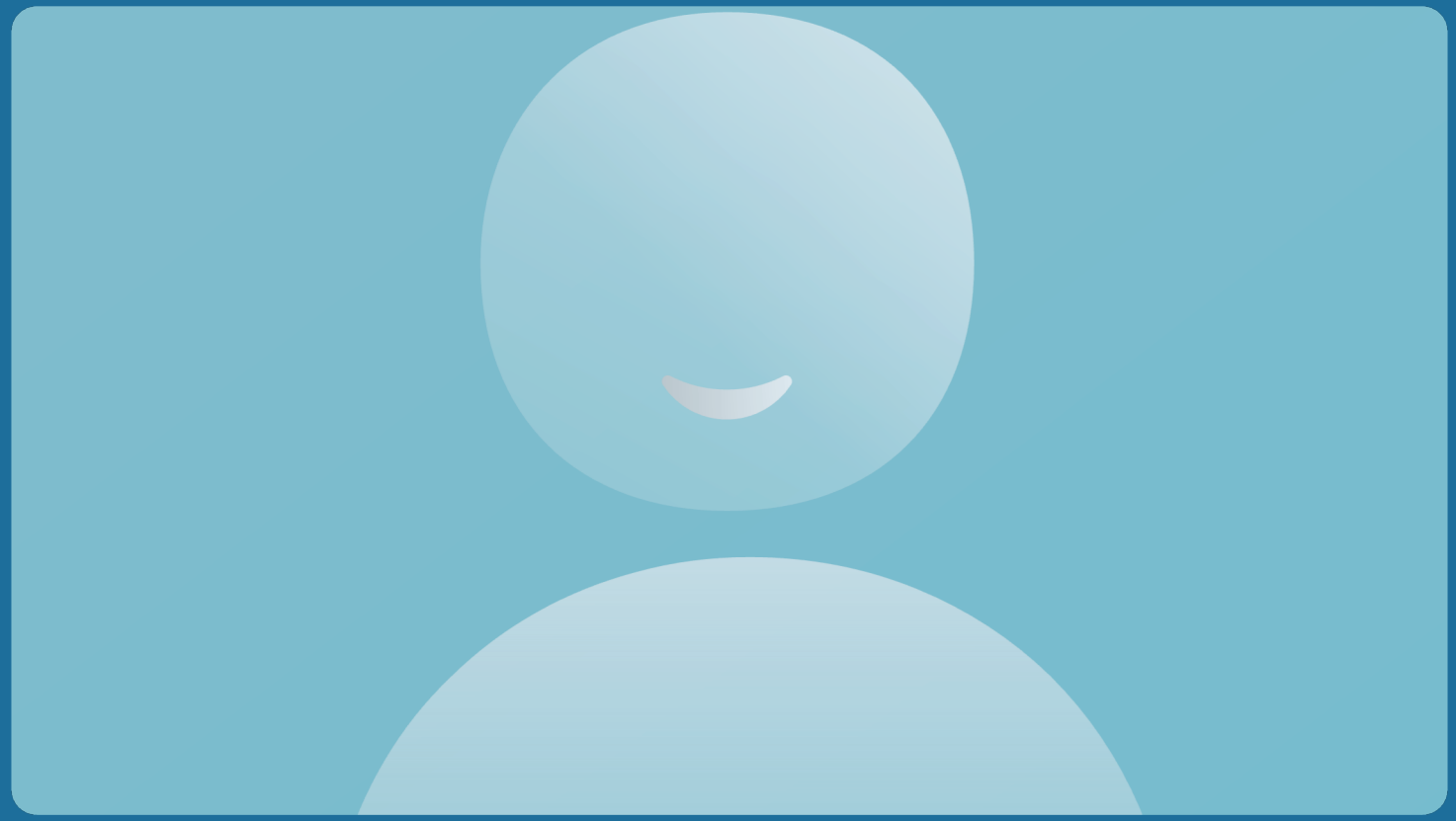
APIs e Web Services

PUC Minas



Prof. Rommel Vieira Carneiro

Tópicos

- Unidade 1 - Introdução
  - Introdução ao Mundo das APIs
  - Protocolo HTTP
- Unidade 2 - Web APIs
  - Aspectos arquiteturais de APIs
  - Estilos: SOAP, REST, GraphQL, Webhooks e Websockets
- Unidade 3 - Segurança em APIs
  - Fundamentos de Segurança
  - Autenticação no Protocolo HTTP
  - Tecnologias e Frameworks: JWT, Oauth e CORS
- Unidade 4 - Gerenciamento de APIs
  - API Lifecycle Management (ALM)
  - Arquitetura de Microserviços e API Gateway
  - Boas práticas no desenvolvimento de APIs

Informações Gerais

Ementa

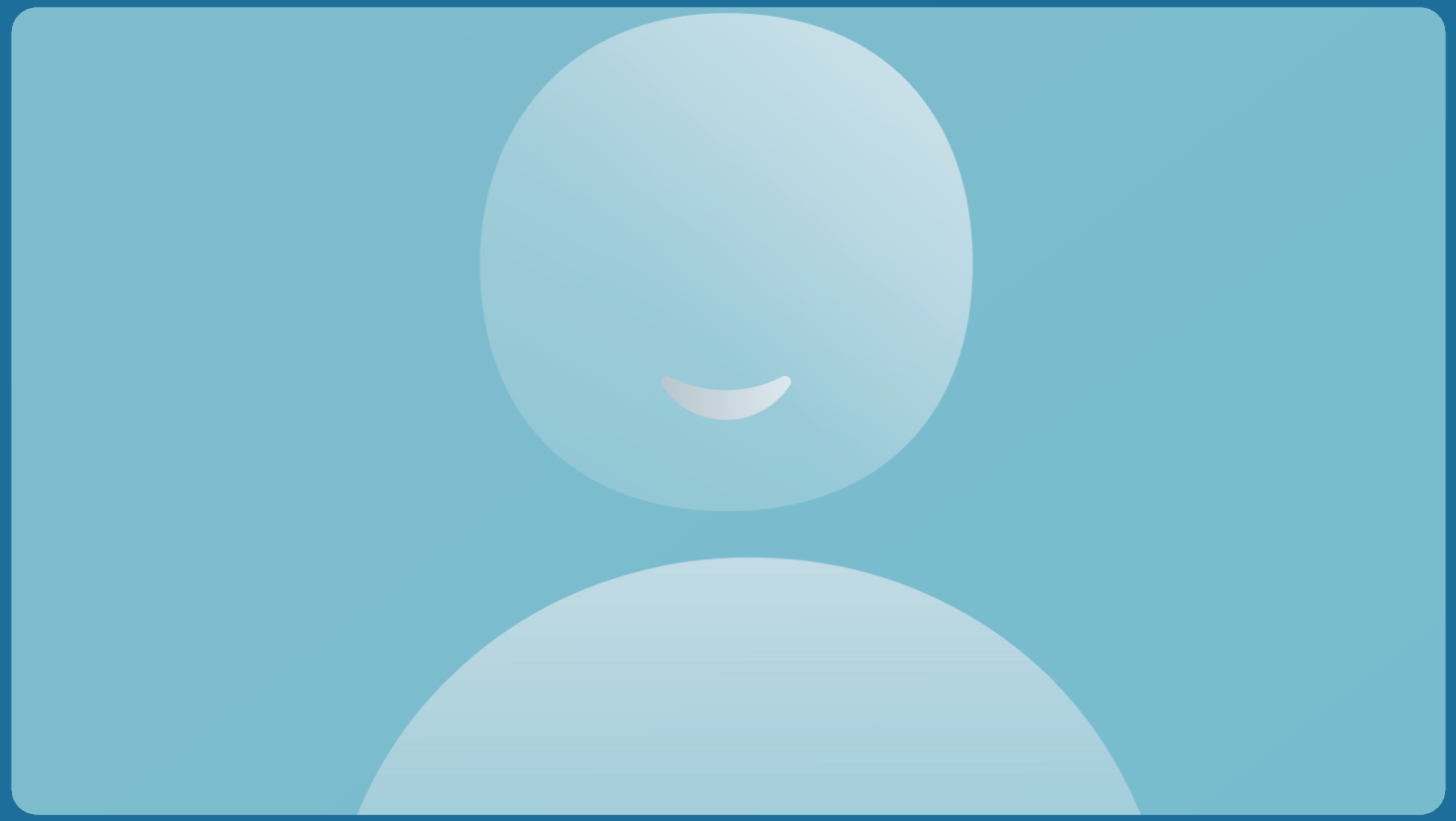
Evolução das APIs. Gestão do ciclo de vida das APIs. Melhores práticas no projeto de API ferramentas para documentação de APIs. Mecanismos de segurança: autenticação, a vulnerabilidades. Abordagens arquiteturais de APIs: RESTful, GraphQL, WebSockets, WebF Streaming.

