Information Systems and Technology Management 2

Marcos William Kaspchak Machado
(Organizador)

Atena Editora
Ano 2019

Marcos William Kaspchak Machado

(Organizador)

Information Systems and Technology Management 2

Atena Editora
2019

2019 *by Atena Editora*

Editora Chefe: Profª Drª Antonella Carvalho de Oliveira
Diagramação e Edição de Arte: Lorena Prestes e Karine de Lima
Revisão: Os autores

Dados Internacionais de Catalogação na Publicação (CIP)
(eDOC BRASIL, Belo Horizonte/MG)

I43 Information systems and technology management 2 [recurso eletrônico] / Organizador Marcos William Kaspchak Machado. – Ponta Grossa (PR): Atena Editora, 2019. – (Information Systems and Technology Management; v. 2)

Formato: PDF
Requisitos do sistema: Adobe Acrobat Reader
Modo de acesso: World Wide Web
ISBN 978-85-7247-202-9
DOI 10.22533/at.ed.029191903

1. Gerenciamento de recursos de informação. 2. Sistemas de informação gerencial. 3. Tecnologia da informação. I. Machado, William Kaspchak. II. Série.

CDD 658.4

Elaborado por Maurício Amormino Júnior – CRB6/2422

APRESENTAÇÃO

A obra denominada "*Information Systems and Technology Management*" contempla dois volumes de publicação da Atena Editora. O volume II apresenta, em seus 26 capítulos, um conjunto de estudos sobre a aplicação da gestão do conhecimento aos processos de gestão organizacional, operacional e de projetos.

As áreas temáticas de gestão organizacional e de projetos mostram a importância da aplicação dos sistemas de informação e gestão do conhecimento na cultura organizacional e no desenvolvimento de novos projetos.

Este volume dedicado à aplicação do conhecimento como diferencial competitivo para inovação em processos produtivos, traz em seus capítulos algumas aplicações práticas de levantamento de dados, gestão da cultura e governança empresarial, além de ferramentas de monitoramento da qualidade da informação.

Aos autores dos capítulos, ficam registrados os agradecimentos do Organizador e da Atena Editora, pela dedicação e empenho sem limites que tornaram realidade esta obra que retrata os recentes avanços científicos do tema.

Por fim, espero que esta obra venha a corroborar no desenvolvimento de novos, e valorosos conhecimentos, e que auxilie os estudantes e pesquisadores na imersão em novas reflexões acerca dos tópicos relevantes na área de gestão do conhecimento e aplicações dos sistemas de informação para formação de ambientes cada vez mais inovadores.

Boa leitura!

Marcos William Kaspchak Machado

SUMÁRIO

CAPÍTULO 7

APIs AND MICROSERVICES

Anaury Norran Passos Rito
ORCID iD 0000-0001-7108-1678
(Universidade Federal de Pernambuco, Brasil) – anpr@cin.ufpe.br

José Carlos Cavalcanti
ORCID iD 0000-0002-6236-3961 (Universidade Federal de Pernambuco, Brasil) – cavalcanti.jc@gmail.com

ABSTRACT: In modern society the use of technology to facilitate conflict resolution has become increasingly popular. Thus, companies have the need to invest in technology to stay in the market and supply the high demand for activities. Therefore, some technologies stand out in relation to others. In this case, APIs and Microservices have emerged. Using these components, applications can perform functions more easily and efficiently, and can generate revenue for the business. For the economic success, using APIs it is necessary to know fundamental business aspects, manage the APIs and trace monetary plans to obtain income. Therefore, the objective of this paper is to present the main concepts related to APIs and Microservices, seeking to understand what can be called here the foundations of the API and Microservice Economics. As a result, it was possible to observe that the use of APIs and Microservices has much to contribute to a business economic growth. However, even their use can contribute positively, some difficulties must be assessed with caution in order to achieve a favorable result.

KEYWORDS: API, Microservice, Business, Monetization, Management.

RESUMO: Na sociedade moderna se tornou crescente o uso de tecnologia para facilitar na solução de conflitos. Dessa forma, as empresas têm a necessidade de investir em informatização para se manterem no mercado e suprir a alta demanda de atividades. Diante disso, algumas tecnologias se destacam em relação a outras. Neste caso, emergem as APIs – *Application Programming Interfaces* e os Microsserviços. Com o uso desses componentes, as aplicações conseguem executar funções de maneira mais fácil e eficiente, além de tornar possível a geração de receita ao negócio. Para se ter sucesso economicamente usando APIs, é necessário conhecer pontos fundamentais do negócio, gerenciar as APIs e traçar planos monetários para se obter renda. Neste sentido, este trabalho tem como objetivo apresentar os conceitos principais associados à relação entre APIs e Microsserviços, buscando entender o que se poderia aqui denominar de os alicerces da *Economia das APIs e dos Microsserviços*. Como resultado, foi possível observar que o uso de APIs e Microsserviços tem muito a contribuir

com o crescimento econômico de um negócio. Todavia, da mesma forma que o seu uso contribui positivamente, algumas dificuldades precisam ser avaliadas com cautela para se obter um resultado favorável.

PALAVRAS-CHAVE: API, Microsserviço, Negócio, Monetização, Gerenciamento.

