FIAP ON

SERVICES ARCHITECTURE,
API E MOBILE ARCHITECTURE

API GATEWAY & SERVICE MESH

CHRISTIANE DE PAULA REIS

4

LISTA DE FIGURAS

SUMÁRIO

4 API GATEWAY & SERVICE MESH

Desenvolver uma aplicação corporativa do lado do servidor significa que ela deve suportar uma variedade de clientes diferentes, como navegadores móveis, navegadores de desktop e aplicativos móveis nativos. A aplicação também pode ser consumida por terceiros através de microsserviços ou através de APIs (*Application Programming Interface*). E pode ainda se integrar a outros aplicativos por meio de serviços da Web ou de um intermediário de mensagens. Para o bom funcionamento de toda essa cadeia de comunicação, padrões devem ser seguidos e também deve haver um bom gerenciamento da plataforma e dos serviços.

Neste capítulo, você vai aprender sobre as APIs, o Gateway de API e Service Mesh, assim como suas funcionalidades e aplicações na Arquitetura de Microsserviços (*MSA – Microservice Arquitecture*).

4.1 Desafios dos Sistemas Distribuídos

Estamos vivenciando uma realidade em que os Sistemas Distribuídos estão crescendo em escala de forma vertiginosa; e nesse cenário, os microsserviços são um assunto em evidência. Em decorrência, temos que lidar a cada dia com problemas que antes não havia em sistemas centralizados.

> “Sistemas distribuídos são uma coleção de computadores independentes que aparentam aos usuários como um único e coerente sistema." (TANENBAUM; VAN STEEN, 2006)

Felizmente, o movimento da disseminação de conhecimento sobre os sistemas distribuídos tem contribuído bastante para o surgimento de diferentes tipos de soluções distribuídas em nossa rede de artefatos. Houve também o aparecimento de novas abordagens para a resolução de problemas em nível da infraestrutura, removendo essa responsabilidade do nível da aplicação. Foi, então, que nasceu o *Service Mesh*.

Entretanto, para implementar uma Arquitetura de Microsserviços (MSA) eficiente, é preciso investimento para que os sistemas sejam mais resilientes. Nesse caminho, surgiram as conhecidas APIs, estruturas de códigos de programação que

funcionam como “pontes” entre sistemas de diversos tipos; e também os Gateways de API, responsáveis por centralizar funcionalidades compartilhadas e administrar chamadas para cada microsserviço.

4.2 APIs: o que são?

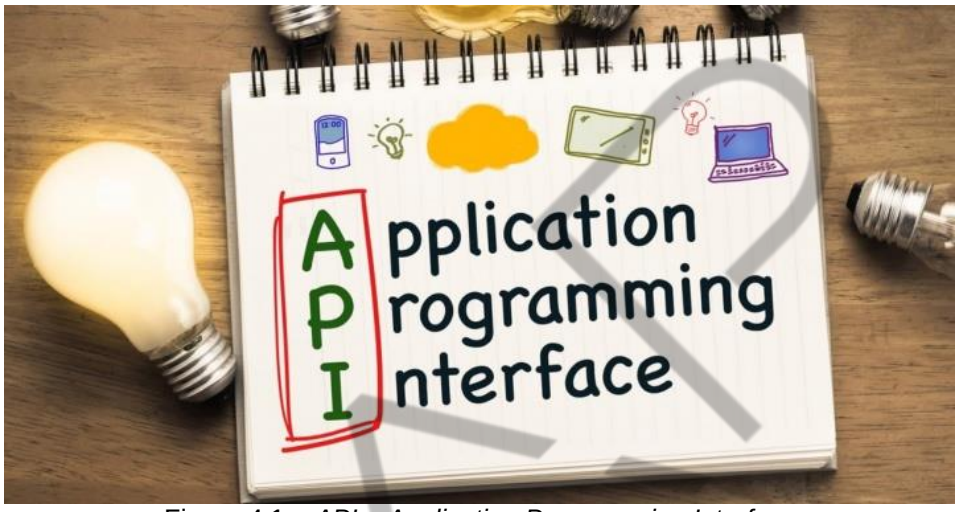

Figura 4.1 – *API – Application Programming Interface*
Fonte: Shutterstock (2019)

De maneira simples, uma Interface de Programação de Aplicativos (APIs) pode ser definida como uma espécie de “ponte” para conectar aplicações com a finalidade de realizar integração e troca de dados. Em termos técnicos, API é uma estrutura de códigos de programação padronizados que permitem a conexão entre sistemas.

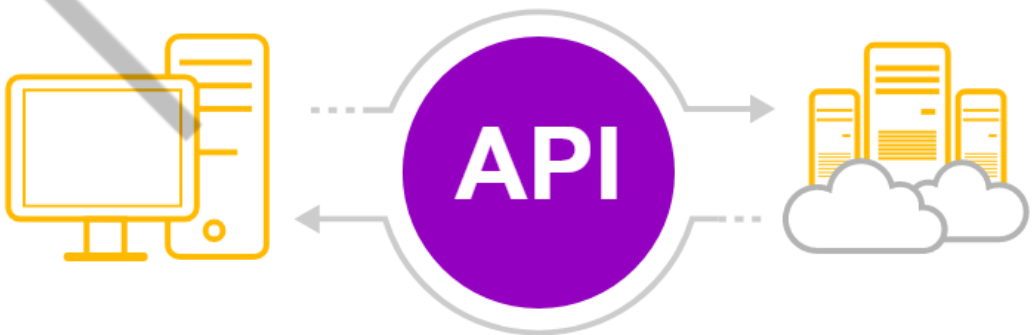

Figura 4.2 – APIs no mundo digital
Fonte: Google Imagens (2019)

O desenvolvimento de uma API passou a ser orientado para o negócio, exigindo execução muito rápida. As APIs se tornaram protagonistas em diversas tarefas que executamos no nosso dia a dia e fazem o mundo digital funcionar. Várias

empresas estão fazendo uso da conectividade conduzida por API, que é uma estratégia de API específica para expor seus serviços como produtos (exemplo: serviços locais ou na nuvem, baseados em microsserviço ou monolíticos).

