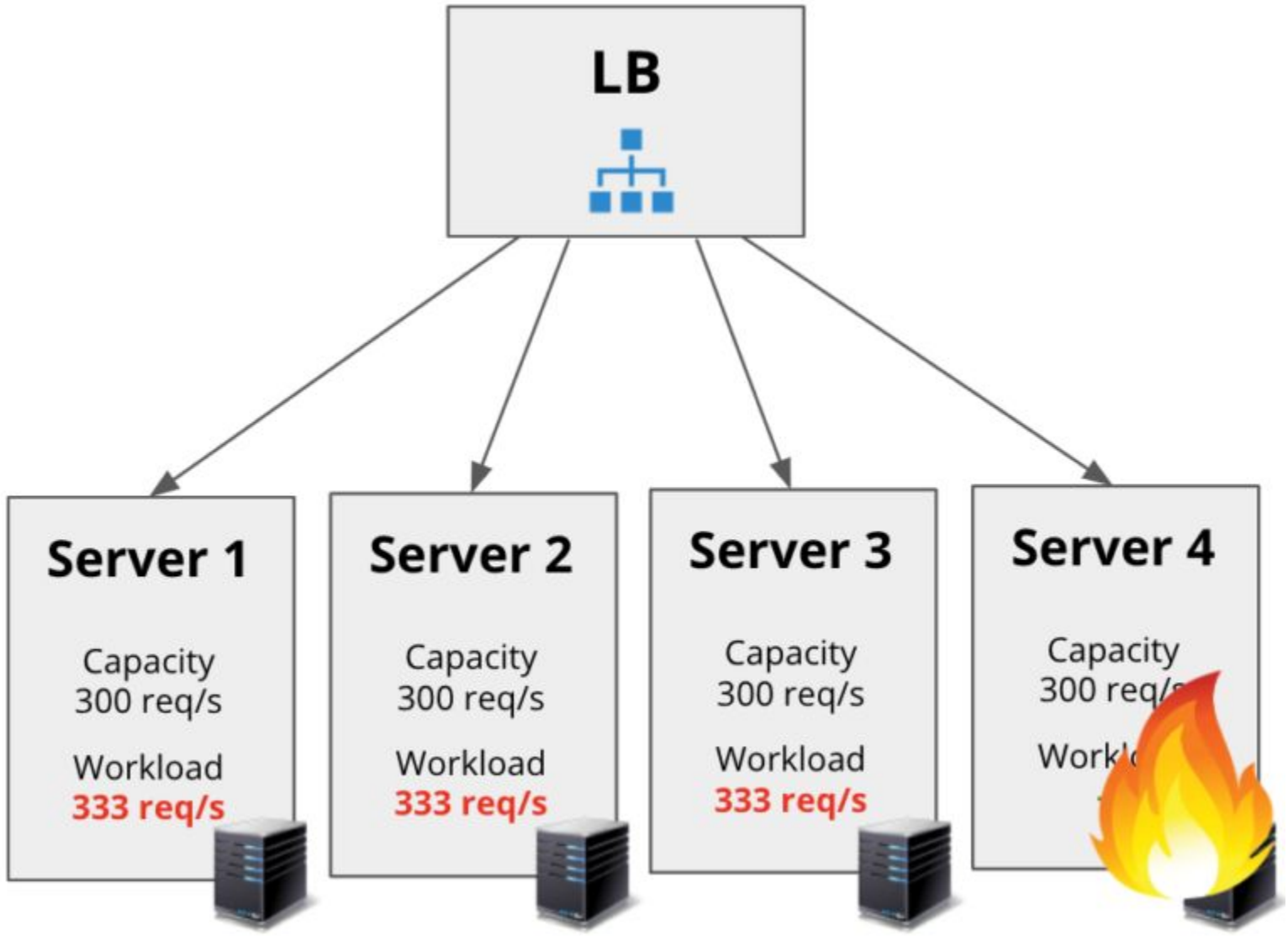
Técnicas

- Redundâncias
  - A redundância deve ser adequada ao funcionamento;
  - Deve ser adequada a quantidade;

Técnicas


LB
Server 1
Capacity
300 req/s
Workload
250 req/s
Server 2
Capacity
300 req/s
Workload
250 req/s
Server 3
Capacity
300 req/s
Workload
250 req/s
Server 4
Capacity
300 req/s
Workload
250 req/s

Técnicas


LB
Server 1
Capacity
300 req/s
Workload
333 req/s
Server 2
Capacity
300 req/s
Workload
333 req/s
Server 3
Capacity
300 req/s
Workload
333 req/s
Server 4
Capacity
Workl

Técnicas

- Redundâncias
  - Não há resiliência sem redundância. Entretanto, esta é apenas a primeira medida e, provavelmente, a mais cara.