1 I INTRODUÇÃO

A cada dia é mais comum constatar aplicações que usam a internet como meio de comunicação, sendo muitas vezes usadas por diversos equipamentos tecnológicos, independentemente de qual plataforma está sendo consumida. Com a crescente integração entre aplicações e sistemas é normal pensar como isso acontece, e é nessa situação onde as APIs se fazem presentes. A sigla API vem do termo em inglês *Application Programming Interface* ou, no português, *Interface de Programação de Aplicações*. Uma *API* disponibiliza dados e serviços para canais de desenvolvedores parceiros, colegas, ou terceiros para a criação de aplicações [1].

Historicamente, a API da Salesforce, criada em 7 de fevereiro de 2000, foi dada como o primeiro caso de sucesso. Mas o "boom" só veio mesmo em 2004, quando a API de fotografias do Flickr foi disponibilizada. Esse lançamento fez com que a empresa crescesse rapidamente e, em menos de um ano, fosse comprada pela Yahoo. Dois anos depois o Facebook lançou a sua API e, depois de alguns meses, o Twitter agiu da mesma forma. Ainda em 2006, a Amazon apresentou sua própria API voltada para o armazenamento de dados e, seis meses após, desenvolveu outra API, dessa vez focada em infraestrutura [2]. Em meados de 2010, as aplicações móveis e os serviços em nuvem começavam a ganhar força. E esse novo padrão de aplicação permitiu que novos modelos de negócios fossem apresentados, dando origem ao termo Economia das APIs [3].

Paralelamente os microsserviços ganhavam força, porém o histórico é um pouco mais breve. Em 2005 Peter Rogers apresentou o termo "*micro web services*", ou, em português, *microsserviços de rede*. Mas a expressão '*Microsserviço*' só ganhou força em 2011, quando foi apresentada uma arquitetura baseada em componentes distribuídos [4]. O conceito de microsserviços, segundo Martin Fowler, da empresa ThoughtWorks, pode ser entendido como pequenos sistemas autônomos que se comunicam entre si [5].

Atualmente é frequente encontrar aplicações que usem as duas tecnologias, e no caso das APIs, elas são usadas em redes sociais, como Facebook ou Twitter, em sites de comércio eletrônico, como Amazon e Ebay, em pagamentos eletrônicos, como PayPal, e em diversas outras aplicações [6]; já os microsserviços são usados, novamente, em empresas como Amazon e Ebay, além de Netflix, Uber, SoundCloud e tantas outras [7], pois ambas trazem benefícios significativos para o negócio onde são inseridas.

Associados às APIs, os Microsserviços permitem que os desenvolvedores, tanto internos quanto parceiros, criem aplicações que possam agregar mais valor ao produto/serviço sem haver uma grande interdependência [1].

De acordo com dados do site ProgrammableWeb, no primeiro quadrimestre de 2017, o número de APIs públicas em seu repositório ultrapassou a marca de 17000, e desde 2014 até a data da análise, foram adicionadas mais de 5900 APIs [8]. Com diversas aplicações voltadas aos clientes, sistemas de comércio eletrônico, redes sociais, sistemas de *Internet Banking,* o crescimento da Internet das Coisas e outros, é que esse número continue a crescer.

Neste sentido, este trabalho tem como objetivo apresentar os conceitos principais associados à relação entre APIs e Microsserviços, buscando entender o que se poderia aqui denominar de os alicerces da *Economia das APIs e dos Microsserviços*. Sendo assim, o trabalho, além desta breve introdução, está sub-dividido em mais quatro seções. A seção 2 contém os principais conceitos relacionados às APIs, desde seu surgimento, passando pela definição, sua cadeia de valor, bem como benefícios e desafios de uso. A seção 3 contempla conceitos relacionados aos Microsserviços, assim como a relação entre eles e as APIs, incorporando seu surgimento, definição, as características principais das arquiteturas baseadas em microsserviços, e, finalmente, benefícios e desafios da abordagem dos microsserviços. Na seção 4 é feita uma breve apresentação dos principais aspectos relativos ao design da Arquitetura de Microsserviços. A seção 5 apresenta as conclusões principais.

2 | APIs

2.1 Surgimento

Segundo Robert Broeckelmann, do site Medium [9] e principal consultor da Levvel (levvel.io), o termo API começou a ser usado para descrever qualquer interface para bibliotecas ou módulos entre grandes sistemas. Porém, nos últimos anos, a expressão passou a se referir, mais comumente, a um estilo arquitetural de cliente-servidor baseado no protocolo HTTP, sem estado e de uso mais simples.

Antes das APIs, era mais comum o uso de *Web Services* para fazer a comunicação entre as máquinas. Os modelos arquiteturais de *Web Services* mais usados em meados da década de 90 até a popularização das APIs foram o SOAP– *Simple Object Access Protocol,* e, posteriormente, o REST- *Representational State Transfer*. O padrão REST se tornou uma popular alternativa aos *Web Services* SOAP, principalmente em aplicações móveis. Ele veio como opção para simplificar e melhorar o paradigma anterior (SOAP) e rapidamente ganhou notoriedade no mercado, devido a sua maior simplicidade e por não se basear em estados definidos, como no SOAP, e sim em requisições, assim como não se preocupar com detalhes de desenvolvimento

[10].