Por exemplo, se você desenvolve um sistema que necessita realizar a consulta de um CEP, pode utilizar a API disponibilizada pelos Correios para encontrar um endereço. Sempre que usamos um aplicativo de smartphone para enviar mensagens ou compartilhar fotos, para verificar e-mails, fazer reservar em restaurante ou comprar ingressos para shows, estamos fazendo uso de APIs.

E para que haja entendimento entre as partes requisitante e solicitante do serviço, é muito importante que exista uma documentação de API do projeto, especificando os métodos para comunicação, os padrões para acesso à interface e as exigências para realizar a troca de dados. APIs oferecem diversas vantagens para o mundo corporativo, é o que vamos saber a partir deste momento.

4.2.1 Quais as vantagens das APIs?

Uma das grandes vantagens no uso e no desenvolvimento de APIs é a possibilidade de conectar tecnologias heterogêneas de maneira ágil e segura, beneficiando o ganho de escala do sistema e ampliando o *market share* da empresa.

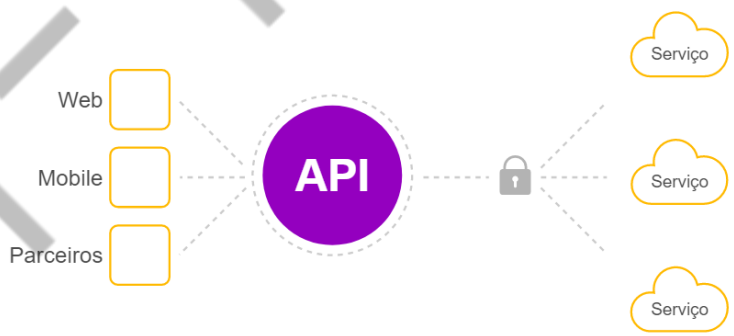

Figura 4.3 – Vantagens da API
Fonte: Google Imagens (2019)

Além disso, uma API tem benefícios relevantes que agregam valor implícito ao projeto, são eles:

- **Segurança na troca de informações** (por exemplo, conversação entre diferentes bancos de dados).

- **Auditoria de acessos –** Possibilita validar o fluxo de dados e identificar detalhes sobre o acesso ao sistema.
- **Controle de tráfego por meio de APIs Privadas**.
- **Autenticação –** Trabalha com a identidade do cliente no sistema através de *token* de acesso.
- **Monetização por acesso** (como, por exemplo, o que é realizado pelo Google ao monetizar os acessos aos serviços de *cloud*).
- **Design –** Define os links para acesso aos recursos e a forma como a API vai disponibilizar os recursos, que podem ser estruturados, por fluxos de *bytes* ou caracteres.
- **Formatos de dados –** Estabelece o formato de mensagem para a troca de dados, que podem ser formatos binários complexos, formatos de texto simples, *SOAP (Simple Object Access Protocol)* ou esquemas.
- **Gerenciamento –** Recurso para fornecer controle, visibilidade e governança sobre os ativos de negócios da empresa.
- **Redução no volume de dados –** Integração de forma específica para otimizar a performance da plataforma.

As APIs suportam o autoatendimento e um relacionamento um-para-muitos entre provedores e consumidores. Foram ganhando espaço entre serviços financeiros, governo, saúde, varejo e, nos últimos anos, um grande número de empresas e instituições governamentais publicou APIs em portais de desenvolvedores para alimentar a inovação B2C (*Business-to-Consumer*), permitir o uso de aplicativos móveis e se beneficiar de interações B2B (*Business-to-Business*) mais diretas com parceiros de negócios.

4.2.2 Qual a importância das APIs para a MSA?

Para responder a esta pergunta, devemos nos questionar sobre os impactos para os negócios de um mundo cada vez mais interconectado. Existem cada vez mais softwares capazes de atender de forma específica uma determinada atividade.

Isso faz com que haja uma grande competição para criar novos modelos de negócios.

O tema principal do Simpósio 2017 da Gartner/ITxpo foi sobre a necessidade de expandir os negócios digitais. Para uma empresa, não é fácil se manter na concorrência e prosperar nos dias de hoje. Assim, as empresas se viram obrigadas a passar por uma evolução, culminando com uma transformação digital, para atender às demandas do mundo atual.

