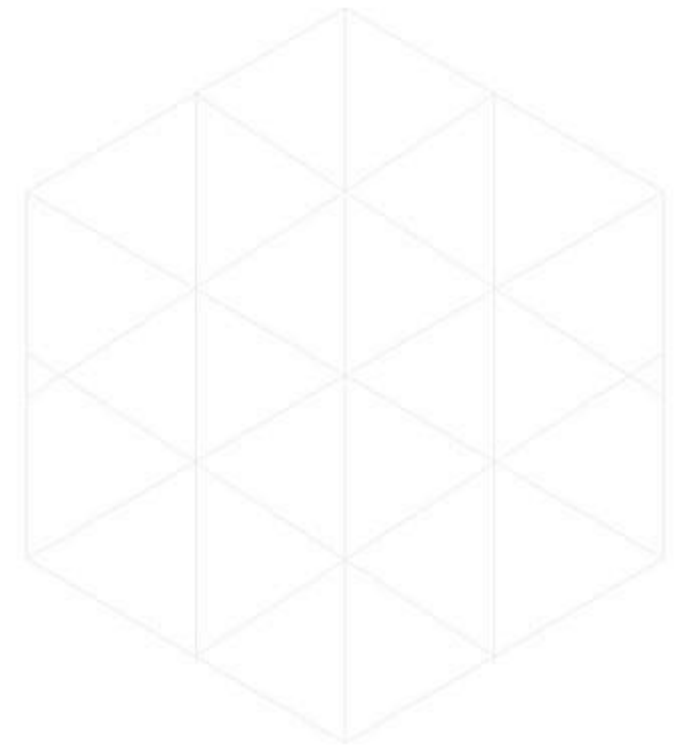




## *Fundamentos de Arquitetura de microserviços*

# ***Apresentação***

- Bacharel em ciência da computação (2003-2007)
- MBA em Arquitetura de soluções (2018)
- Arquiteto de softwares do time 7COMm (2021-...)

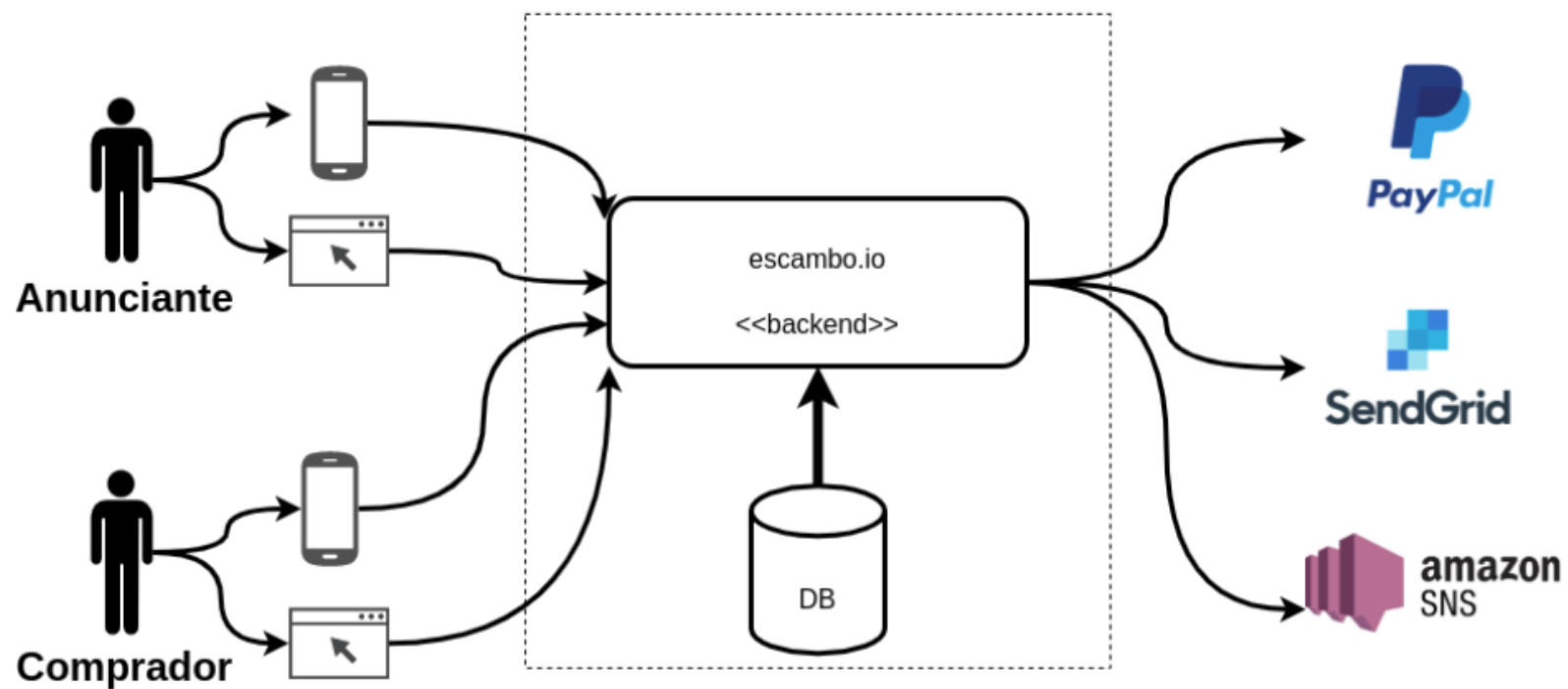




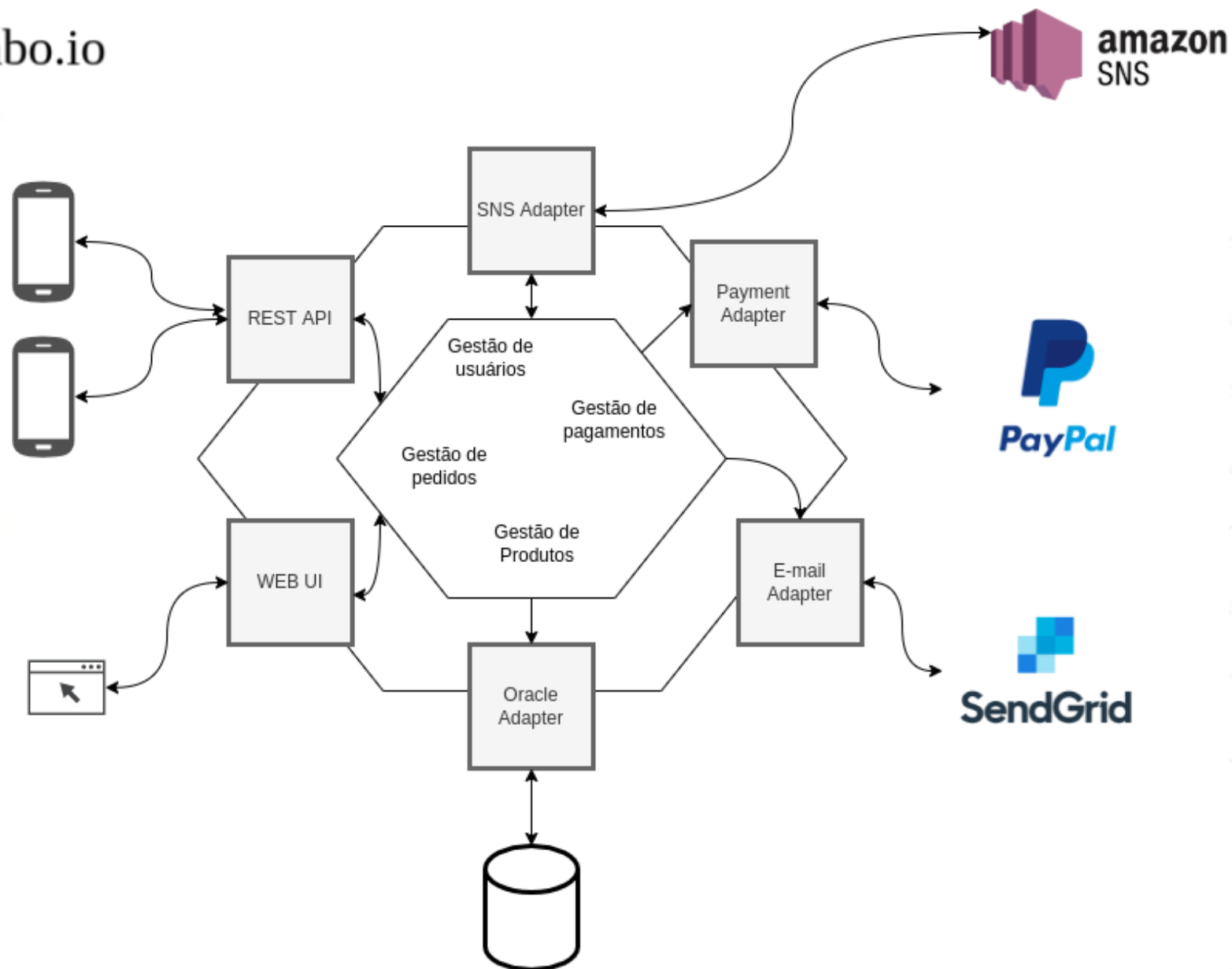
escambo.io

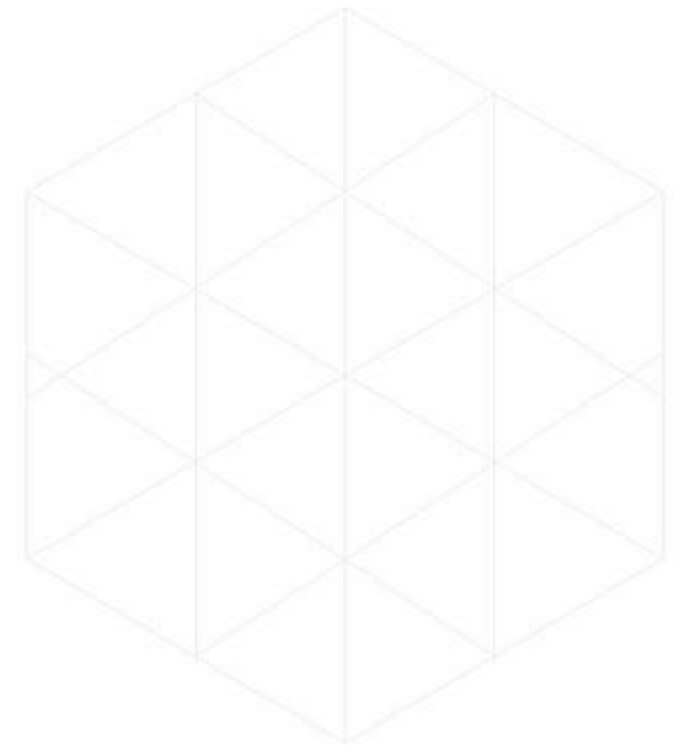
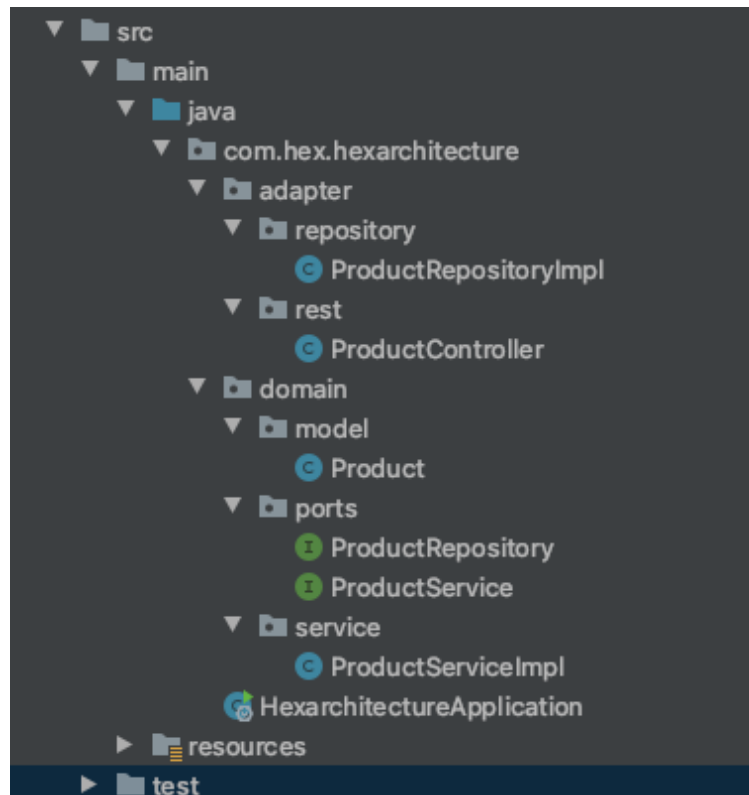
(empresa fictícia)

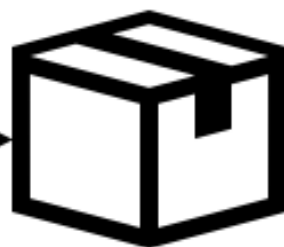
- Uma aplicação para **compra/venda** de mercadorias e serviços.
- Gerencia os **anúncios, pagamentos, mediação, contas de usuário**, etc.
- Desenvolvido com as melhores tecnologias do mercado.
- Aplicação criada em **2015**.
- Oferece soluções de comércio eletrônico para que pessoas e empresas possam comprar, vender, pagar, anunciar e enviar produtos por meio da internet.



