

**INSTITUTO
FEDERAL**

Paraíba

Campus
Cajazeiras

PROGRAMAÇÃO P/ WEB 2

INTRODUÇÃO

PROF. DIEGO PESSOA

✉ DIEGO.PESSOA@IFPB.EDU.BR

🐱 [@DIEGOEP](https://github.com/DIEGOEP)



CST em Análise e
Desenvolvimento de
Sistemas

APRESENTAÇÃO DA DISCIPLINA

EMENTA

- ▶ Arquitetura, ciclo de vida, conceitos e ferramentas para a construção de Aplicações Web avançadas.
- ▶ Novos padrões arquiteturais e paradigmas de desenvolvimento.
- ▶ Tópicos avançados e tendências.

OBJETIVO GERAL

- ▶ Exposição prática a **conceitos, ferramentas e princípios** do desenvolvimento de **aplicações Web** baseadas na arquitetura de **microsserviços**, juntamente com as **boas práticas e técnicas** de implantação utilizando-se dos princípios de **DevOps**.

OBJETIVOS ESPECÍFICOS

- ▶ Tornar o aluno capacitado a entender os fundamentos, desenvolver e gerenciar uma **aplicação Web avançada** baseada em **microserviços**.
- ▶ Aplicar e gerenciar os **principais frameworks** utilizados no desenvolvimento de aplicações Web.
- ▶ Apresentar todo o **fluxo de desenvolvimento** de uma aplicação Web avançada, da **concepção, desenvolvimento, configuração, implantação e disponibilização** para o usuário final.

PLÁGIO & CÓDIGO DE HONRA

- ▶ Não trabalhe em exercícios que não os seus
- ▶ Não copie e cole partes do seu exercício de fontes de terceiros
- ▶ Não copie código de terceiros
- ▶ Cite as referências





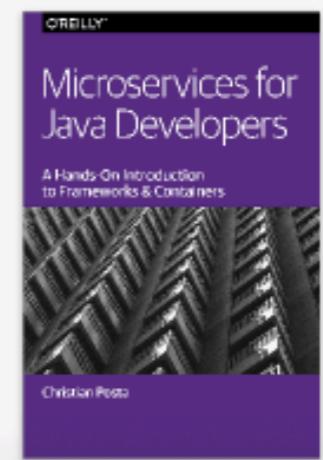
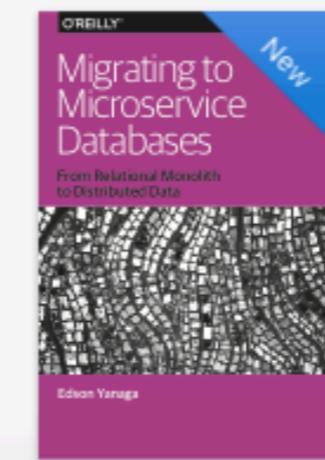
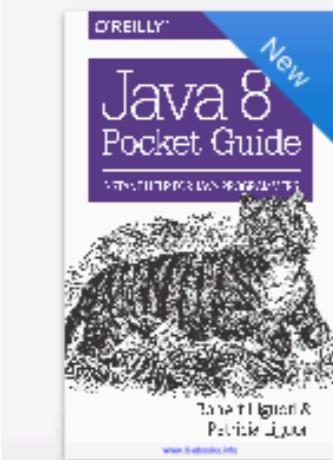
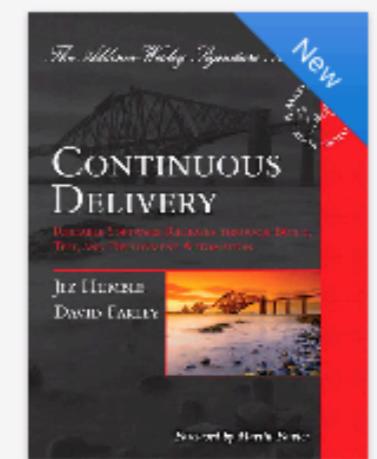
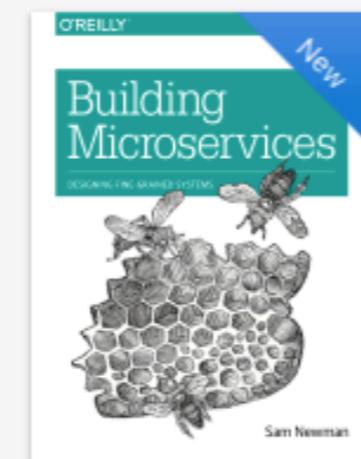
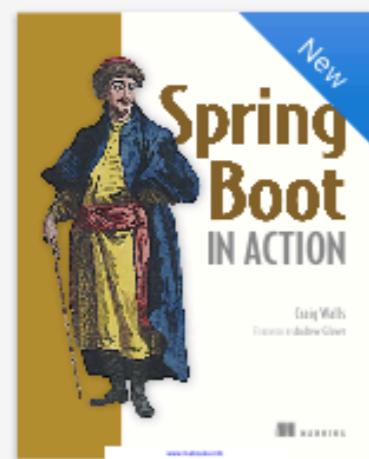
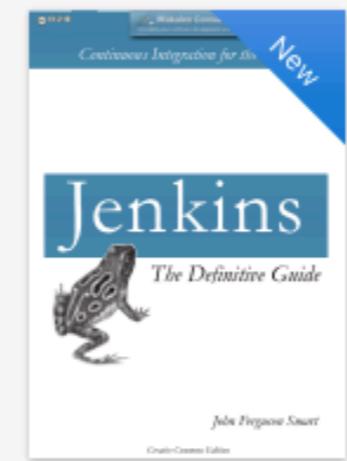
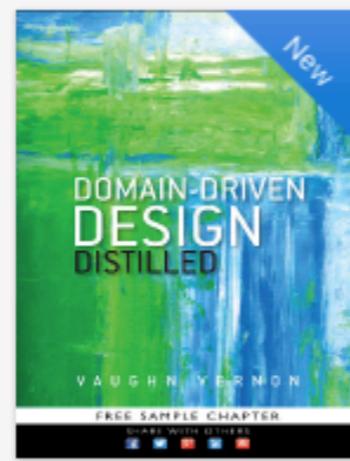
AVALIAÇÃO

- ▶ **Homeworks** semanais individuais de acordo com tópicos de aula
- ▶ **Projeto** incluindo a concepção, desenvolvimento, implantação e documentação de aplicação Web baseada em microsserviços desenvolvida ao longo do semestre
- ▶ Grupos de no máximo duas pessoas