Posteriormente, as APIs, evoluídas do REST, chegaram para formalizar o estilo. Porém, elas podem ser baseadas tanto em REST quanto em SOAP. Uma grande diferença entre o contexto onde eram usados os *Web Services*, antes do surgimento das APIs, e o contexto onde elas são usadas agora, é em relação à exposição. Os *Web Services* são mais expostos dentro da empresa, enquanto as APIs têm a opção de expor para fora dela, a uma base maior de usuários, sujeito a usos inesperados e inovadores, permitindo, de acordo com o público atingido, a criação de diferentes estratégias de monetização sobre as funções expostas. Dessa forma, as APIs também podem ser tratadas como produtos, além de apenas uma forma de estabelecer comunicação entre aplicações [11].

2.2 Definição

Assim como foi mencionado na introdução deste trabalho, uma *Application Programming Interface*, no português, Interface de Programação de Aplicações ou simplesmente API é, segundo Daniel Jacobson, Greg Brail e Dan Woods, uma forma de dois programas se comunicarem, independentemente da plataforma usada, usando uma linguagem comum [1]. Ou, de forma mais técnica, uma API é um conjunto de rotinas, ferramentas e protocolos para se criar uma aplicação [12].

Ela tem por objetivo conectar times internos, parceiros externos, processos e serviços para a criação de aplicações, como citado na introdução, ou seja, transmitir dados. Dessa forma, é possível imaginar uma API como uma “cola” digital (ver Figura 1) [13].

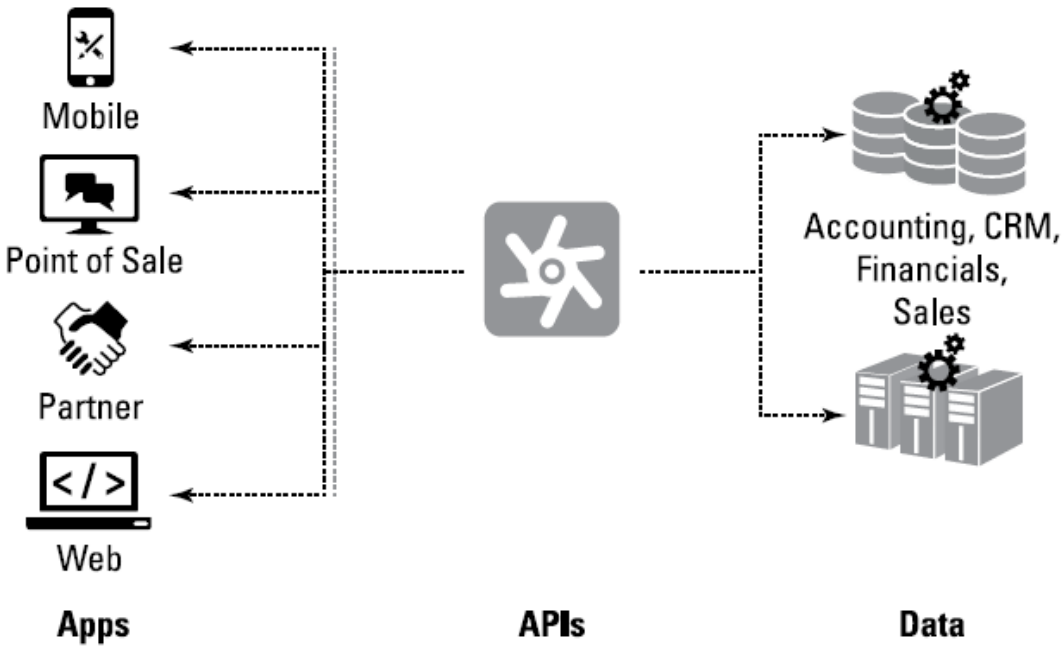

Figura 1: API como uma cola digital.

Fonte: API for Dunmies [13]

2.3 Entendendo a Cadeia de Valor de uma API

Para desenvolver uma API é necessário ter o conhecimento da sua cadeia de valor e os impactos que ela pode causar no negócio. Numa análise a ser feita previamente, busca-se entender os seguintes fatores [1]:

1. **Para quem os ativos comerciais que serão expostos?**
2. **Quem é o provedor da API?**
3. **Quem serão os desenvolvedores?**
4. **Quais aplicações farão uso dela?**

Analisando esses quatro pontos básicos, é possível obter uma noção de qual o segmento essa API vai se encontrar. Elas são comumente classificadas em três grupos, da forma: 1) Privadas, 2) Públicas ou 3) Privadas/De parceiros [14], onde cada uma delas tem públicos alvo, objetivos e cadeias de valor distintas.