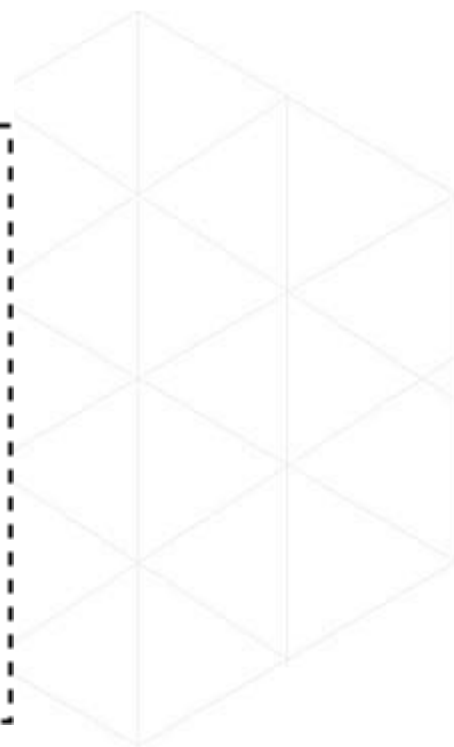








Apache  
Tomcat

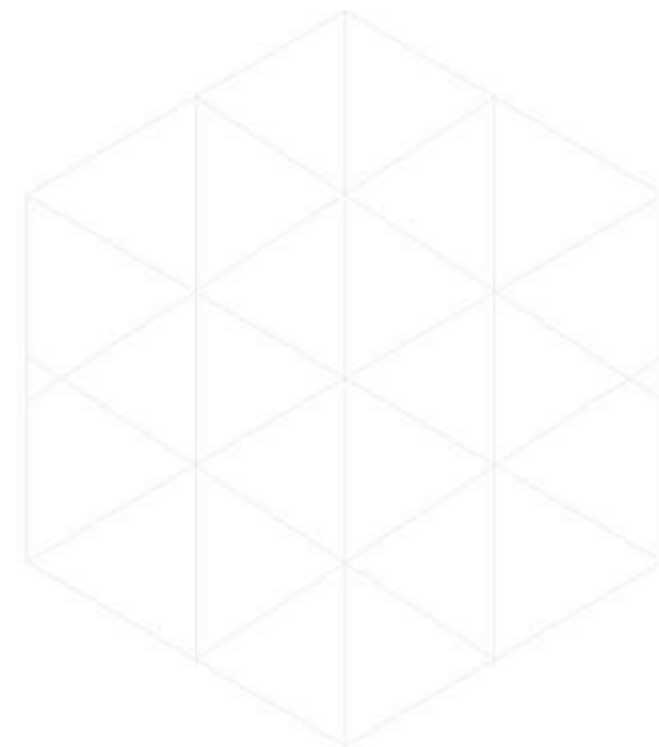


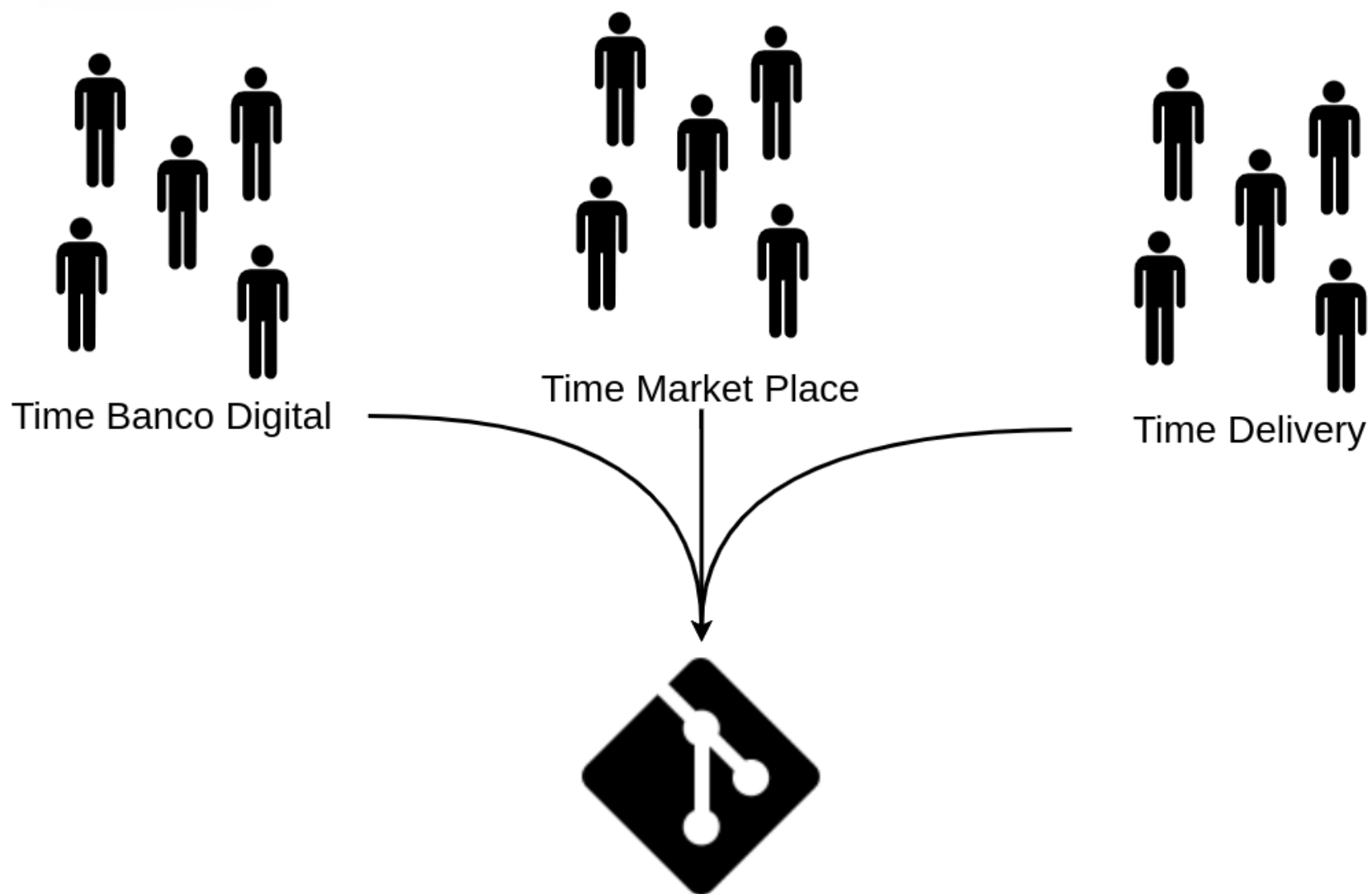
## ***Novos Requisitos***

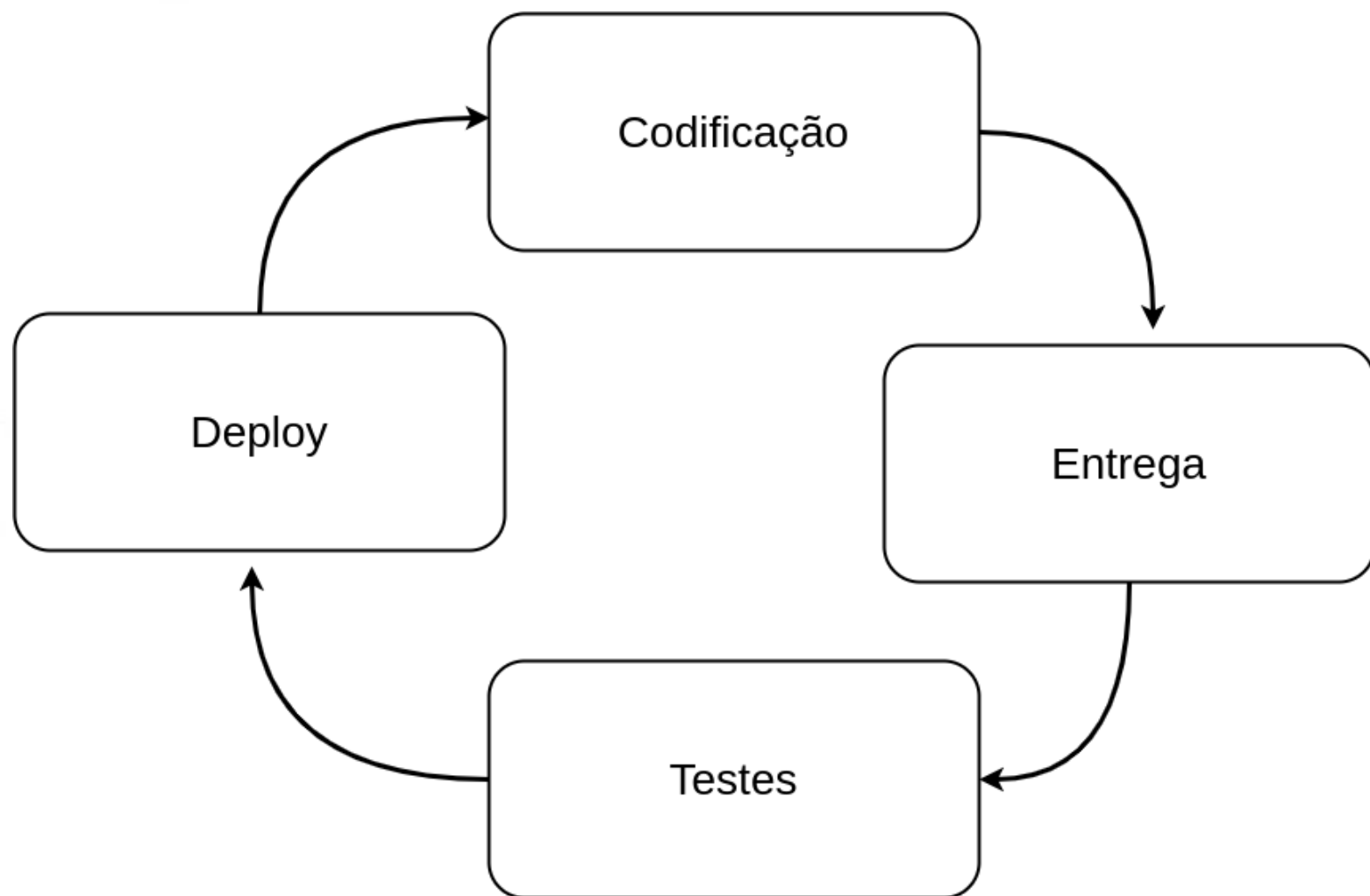


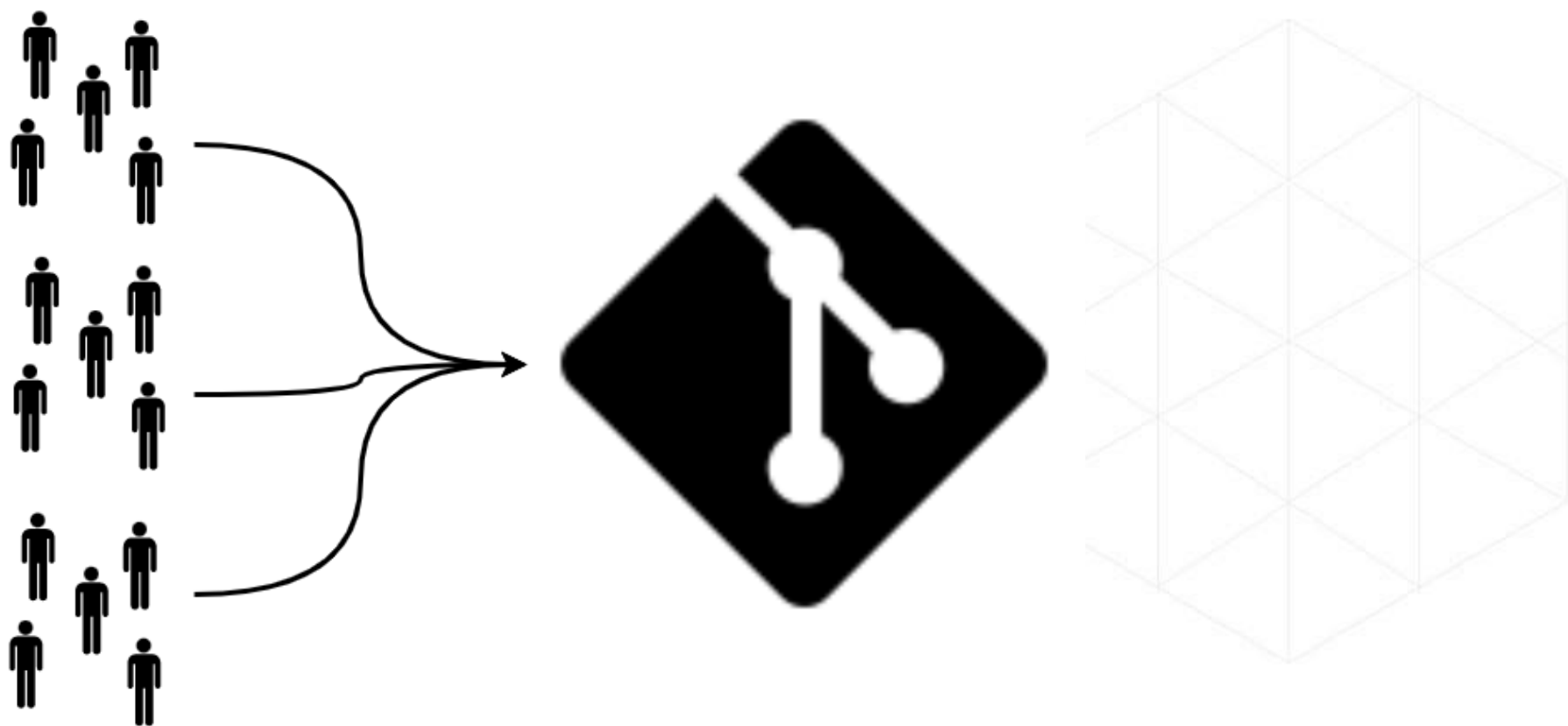


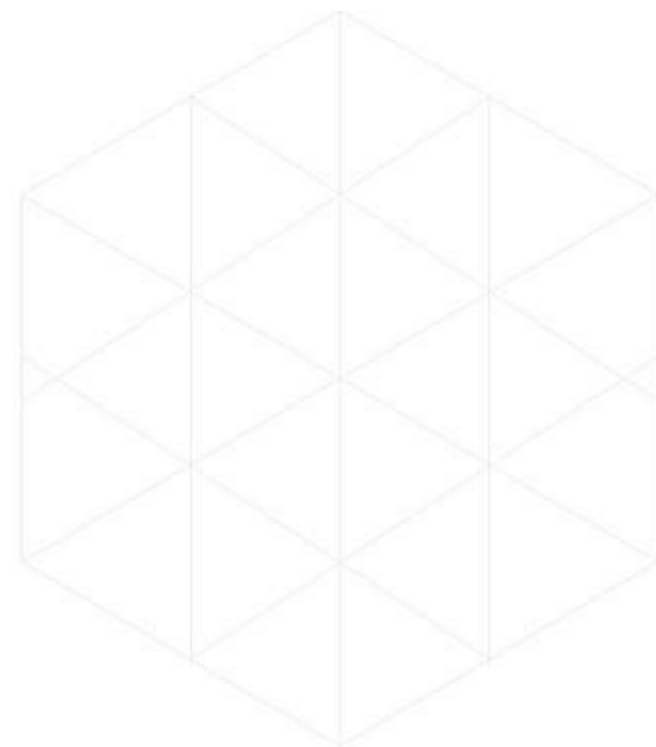
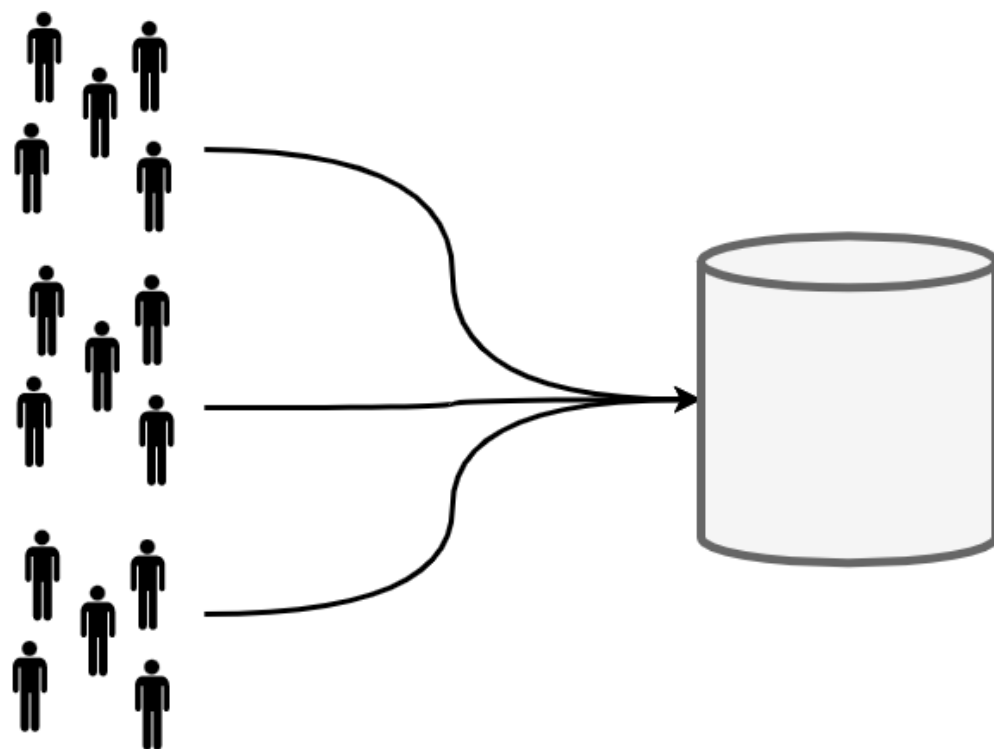
- Banco Digital
- Market Place
- Delivery por parceiros