TÓPICOS PRINCIPAIS

- ▶ *1. Revisão e aprofundamento de conceitos fundamentais*
- ▶ *2. Migrando de aplicações monolíticas para microserviços*
- ▶ *3. Desenvolvendo lógica de negócio*
- ▶ *4. Comunicação entre microserviços*
- ▶ *5. Gerenciamento de Consultas*
- ▶ *6. Padrões para consumo como APIs externas*
- ▶ *7. Testes em microserviços*
- ▶ *8. Desenvolvendo serviços prontos para produção*
- ▶ *9. Implantação de microserviços*
- ▶ *10. Tópicos avançados e tendências*

BIBLIOGRAFIA



AMBIENTE DO CURSO

- ▶ Página principal: <https://github.com/ifpb/pweb2>
 - ▶ Agenda, *Lectures, homeworks, source code*
- ▶ Canal de comunicação: [slack](#) (#ads-pweb2)
- ▶ Compartilhamento de material complementar, discussões, dúvidas

DICAS

- ▶ E-mail @academico.ifpb.edu.br
- ▶ Github student pack: <https://education.github.com/pack>
- ▶ Licença IntelliJ: <http://intellij-support.jetbrains.com>

VISÃO GERAL

Baseado em:

<https://github.com/vinicius3w/i>

https://resources.sei.cmu.edu/asset_files/Presentation/2017_017_001_497911.pdf



Dilbert by Scott Adams, 2006

DESENVOLVIMENTO DE APLICAÇÃO WEB MODERNA

PROCESSO: INTEGRAÇÃO/DEPLOYMENT CONTÍNUO



ORGANIZAÇÃO: PEQUENA, TIMES AUTÔNOMOS

ARQUITETURA: ??????????????

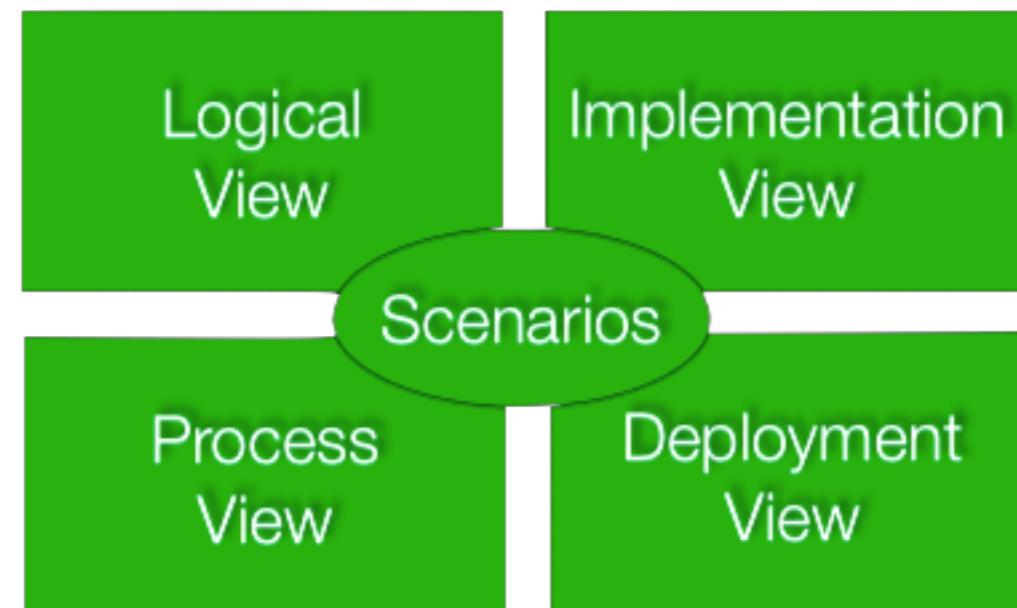
MAS O QUE É UMA ARQUITETURA DE SOFTWARE?

""THE SOFTWARE ARCHITECTURE OF A COMPUTING SYSTEM IS THE SET OF STRUCTURES NEEDED TO REASON ABOUT THE SYSTEM, WHICH COMPRIZE SOFTWARE ELEMENTS, RELATIONS AMONG THEM, AND PROPERTIES OF BOTH."""

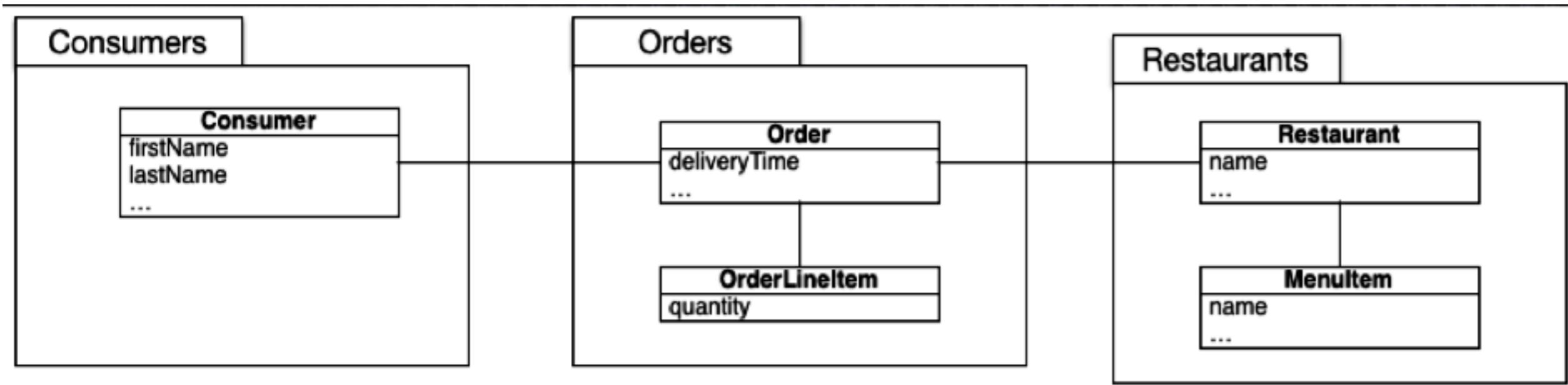
Documenting Software Architectures, Bass et al

EM OUTRAS PALAVRAS...

