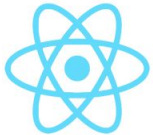
Microserviços: como **não** dar um tiro no próprio pé

Rodrigo Nonose

EmCasa



React

elixir

Qualquer organização que projeta um sistema irá produzir um design onde a estrutura é a cópia da estrutura comunicacional da organização

Resumidamente, o estilo ou padrão arquitetural de microserviços é uma abordagem para desenvolver uma única aplicação como uma suíte de serviços pequenos, cada um rodando em seu próprio processo e comunicando por mecanismos leves, geralmente uma API HTTP. Esses serviços são construídos em torno de capacidades de negócio e independentemente implantáveis por mecanismos completamente automatizados. Existe uma mínima gerência centralizada desses serviços, que podem ser escritas em linguagens de programação diferentes e utilizar tecnologias de armazenamento diferentes

Problemas que propõe resolver

Baixa produtividade

Baixa produtividade

- Ramp up
- Fragilidade
- Testabilidade

Limite de escalabilidade vertical

Limite de escalabilidade vertical

- Limite de single cores
- Memória não escala linearmente

Alta duração do ciclo de vida

Alta duração do ciclo de vida

- O tempo que uma modificação demora pra estar em produção
- Ciclos altos de entrega
- Demora de feedback (retrabalho)

Dívida técnica

Alta duração do ciclo de vida

- Ferramentas antigas
- Versões antigas de ferramentas
- Dificuldade de se corrigir bugs antigos

Benefícios

Interface modular bem definidas

Interface modular bem definidas

- Módulos definidos por modelo de negócio
- Fácil de testar/integrar/comunicar entre módulos diferentes
- Ramp-up rápido

Deploy independente

Deploy independente

- Escalado independentemente
- Facilita técnicas como teste A/B, canary release, manter versões múltiplas de serviços etc

Diversidade Tecnológica (?)

Diversidade Tecnológica (?)

- Escolha a ferramenta certa para o trabalho certo
- Faca de dois gumes: se abusado, pode virar um caos

Custos

Distribuição

Falácias de computação distribuída

- A rede é confiável
- Latência é zero
- Banda é infinita
- A rede é segura
- Topologia não muda
- Existe um administrador
- Custo de transporte é zero
- A rede é homogênea

Complexidade Operacional

Complexidade Operacional

- DevOps
- Balanceador de carga
- Versionamento de APIs
- Stacks diversas

Consistência Eventual

Concistência Eventual

- **Síncrono x Assíncrono**
- **Mensagerias**
- **Transações distribuídas**

Devo utilizar microserviços?

Nope

NO


No
No
No
No
No
No
No

Obrigado

rhnonose

RodrigoNonose

Requisitos

- Provisionamento rápido
- Monitoramento básico, logging, instrumentação, *health-check*
- Desenvolvimento rápido de aplicação
- Governança decentralizada
- Alto grau de maturidade técnica e organizacional

Esforços

- Separação de lógica: módulos auto-contidos
- Distriuição de responsabilidades
- Automatização de infraestrutura
- *DevOps* como cultura
- *Cookie-cutter scaling*
- *Sharding*
- Integração/Deploy contínuo
- Testes automatizados

Microserviços como ponto de partida

Microserviços como objetivo

Obrigado

rhnonose

RodrigoNonose

Links

https://emcasa.com
https://github.com/em-casa/
TG microserviços
Building Microservices - Sam Newman
Microservices resource guide
Video spotify
Fallacies of Distributed computing