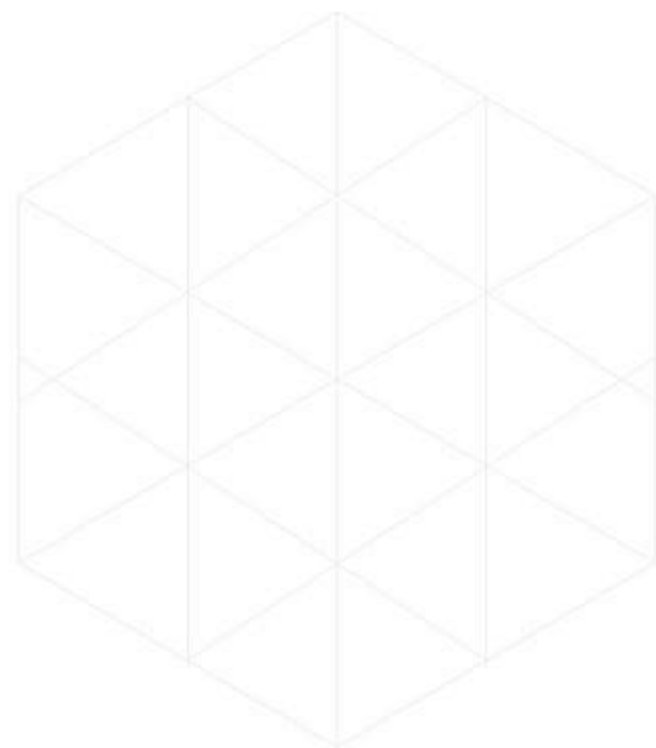
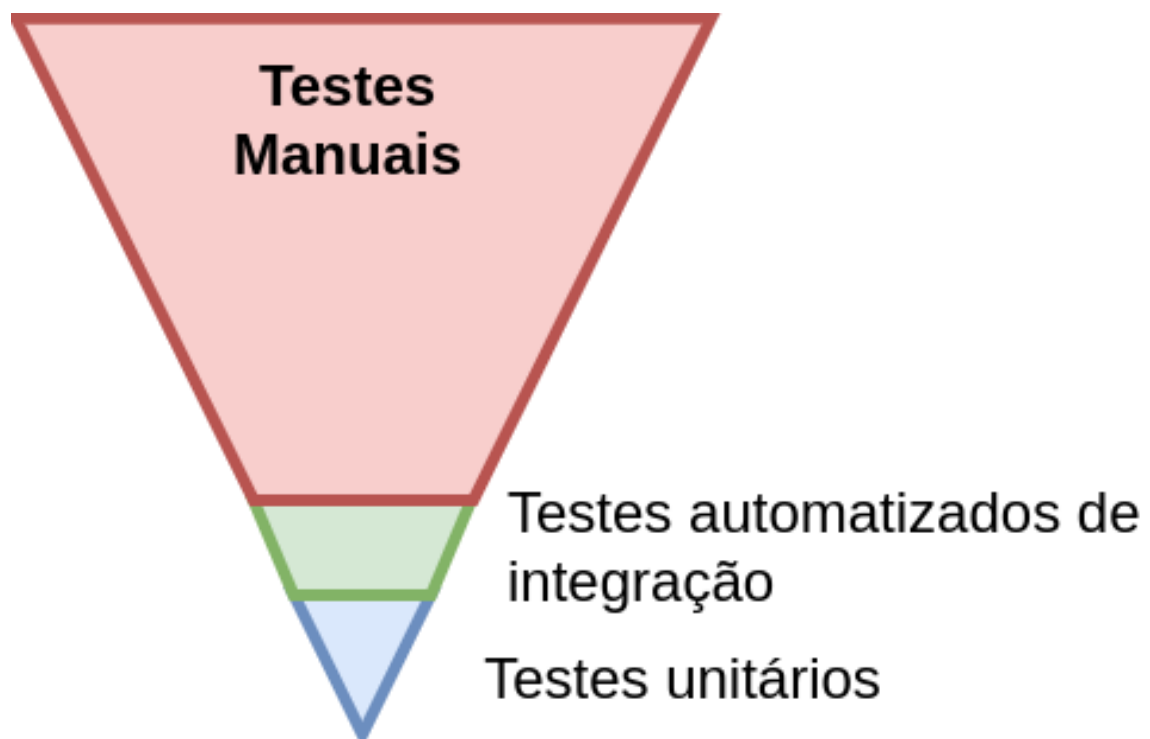


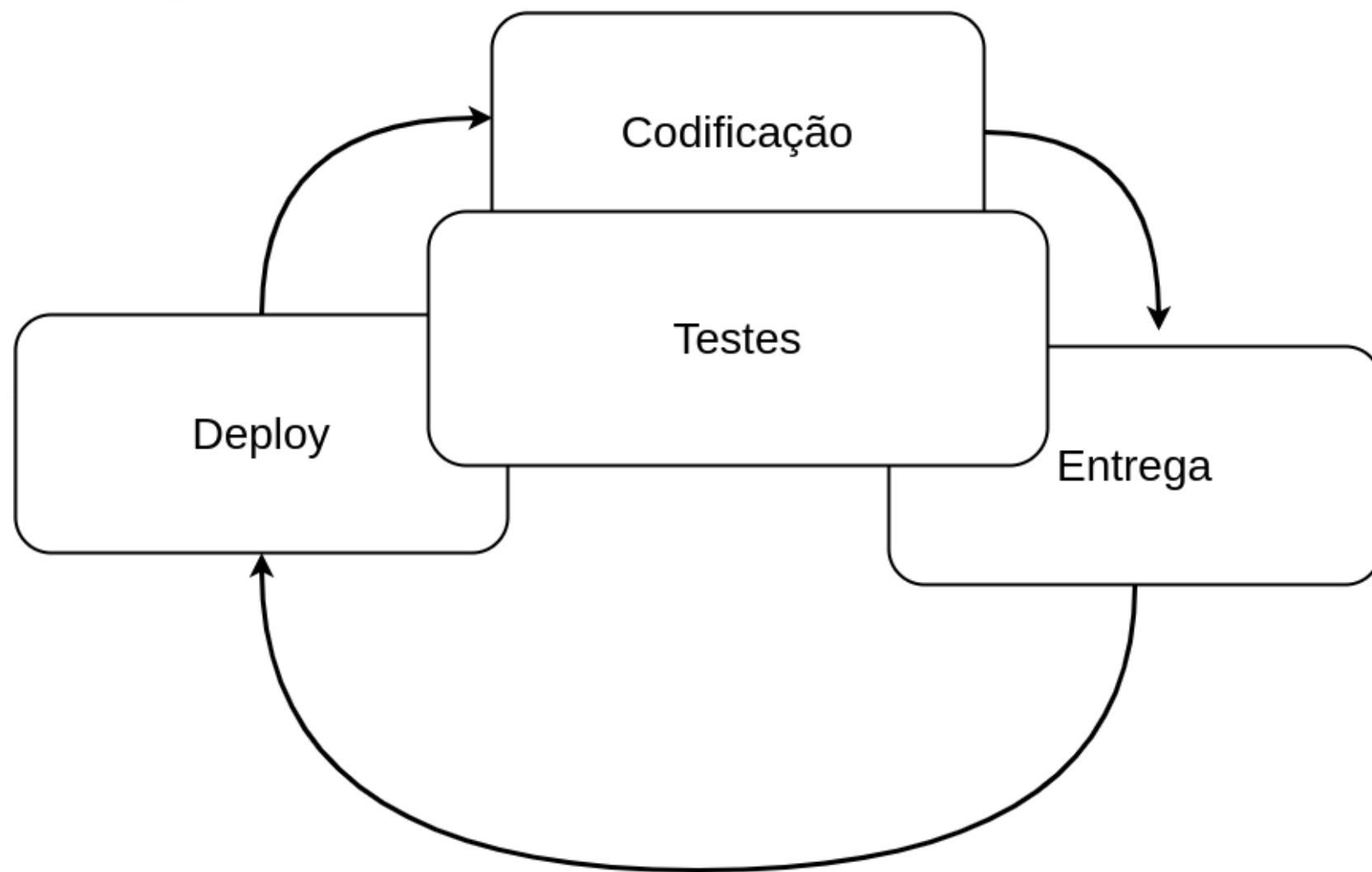
**Scrum.org**

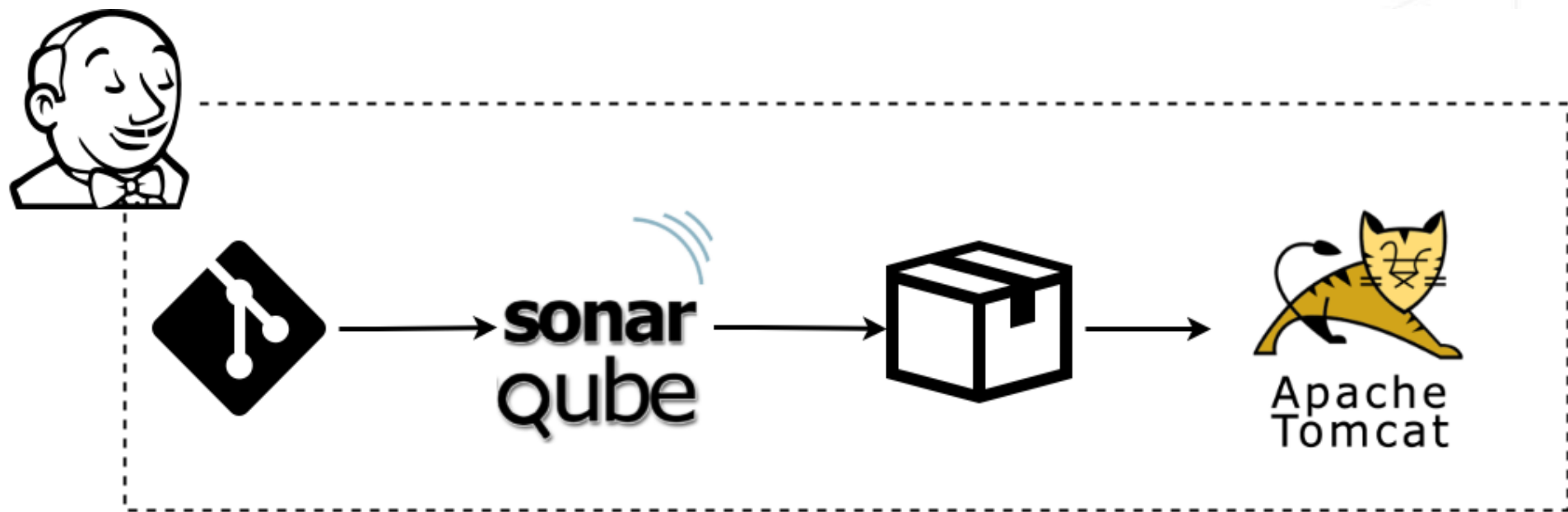
Improving the Profession of Software Development