- ▶ Arquitetura de Software = (elementos, relações, propriedades)
- ▶ São multi-dimensionais (descritas por muitas visões)
- ▶ Visão = Arquitetura aplicada num contexto específico

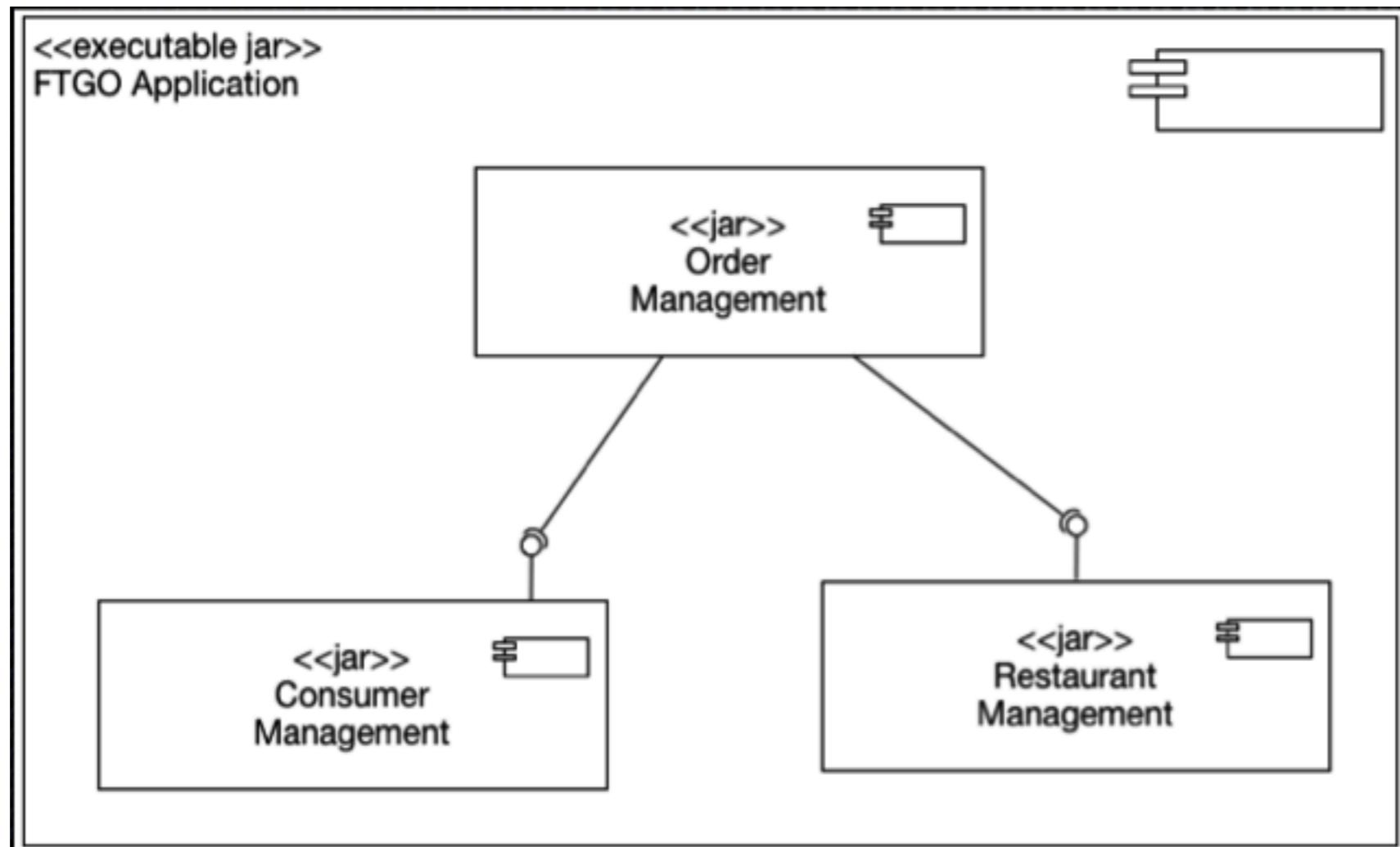


VISÃO LÓGICA



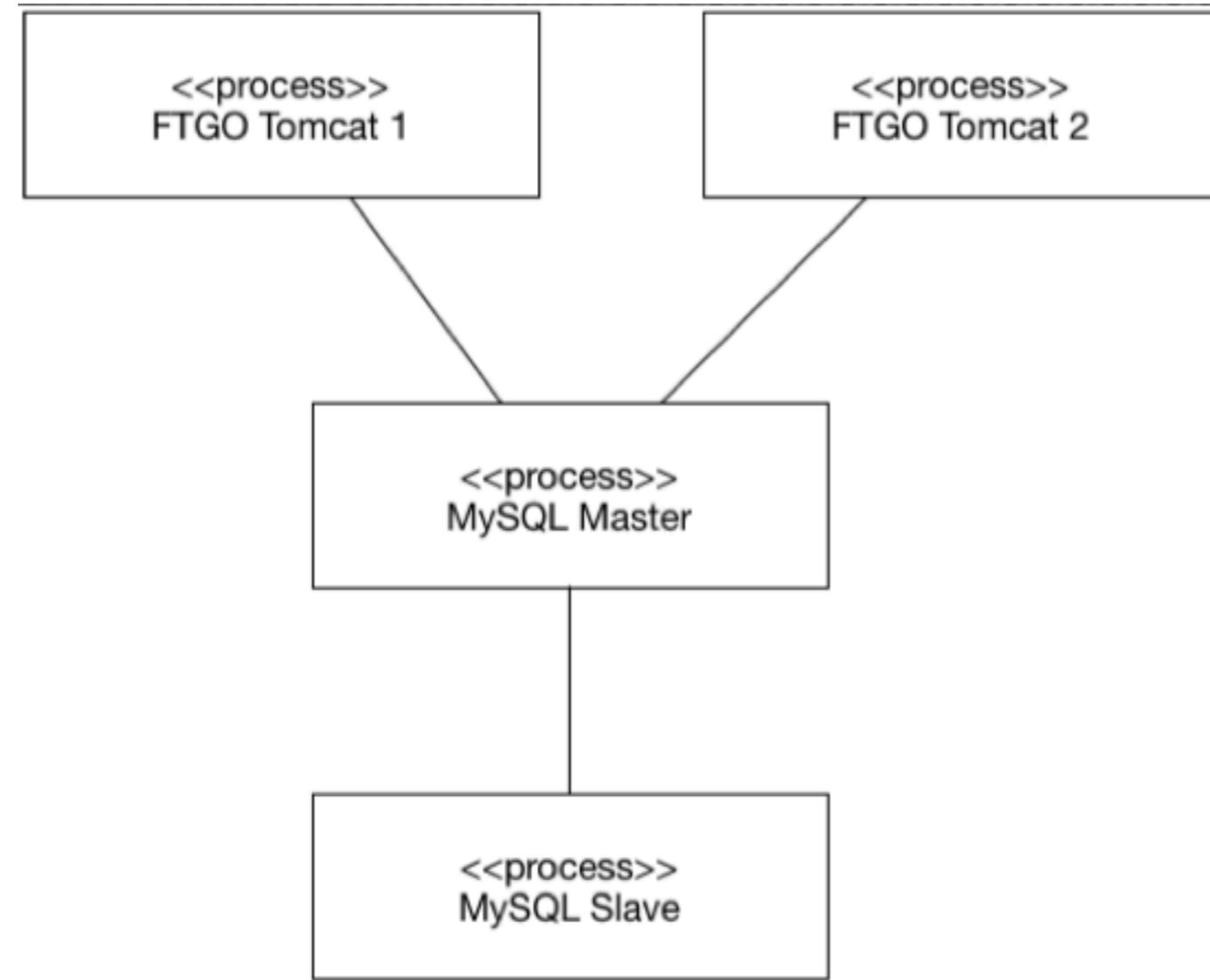
- ▶ Elementos: classes e pacotes
- ▶ Relações: herança e associações