Bibliografia

- Designing Web APIs. Brenda Jin, Saurabh Sahni, Amir Shevat. O'Reilly Media, Inc. (2018)
- Mastering API Architecture. James Gough, Daniel Bryant, Matthew Auburn. O'Reilly Media, Inc. (2022)
- API Design Patterns. John J. (JJ) Geewax. Manning Publications (2021)
- Designing APIs with Swagger and OpenAPI. Josh Ponelat, Lukas Rosenstock. Manning Publications (2022)
- Microservice APIs. Jose Haro. Manning Publications (2023)

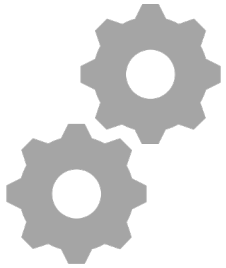
Gerenciamento de APIs

PUC Minas



Prof. Rommel Vieira Carneiro

Gerenciamento de APIs



Projeto e Desenvolvimento de APIs

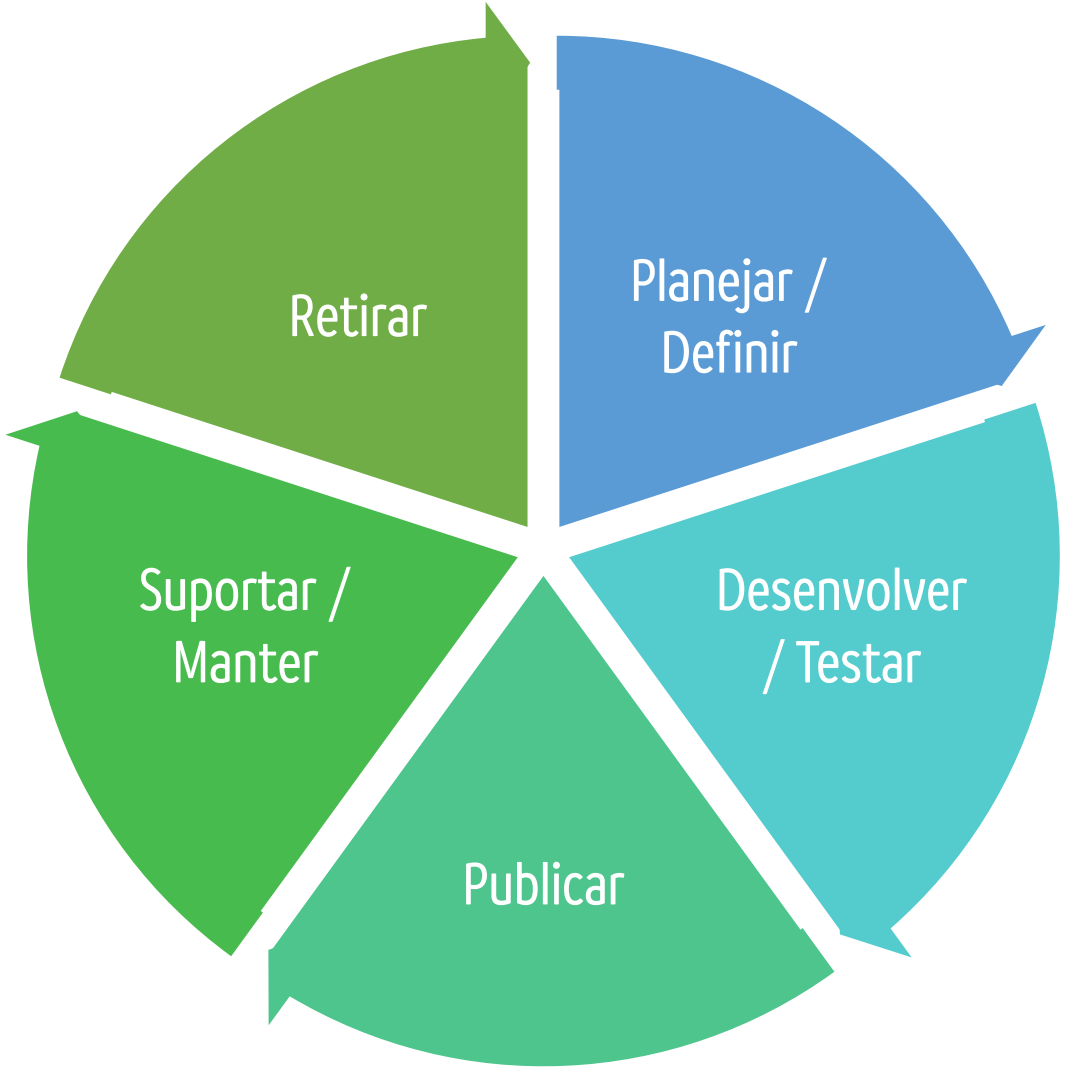
Plataforma de API Gateway

Gerenciamento do ciclo de vida de APIs

Projeto e Desenvolvimento de APIs

- Design de APIs
  - Definição de endpoints, recursos e operações
  - Uso de padrões e estilos arquiteturais
- Especificação e Documentação de APIs
  - Abrangência e clareza na documentação de APIs
  - Uso de ferramentas e padrões de documentação (OpenAPI, WSDL)
- Projeto da segurança
  - Uso de padrões consistentes de autenticação e autorização (OAuth, JWT, CORS)
  - Prevenção de ameaças (SQL Injection, XSS, CSRF)
- Desenvolvimento e Testes com foco em APIs
  - Abordagem API First
  - Testes unitários e testes de integração
  - Ferramentas de produtividade para APIs

API Lifecycle Management



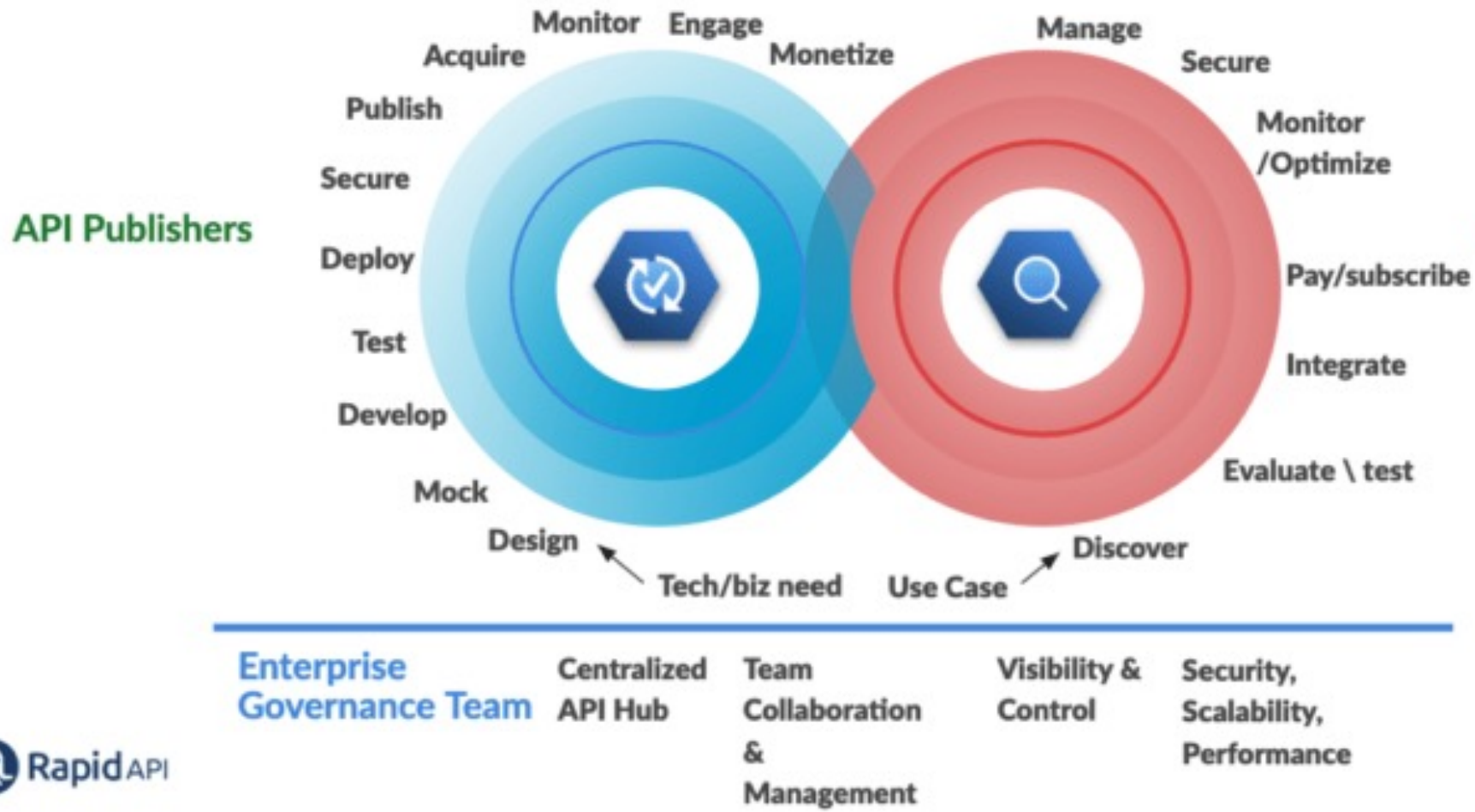
Life Cycle API Management

- Planejar / Definir
- Desenvolver / Testar
- Publicar
- Suportar / Manter
- Retirar