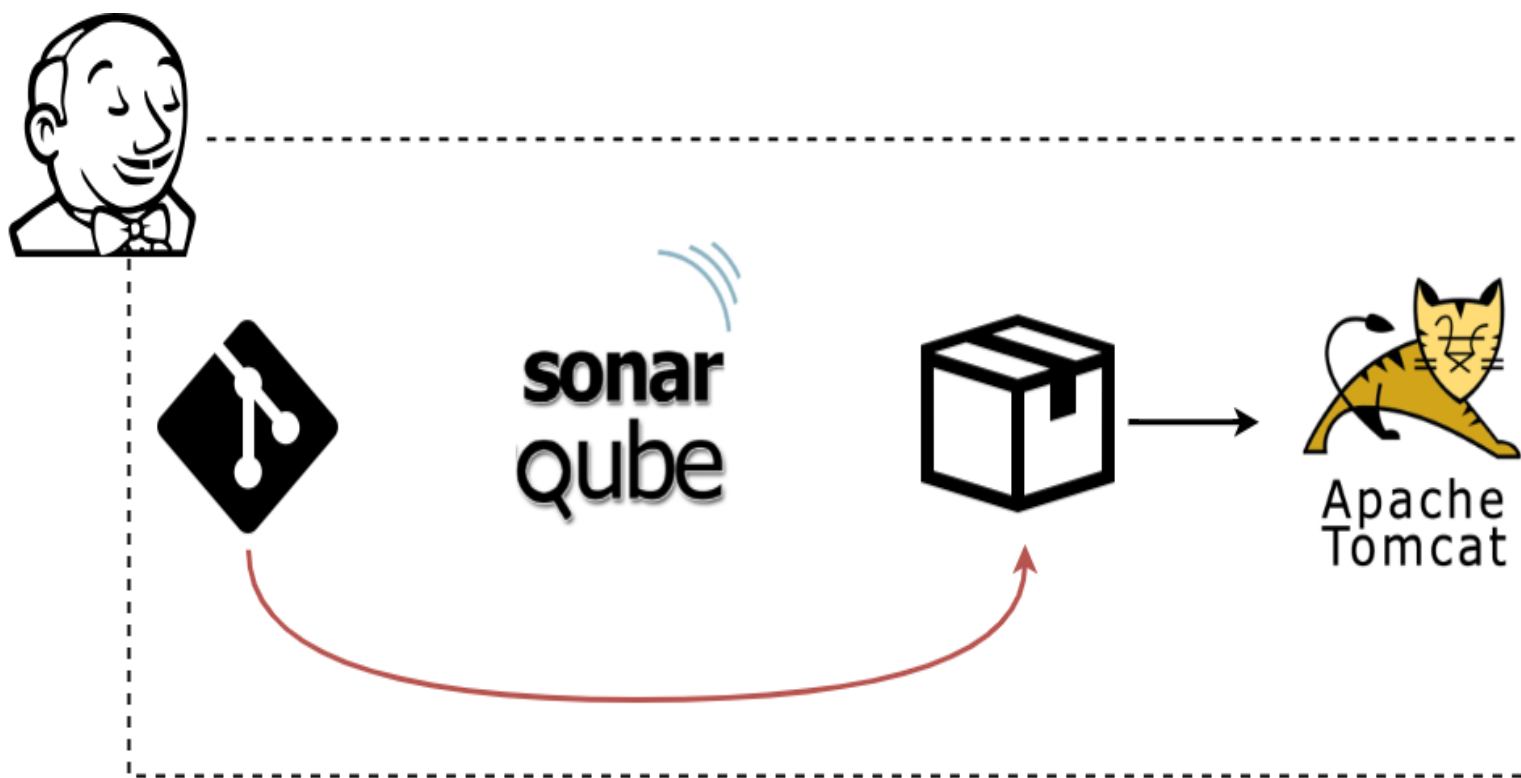


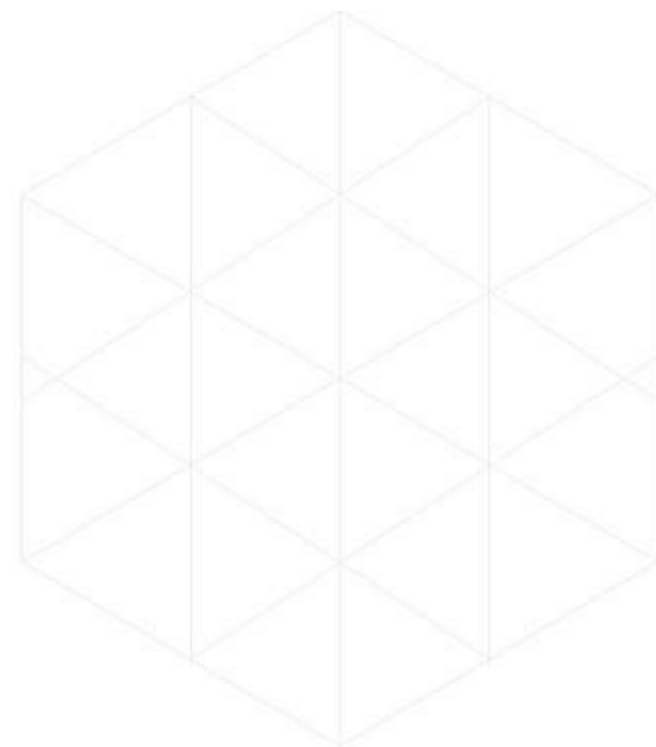
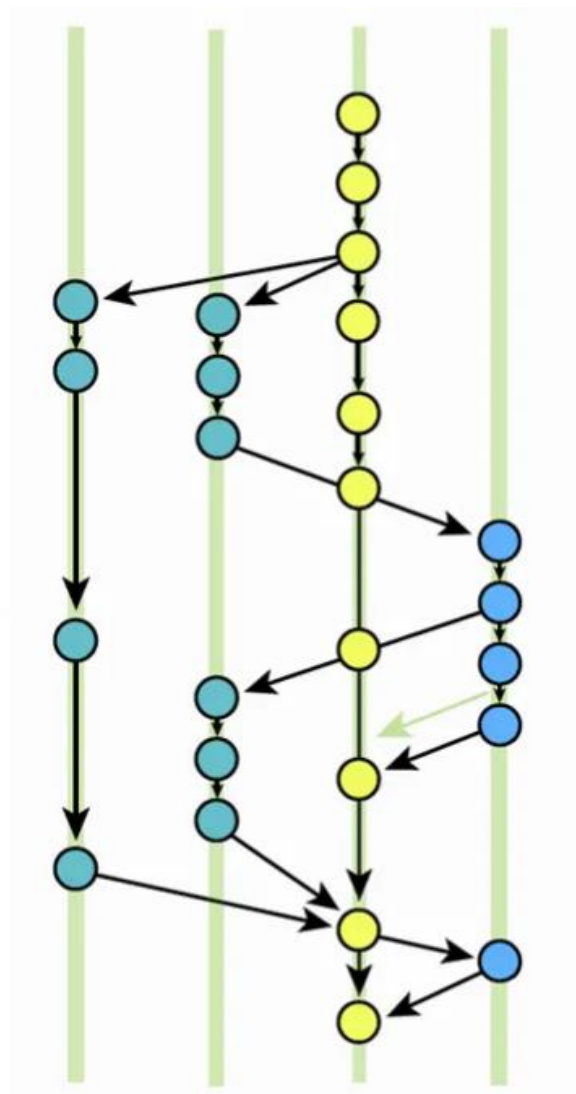


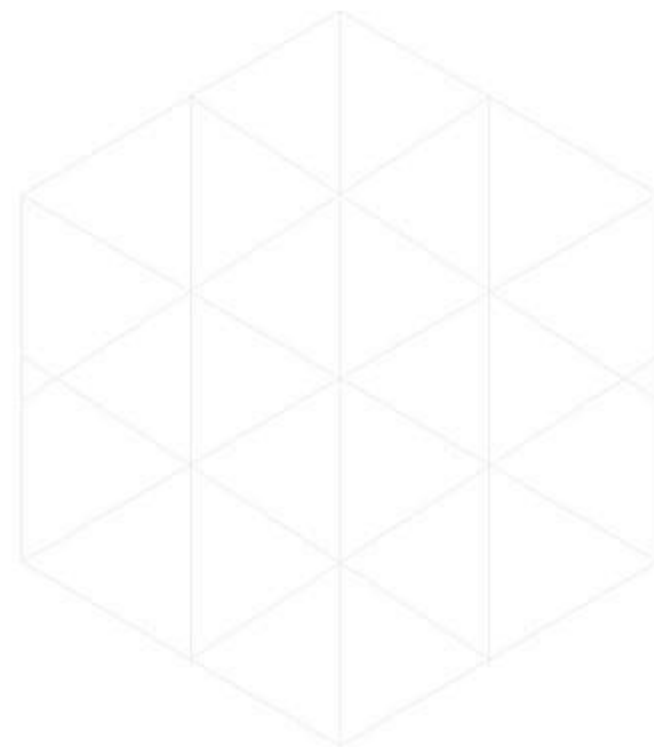
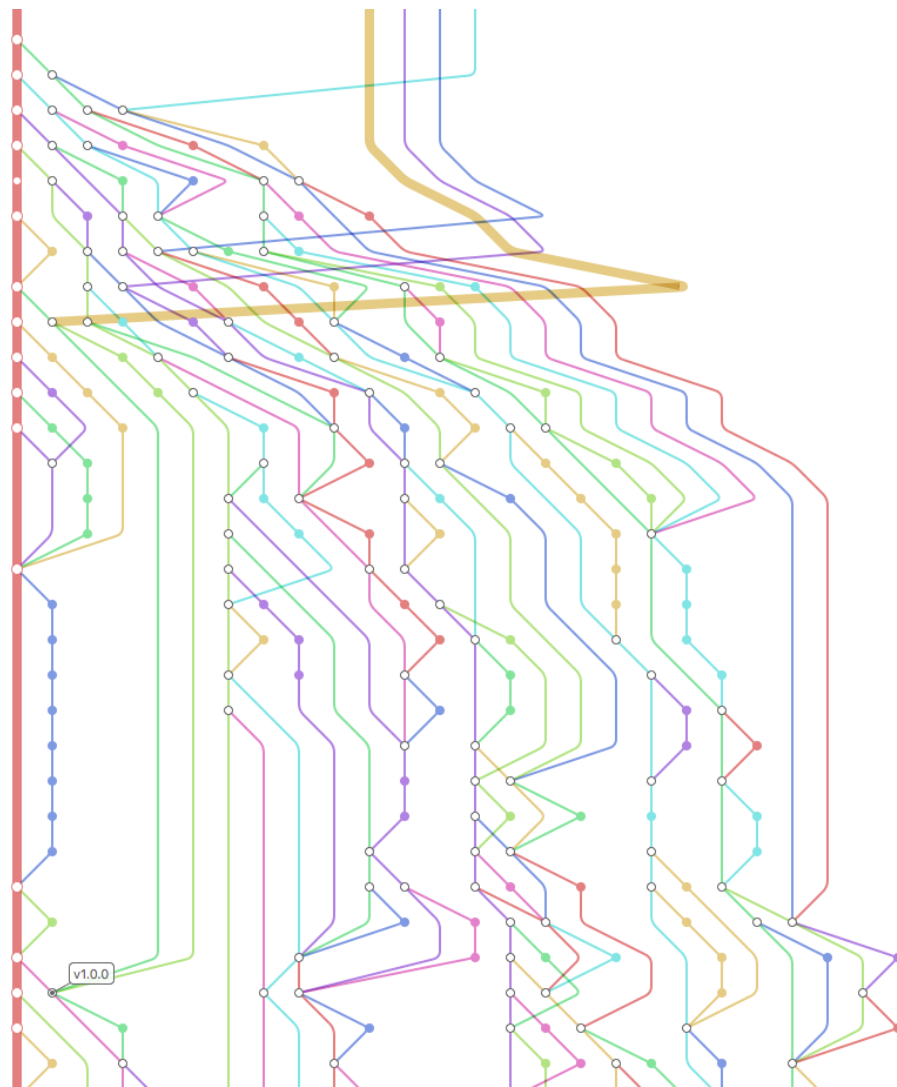