Técnicas

- Fallback
  - Fallback é uma alternativa de substituição para uma determinada unidade de mitigação, capaz de cumprir sua premissa funcional enquanto ela estiver indisponível ou inviável.

Técnicas

- Interfaces administrativas:
  - As verificações podem incluir as principais dependências dos componentes, como bancos de dados, serviços remotos, etc.

Técnicas

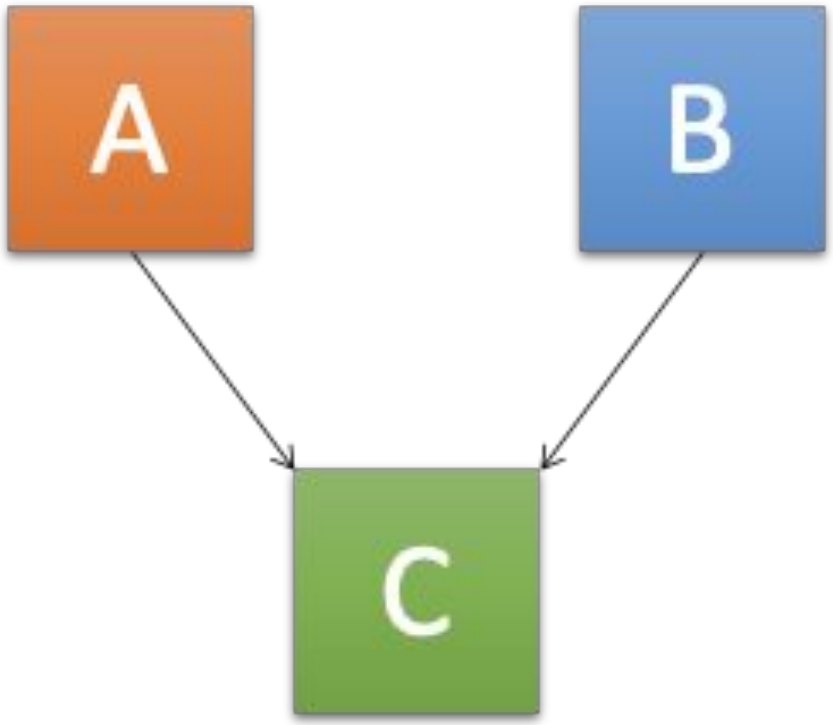
- Watchdogs
  - Projetar agentes e critérios de decisão e escalation;
  - Watchdogs atuam ativamente, muitas vezes acionando health checks, para, sob determinadas circunstâncias, disparar algum tipo de ação.

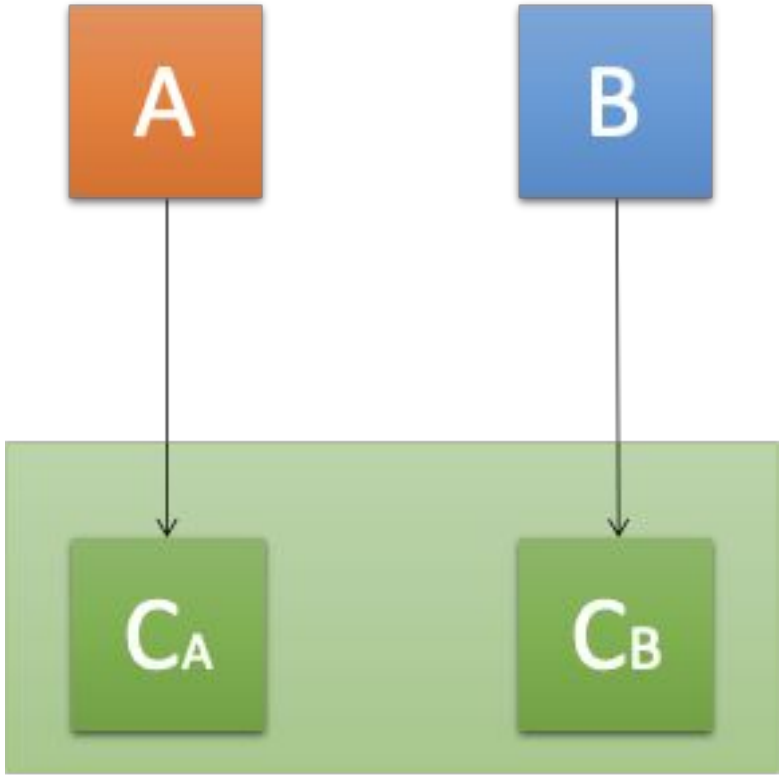
Técnicas

- Comunicação assíncrona:
  - A ideia é substituir chamadas a componentes potencialmente instáveis por mecanismos de mensageria comprovadamente sólidos e estáveis.