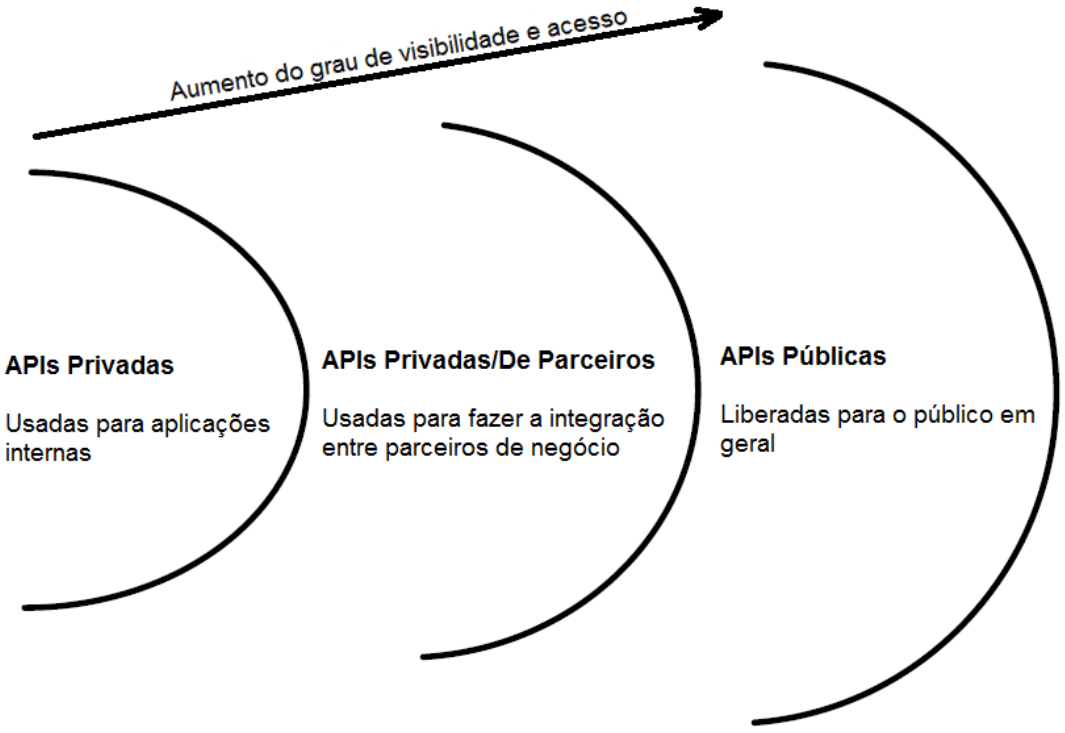

Figura 2: Tipos de API

Fonte: Adaptada de API Management [14]

2.3.1 APIs Privadas

Como podem ser observadas na Figura 2, as APIs Privadas são as da camada mais interna, onde apenas os times internos têm acesso às funcionalidades. Nesse

caso, essas APIs funcionam como uma interface entre a aplicação e o *Back-End* da empresa ou como meio de conectar diversos setores locais.

Esse segmento tem uma grande influência na Economia das APIs, além de ser, junto com as APIs de Parceiros (vistas à frente na seção 2.3.2), a maioria em número de APIs. Elas podem trazer, muitas vezes, mais benefícios ao negócio que uma API Pública. Elas possibilitam transformar a estrutura negocial, reduzindo custos de operação, permitindo a colaboração entre times, simplificando a infraestrutura de TI, trazendo mais segurança aos dados compartilhados, e permitindo a modularização de componentes, entre outros benefícios [14] [15].

Respondendo os quatro pontos citados anteriormente, é possível ter uma ideia da cadeia de valor relacionada a uma API privada [1]:

1. Para quem os ativos comerciais que serão expostos?

A empresa que usa uma API privada não tem interesse em expor seus ativos para fora dela. Logo, tais ativos só serão usados internamente;

2. Quem é o provedor da API?

Por ser privada, geralmente o provedor da API é a própria empresa;

3. Quem serão os desenvolvedores?

Seguindo o mesmo pensamento do item 2, os desenvolvedores são internos;

4. Quais aplicações farão uso dela?

Geralmente aplicativos e sistemas criados para o uso interno, mas também podendo ser usadas para a criação de aplicações (de criação própria) para serem disponibilizadas ao público interno ou externo.

2.3.2 APIs Privadas/De Parceiros

Em seguida, existem as APIs de Parceiros (ou *Partner* APIs). Alguns autores, como por exemplo, Daniel Jacobson, Greg Brail e Dan Woods, autores do livro "*APIs: A Strategy Guide"*, as encaixam como APIs Privadas, mesmo que também pareçam com as APIs Públicas, pelo fato de não serem usadas apenas internamente. Nesse caso, as funcionalidades e informações, além de serem acessadas pelas equipes internas, elas são disponibilizadas também para os parceiros de negócio [1].

Junto com as APIs Privadas, elas somam a maior fatia do mercado de APIs e têm como benefícios a criação de novos canais, a extensão de produtos e negócios, além de serem agregadores de novos valores ao produto/serviço. Fazendo a mesma associação que foi feita no item 2.3.1, as APIs de Parceiros também geram uma cadeia de valor:

1. Para quem os ativos comerciais que serão expostos?

Uma empresa que possui uma API de Parceiros pretende expor ativos apenas para parceiros comerciais e internos;

2. Quem é o provedor da API?

Nesse caso, o provedor é a própria empresa;

3. Quem serão os desenvolvedores?

Nesse ponto, as APIs de Parceiros se parecem mais com as APIs Públicas, que serão vistas na seção seguinte, porém com uma pequena diferença: o público externo é restrito a afiliados;

4. Quais aplicações farão uso dela?

As APIs de parceiros se comportam tanto como uma API Privada, quanto como uma API Pública, porém, nesse último caso, com um alcance menor. Portanto, elas podem ser usadas em aplicações internas e por aplicações externas, mas que tenham relação com o negócio do provedor.