### Sprint 1

Primeira Semana: Codificação

Segunda Semana: Testes

### Sprint 2

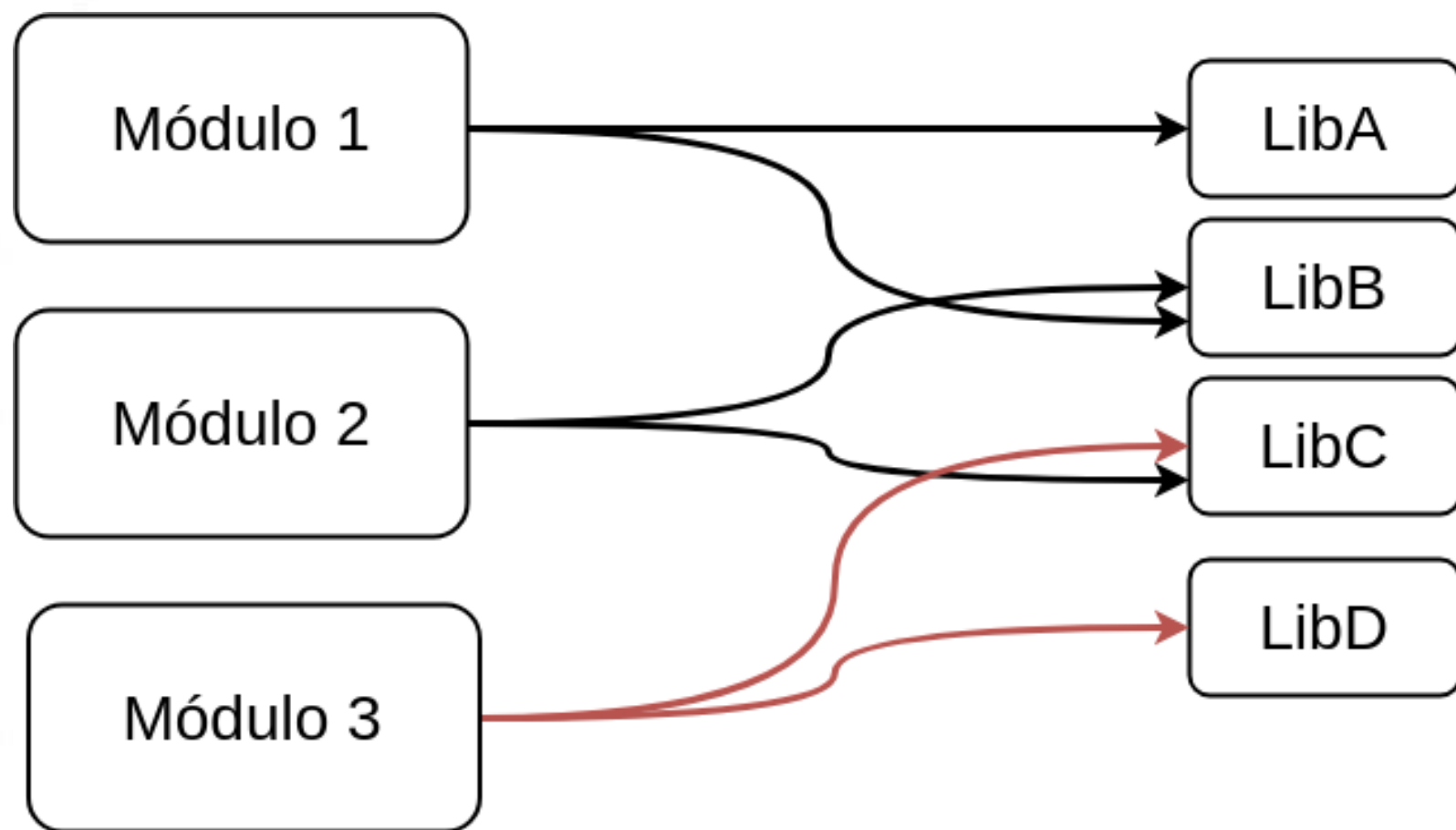
Primeira Semana: Codificação

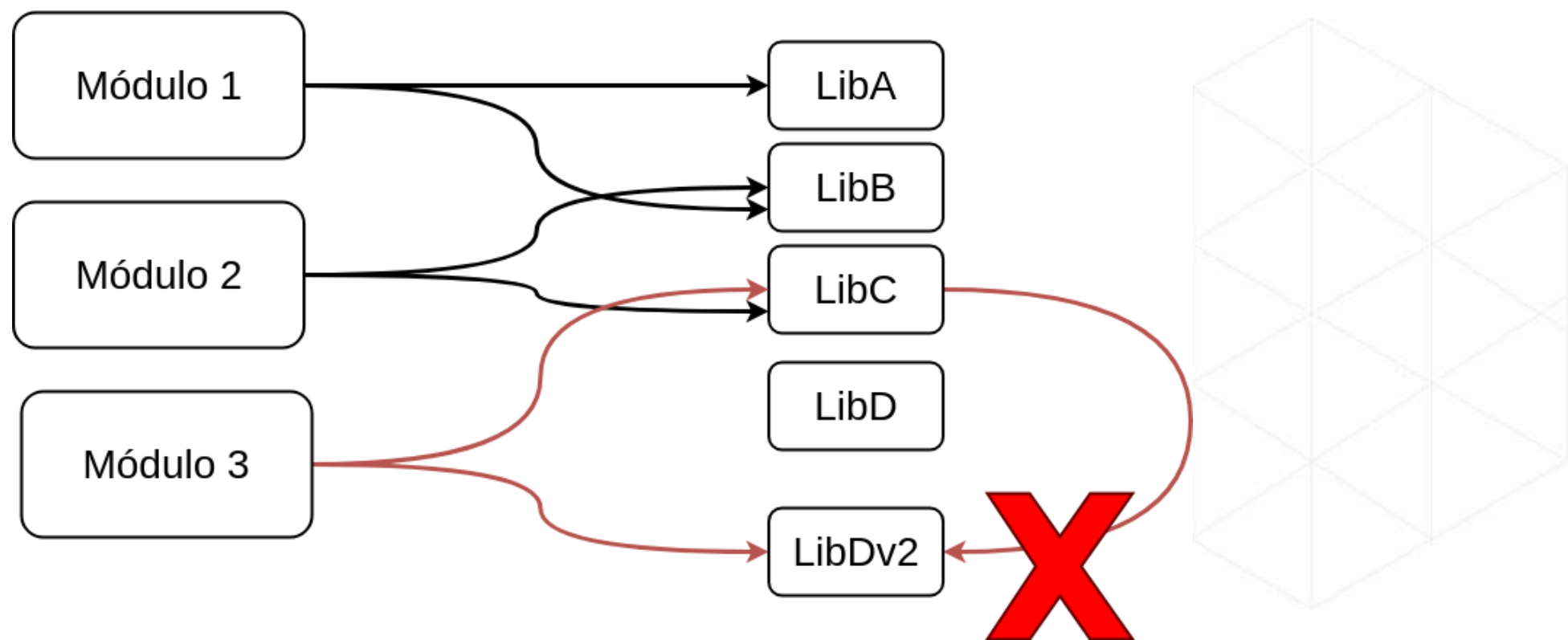
Segunda Semana: Testes

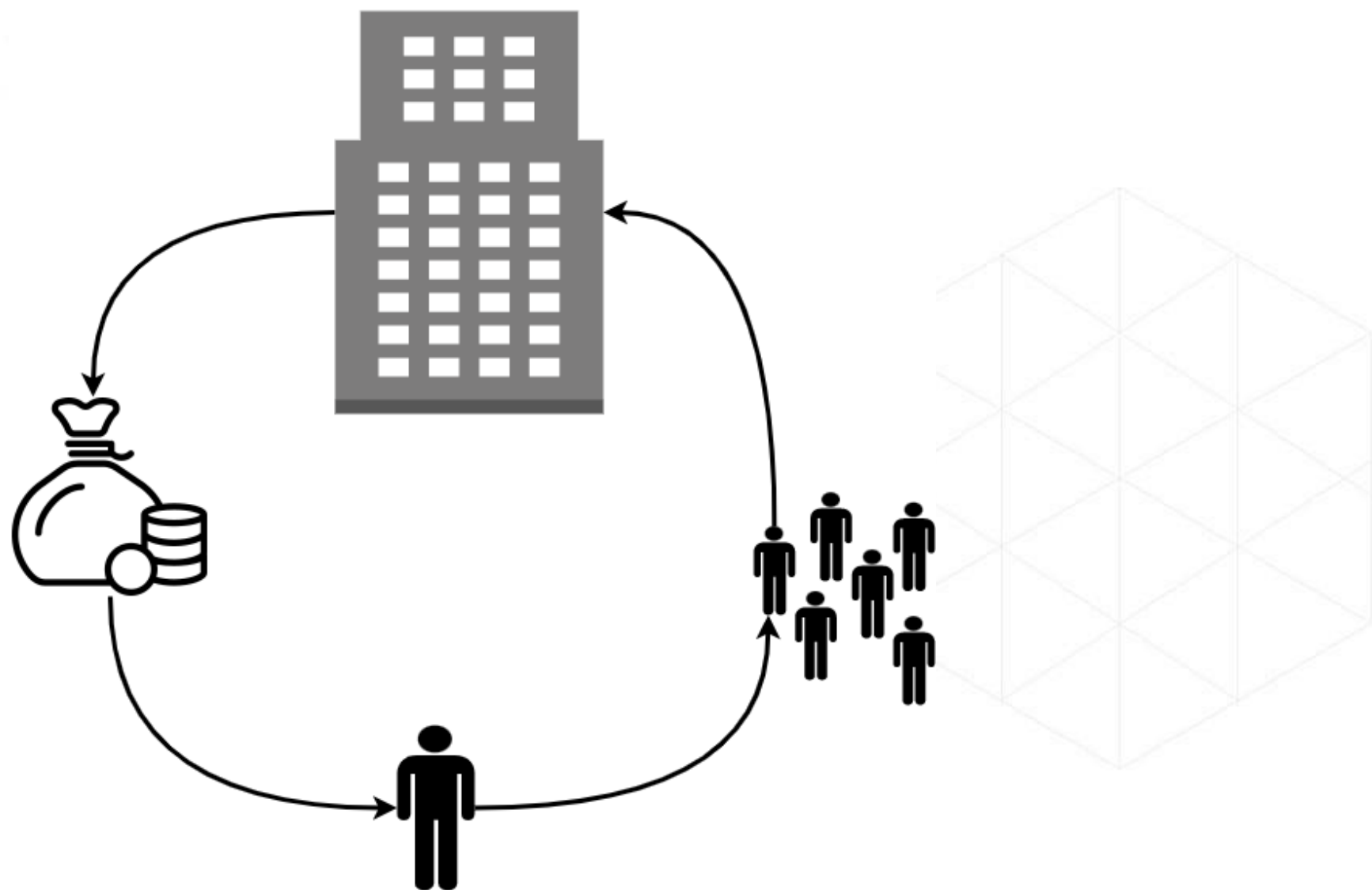


Última Sexta-feira  
Deploy















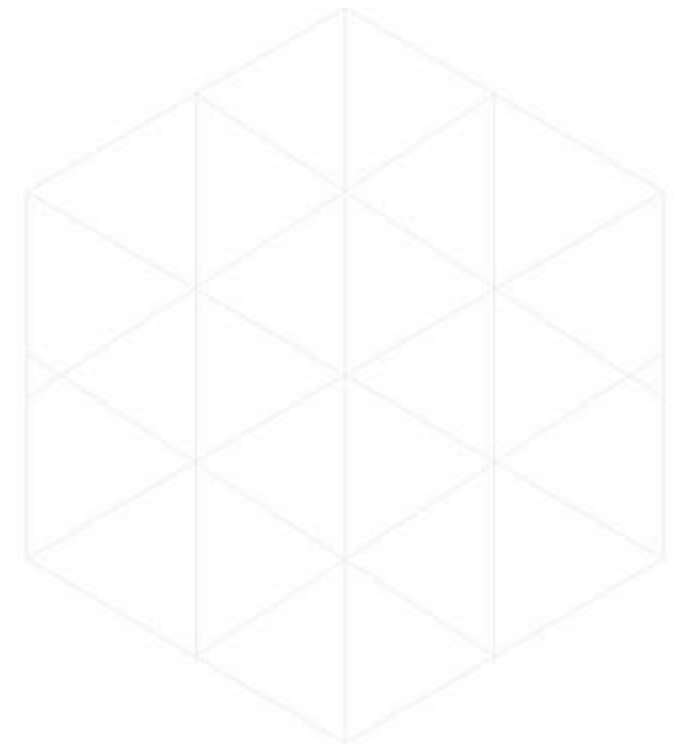
# Agenda do Curso

- *Módulos e detalhes de cada módulo*



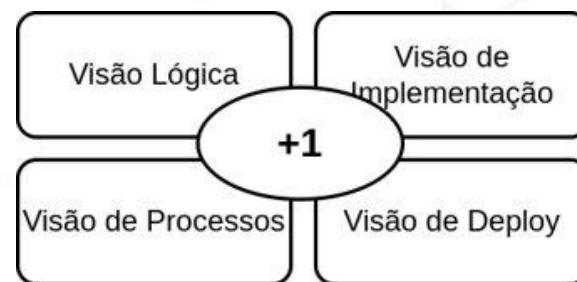
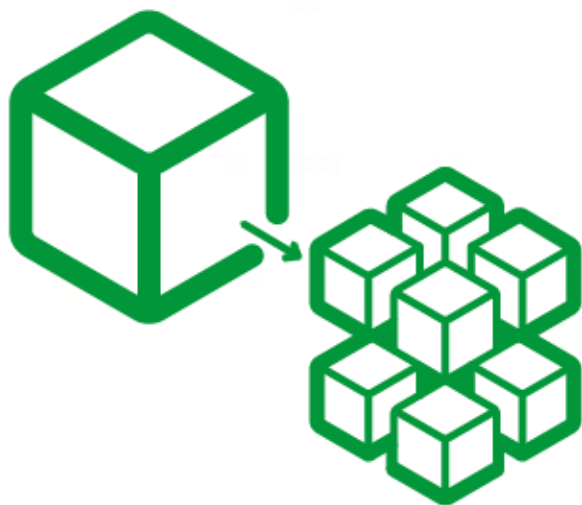
# ***Módulos***

1. Fundamentos de Arquitetura de microserviços
2. Microservices Patterns
3. Tópicos Práticos em microserviços



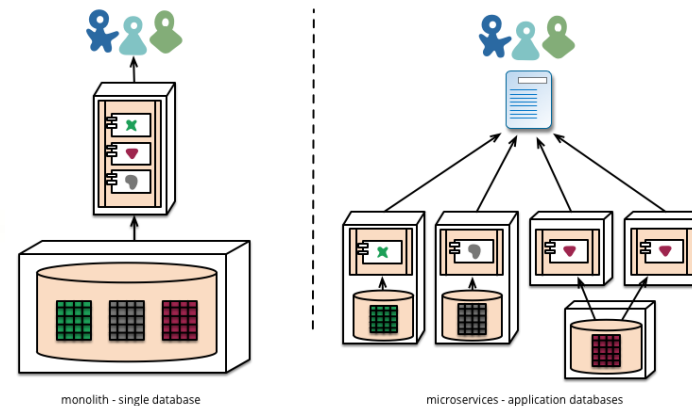
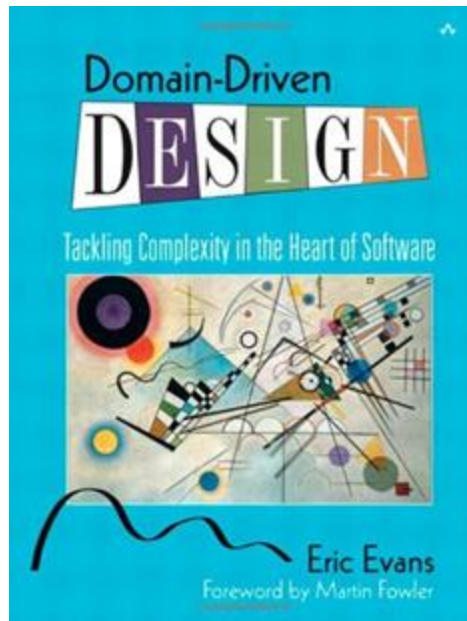
# ***Fundamentos de Arquitetura de microserviços***

- Bases de uma arquitetura distribuída



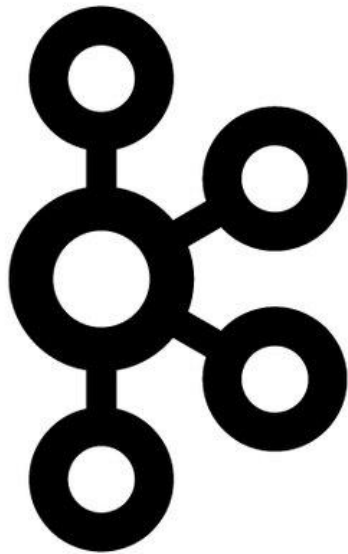
# ***Fundamentos de Arquitetura de microsserviços***

- Estratégias de decomposição e o monólito distribuído

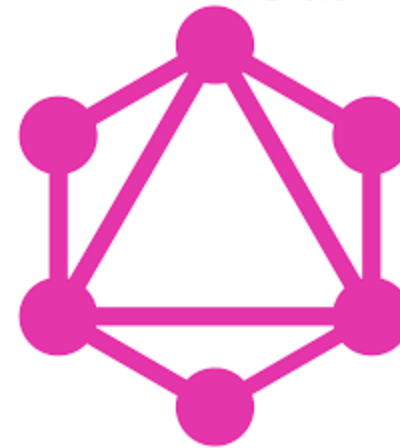
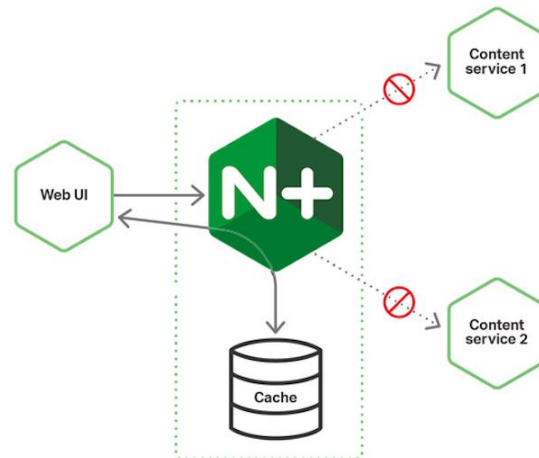


# *Fundamentos de arquitetura de microserviços*

- Integração de componentes



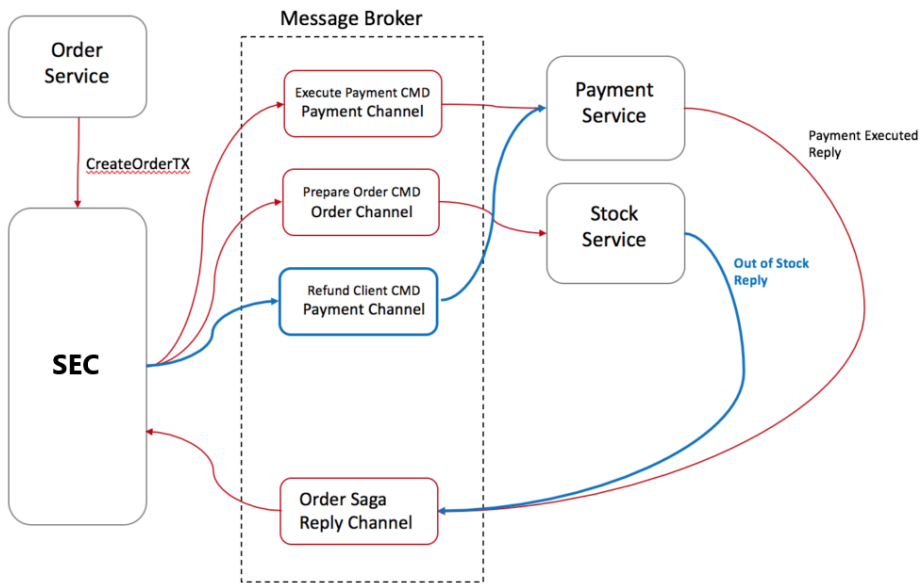
{ REST }



**NETFLIX**  
**OSS**  
Eureka

# Microservices Patterns

- Gerenciando transações com sagas



A

**Atomic**

All changes to the data must be performed successfully or not at all

C

**Consistent**

Data must be in a consistent state before and after the transaction

I

**Isolated**

No other process can change the data while the transaction is running

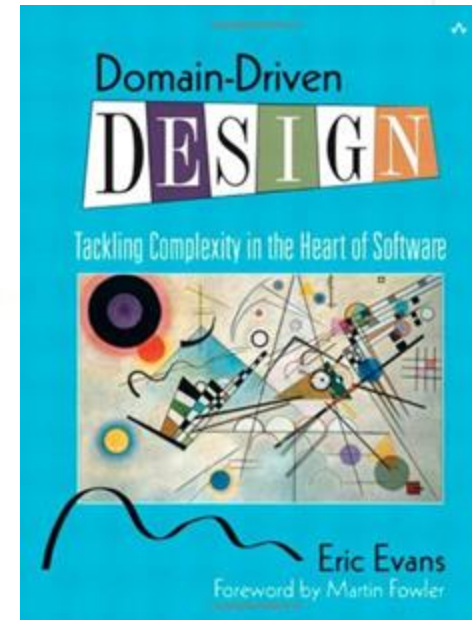
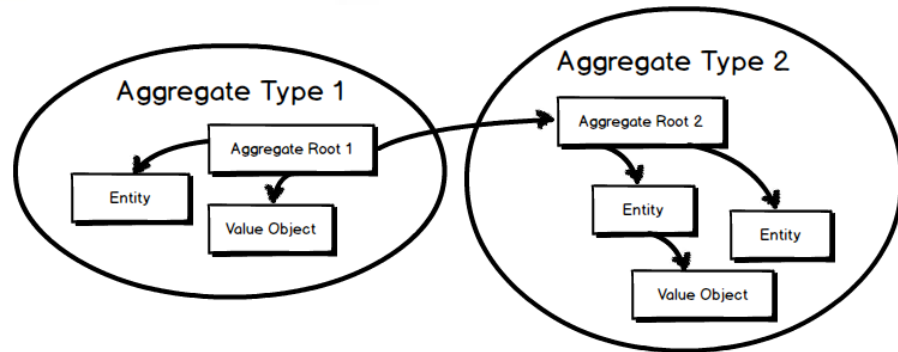
D