Fontes:

- What is API Lifecycle Management?
- API and APP Ecosystems

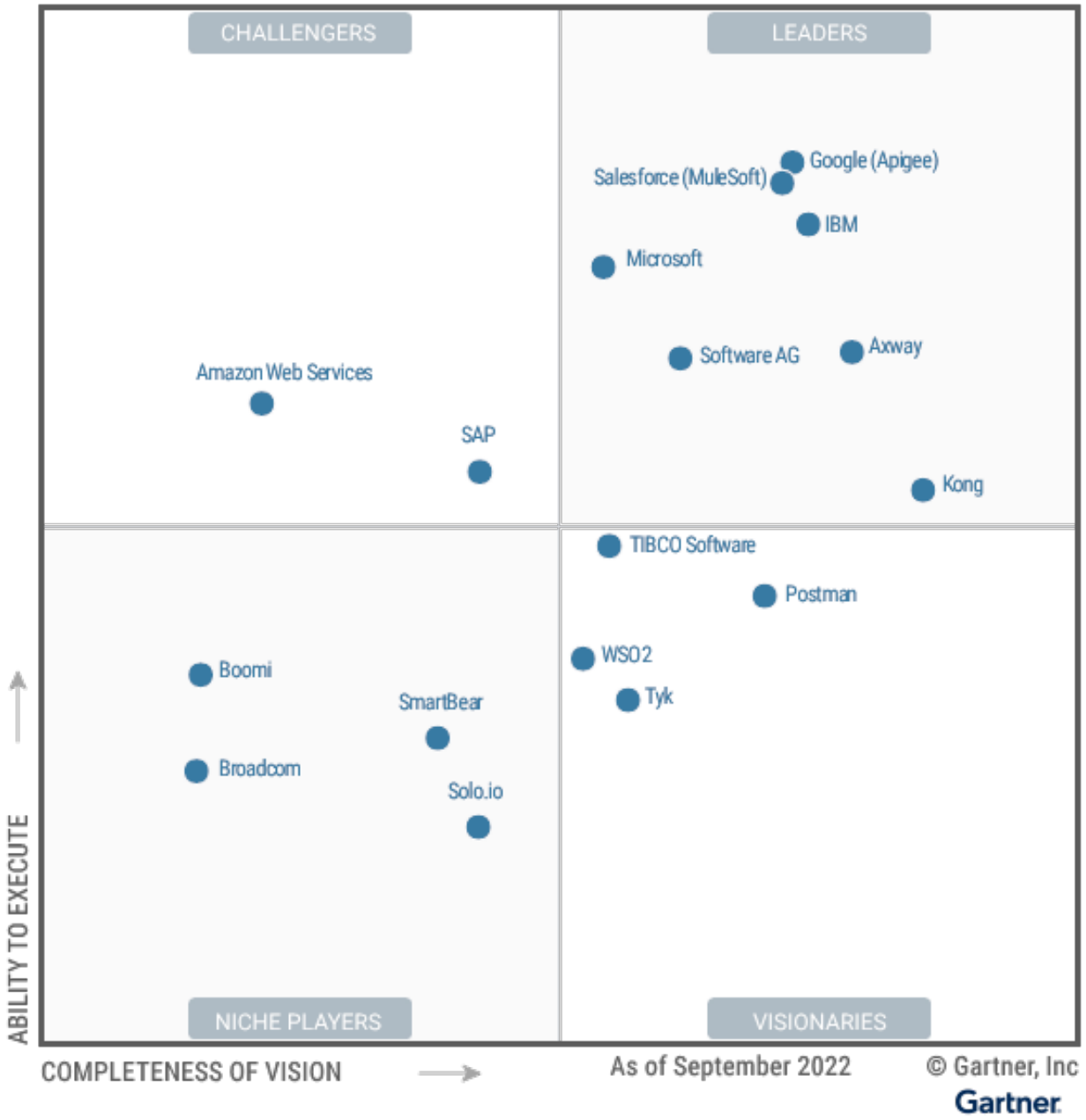
API Lifecycle Management



Fonte: [What is API Lifecycle Management? Design, Develop, Secure, Test, & More]

ALM Tools

- Quadrante Mágico (Gartner)
  - Google Apigee
  - Salesforce MuleSoft
  - IBM API Connect
  - Microsoft Azure API Mgmt
  - Axway
  - Software AG
  - Kong
  - Postman



ALM Tools

- Google apigee
- SmartBear – Swagger | Soap UI
- Kong
- apiary (Oracle)
- apimatic
- 3scale (RedHat)
- Postman