2.3.3 APIs Públicas

Na outra extremidade estão as APIs Públicas, onde essas são desenvolvidas para expor funcionalidades e informações de um ou vários sistemas para pessoas fora do negócio, com o mínimo de arranjos contratuais.

Da mesma forma que o público externo pode acessar essa API, o interno tem a mesma possibilidade, podendo fazer o papel de uma API privada. Por terem um alvo maior, as APIs públicas têm grande potencial para agregar valor ao negócio sem investimento direto e reduzindo o custo no desenvolvimento de funcionalidades [16].

É dito que é nesse tipo onde existem as maiores chances de inovação [17]. Mas para tal objetivo ser alcançado, é necessário adotar medidas para chamar a atenção desses desenvolvedores e fornecer suporte e uma interface intuitiva para a execução das tarefas. Do outro lado, é indispensável que também sejam adotados métodos de segurança no acesso, mecanismos para resolver sobrecargas, além de tentar reduzir o impacto nos serviços quando houver alguma alteração na API [14].

Uma forma comum de se chegar a essa etapa, é seguir o caminho de uso de uma API Privada, compartilhar essa API com parceiros de negócios e, então, tornar essa API aberta.

Da mesma maneira feita nos dois outros casos de APIs, é possível analisar as mesmas questões e encontrar uma cadeia de valor para as APIs públicas [1]:

1. Para quem os ativos comerciais que serão expostos?

Nesse caso os provedores da API desejam que elas alcancem uma maior escala de pessoas;

2. Quem é o provedor da API?

Os usuários das APIs públicas podem usar as APIs disponibilizadas pela própria empresa ou usar APIs de terceiros;

3. Quem serão os desenvolvedores?

Os desenvolvedores serão tanto do público interno, quanto do público externo;

4. Quais aplicações farão uso dela?

Como, nessa ocasião, a API é pública, diversas aplicações podem ser criadas com vários objetivos distintos, internamente ou externamente.

2.4 Benefícios e Desafios do Uso

2.4.1 Benefícios

O desenvolvimento de uma API pode parecer uma tarefa complicada, porém o uso dessa tecnologia traz consigo diversos benefícios, tanto para os desenvolvedores, quanto para o empreendimento. Benefícios esses que variam de mercado para mercado, e de negócio para negócio. Todavia, é possível citar alguns que são comuns a qualquer negócio [18] [19]:

- **Eficiência:** Uma API permite que o conteúdo criado esteja automaticamente disponível para todos os canais;
- **Automatização:** As APIs permitem que as máquinas também lidem com as cargas de trabalho;
- **A criação de aplicativos:** Por ter acesso à informação e por ser uma interface, uma API possibilita que essas informações sejam usadas por aplicativos móveis;
- **Parceria:** Um caminho natural a se chegar a uma API Pública é sair de uma API Privada, serem feitas parcerias, passando pelo segundo tipo e, por fim, alcançar o título de API Pública. Dessa forma, as APIs proporcionam que sejam feitas parcerias entre o provedor e os desenvolvedores;
- **Integração e experiência para o usuário:** As APIs permitem que seu conteúdo seja mais facilmente acessado por sites ou diferentes aplicações;
- **Personalização:** Usuários, funcionários, desenvolvedores, empresas, etc. têm a capacidade de personalizar as sessões com as informações e serviços que sejam mais interessantes para eles;
- **Transparência:** Para as APIs, todos os consumidores são mostrados da mesma forma, independente das especificações do dispositivo usado.

2.4.2 Desafios

Assim como qualquer outra tecnologia, o uso de APIs também traz desafios ao

negócio ao qual elas são implantadas. Dessa forma, é viável citar [20] [21]:

- **Custo:** Desenvolver e fornecer recursos de uma API pode ser custoso em termos de desenvolvimento, tempo, manutenção e suporte;
- **Necessidade de conhecimento:** Para o desenvolvimento de uma API é necessário ter um bom conhecimento de programação e a curva de aprendizado pode ser maior, quando comparada com outras tecnologias;
- **Segurança:** Por uma API ser uma interface, ela está mais suscetível à conexão de aplicações e módulos os quais podem trazer potenciais ataques;
- **Lidar com diferentes públicos:** O nível de acesso, funções, documentação, etc., variam de acordo com o tipo de API escolhido;
- **Integração com sistemas legados:** Em empresas menores ou *start-ups*, onde há uma grande dependência de funções externas, o uso de APIs é um caminho natural. Porém isso muda quando uma empresa é dependente de um sistema legado, é necessário mais esforço, tanto para integrar uma API ao sistema, quanto criar uma.