**Durable**

The changes made by a transaction must persist

# Microservices Patterns

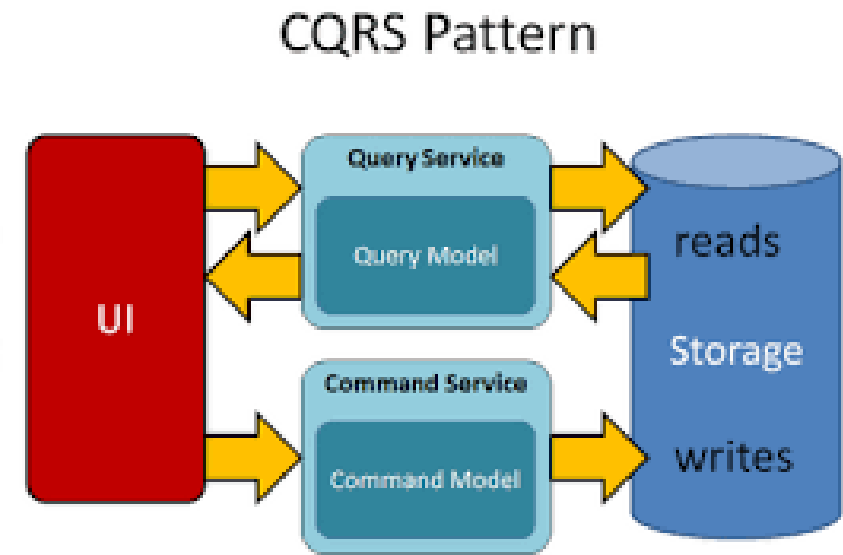
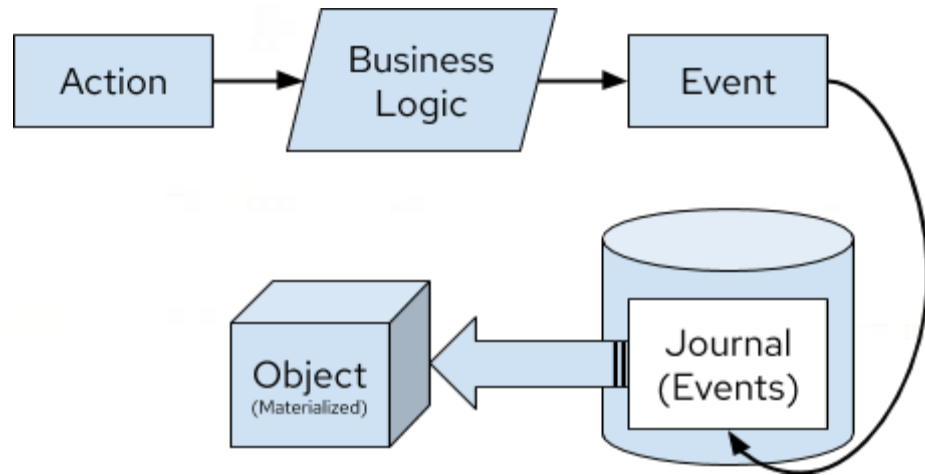
- Regras de negócio e estratégias de descentralização





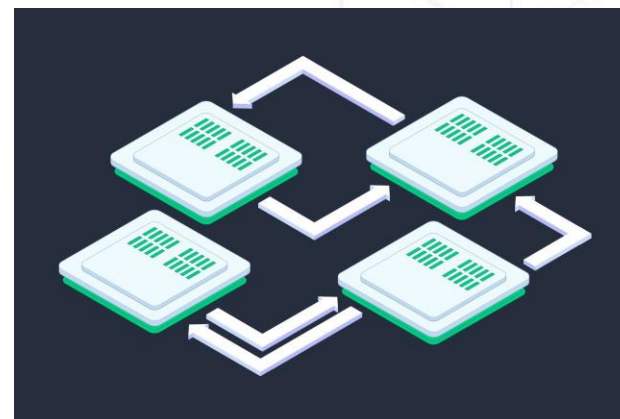
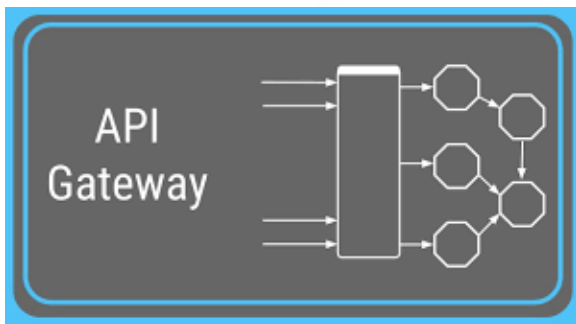
# Microservices Patterns

- Event Sourcing e CQRS



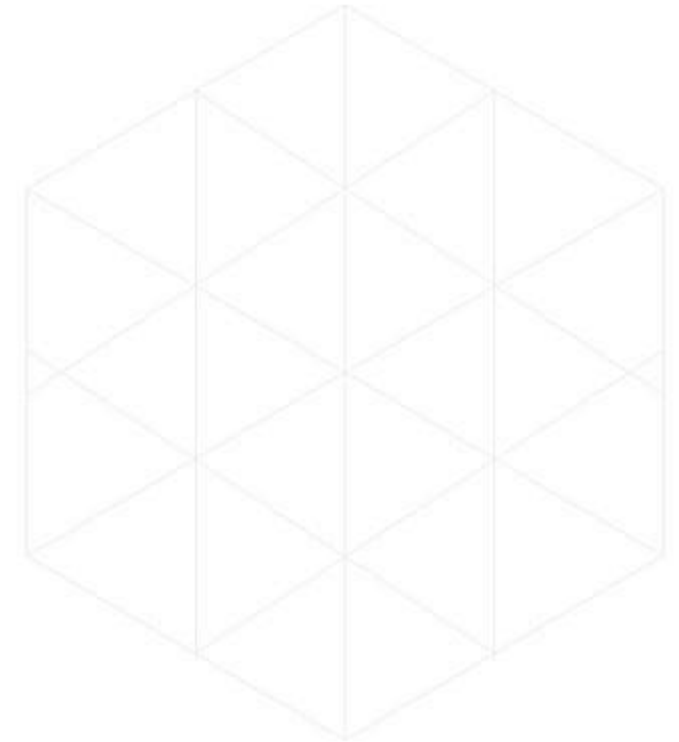
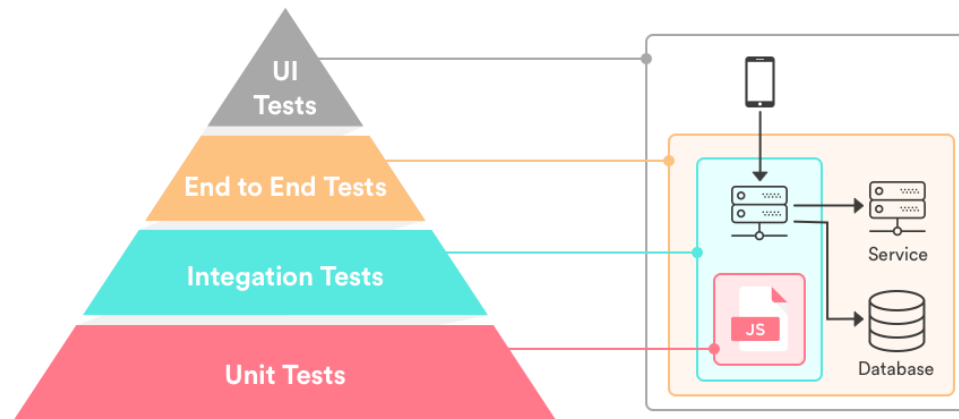
# *Microservices Patterns*

- Padrões para APIs Externas



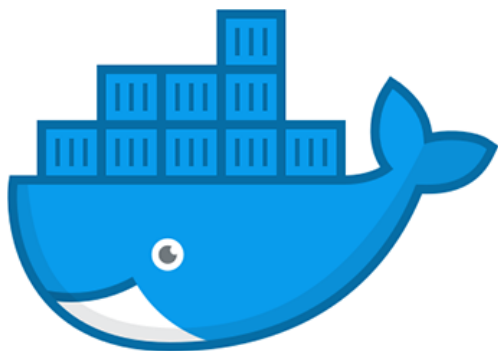
# ***Tópicos Práticos em microserviços***

- Testando microserviços



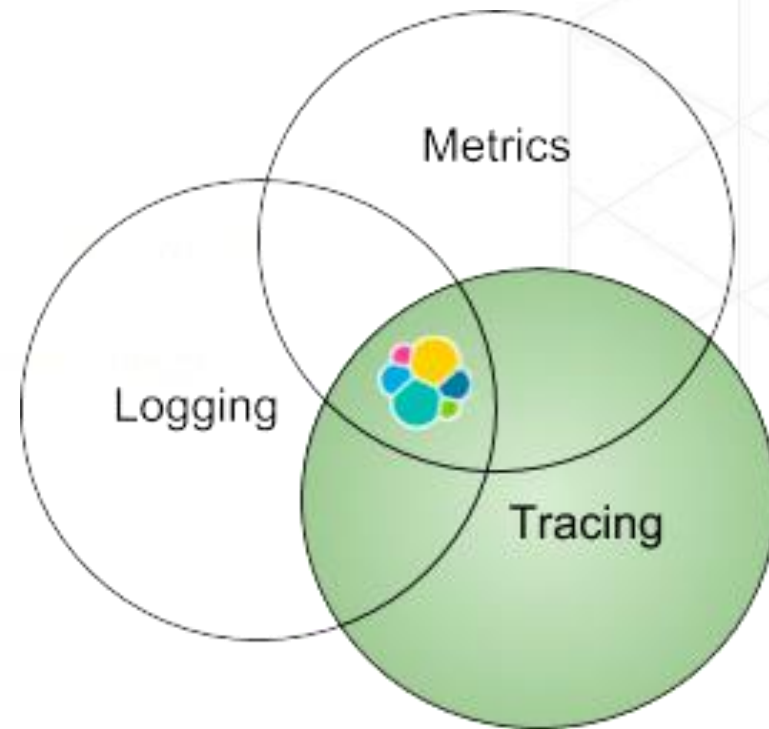
# ***Tópicos Práticos em microserviços***

- Deploy de microserviços e produção



# ***Tópicos Práticos em microserviços***

- Monitoramento de uma aplicação em microserviços



# ***Dinâmica do Curso***

- Apresentação no *teams*
- Exemplos práticos no *github*
- Apresentação de soluções (no caso de frameworks ou ferramentas relevantes)
- ...
- Pré-requisitos ***Desejáveis***
  - Git
  - Docker
  - JDK + IDE
  - Acesso ao github







# *A arquitetura em microserviços*

# *Vamos do começo*

- Micro-serviço
- Microserviço
- Micro serviço
- Microsserviço





# *Vamos do começo*

- O falso prefixo micro é um elemento de composição na formação de palavras. Pela nova ortografia, somente é separado do segundo elemento por hífen nos casos em que este inicia por "o" ou "h".
- Caso o segundo elemento inicie com a consoante "s" ou "r", é necessário dobrá-la, sem usar hífen. Nos demais casos, quando o segundo elemento inicia por outras consoantes ou vogais, não há hífen.
- <http://www.portaldalinguaportuguesa.org/?action=acordo&version=1990>