Em uma pesquisa realizada pela empresa Freeform Dynamics (Out/2015), foram identificados os principais motivadores da transformação digital:

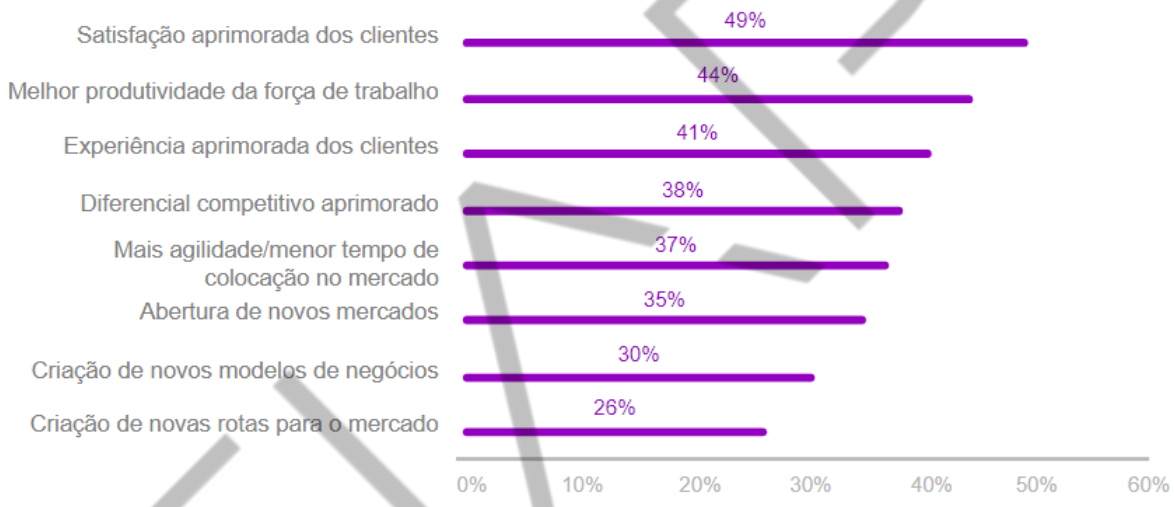

Figura 4.4 – Motivadores da transformação digital
Fonte: Freeform Dynamics (2015)

A empresa Gatepoint Research realizou outra pesquisa em que foram apontadas as iniciativas digitais mais importantes:

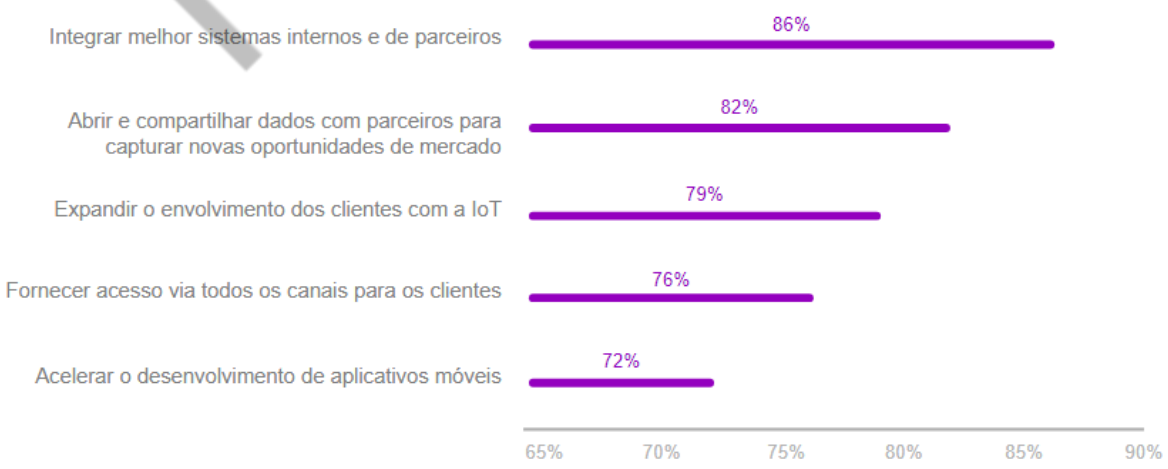

Figura 4.5 – Como implementar a transformação digital

Fonte: Gatepoint Research (2015)

Com essas informações, foi possível perceber que há uma grande necessidade de conectar cada vez mais pessoas, dados, dispositivos e sistemas, levando à conclusão de que o caminho para promover a expansão dos negócios digitais é através do uso de APIs (que oferecem integrações robustas e flexíveis que o mercado demanda) juntamente com a implementação de Microsserviços (que oferecem agilidade, alta produtividade e maior performance).

Diante desse cenário, as APIs passaram a ser pré-requisito de negócios, deixando de ser apenas modelo arquitetural de software para se tornarem produtos. Os Microsserviços passaram a ter lógica de negócios, sendo conhecidos como "*endpoints* inteligentes" e agregando grande valor aos negócios, pois fazem parte de produtos específicos com base em um ou mais serviços individuais. Estes são os motivos pelos quais as APIs e os Microsserviços estão cada vez mais em evidência.