3 I MICROSSERVIÇOS

3.1 Surgimento

A criação de aplicações hoje consideradas *monolíticas* (tais como grandes sistemas ERP, CRM, dentre outros) sempre teve suas dificuldades, principalmente com o seu crescimento. Com a evolução da tecnologia, o modelo de arquitetura monolítica tradicional começou a apresentar problemas. Devido à sua grande complexidade interna, tornou-se difícil definir a granularidade de um projeto. O fato de todas as funcionalidades serem agregadas ao sistema principal, faz com que o sistema cresça, aumente de complexidade e aumente a dependência entre as funções.

Como consequências de uma forte interdependência entre funções, é comum verificar que, o sistema como um todo, apresente dificuldades na mudança de tecnologias, no desenvolvimento de funções, na escalabilidade do sistema, na instalação e na configuração do sistema, etc. Além do que foi citado, há também: problemas de impacto das mudanças feitas, adversidade nas entregas continuas e problemas em módulos isolados que afetariam a aplicação como um todo. Porém, um grande ponto positivo dessa arquitetura seria a arquitetura externa simplificada [22].

A arquitetura de microsserviços veio para facilitar na criação e desenvolvimento de aplicações em torno do negócio, decompondo o sistema maior em partes menores que se comunicam entre si. Nesse caso, há uma troca válida entre simplicidade na arquitetura interna do componente, com uma arquitetura externa mais complexa. A simplicidade interna proporciona diversos benefícios que possibilitam a diminuição do tempo de desenvolvimento e, consequentemente, na oportunidade de entrega de

novas funcionalidades em menor espaço de tempo [23].

3.2 DEFINIÇÃO

Segundo Eberhard Wolff, autor do livro "*Microservices: Flexible Software Architecture*" [24], microsserviço é um conceito de modularização, com propósito de dividir um sistema maior em módulos menores. Nesse caso, esses componentes podem ser desenvolvidos separadamente, em diferentes tecnologias, com seus próprios recursos e suporte, trazendo maior independência entre os módulos. Em outras palavras, microsserviços são pequenos sistemas autônomos que podem compor um maior e estabelecer comunicação entre si.

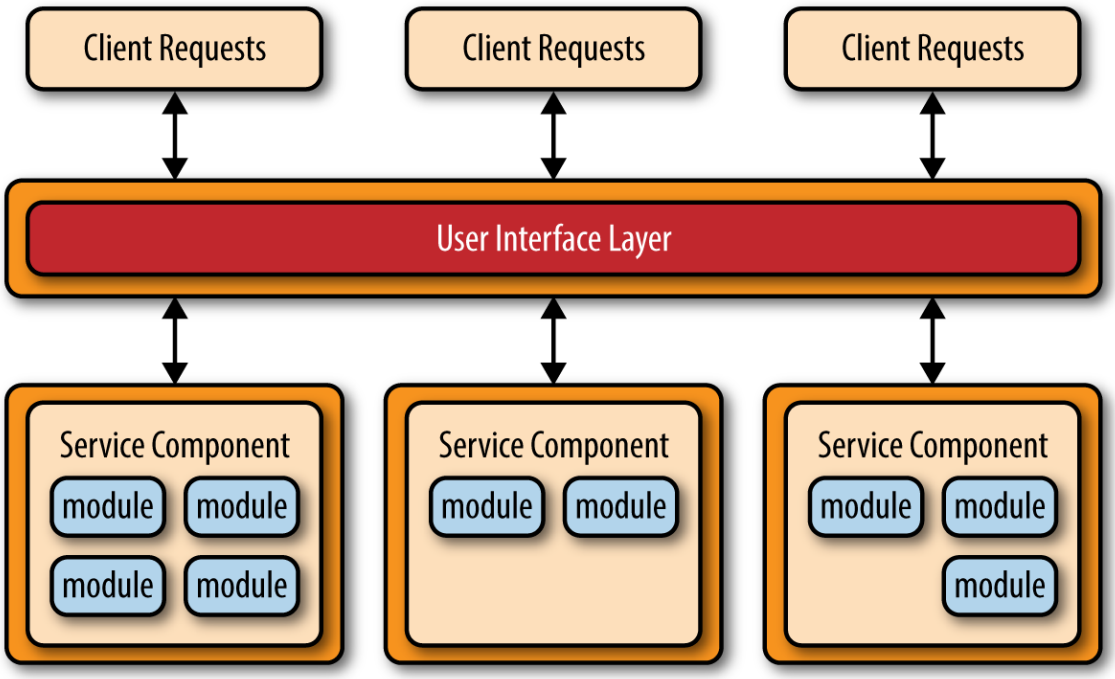

Figura 3: Modelo básico da Arquitetura de Microsserviços

Fonte: Software Architecture Patterns [25]

3.3 Características das Arquiteturas Baseadas em Microsserviços

A Figura 3 é uma representação básica de uma arquitetura baseada em microsserviços. Assim como qualquer outro tipo de arquitetura, ela tem suas próprias características, as quais proporcionam vantagens e desvantagens ao negócio (vistas à frente na seção 3.5). Podem ser vistos como traços principais [26]:

1. **Componentização:** Como característica mais aparente, é observado que, diferente de outras formas arquiteturais, a arquitetura baseada em microsserviços propõe que as funcionalidades de uma aplicação sejam divididas em serviços menores, contanto que haja comunicação entre eles, e que cada um possa ser executado independente de outro;