apigee

SoapUI Supported by SMARTBEAR

Kong

apiary.io

swagger

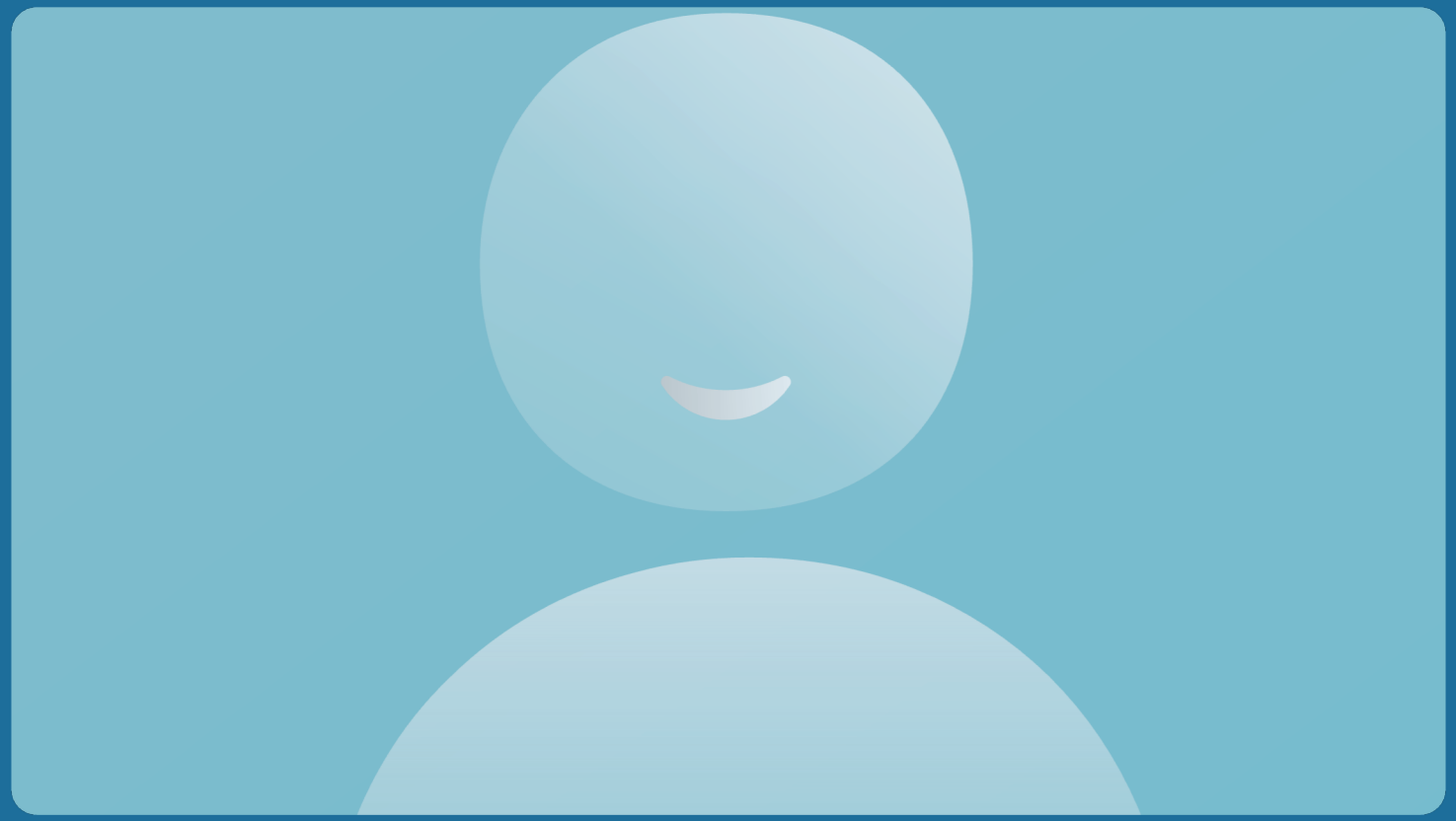
POSTMAN

APIMATIC

3scale BY RED HAT

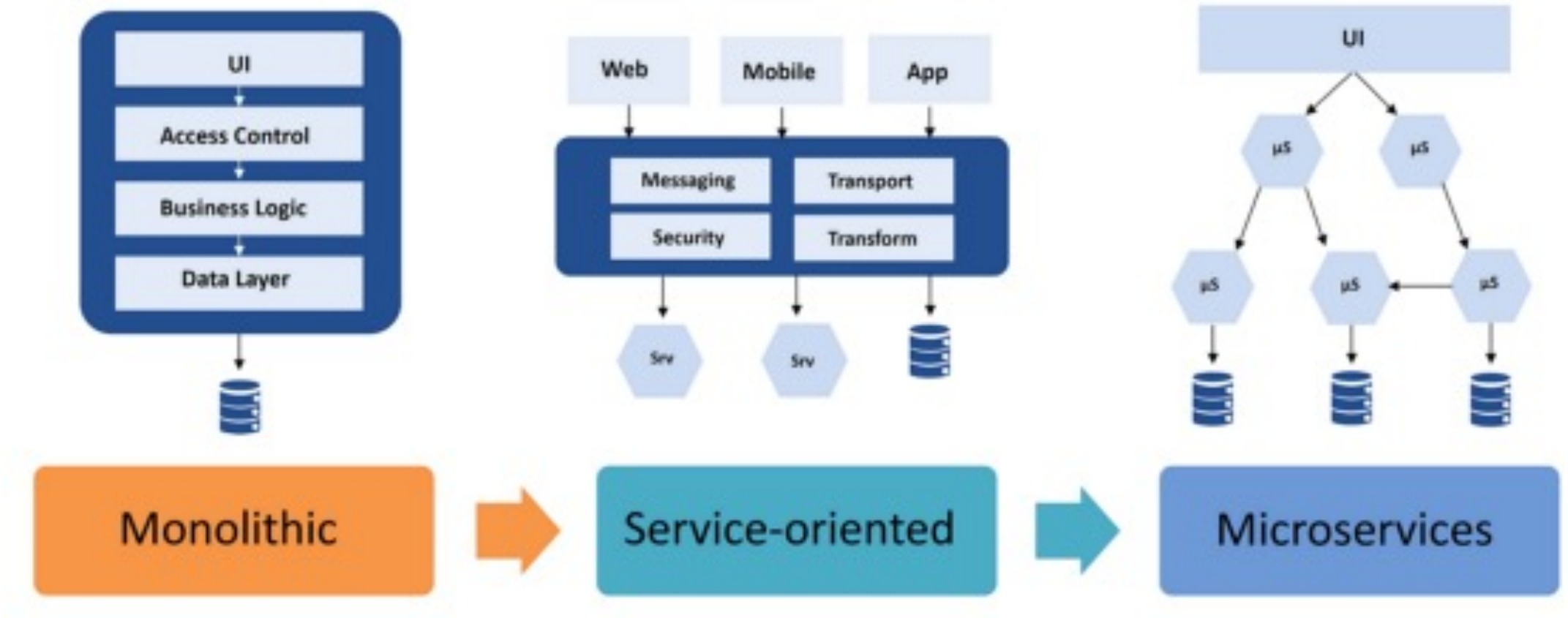
Arquitetura de Microserviços

PUC Minas



Prof. Rommel Vieira Carneiro

Evolução da Arquitetura de Software



Fonte: Software architecture evolution

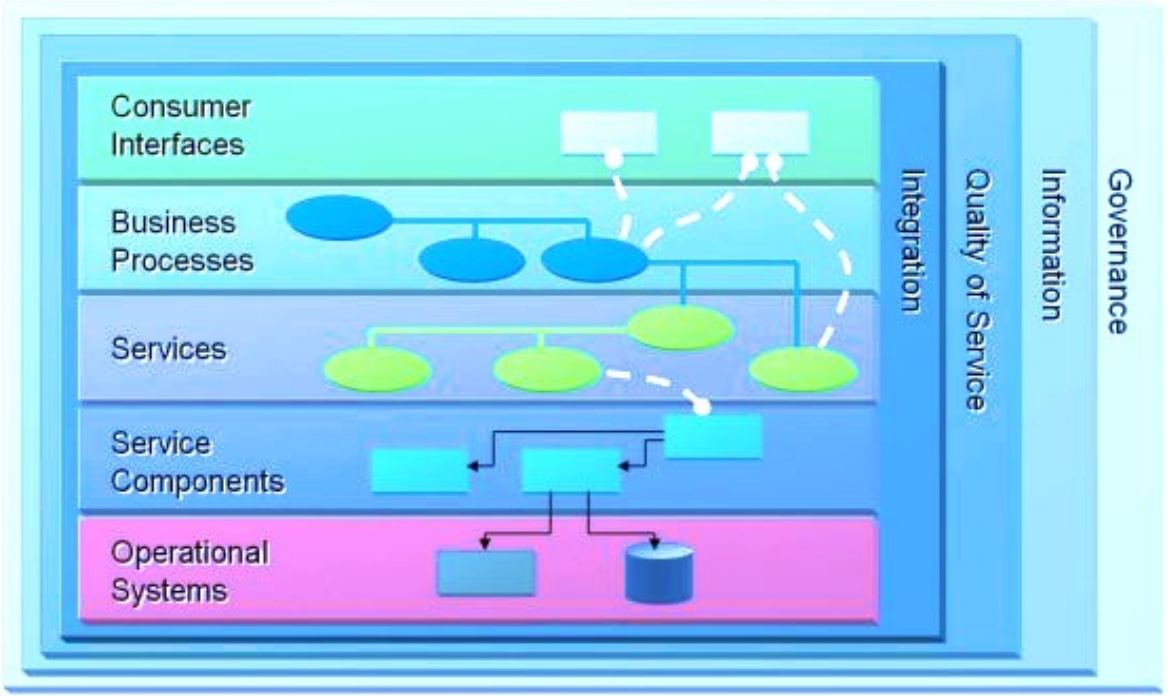
Arquitetura Orientada a Serviços (SOA)

Estilo arquitetural de software baseado na estruturação das aplicações de negócios por meio de um conjunto de serviços.

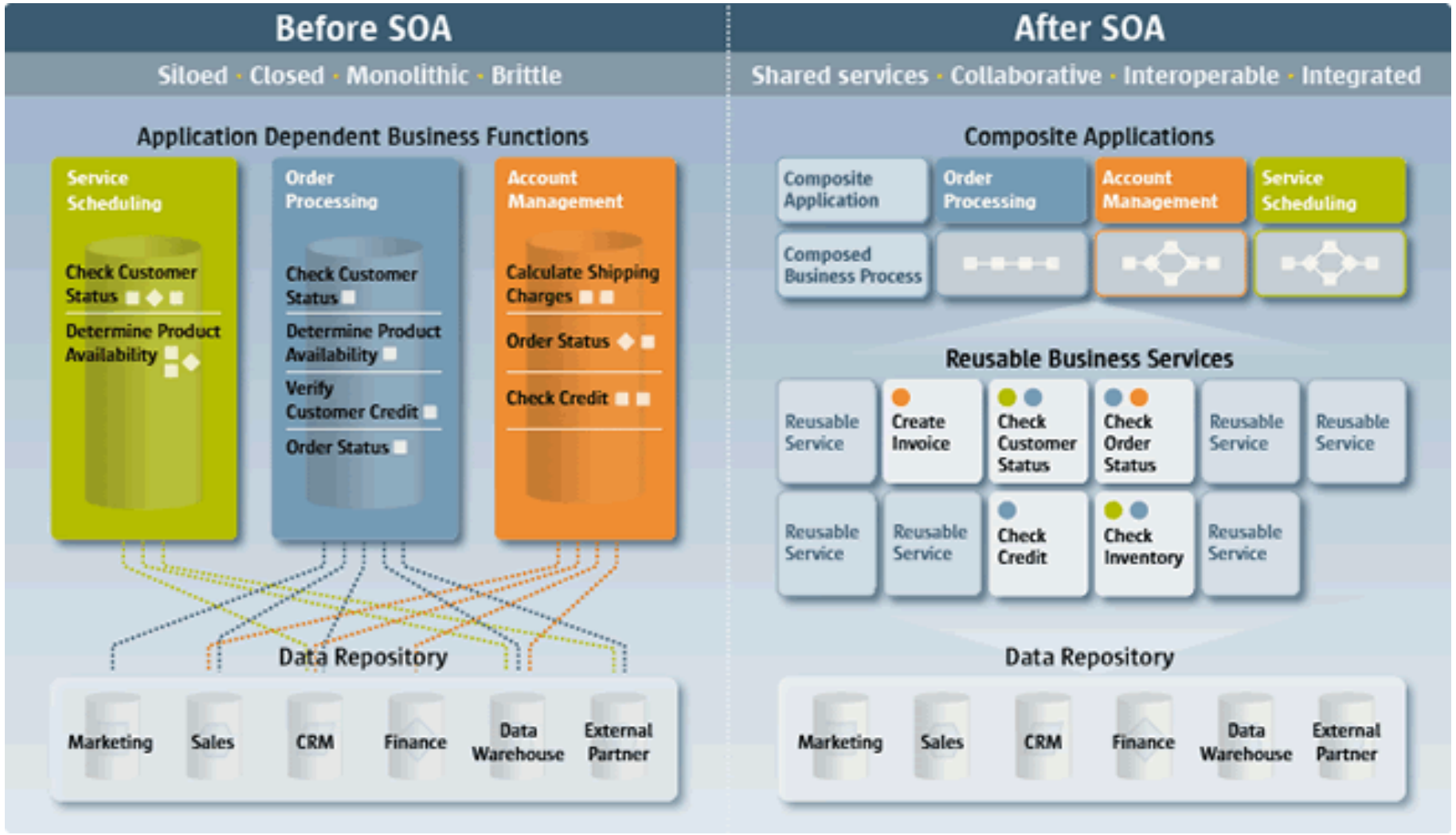
Cada serviço provê uma funcionalidade específica do negócio e permite a criação de aplicações distribuídas utilizando diferentes tecnologias e plataformas.