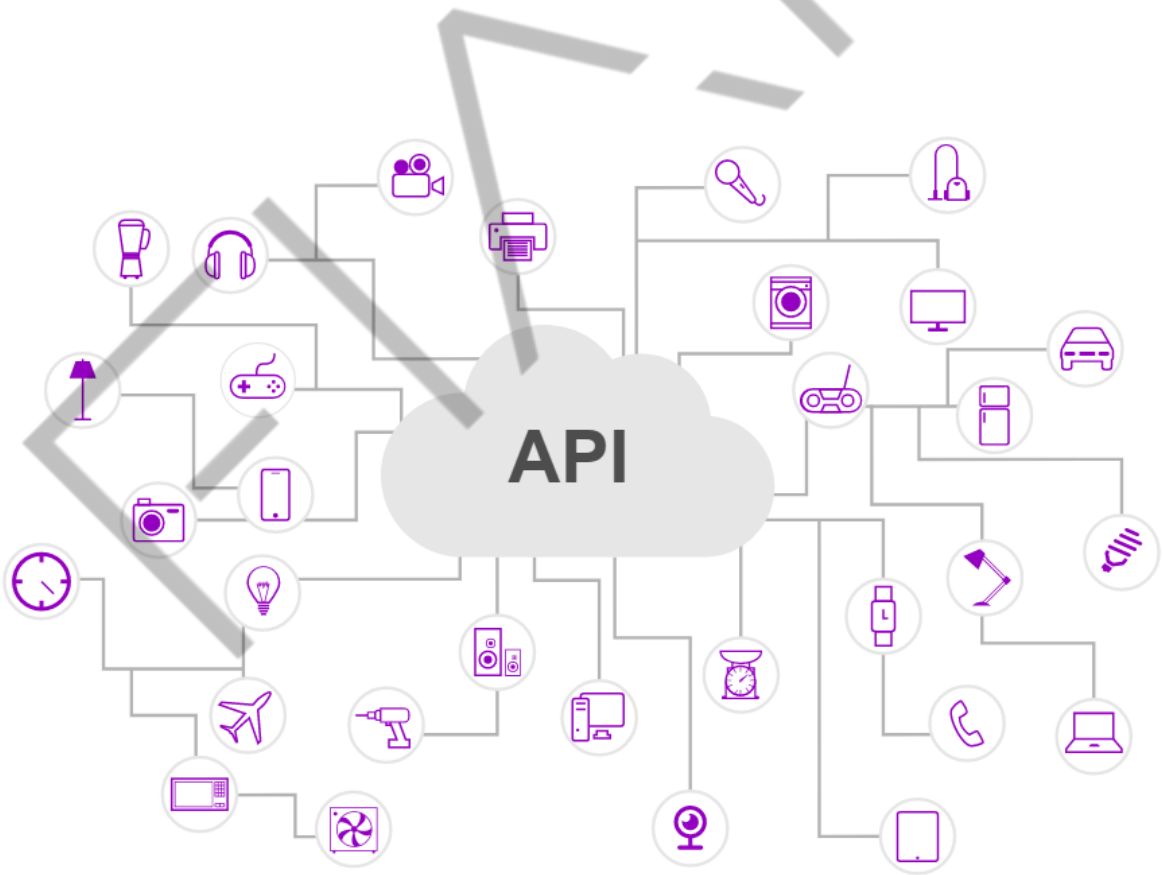

Figura 4.6 – Importância das APIs
Fonte: Shutterstock (2019)

Vejam o exemplo da Netflix, serviço de *streaming* de vídeo muito popular e responsável por até 30% do tráfego da Internet que tem uma arquitetura orientada a

serviços em larga escala. Eles lidam com mais de 1 bilhão de chamadas por dia com sua API de *streaming* de vídeo em mais de 800 tipos diferentes de dispositivos. Cada chamada de API alcança uma média de seis chamadas para serviços de *back-end*. (RICHARDSON, 2017. Tradução livre).

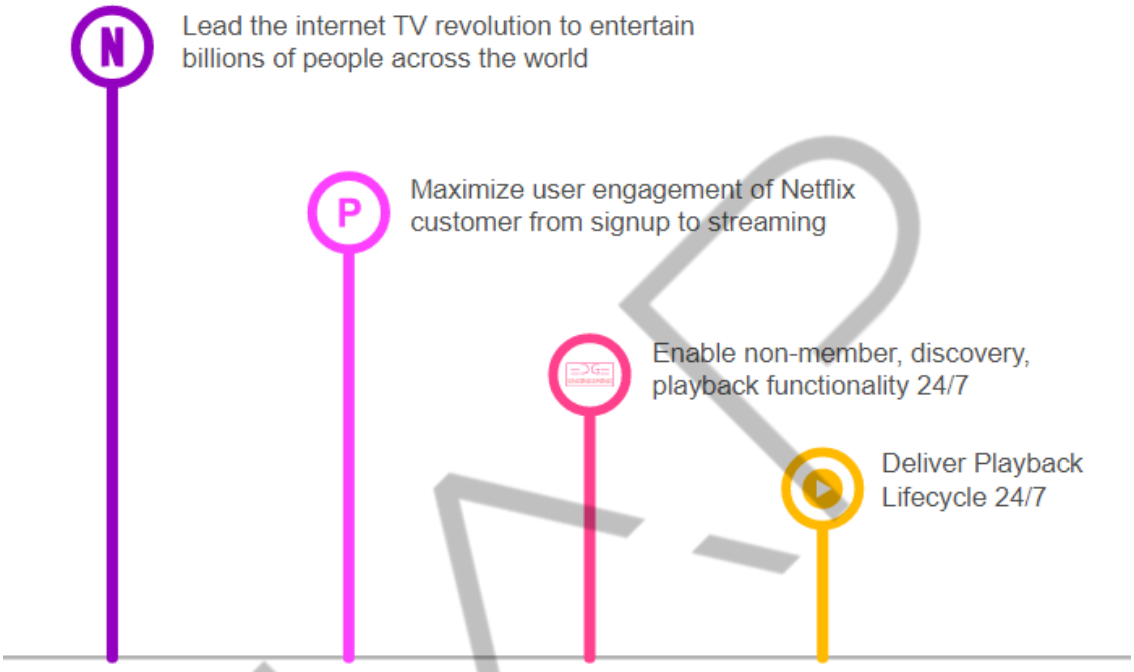

Figura 4.7 – Exemplo da Netflix usando APIs
Fonte: Google Imagens (2019)