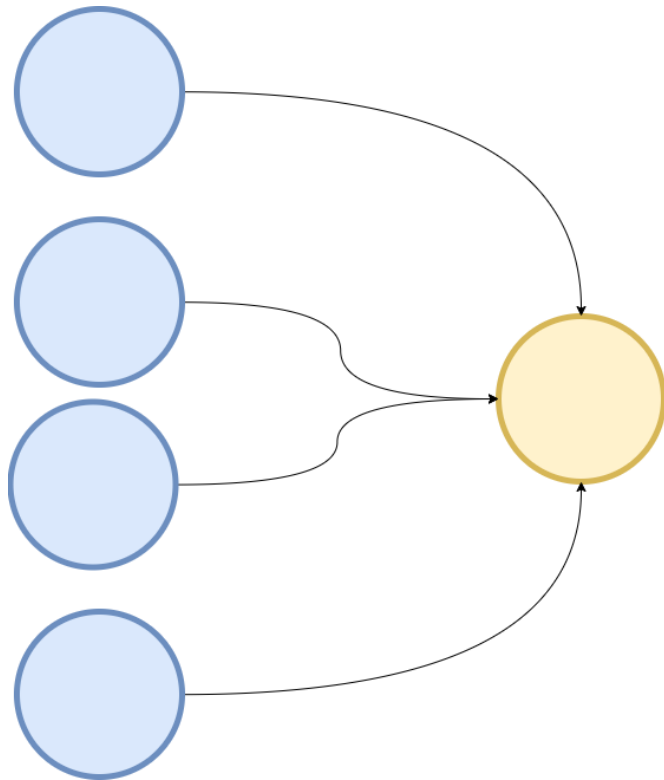
# *Vamos do começo*

- ~~Micro-serviço~~
- ~~Microserviço~~
- ~~Micro-serviço~~
- **Microserviço**

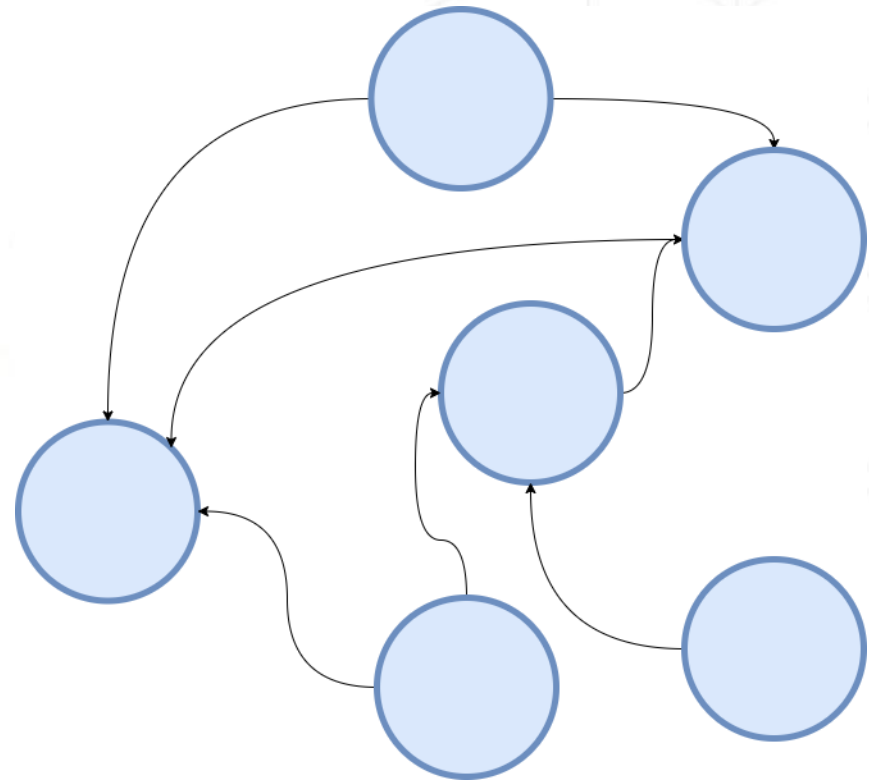


# *Arquitetura de Sistemas*

**Arquitetura Centralizada**

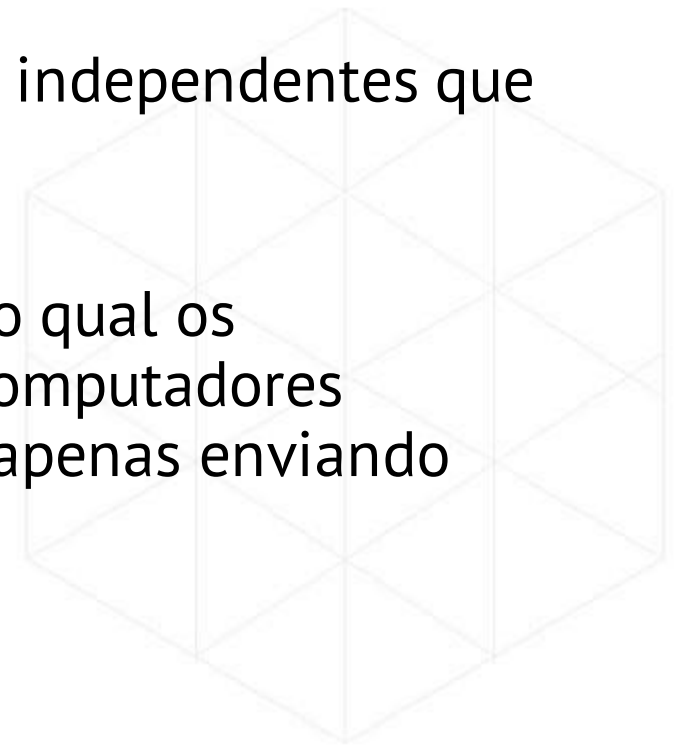


**Arquitetura distribuída**

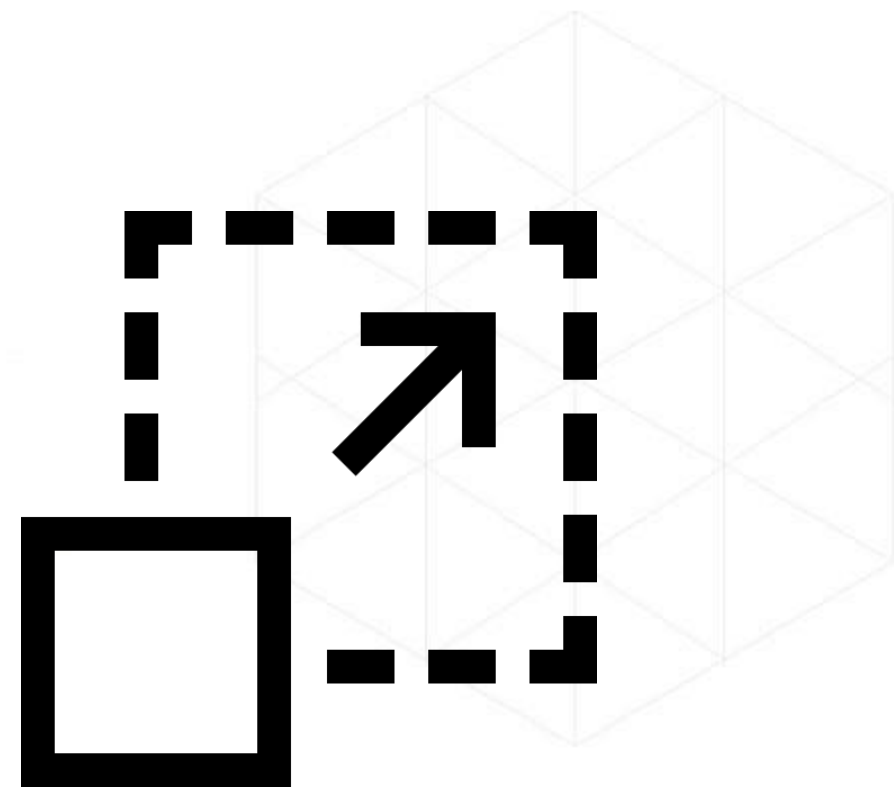


# ***Sistema distribuído***

- “Um sistema distribuído é um conjunto de computadores independentes que se apresenta a seus usuários como um sistema único e coerente”[TANENBAUM]
- “Definimos um sistema distribuído como sendo aquele no qual os componentes de hardware ou software, localizados em computadores interligados em rede, se comunicam e coordenam ações apenas enviando mensagens entre si.” COULORIS[2]



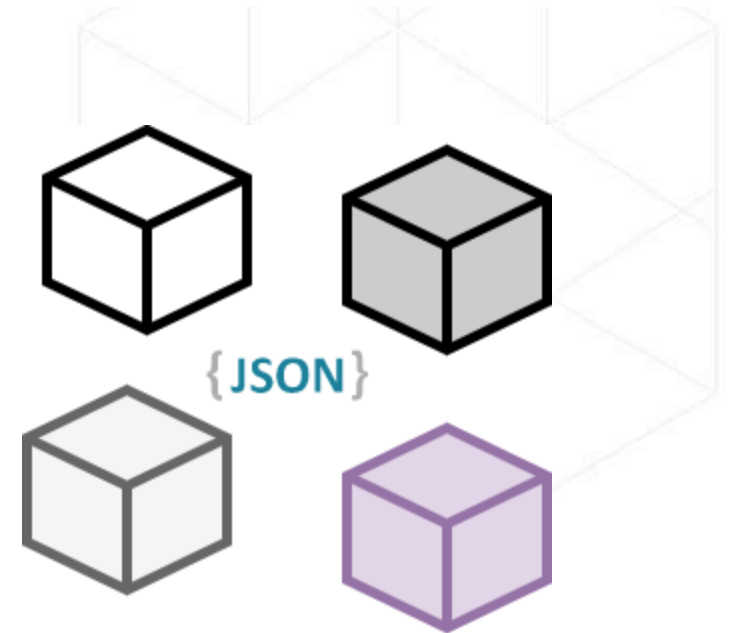
# ***Pilares de uma arquitetura distribuída (no contexto de microsserviços)***



# Modularidade

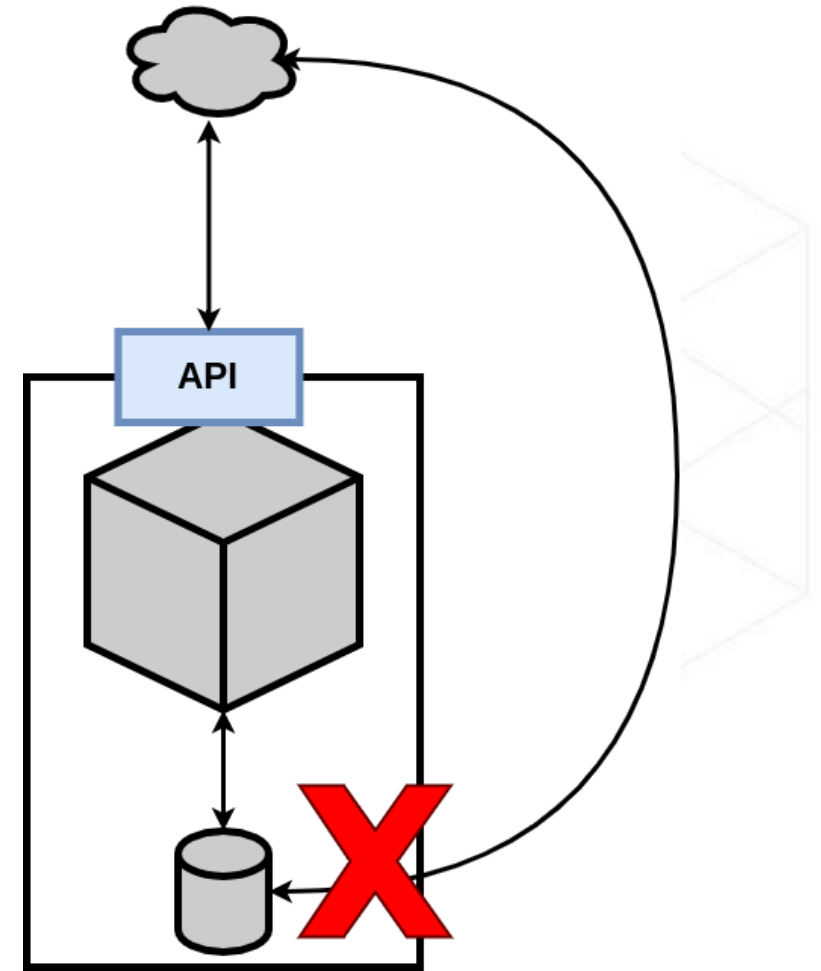
- Conceito popular e importantíssimo no desenvolvimento de software.
- Aplicações desenvolvidas em módulos menores conceitualmente ou tecnicamente próximos.
- Em microsserviços, os módulos podem funcionar como serviços, com uma API bem definida.

**S.O.L.I.D.**



# Modularidade

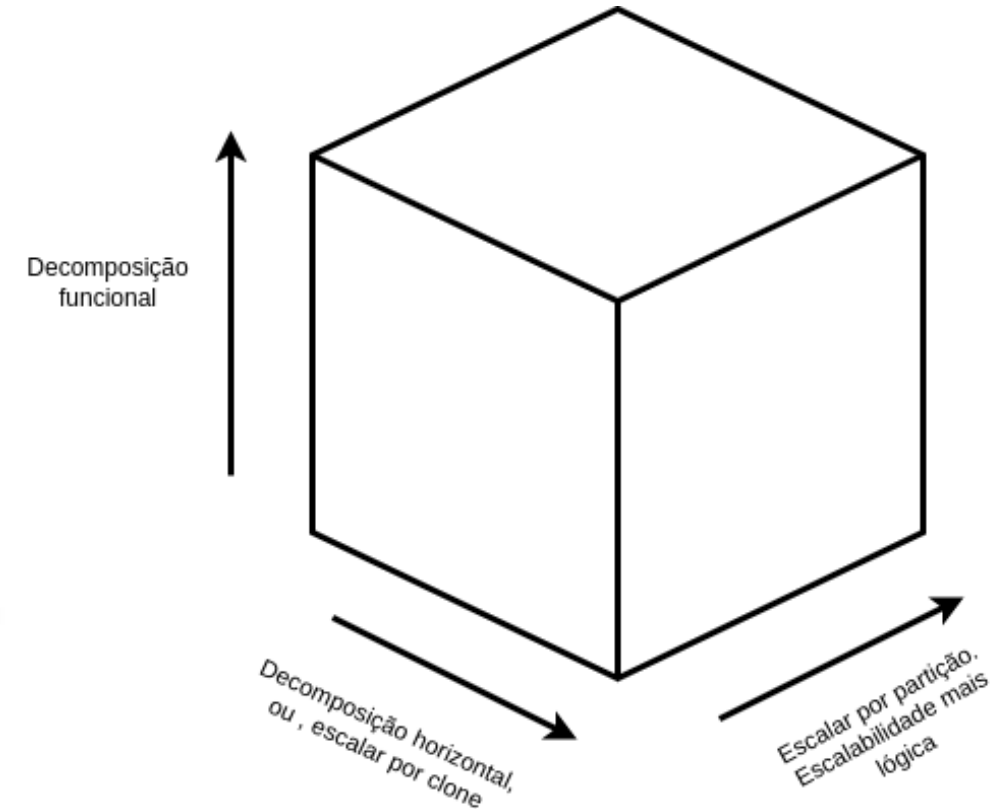
- Em uma arquitetura de microsserviços, o serviço, e somente ele, pode acessar diretamente sua base de dados.
- 100% do acesso externo deve ser feito através de APIs bem definidas.
- Em microsserviços não existe a integração via Banco de dados!





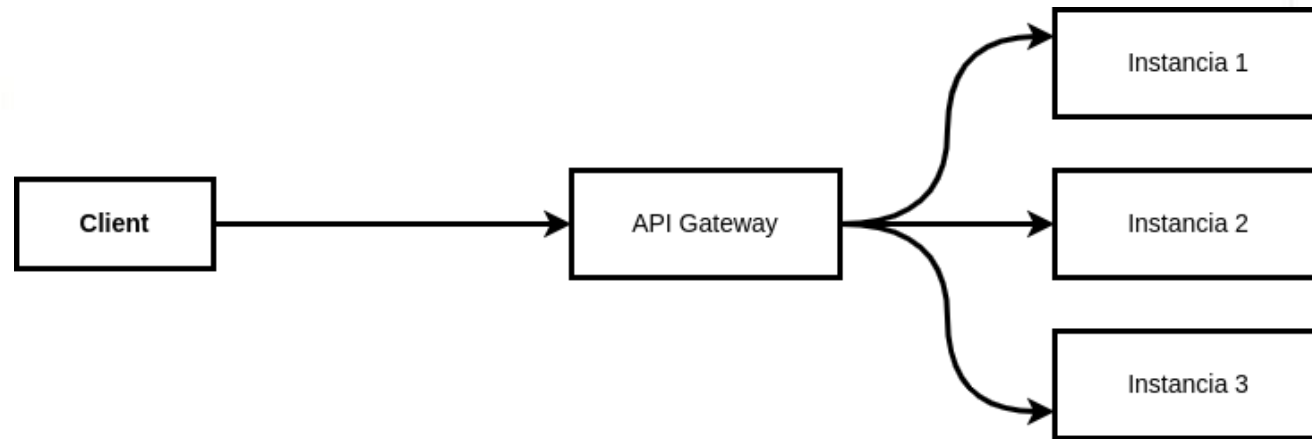
# Escalabilidade

- Escalabilidade é a capacidade que um sistema tem de crescer para atender as demandas sem perder as qualidades que lhe agregam valor.
- Com boa escalabilidade uma companhia consegue atender mais clientes por um custo menor, sem ter de substituir nenhum dos seus sistemas.
- O cubo da escalabilidade define três maneiras de se escalar uma aplicação.