Camadas

- Interface de Usuário
- Processos de Negócios
- Serviços
- Componentes de Serviços
- Sistemas Operacionais



Arquitetura Orientada a Serviços (SOA)

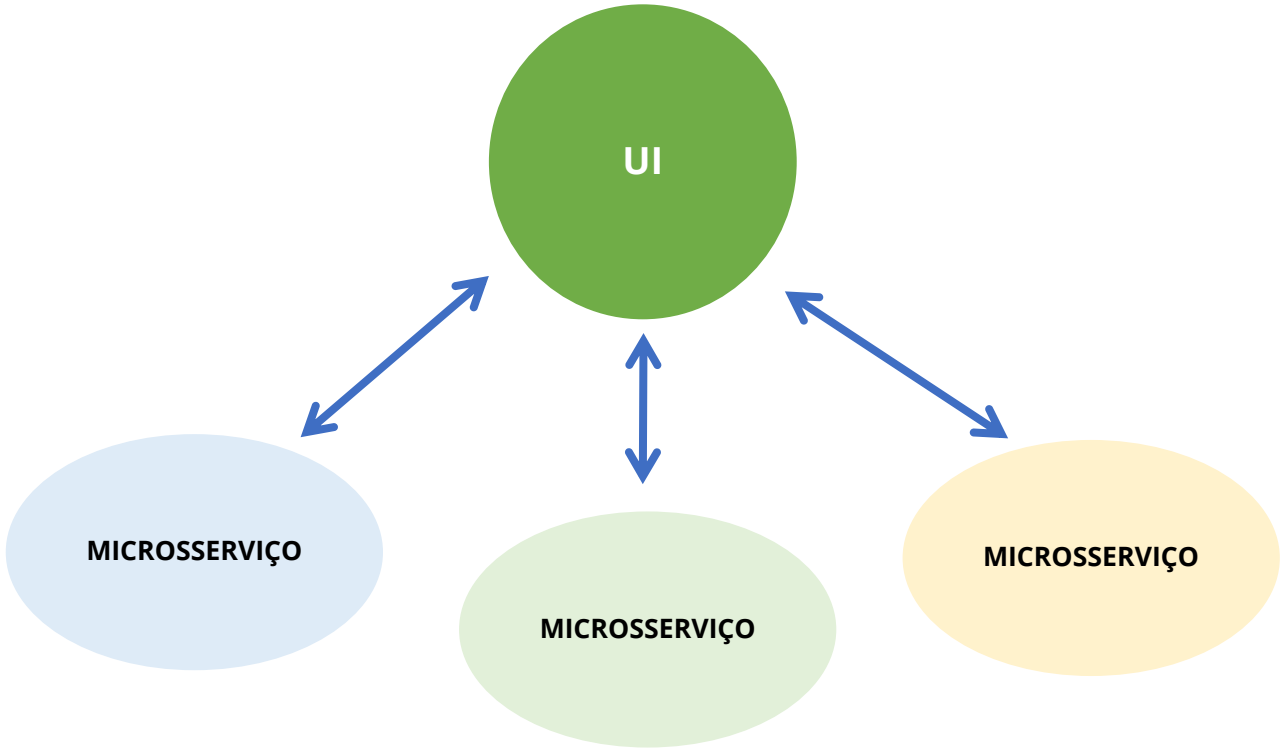


Fonte: [Service oriented architecture (SOA) deserves service oriented data]

Microsserviços

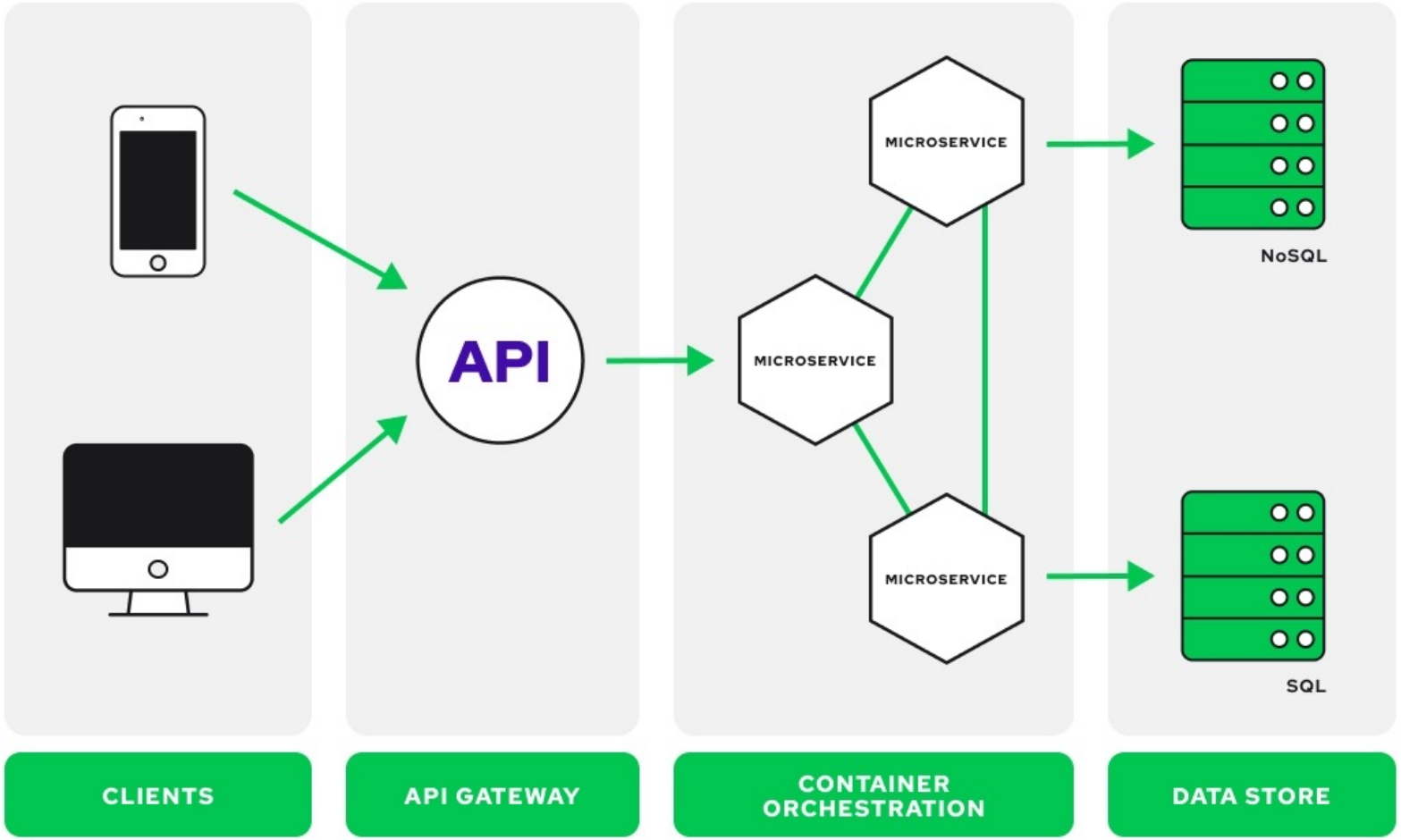
Microsserviços é uma abordagem arquitetural que define um conjunto de serviços independentes com escopo limitado.