Para se implementar efetivamente uma MSA, é necessário ter APIs como peça fundamental para gerenciar a complexidade entre velocidade e flexibilidade que os microsserviços fornecem. Uma estratégia de API eficaz também permite que os microsserviços coexistam com sistemas legados existentes e possibilita que organizações estejam mais preparadas para lidar com aumento de volume, escala e volatilidade de aplicações voltadas para o cliente.

Vamos avançar para descobrir o funcionamento das APIs com os microsserviços.

4.2.3 Como APIs e Microsserviços funcionam juntos?

O nosso grande desafio aqui é demonstrar de que forma podemos orquestrar a comunicação entre APIs e Microsserviços.

Como já escrito, a API deixou de ser apenas interface de código de programação de baixo nível para se tornar produto em si mesma. Em sua evolução, a API se tornou mais amigável para o desenvolvedor a partir do momento em que aderiu a padrões, como o REST (*Representational State Transfer*), e passou a oferecer um esquema mais forte de gerenciamento e governança. Desde esse momento., surgiu uma economia de APIs, possibilitando a troca de valores entre provedores e consumidores de APIs.

Devido às alterações em seus requisitos técnicos, as APIs passaram a necessitar de portais sofisticados para possibilitar gerenciamento pelos desenvolvedores. E como são expostas para o mundo externo da empresa, também necessitam de fortes recursos de segurança, que podem ser adquiridos através do Gateway de API. O Gateway de API disponibiliza o gerenciamento de API, recurso para fornecer controle, visibilidade e governança sobre os ativos de negócios da empresa.

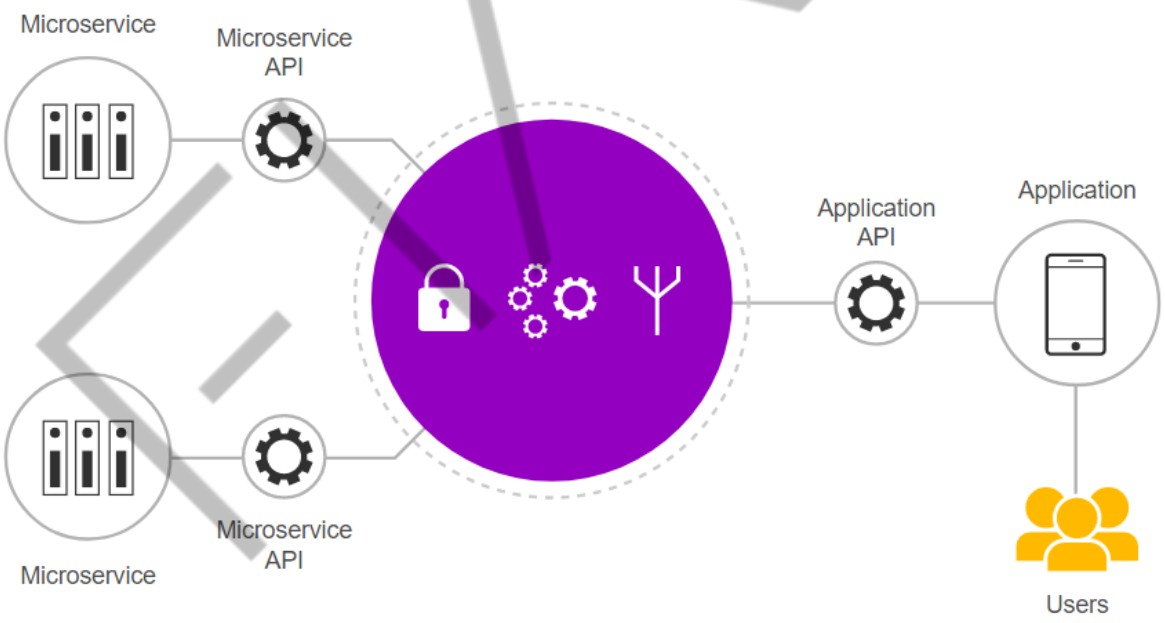

Figura 4.8 – Relacionamento das APIs com os Microsserviços
Fonte: API Academy (2016)

À medida que a MSA vai se difundindo dentro da empresa, o valor de se utilizar uma API interna é de suma importância, pois é extremamente difícil controlar a conectividade entre diversos *endpoints*. Uma receita para o fracasso é tentar criar integrações ponto a ponto entre todos esses *endpoints*.

Para tornar a aplicação mais eficiente, é muito importante implementar uma estratégia de integração liderada por API padronizada, principalmente para organizações que possuem uma pilha de TI legada. Isso permite que os microsserviços sejam conectados e/ou desconectados rapidamente, de acordo com as necessidades do negócio, e reduz o custo associado à construção de integração ponto a ponto entre sistemas legados e aplicações SaaS (*Software as a Service*).