VISÃO DE IMPLEMENTAÇÃO



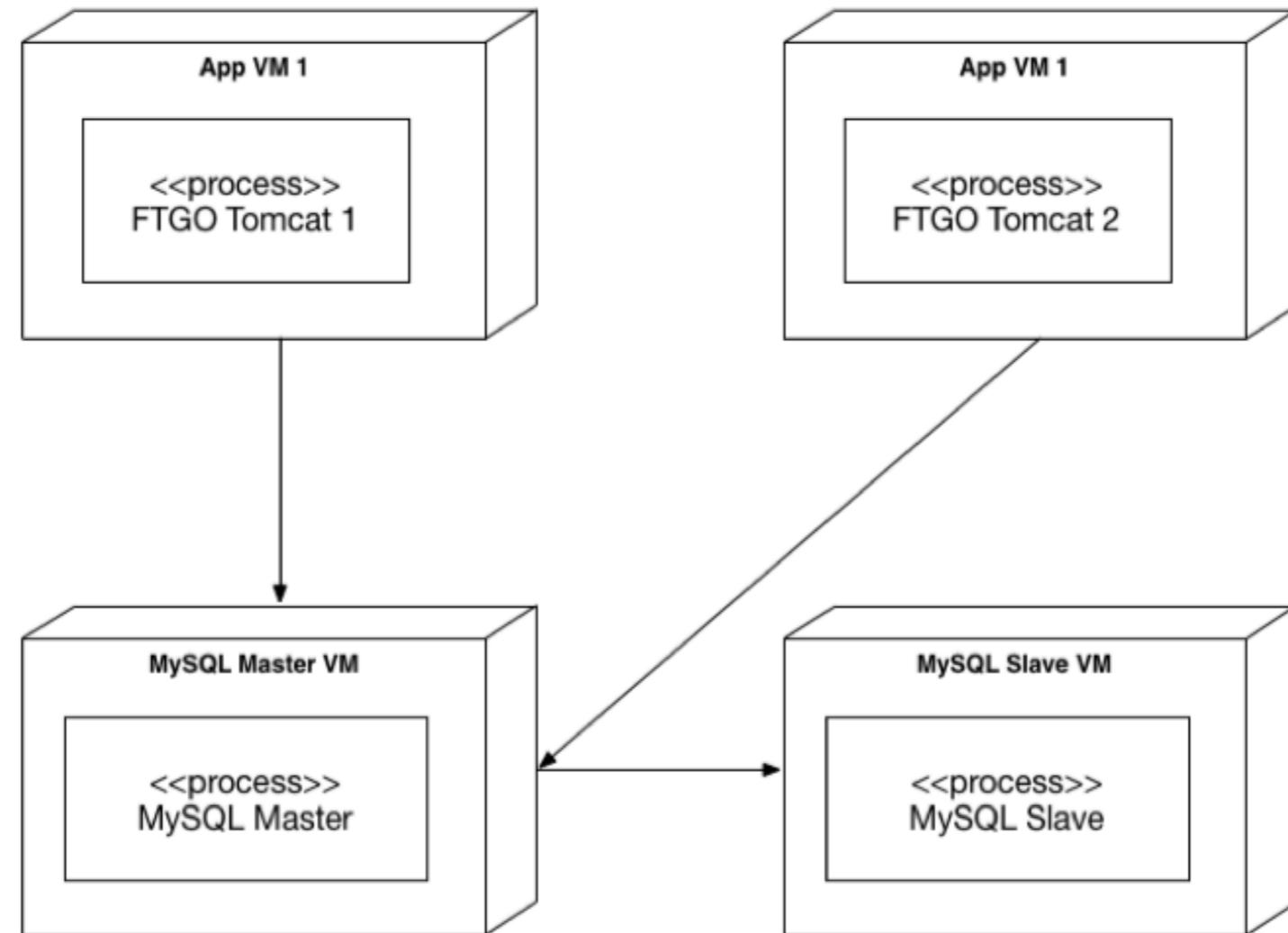
- ▶ Elementos: módulos e componentes
- ▶ Relações: dependências