2. Contexto limitado: Do inglês *Bounded Context*, esse princípio é um dos mais importantes nesse tipo de arquitetura. É fundamental que cada componente execute apenas uma função, de forma satisfatória, do sistema maior;

3. Desenhado visando falhas: Como consequência da componentização, a estrutura dos microsserviços tem que ser confiável e tolerante a falhas. Caso ocorra alguma falha em algum componente, não haverá impactos em outros componentes, e ocorrendo, será de mínima importância;

4. Tratados como produtos: Também em decorrência da componentização, cada componente é visto, na maioria das vezes, como um produto único, que ao fim do desenvolvimento, quando o item é entregue, tem-se o objetivo como concluído e grupo de desenvolvimento é desfeito;

5. Gerenciamento de dados descentralizados: Diferente da arquitetura monolítica, os microsserviços têm características de sistemas distribuídos. Faz-se necessário o gerenciamento de transações entre eles, e o acompanhamento de diferentes bancos de dados. Caso seja usado apenas um banco de dados compartilhado, isso violaria a independência entre os componentes;

6. Comunicação entre componentes: Contrapondo-se à arquitetura monolítica, onde a comunicação entre as "partes" do sistema é feita de forma mais simples, na arquitetura baseada em microsserviços é importante que seja estabelecido um meio de comunicação entre os componentes, assegurando que não haja dependência entre eles;

7. Complexidade: Por ser um sistema distribuído, e devido à divisão em componentes, é natural que esse tipo de arquitetura necessite de mais recursos para realizar diversas atividades. Por ter, muitas vezes, diversos processos sendo executados, é fundamental desenvolver mecanismos de execução de testes, monitoramento, balanceamento, comunicação e determinar uma infraestrutura para que esses mecanismos sejam executados;

8. Design evolutivo: Pelo fato dos elementos serem componentizados e possuírem independência entre si, essa arquitetura permite que a aplicação se comporte de forma evolutiva, onde novas funcionalidades podem ser adicionadas e funções obsoletas podem ser removidas. Dessa forma, é possível realizar mudanças de forma que haja mínimo impacto entre os elementos.

3.4 Benefícios e Desafios da Abordagem de Microsserviços

Diante das características apresentadas na sub-seção anterior, diversos impactos são trazidos por essa estrutura ao negócio, e cada um traz consigo benefícios e desafios que precisam ser ponderados no momento em que é pensado sobre a adesão da estrutura arquitetural. Dessa forma, são apresentados esses efeitos nos itens a seguir.

3.4.1 Benefícios

Para se adotar um modelo de arquitetura é intuitivo que ela tenha que trazer benefícios ao sistema como um todo. E com essa estrutura de arquitetura não é diferente. Esse formato é capaz de trazer vantagens, tanto técnicas, quanto negociais, agregando valor ao resultado final.

Entre os benefícios, é possível citar [23] [26] [27]:

- **Forte modularização:** Como consequência da característica de componentização, os módulos dos microsserviços desempenham suas funções havendo comunicação com outros componentes apenas por meio de interfaces, diminuindo a dependência entre eles;
- **Facilita a substituição das partes e o desenvolvimento em diferentes tecnologias:** Por serem serviços distintos, com mínima interação com outros componentes e por serem acoplados a uma interface, essa arquitetura facilita a troca de módulos, caso haja mudanças, ou, seja necessária haver a troca, além de permitir que cada serviço seja desenvolvido em diferentes tecnologias;
- **Códigos menos complexos:** Diferindo dos sistemas monolíticos, onde o sistema executa todas as funções e pode haver milhões de linhas de código para que isso seja realizado, na arquitetura de microsserviços os módulos executam apenas a função a eles designada (característica de Contexto Limitado, vista na sub-seção 3.3) e, consequentemente, diminuindo o tamanho e a complexidade dos códigos;
- **Falhas isoladas:** Como cada componente tem dependência mínima de outro, caso haja algum problema, esse acontecimento vai ter maior impacto somente no módulo em que aconteceu;
- **Desenvolvimento contínuo:** O desenvolvimento de partes especializadas em realizar apenas uma função facilita que o sistema esteja em constante crescimento;
- **Facilidade de escalar os serviços:** Os microsserviços são oferecidos através de interfaces ligadas a uma rede, a qual pode ser acessada de diversos locais e dispositivos.