Você vai ver, em seguida, mais detalhes sobre API Gateway e como ele está inserido na MSA.

4.3 API Gateway

Aplicações do lado cliente, normalmente, precisam consumir funcionalidades de um ou mais microsserviços. Como possibilitar que clientes de uma aplicação ou aplicativo em nuvem, baseados em microsserviços, acessem serviços individuais?

Para evitar chamada direta para cada um dos microsserviços que compõem a aplicação, foi criado um nível intermediário ou indireto denominado Gateway. Na MSA, deve ser implementado um Gateway de API como único ponto de entrada para todos os clientes. O Gateway de API é essencial para qualquer tipo de aplicação, mas, principalmente, para aplicativos móveis.

Entre suas responsabilidades, temos:

- Manipular as solicitações e rotear para o serviço apropriado.
- Isolar os clientes do acesso direto aos microsserviços.
- Determinar os locais das instâncias de cada serviço.
- Reduzir o número de solicitações / “*round trips”* (viagens de ida e volta), melhorando a experiência do usuário e resultando em menos sobrecarga.
- Implementar segurança (exemplo: verificar se o cliente está autorizado a executar uma determinada solicitação).
- Mover a lógica do lado do cliente para o Gateway de API.
- Fornecer uma API ideal para cada necessidade do cliente.

O objetivo principal de uma implementação com API Gateway é expor um microsserviço como uma API gerenciada. Existe uma variedade de entradas (*front-ends*) e uma variedade de serviços a serem consumidos (*back-ends*) e essas partes precisam conversar. Aí entra a necessidade de expor uma API diferente para cada cliente, fazendo o papel de um Gateway de API. Como exemplo, podemos ter três tipos de clientes (aplicação web, aplicativo móvel e aplicações externas de terceiros) consumindo serviços da sua empresa e cada um deles deve acessar um Gateway de API específico.

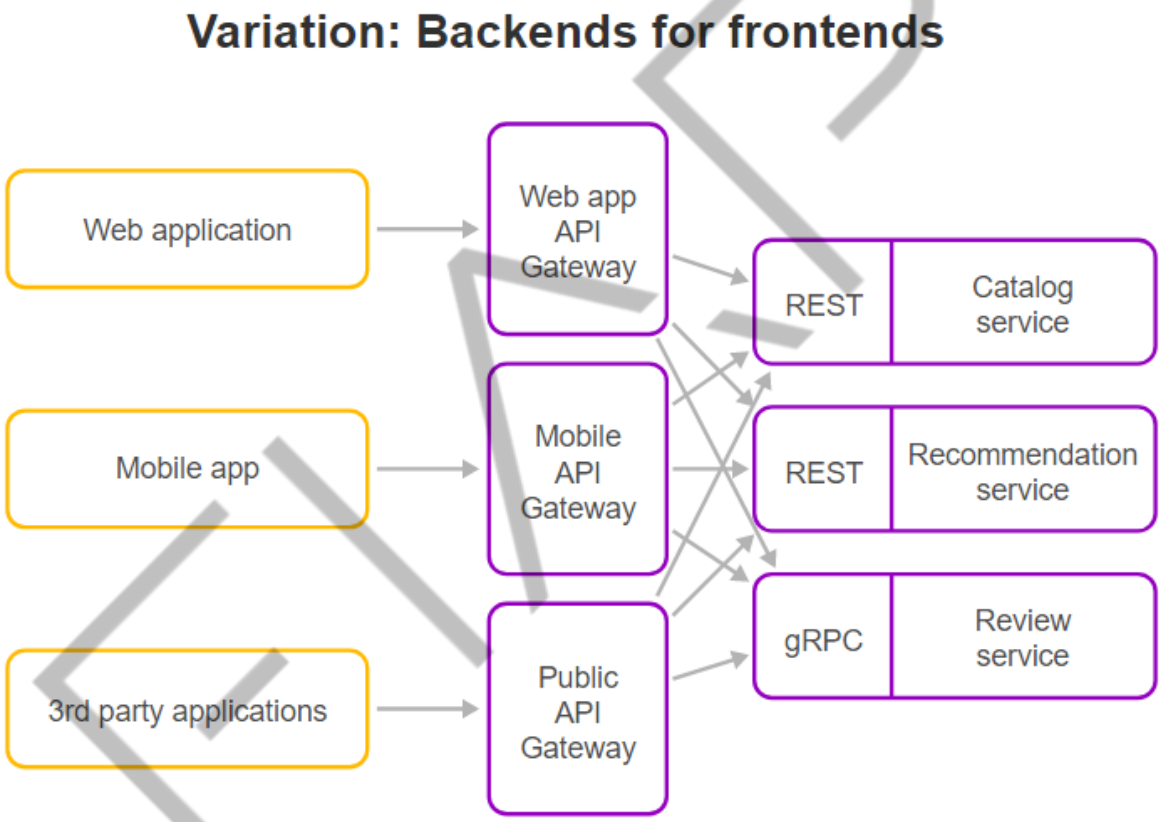

Figura 4.9 – API Gateway personalizado por Frontend
Fonte: Google Imagens (2019)

A Netflix faz muito bem o uso deste tipo de benefício. O Gateway de API da Netflix (ou Netflix API Gateway) executa o código do adaptador específico de cada cliente e fornece a ele uma API mais adequada aos seus requisitos.