VISÃO DE PROCESSO



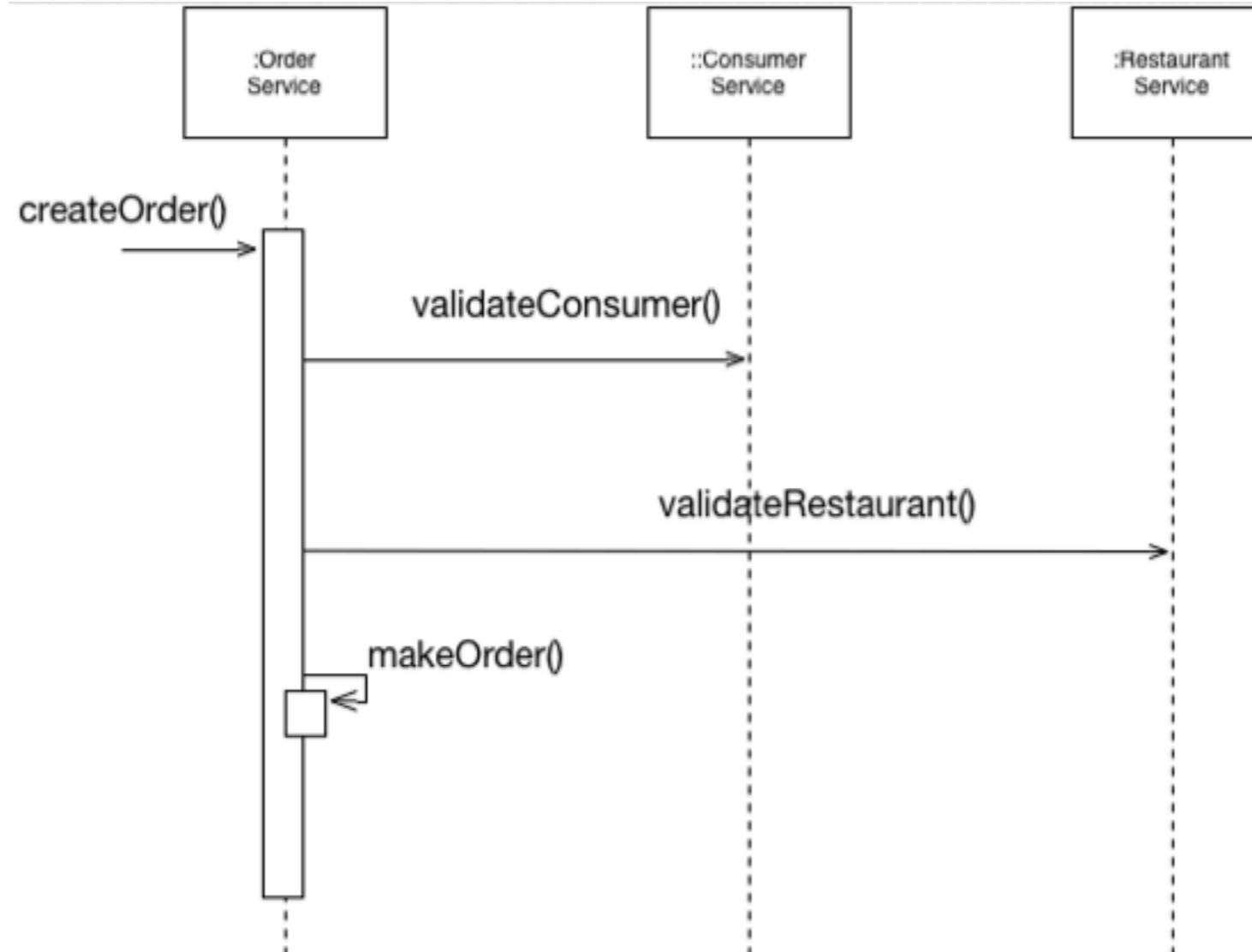
- ▶ Elementos: processos
- ▶ Relações: comunicação inter-processo (IPC)

VISÃO DE DEPLOYMENT



- ▶ Elementos: "máquinas"
- ▶ Relações: rede

4+1 CENÁRIOS



- ▶ Derivado dos casos de uso
- ▶ Integra as visões

MAS ENTÃO... QUAL ESTILO ARQUITETURAL SEGUIR?

"" ... AN ARCHITECTURAL STYLE DETERMINES THE VOCABULARY OF COMPONENTS AND CONNECTORS THAT CAN BE USED IN INSTANCES OF THAT STYLE, TOGETHER WITH A SET OF CONSTRAINTS ON HOW THEY CAN BE COMBINED.... ""

David Garlan and Mary Shaw, An Introduction to Software Architecture

O PAPEL DA ARQUITETURA



"ADES"

Depende das...

- ▶ Manutenibilidade, Evolabilidade, Testabilidade,
Implantabilidade
- ▶ Escalabilidade, Segurança, Confiabilidade

AFETA VELOCIDADE
DE IMPLANTAÇÃO

NOS DIAS DE HOJE...



NEGÓCIOS PRECISAM
INOVAR MAIS RÁPIDO!

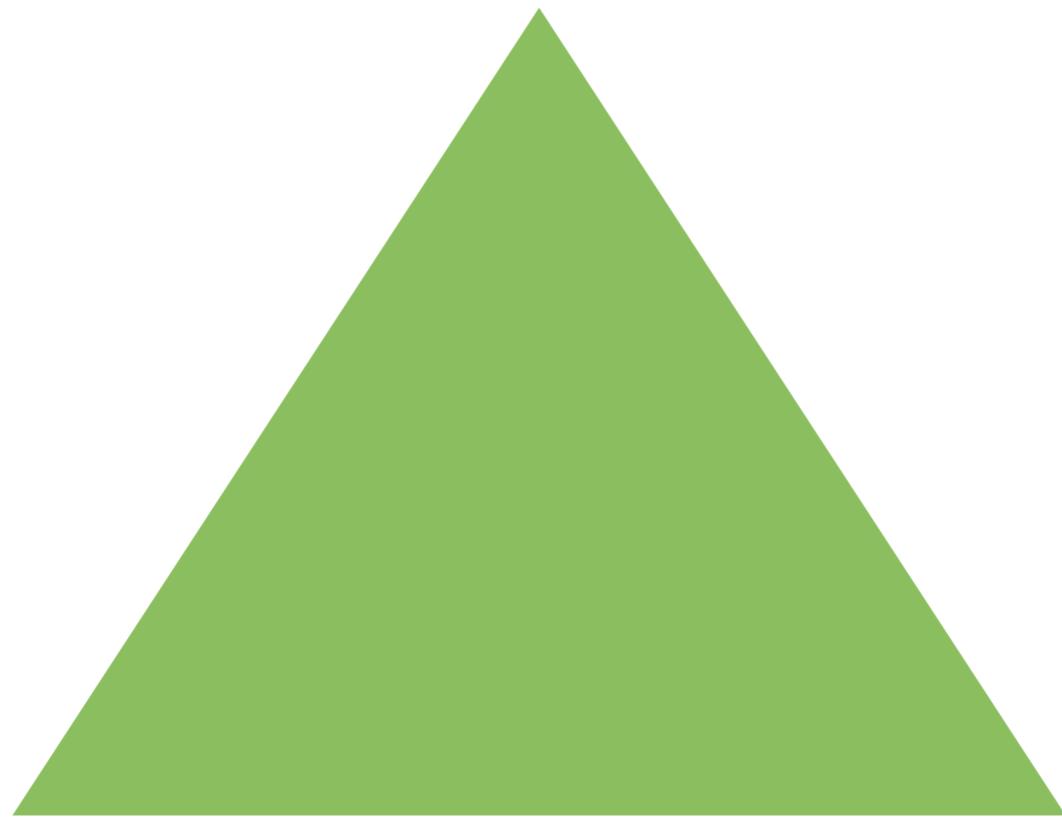
=
SOFTWARES PRECISAM
SER CONSTRUÍDOS
MAIS RÁPIDO!