Monolito



Microsserviços

Microsserviços



Fonte: Microservices in eCommerce Explained

Microsserviços

Aspecto	Monolito	Microsserviços
Deployment	Deploy simples e rápido do sistema inteiro	Requer recursos distintos tornando a orquestração mais complexa
Escalabilidade	Difícil de manter e aplicar mudanças.	Cada elemento pode ser escalado independente sem downtime
Agilidade	Dificuldade de adotar novas tecnologias	Adota novas tecnologias resolvendo problemas de negócios
Resiliência	Um erro pode afetar o sistema inteiro	Uma falha em um serviço não afeta os demais
Teste	Teste fim a fim	Componentes requerem testes individuais
Segurança	Comunicação dentro de uma unidade única	Comunicação interprocessos requer API gateways e maior complexidade
Desenvolvimento	Dificuldade de distribuir os esforços dos times em função da estrutura	Os times podem trabalhar de forma independente em cada componente

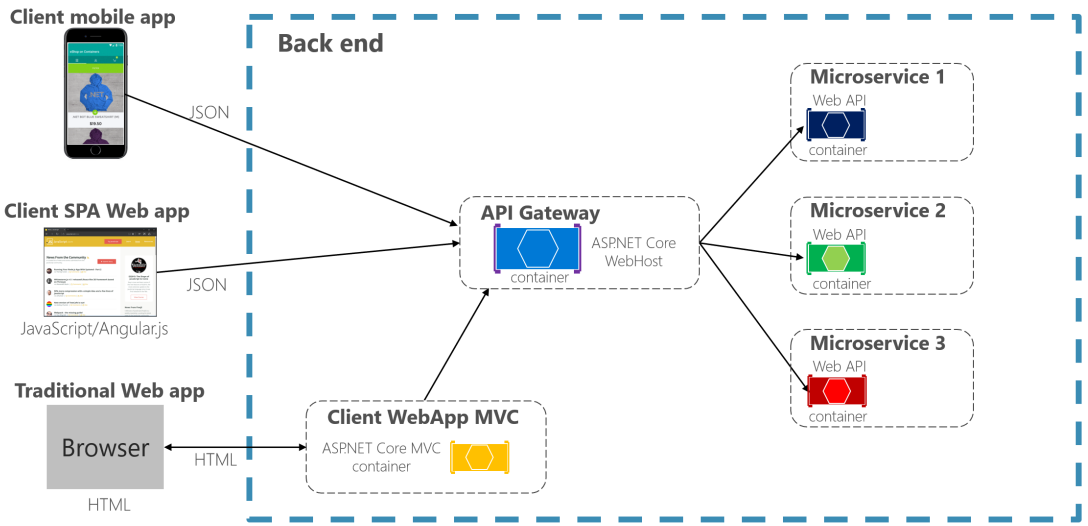
Fonte: Monolithic Architecture vs. Microservices

API Gateway

API Gateway é um *Design Pattern* que oferece um único ponto de entrada para APIs numa arquitetura de microsserviços, atuando como um proxy reverso que trata solicitações de clientes e encaminha ao serviço apropriado.

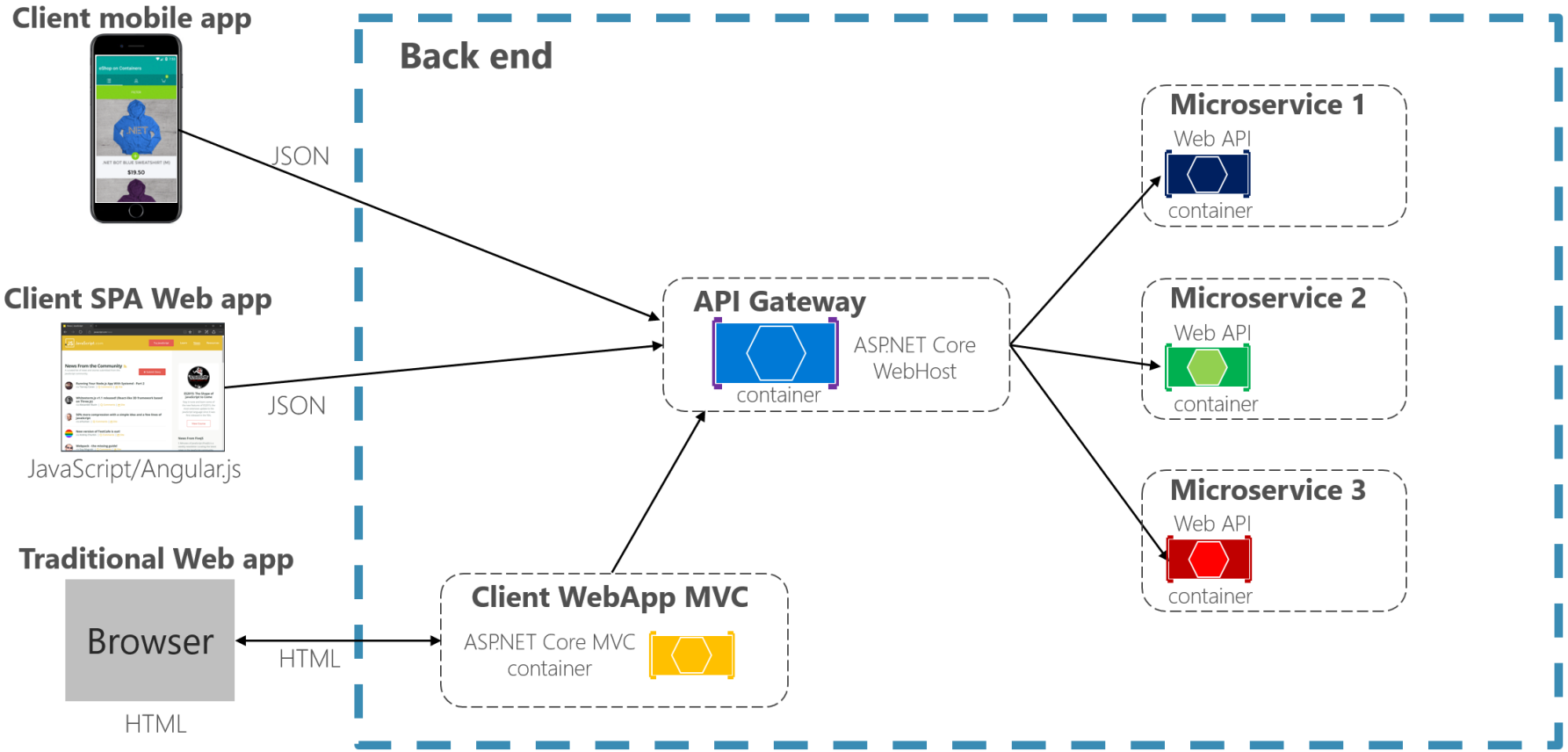
Funcionalidades de um **API Gateway**:

- Cache de Respostas
- Balanceamento de carga
- Roteamento de requisições
- Gerenciamento da autenticação e autorização
- Monitoramento e análise de dados
- Documentação de APIs



Fonte: [What is API Gateway Design Pattern in Microservices?](#) (Java67)