Para suprir a necessidade de alta disponibilidade para sua aplicação, uma alternativa é executar vários Gateways de API (exemplo: implantação do Kubernetes com várias réplicas) por trás de um balanceador de carga gerenciado (exemplo: *AWS ELB – Elastic Load Balancing*).

Não diferente de outras APIs, o API Gateway deve ser desenvolvido, implantado e gerenciado, ou seja, é mais uma “parte móvel” da empresa. A implementação de um API Gateway pode ser com NodeJS ou pode-se fazer uso de bibliotecas da JVM (*Java Virtual Machine*) baseadas em NIO (*Non-blocking I/O*), como Netty, Spring Reactor etc.

4.4 Service Mesh

Service Mesh é uma rede de microsserviços que surgiu como um padrão (*pattern*) para superar o desafio advindo da MSA de administrar a complexa comunicação entre diversos microsserviços, após a eliminação do barramento ESB (*Enterprise Service Bus*) existente na Arquitetura Orientada a Serviços (SOA).

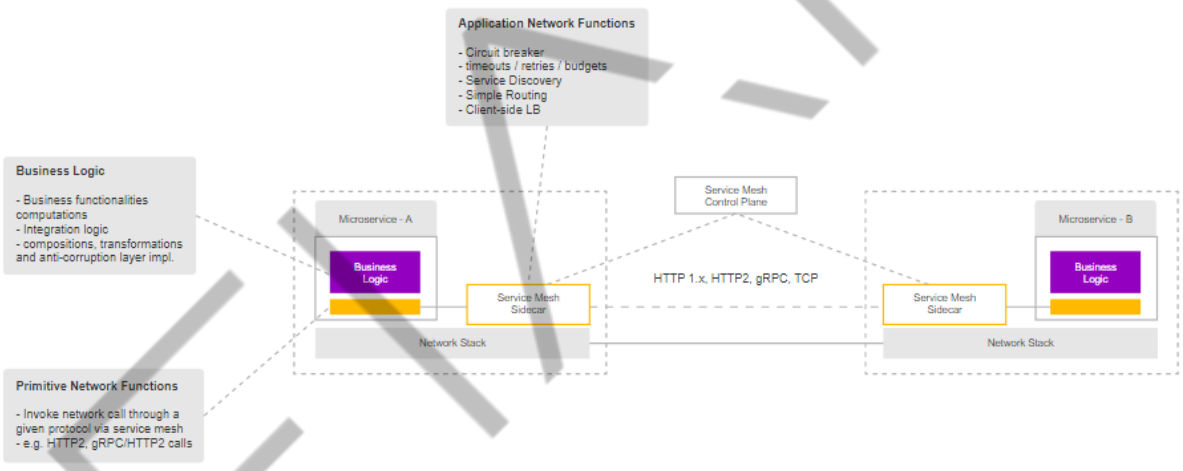

Figura 4.10 – Serviço para serviço de comunicação com o Service Mesh
Fonte: Indrasiri (2018)

O **Service Mesh Control Pane** é responsável por gerenciar todos os *proxies* de *Service Mesh* com a finalidade de suportar recursos, como descoberta de serviço, controle de acesso, observabilidade, entre outros.

O **Service Mesh Sidecar** (ou *Protocol Proxy Sidecar*) fornece uma camada distribuída genérica que encapsula os recursos básicos da comunicação entre serviços que não fazem parte da lógica de negócios do serviço. Essa camada é um *proxy* para todos os eventos de entrada e saída dos consumidores e produtores de serviços responsável por garantir que um microsserviço não se comunique diretamente com outro microsserviço.

IMPORTANTE: As funções de rede devem ser responsabilidade do Service Mesh, não devem ser implementadas em cada nível de microsserviço.

Como principais funções do Service Mesh, temos:

- Segurança no nível de transporte (*TLS – Transport Layer Security*) e gerenciamento de chaves.
- Service Discovery (Descoberta de Serviço) através de um registro de serviço dedicado.
- Observabilidade: monitoramento, métricas, logs, registro distribuído e rastreamento distribuído.
- Controle de tráfego: *load balance*, limitação de taxa e qualidade do serviço.
- Controle de acesso baseado em listas.
- Tolerância a falhas sem a necessidade de um recurso compartilhado (usando um API Gateway, por exemplo).
- Resiliência para comunicações entre microsserviços: *circuit breaking*, autenticação, novas tentativas e *timeouts*, roteamento simples, *fault-injection* (injeção de falhas), gerenciamento de falhas e *load balance*.
- Implantação com suporte nativo para containers (Docker e Kubernetes).

É válido salientar que o Service Mesh aborda apenas um subconjunto de problemas de comunicação entre microsserviços. Problemas complexos, como integração com outros serviços e sistemas, roteamento complexo e outros mais, devem ser tratados na lógica de negócios de cada microsserviço.

Algumas implementações populares de Service Mesh são o Istio e o Linkerd. E como proxy para implementação dos Sidecars, temos o Envoy, Nginx, Traefik, entre outros. O proxy atua como o balanceador de carga principal para solicitações entre serviços internos.

4.5 API Gateway vs. Service Mesh

API Gateway e Service Mesh são conceitos que abordam requisitos essencialmente distintos. A diferença principal é que o Service Mesh nada mais é do que uma infraestrutura de comunicação entre os microsserviços que não contém nenhuma lógica embutida. Diferentemente do API Gateway, que implementa em sua camada o API Service, que contém a lógica de negócios própria que cria composições / *mashups* de vários serviços *downstream* e é responsável por chamar microsserviços compostos e atômicos de forma *downstream* (INDRASIRI, 2018).