REDUÇÃO DO TEMPO DE
ENTREGA

AUMENTO DA FREQUÊNCIA
DE IMPLANTAÇÃO

VOLTANDO... DESENVOLVIMENTO DE APLICAÇÃO WEB MODERNA

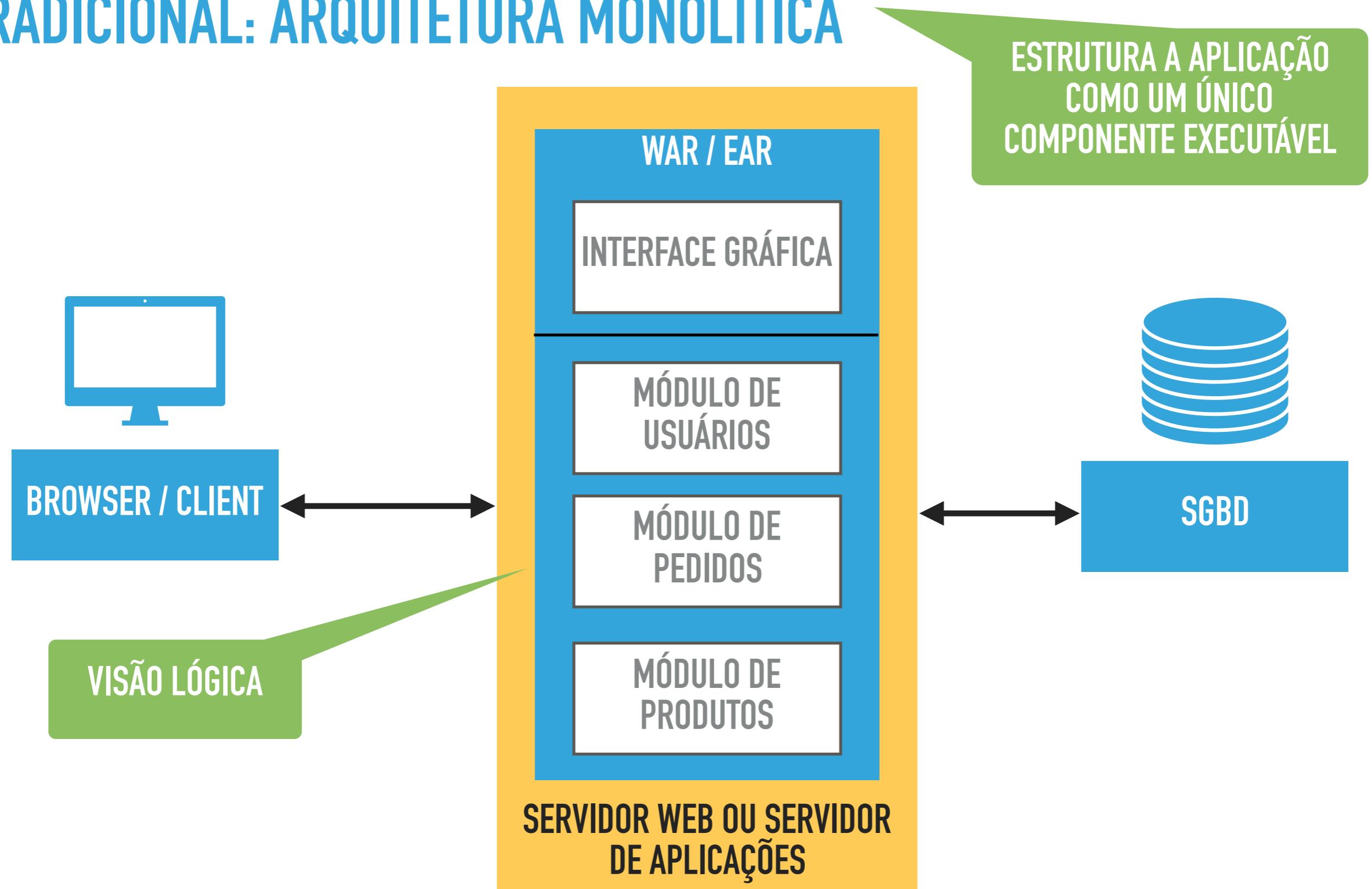
PROCESSO: INTEGRAÇÃO/DEPLOYMENT CONTÍNUO



ORGANIZAÇÃO: PEQUENA, TIMES AUTÔNOMOS

ARQUITETURA: ??????????????