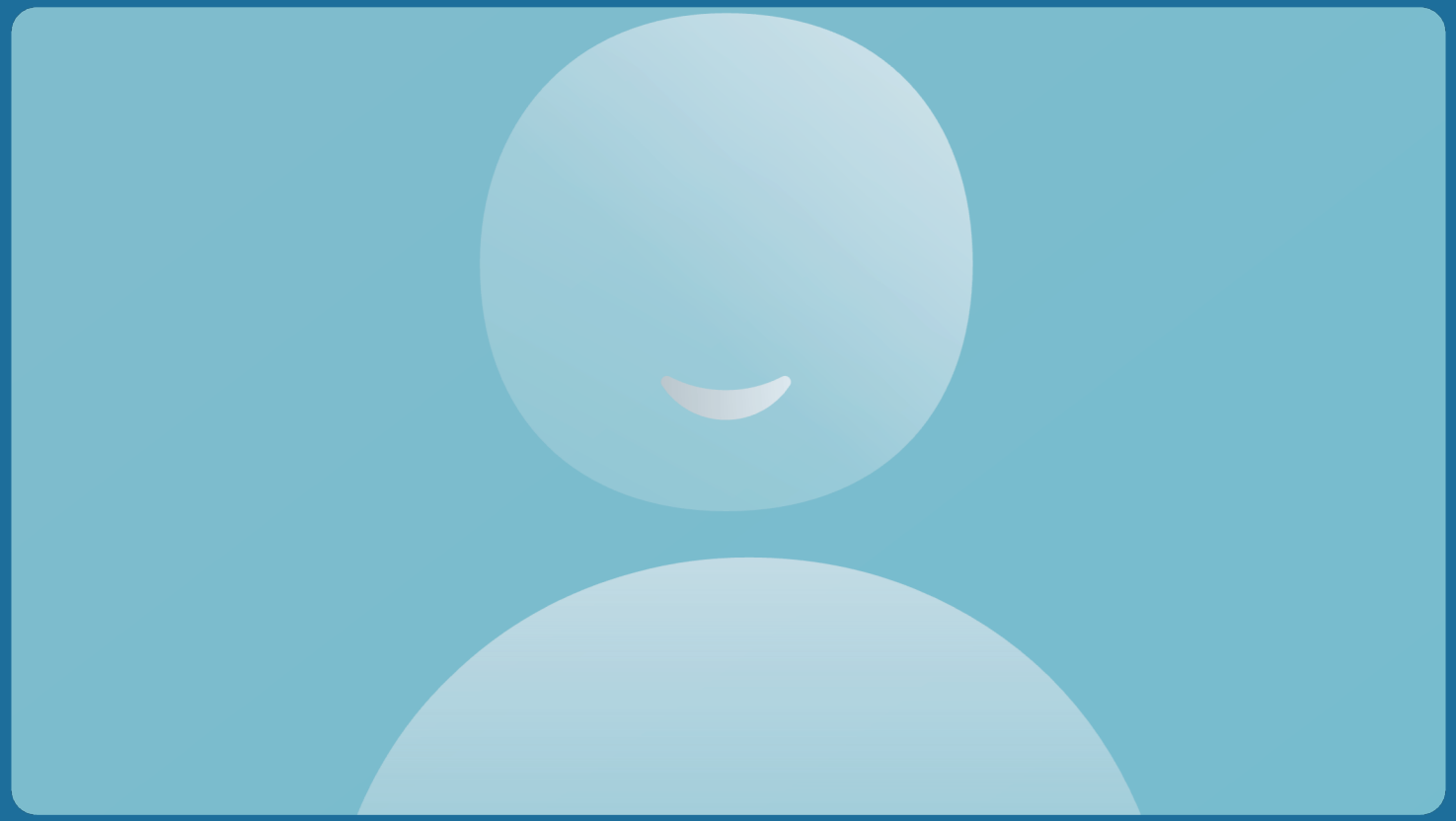
API Gateway



Fonte: [What is API Gateway Design Pattern in Microservices?](#) (Java67)

Estratégia API First

PUC Minas



Prof. Rommel Vieira Carneiro

Estratégia API First

A estratégia **API First** está no projeto e criação da API antes do desenvolvimento do sistema, estabelecendo um contrato prévio entre equipes de backend e frontend.

Vantagens

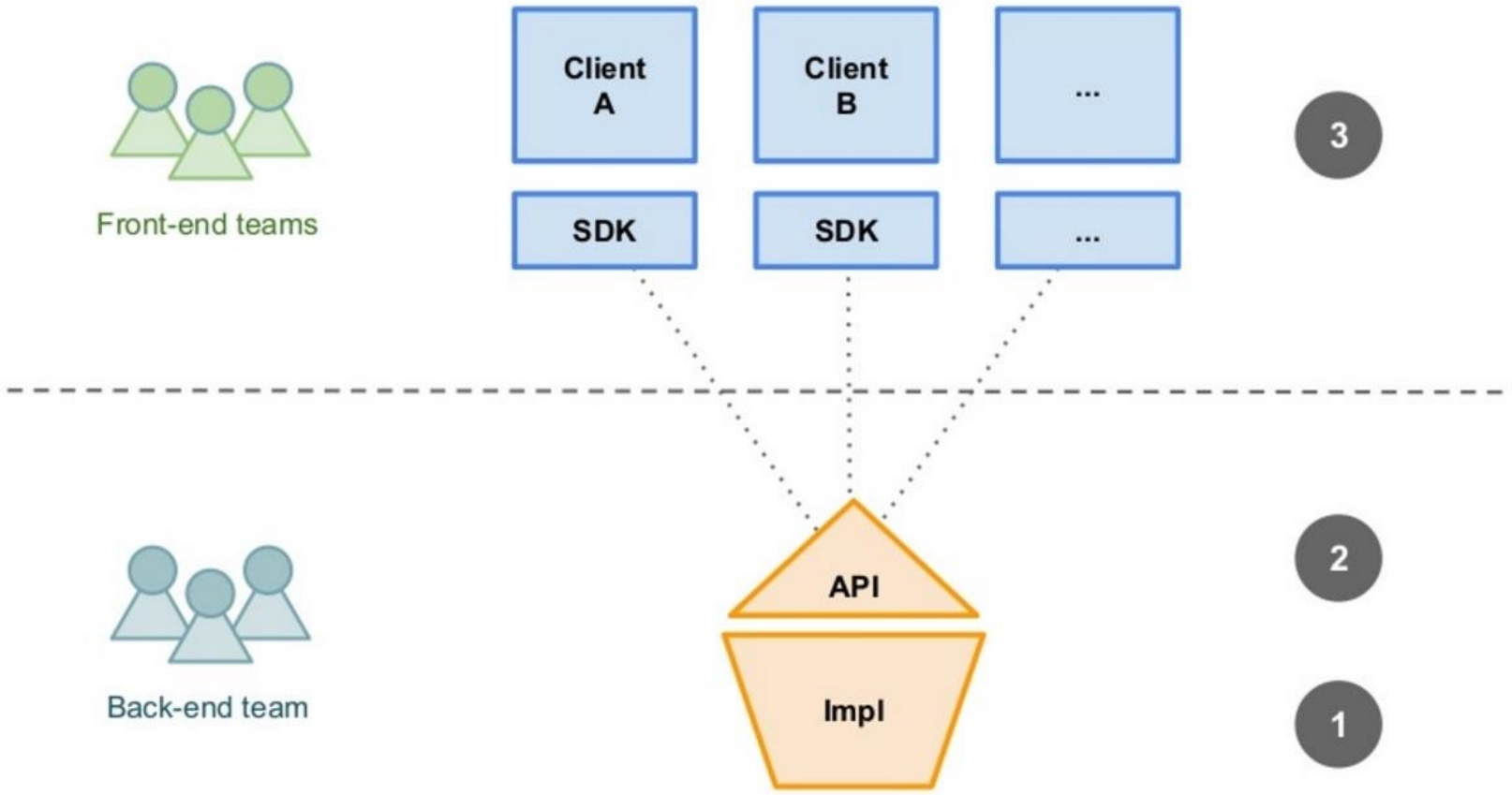
- Separação de preocupações (frontend *vs* backend)
- Redução de interdependências
- Expansão facilitada para diferentes dispositivos
- Rapidez no desenvolvimento e entrega – Times trabalhando em paralelo

Estratégia API First

Modelo Tradicional

Processo Síncrono

1. Equipe de backend constrói a base do sistema
2. Equipe de backend disponibiliza a API
3. Equipe de frontend começa a trabalhar