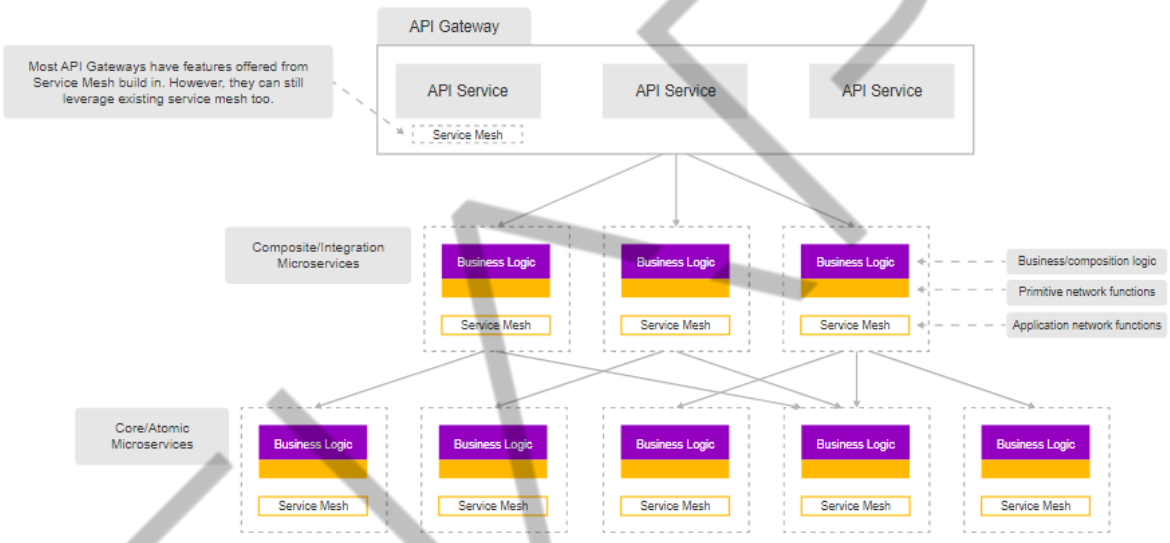

Figura 4.11 – API Gateway e Service Mesh
Fonte: Indrasirl (2018)

Um *Service Mesh* é implementado junto ao microsserviço como um *sidecar* (padrão lateral), fornecendo recursos de suporte, mas é totalmente independente da lógica do microsserviço.

O API Gateway hospeda os API Services e deve ser implementado como parte da lógica de negócio de seu microsserviço. É possível usar Service Mesh para chamar serviços *downstream*, transferindo as funções de rede do aplicativo para a malha de serviço; ou implementar recursos internos de comunicação entre serviços.

4.6 Desafio

Lançamos aqui um desafio: procure incrementar o projeto que foi criado no capítulo anterior para utilizar o API Gateway e trabalhar com Service Mesh.

1. Manter armazenado em container Docker.
2. Fazer *deploy* na AWS ou IBM Cloud.

CONCLUSÃO

Na evolução do mercado com uso de APIs, já é unânime que o uso comercial de APIs e o valor obtido com a "economia de APIs" são muito importantes para prover abertura para novos canais de negócios, integrações com parceiros e até mesmo novos mercados. Também já está evidente que APIs são o elo entre plataforma e ecossistema e a publicação de APIs funciona como gatilho para uma transformação digital.

A tendência é haver cada dia mais produtos que facilitam a transformação digital de uma empresa, provendo ganhos significativos para empresas envolvidas com o mercado digital.

O aprendizado não para por aqui! No próximo capítulo, vamos abordar o conhecimento de implementação para aplicativos *mobile*.

REFERÊNCIAS

API ACADEMY. **API Management 302**: using an API Gateway in Microservice Architecture. 2016. Disponível em: <https://www.apiacademy.co/lessons/2016/09/api-management-302-using-an-api-gateway-in-microservice-architecture>. Acesso em: 10 out. 2019.

FREEFORM DYNAMICS. **Exploiting the Software Advantage**. 2015. Disponível em: <http://www.the-digital-insurer.com/wp-content/uploads/2015/12/629-Exploiting-the-Software-Advantage_final.pdf>. Acesso em: 10 out. 2019.

GATEPOINT RESEARCH. **Digital transformation with APIs and Microservices**, 2015. Disponível em: <https://simplydirect.com/wp-content/uploads/Digital-Transformation-with-APIs-and-Microservices.pdf>. Acesso em: 10 out. 2019.

INDRASIRI, Kasun; SIRIWARDENA, Prabath. **Microservices for the enterprise: designing, developing, and deploying**. Apress, 2018.

RICHARDSON, Chris. **Pattern**: Microservice Architecture. 2017. Disponível em: <https://microservices.io/patterns/microservices.html>. Acesso em: 10 out. 2019.

TANENBAUM, Andrew S.; VAN STEEN, Maarten. **Distributed Systems: principles and paradigms**. 2. ed. Amsterdam: Pearson Education, 2006.

GLOSSÁRIO

API	Application Programming Interface é uma estrutura de códigos de programação padronizados que permitem a conexão entre sistemas.