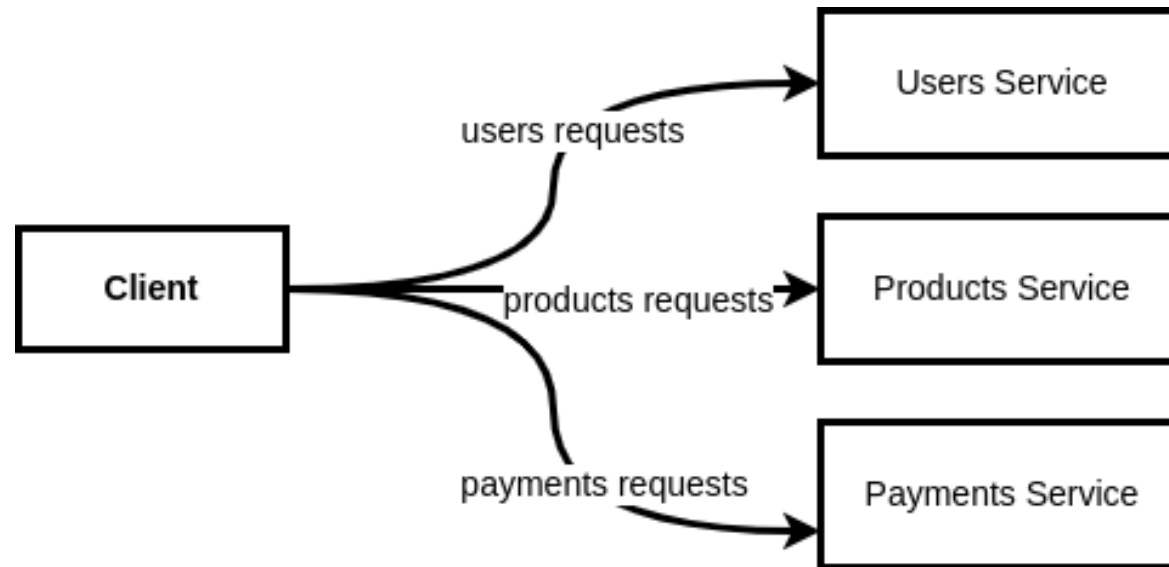
# *Decomposição horizontal*

- Maneira comum de se escalar uma aplicação monolítica, através de múltiplas instancias



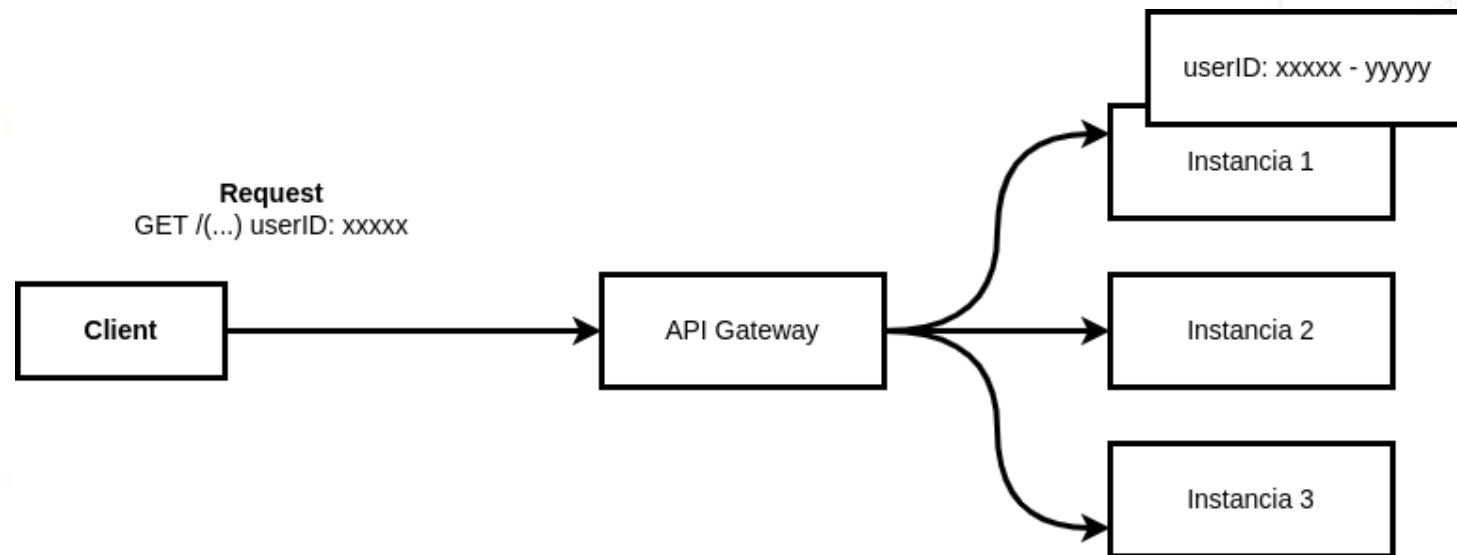
# *Decomposição Funcional*

- Maneira de escalar comum em uma arquitetura de microsserviços, através da decomposição da aplicação em serviços menores

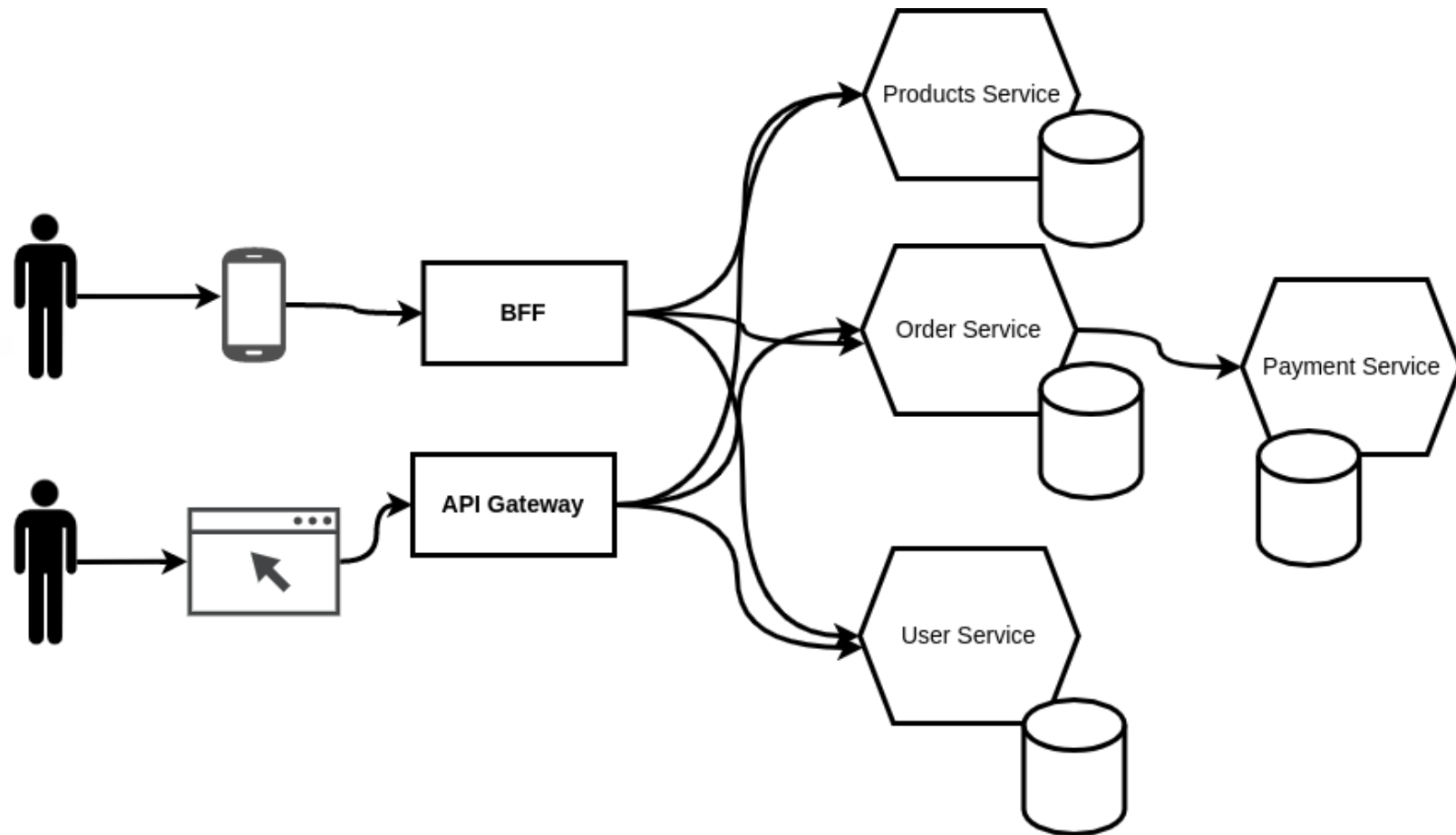


# Decomposição lógica

- É utilizada em conjunto com outra maneira de escalabilidade, através da qual é criado um critério lógico dentro do roteador de requisições



# *A nova arquitetura do escambo.io (provisória)*



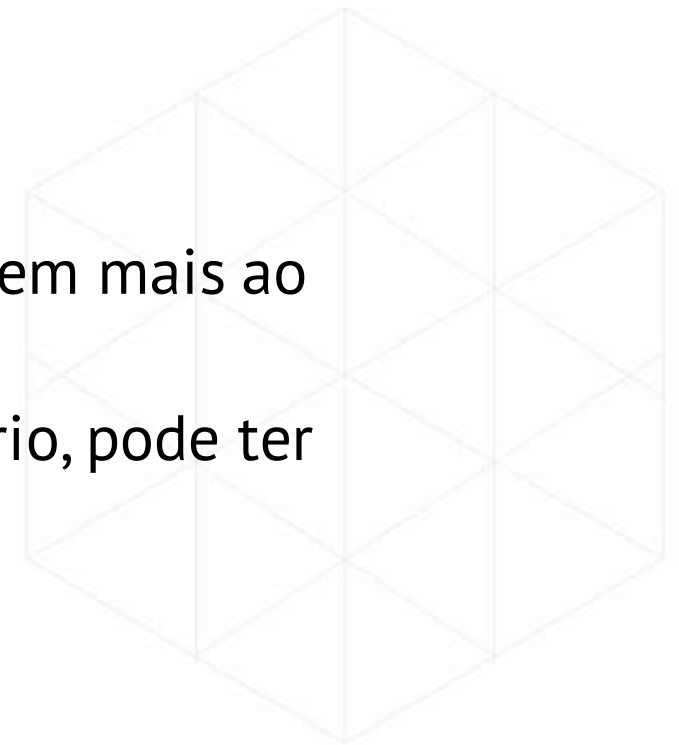
# ***Benefícios do uso de microsserviços***

- CI/CD e consequentemente, automação.
- Divisão de trabalho em times mais concisos.
- Baixa dependência entre os desenvolvimentos.
- Simplicidade nos testes automatizados.
- Base de código pequena, o que permite entregas bem mais ágeis.
- Serviços muito mais escaláveis.



# ***Benefícios do uso de microsserviços***

- Serviços consumindo hardware compatível com seu uso.
- Melhor recuperação de falhas (*memory leak*).
- Facilidade na adoção de novas tecnologias que se adequem mais ao propósito do serviço, introduzindo um baixo risco.
- Microsserviços são independentes (Banco de dados próprio, pode ter tecnologias/linguagens próprias).





# ***Desvantagens do uso de microsserviços***

- Apesar de todas as suas vantagens, temos limitações, que são muitas.
- O desenho pode ser complicado de se fazer sem saber ainda quais serviços serão disponibilizados.
- Caso os serviços sejam decompostos de maneira errada, temos o monólito distribuído.
- São complexos de se desenvolver e devem ter uma boa coordenação.
- Alguns conceitos, já amplamente resolvidas no desenvolvimento tradicional, como transações e consistência de dados, precisam de uma nova abordagem.
- Dificuldade em transações distribuídas, por ser complexo garantir que operações que operam em dois ou mais banco de dados sejam atômicas.

# *Linguagem de microserviços*

- <https://microservices.io/patterns/index.html>
- Linguagem para arquitetura de microserviços desenvolvida por Chris Richardson

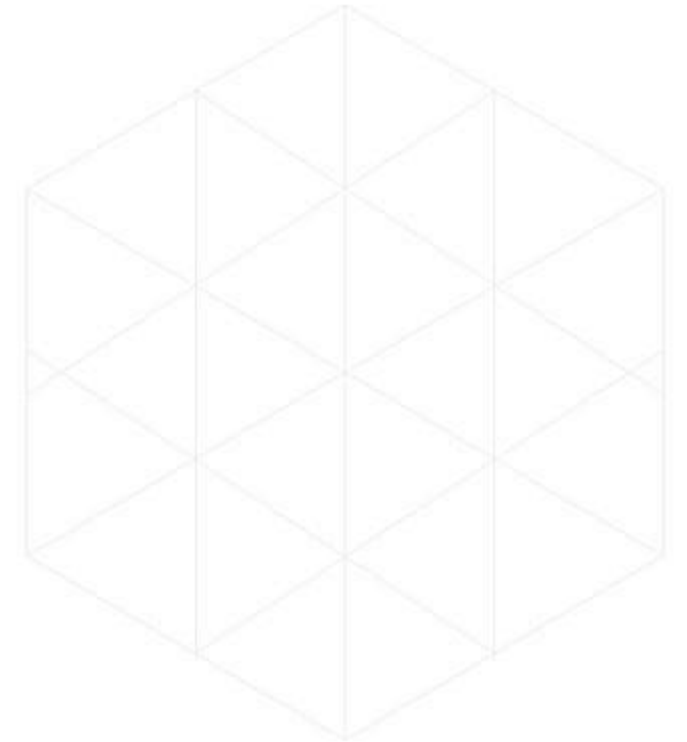


# *Além da tecnologia...*

- Em aplicações grandes e complexas, a adoção de microsserviços pode ser uma boa abordagem, mas, muito além da tecnologia, existem os processos e organização.
- Em um time, o overhead de comunicação é de  $O(n^2)$ . Ou seja, se o time cresce muito, cresce os ruídos da comunicação. (*Chris Richardson, Microservices Patterns With examples in Java, pg. 29*)
- A solução: diminuir os times.
- Times devem ser multidisciplinares e autônomos: desenvolver, testar e entregar sem dependência frequente de outros times

# ***Processos de desenvolvimento***

- Ágil é essencial.
- Muito desejável CI/CD (<https://continuousdelivery.com/> )
  - Entregar software confiável e rápido.
- Métricas
  - Frequência de deploy
  - Lead Time
  - Tempo médio de recuperação
  - Taxa de falhas



# ***Algumas estatísticas***

- A Amazon, ainda em 2014, fazia um deploy em produção a cada 11 segundos.
- Netflix tinha um lead time de 16 minutos.
- Desde que optou por um modelo de entrega contínua, a Digital Media Group da Sony Pictures reduziu sua entrega de softwares de meses minutos.
- <https://netflixtechblog.com/how-we-build-code-at-netflix-c5d9bd727f15>
- <https://www.youtube.com/watch?v=dxk8b9rSKOo>
- <https://www.prolifics.co.uk/which-companies-are-using-devops/>

# ***Para saber mais***

- <https://microservices.io/>
- <https://continuousdelivery.com/>
- <https://microservices.io/patterns/index.html>
- <https://netflixtechblog.com/how-we-build-code-at-netflix-c5d9bd727f15>
- <https://www.youtube.com/watch?v=dxk8b9rSKOo>
- <https://www.prolifics.co.uk/which-companies-are-using-devops/>



# OBRIGADO!

## **Centro**

Rua Formosa, 367 - 29º andar Centro, São Paulo - SP, 01049-000

## **Alphaville**

Avenida Ipanema, 165 - Conj. 113/114 Alphaville, São Paulo - SP, 06472-002

**+55 (11) 3358-7700**

[contact@7comm.com.br](mailto:contact@7comm.com.br)

**7comm**  
Serviços e Soluções em TI