Fonte: Uma abordagem de desenvolvimento API-First

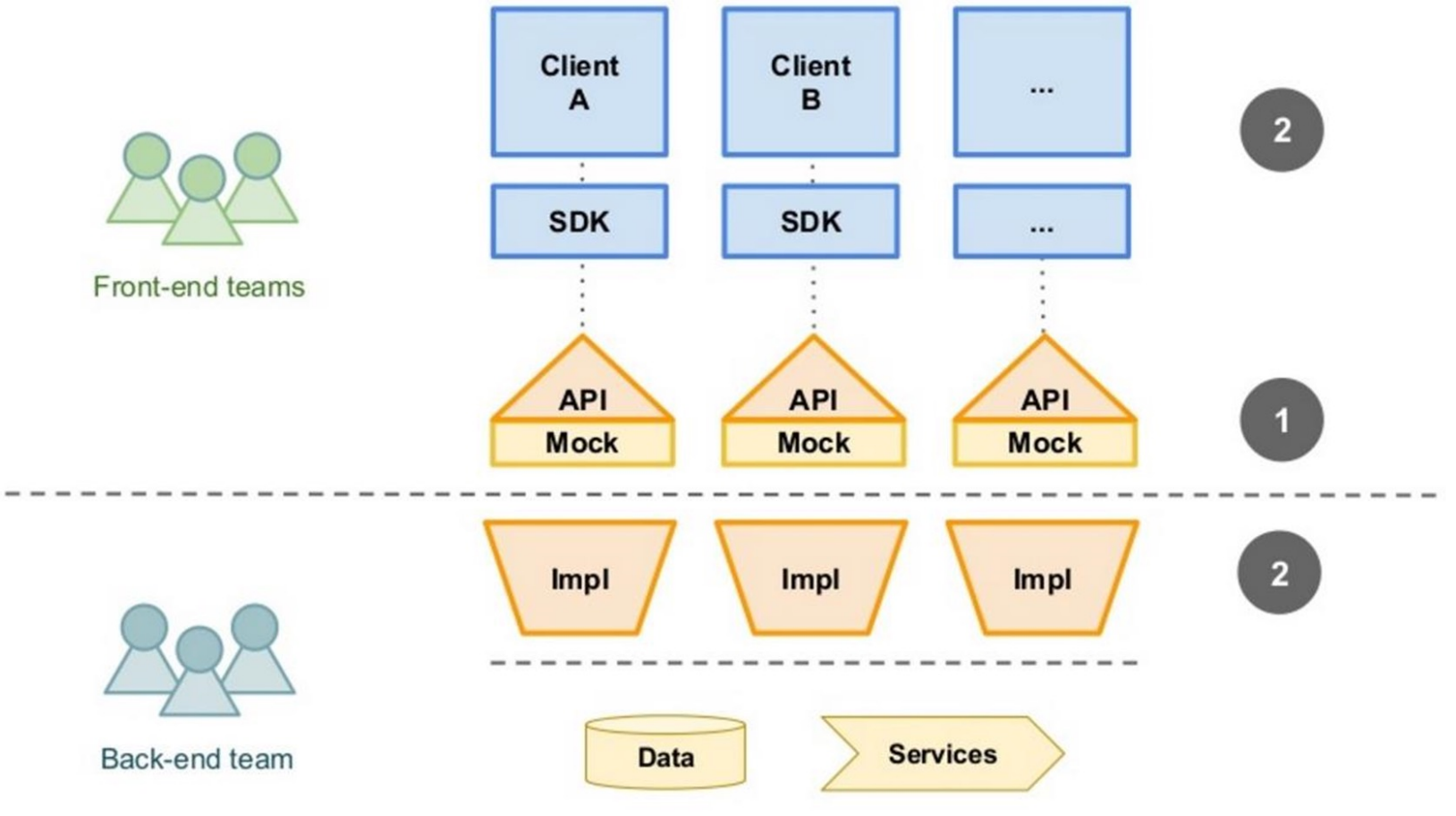
Estratégia API First

Abordagem API First

Processo Paralelo

1. Equipes definem uma API comum e dados simulados (Mock)

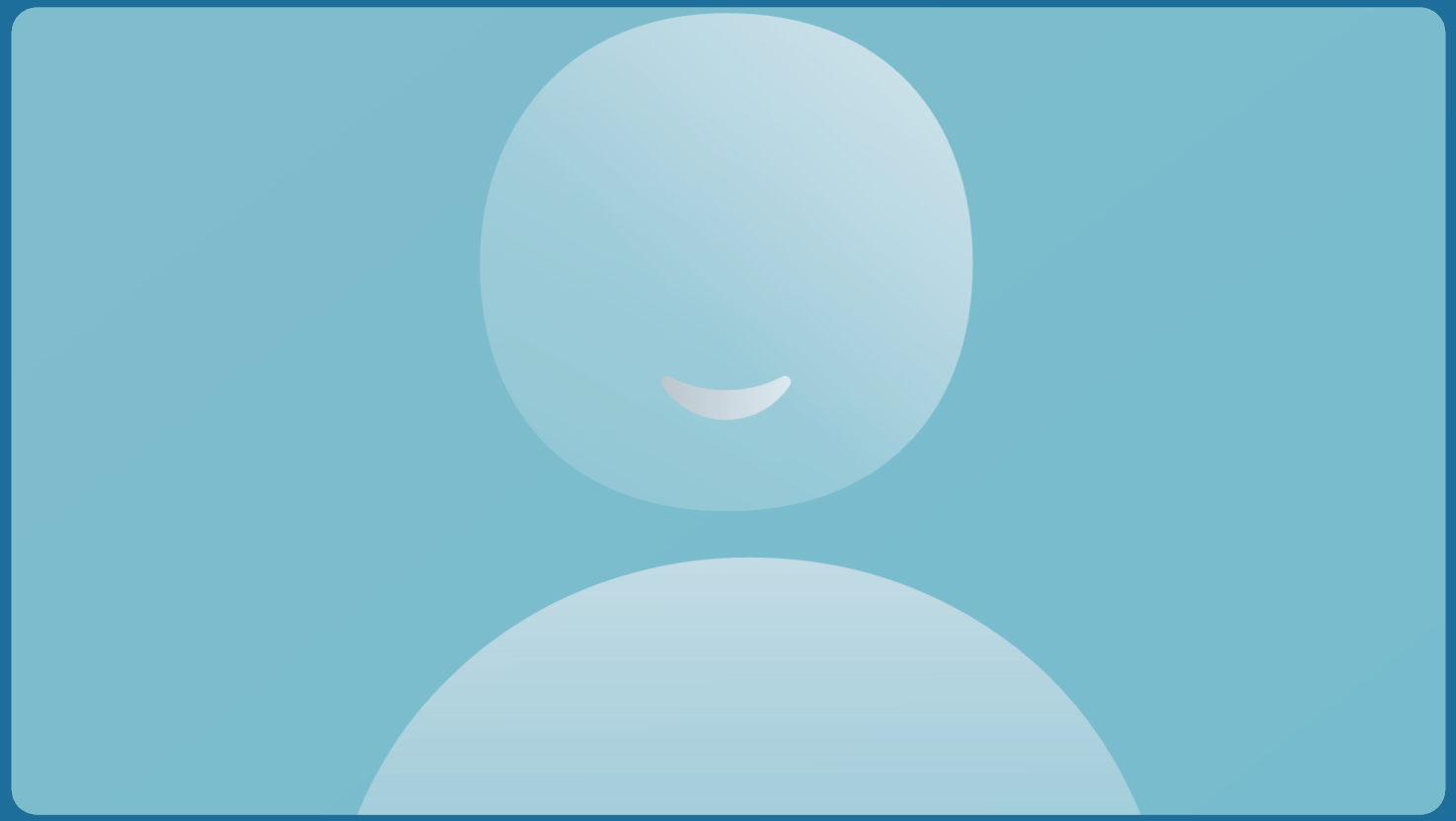
2. Início das tarefas de desenvolvimento em frontend e backend



Fonte: Uma abordagem de desenvolvimento API-First - https://blog.mandic.com.br/artigos/uma-abordagem-de-desenvolvimento-api-first/

Versionamento de APIs

PUC Minas



Prof. Rommel Vieira Carneiro

Versionamento de APIs

A lógica de negócio de uma aplicação ou os recursos disponíveis podem sofrer alterações com o tempo. Isso acaba refletindo na API desta aplicação.

Para garantir que os clientes tratem estas mudanças adequadamente, é fundamental controlar o número da versão da API e refletir isso na forma de acesso garantindo o uso correto das funcionalidades disponíveis.

Diante disso, existem duas questões que precisamos entender

1. Como numerar novas versões?
2. Como refletir os novos números na forma de acesso?

Versionamento de APIs

Para especificar a versão de uma API é comum utilizar a URL de acesso ou, ainda, cabeçalhos HTTP.

Via URL

- Domínio do site → `https://api-v1.servidor.com/recurso`
- Caminho do recurso → `https://api.servidor.com/v1/recurso`
- Parâmetro na query string → `https://api.servidor.com/recurso?v=1`

Via cabeçalho HTTP

- Cabeçalho Accept → `Accept: application/json;versao=1`
- Cabeçalho Customizado → `Accept-Version: v1`

Versionamento de APIs

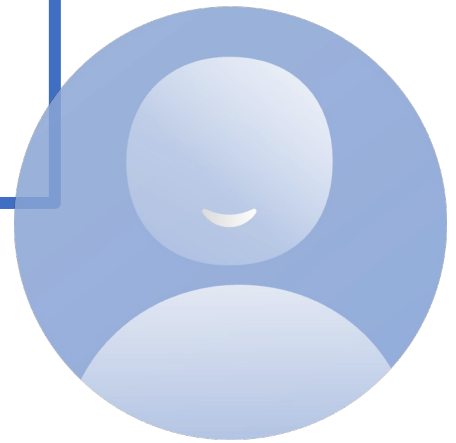
Versionamento Semântico

```
versão: "4 . 16 . 3"
```

Versão Maior
- Possível quebra de compatibilidade

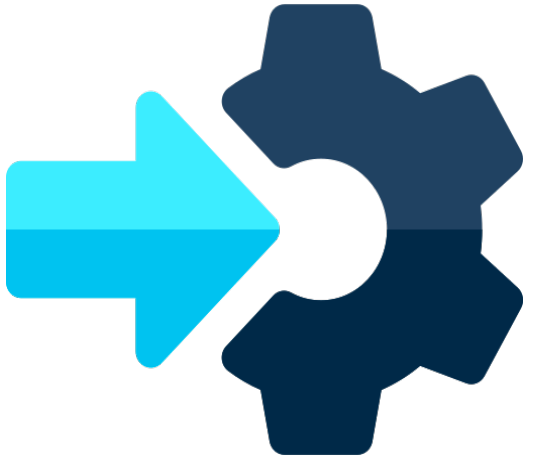
Versão Menor
- Compatibilidade retroativa
- Funcionalidade depreciada, mas funcional
- Refatoração interna

Patch
- Correção de bugs



Fonte: Semantic Versioning - https://semver.org

Obrigado!



APIs & Web Services

Prof. Rommel Vieira Carneiro

in /rommelcarneiro

http://rommelcarneiro.me/