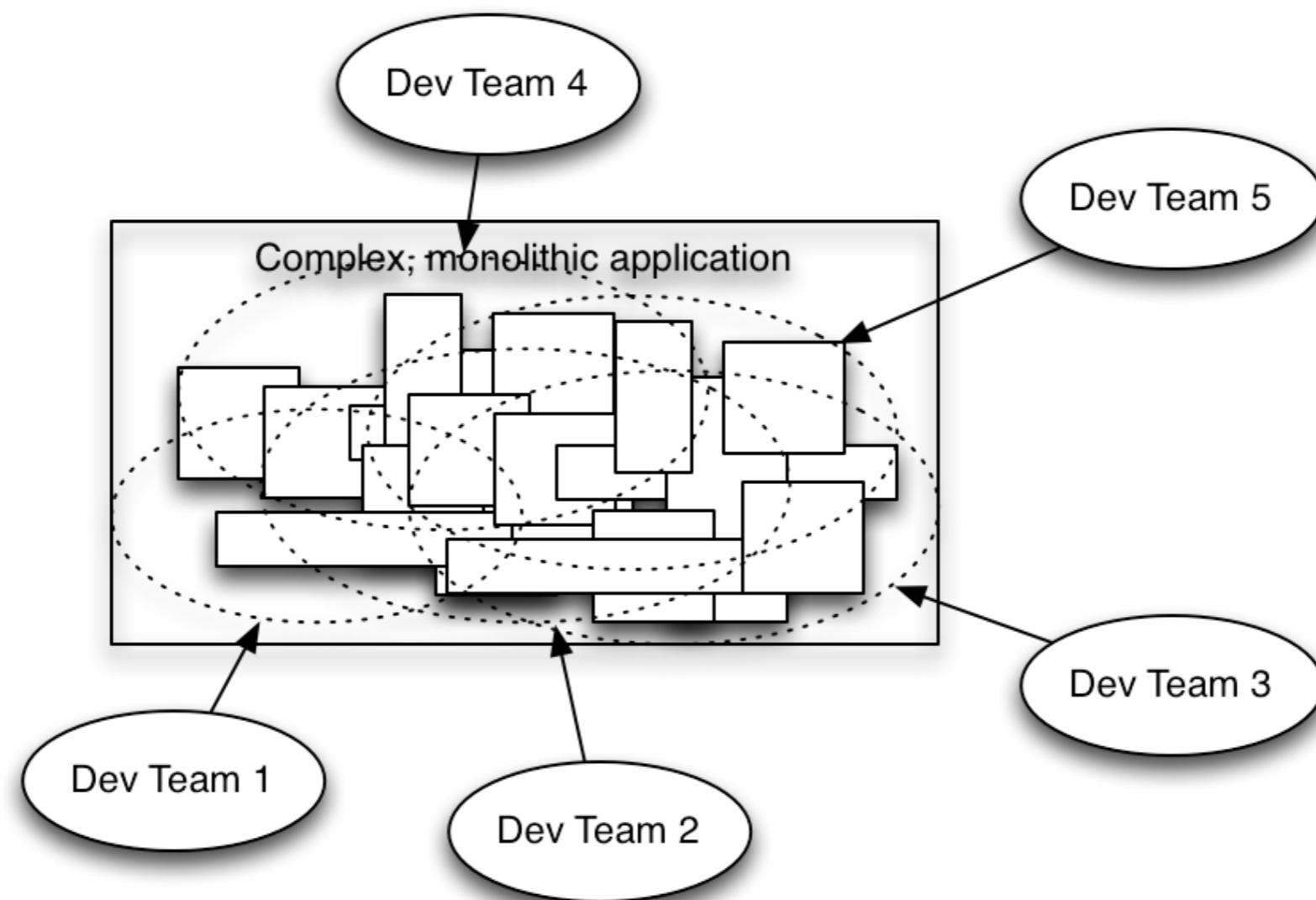
TRADICIONAL: ARQUITETURA MONOLÍTICA



APLICAÇÃO MONOLÍTICA DE SUCESSO... [SERÁ?]



APLICAÇÃO SEGUE CRESCENDO, CRESCENDO, CRESCENDO...

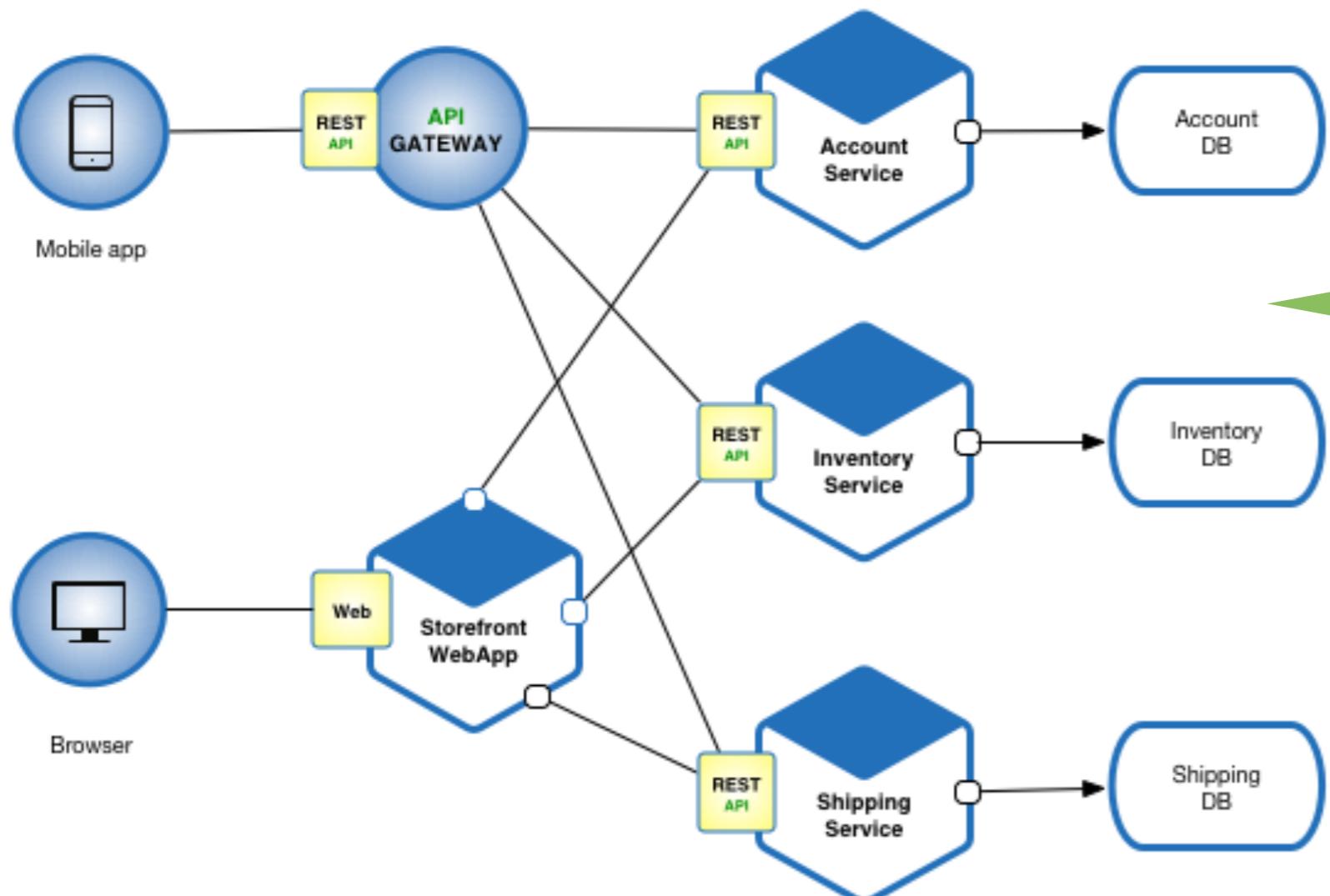


BEM-VINDO AO MONOLITHIC HELL

- ▶ Desenvolvimento e implantação ágil se torna impossível
- ▶ Tecnologias usadas começam a ficar obsoletas, mas...
- ▶ Refatorar não é mais viável!