Técnicas

- Bulkheads





Técnicas

- Determinar Timeouts:
  - Componentes que demoram a responder são mais nocivos que componentes que não respondem.

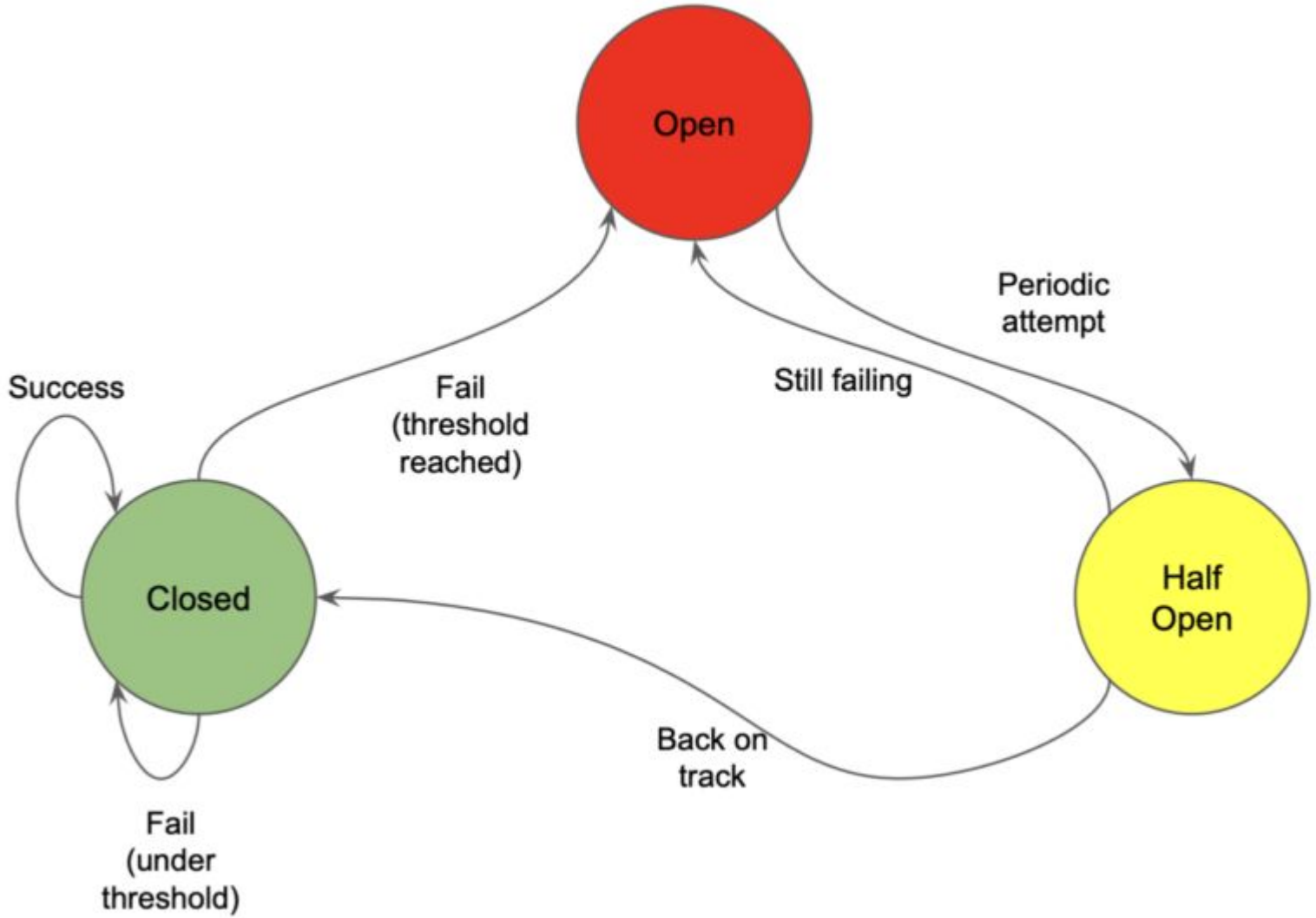
Técnicas

- Transaction Outbox
  - Adicionar uma tabela adicional, no banco de dados onde as transações estão sendo processadas, para “registrar” a demanda do envio de uma mensagem.

Técnicas

- Circuit breakers
  - O objetivo de um Circuit breaker é proteger o “servidor” de requisições enquanto este estiver enfrentando dificuldades (potencial saturação).

Técnicas


Open
Periodic attempt
Still failing
Success
Fail (threshold reached)
Closed
Half Open
Back on track
Fail (under threshold)

Encerramento

“A esperança não é uma estratégia…”

Encerramento

- Microsserviços prontos para a produção, Susan J. Fowler;
- Manual do arquiteto de software, Elemar Júnior.