ALTERNATIVA: MICROSERVIÇOS



microservices.io

ESTILO ARQUITETURAL
QUE ESTRUTURA UMA
APLICAÇÃO COMO UM
CONJUNTO DE SERVIÇOS
FRACAMENTE
ACOPLADOS,
ORGANIZADOS A PARTIR
DA LÓGICA DE NEGÓCIO

MICROSERVIÇOS - FUNCIONAMENTO

- ▶ Serviços se comunicam entre si através de protocolos síncronos como HTTP/REST ou protocolos assíncronos como AMQP.
- ▶ Serviços podem ser desenvolvidos de maneira independente um do outro.
- ▶ Cada serviço possui seu próprio banco de dados, evitando o acoplamento entre serviços
- ▶ Consistência de dados é mantida através do uso do padrão Saga.

MICROSERVIÇOS - BENEFÍCIOS

- ▶ Permite a integração e implantação contínua de aplicações grandes e complexas
- ▶ Melhor estabilidade - serviços menores são mais fáceis de testar
- ▶ Melhor "implantabilidade" - serviços podem ser implantados de maneira independente
- ▶ Permite a organização do desenvolvimento em múltiplos times com responsabilidades específicas e independentes
- ▶ Cada microserviço é relativamente pequeno - fácil de entender e mais rápido de manipular na IDE
- ▶ A aplicação inicia mais rápido, o que diminui o tempo de implantação e torna o desenvolvimento mais produtivo
- ▶ Melhor isolamento de falha: se há vazamento de memória em um serviço, só ele será afetado. Os outros continuarão a responder as requisições normalmente.
- ▶ Elimina o comprometimento a longo prazo com um stack de tecnologias. Quando um novo serviço é desenvolvido, ele pode utilizar novas tecnologias. Reescrever um serviço existente também torna-se viável.

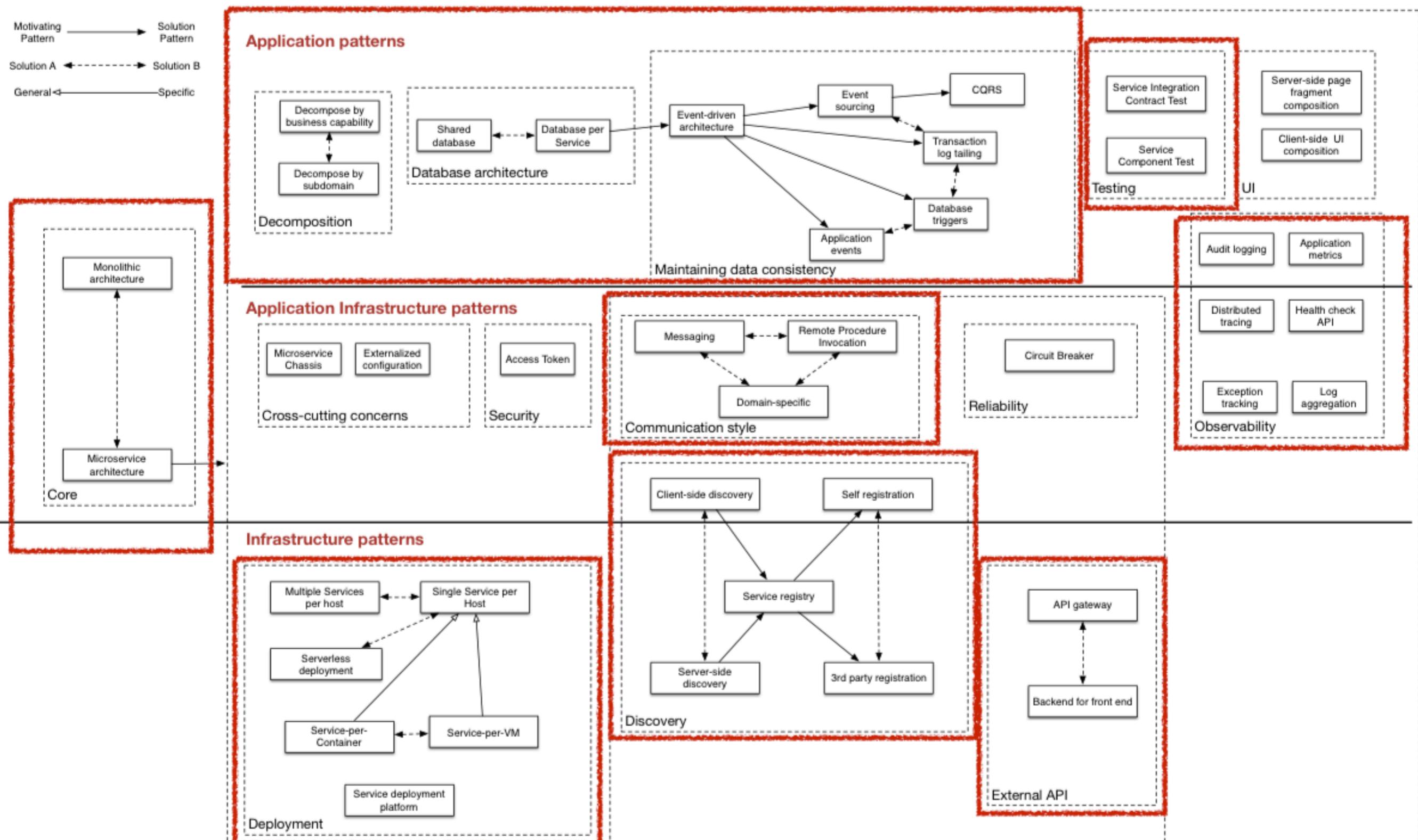
MICROSERVIÇOS - DESVANTAGENS

- ▶ Desenvolvedores precisam lidar com a complexidade adicional de criar um sistema distribuído
- ▶ Ferramentas/IDEs são orientadas a construir aplicações monolíticas e não proveem suporte explícito para aplicações distribuídas
- ▶ Testes integrados são mais difíceis de executar (são muitos serviços/componentes)
- ▶ Desenvolvedores precisam implementar um mecanismo de comunicação inter-serviços
- ▶ Implementar casos de uso que contenham com múltiplos serviços sem usar transações distribuídas é difícil
- ▶ Implementar casos de uso que contenham múltiplos serviços requer uma coordenação cuidadosa entre os times
- ▶ Maior complexidade de implantação e configuração.
- ▶ Aumento do consumo de memória. A arquitetura de microserviços substitui as N instâncias monolíticas por NxM instâncias de serviços. Cada uma roda na sua própria JVM (ou equivalente), o que é necessário para isolar as instâncias.

PARA MITIGAR AS DESVANTAGENS...



PADRÕES RELACIONADOS A MICROSERVIÇOS



EXEMPLO DE STACK DE TECNOLOGIAS