3.4.2 Desafios

Por outro lado, se os provedores do negócio decidirem adotar essa estratégia de arquitetura, é necessário que eles tenham em mente que além dos benefícios que a empresa pode ter, serão trazidos, também, alguns desafios que eles devem ficar em alerta para que a estrutura não se torne um grande problema. Como aspectos a serem observados, são destacados os seguintes pontos [23] [26] [27]:

- **Complexidade:** A partir do momento em que a decisão de usar uma arquitetura distribuída é feita, o aumento na complexidade é inevitável. E com

a abordagem de microsserviços não é diferente. Mesmo que a estrutura dos componentes seja, de certa forma, mais simples que nos sistemas monolíticos, ao agregar muitas dessas estruturas, a complexidade se torna maior;

- **Teste:** Se há, de um lado, nos testes unitários, menor complexidade, do outro lado, nos testes de integração, a dificuldade aumenta. Pois, diferentemente dos sistemas monolíticos, onde a integração é feita em poucas aplicações, no modelo de microsserviços, a integração é feita em diversos componentes;
- **Consistência de dados:** Como cada componente é autônomo e responsável por seu próprio banco de dados, a consistência dos dados pode se tornar um problema;
- **Monitoramento:** Da mesma forma que nos testes, os microsserviços trazem benefícios, eles também trazem questões sobre monitoramento a serem avaliadas. Verificar a saúde de um microsserviço isolado é, relativamente, fácil. A grande preocupação é avaliar a saúde do sistema como um todo, além de monitorar falhas e seus possíveis impactos;
- **Latência e congestionamento na rede:** Isso se torna possível devido ao uso de grandes números de componentes, onde esses se comunicam através de requisições apenas por meio da rede;
- **Versionamento e atualização:** Por estarem em partes separadas, cada componente pode sofre atualizações a qualquer momento, possibilitando a incompatibilidade entre versões, além de, por exemplo, bibliotecas usadas por diversos serviços serem atualizadas e se tornarem incompatíveis com eles;
- **Tamanho do componente:** É necessário ter noção da funcionalidade do módulo, para, caso seja preciso, dividir o componente em partes menores, com objetivo de que cada parte execute uma função diferente;
- **Manter a coordenação entre os módulos:** Como cada parte possui independência, é preciso que seja coordenada a execução dos componentes, visando cumprir o objetivo final de forma satisfatória;
- **Custo:** Por se tratar de uma arquitetura distribuída, onde cada componente tem sua própria infraestrutura, a manutenção desses serviços pode se tornar mais custosa.

3.5 A Relação entre Microsserviços e APIs

Uma maneira apropriada para observar melhor o uso da integração entre microsserviços e APIs, é a partir do exame dos diversos padrões de arquitetura de software no mercado atualmente, estudando suas vantagens, desvantagens e situações as quais cada tipo se aplica melhor. Existem quatro modelos de arquitetura mais usuais: em camadas, baseada em eventos, baseada em *microkernel* e a baseada em microsserviços, seguindo o livro "*Software Architecture Patterns"*, de Mark Richards [25]. Em função da exiguidade de espaço, neste trabalho se examina apenas o modelo

onde é usada uma API para fazer a integração entre os componentes e o *Front-End* da aplicação.

É válido deixar claro que a comunicação entre os componentes é feita, na maioria das vezes, através de uma API REST, usando o padrão de API *Gateway* (melhor apresentado na seção 4.4 à frente), do inglês API *Gateway Pattern*, como forma de arquitetura entre eles [28], a qual está contida no modelo apresentado na seção 4.1.1. É importante também entender que um API Gateway de microsserviços é um mecanismo que faz a conexão entre o *Front-End* da aplicação e os módulos. Porém ele é mais que apenas um ponto de entrada. O mecanismo é responsável por orquestrar as requisições entre os componentes junto à aplicação, além de abstrair a complexidade dos componentes, criar pontos de acesso aos módulos, prover segurança, e fazer a comunicação entre as partes, usando uma linguagem comum [29].

4 I ARQUITETURA BASEADA EM MICROSSERVIÇOS

4.1 Visão Geral

O modelo de arquitetura, que é apresentado nessa sub-seção, vem ganhando espaço nas empresas por ser uma alternativa viável às aplicações monolíticas.

A evolução dessa arquitetura se deu para resolver problemas associados a outros modelos. Pelo lado da estrutura monolítica, evoluiu-se para corrigir o problema na adição de funcionalidades, para buscarem seguir o conceito das entregas de funções de forma contínua, e para facilitarem o desenvolvimento e os testes; e, do lado da arquitetura SOA, mesmo tendo seus benefícios ao negócio, o padrão de microsserviços busca melhorar a complexidade interna, facilitando o desenvolvimento, reduzindo o custo (tanto de desenvolvimento, quanto de manutenção e testes) e a facilitando no entendimento da estrutura [25].

Independentemente da topologia escolhida, é necessário ter um entendimento sobre os principais conceitos associados ao modelo geral. Esses conceitos já foram apresentados na seção 3, porém são retomados, de forma mais simplificada, para seguir com o raciocínio. Dessa forma, é importante ter conhecimento de que cada unidade tem uma função distinta, e que são entregues como produtos únicos. Outro ponto a levar em consideração, é o fato da sua natureza distribuída, onde cada componente é independente dos outros.

Existem diversas formas de uma arquitetura ser implementada; porém, três modelos têm maior relevância entre todos os outros [25]: I) os baseados em aplicações REST, II) os baseados em mensagens centralizadas e III) os baseados em API REST.

4.1.1Topologia Baseada em REST API

Essa topologia tem maior utilidade em aplicações web que disponibilizam suas funções através de uma interface de programação de aplicações (API). Esse modelo (mostrado na Figura 8) é constituído de diversos componentes independentes, com funções distintas, os quais são acessados por uma interface REST através de uma API.

Todos os componentes devem se comunicar com os outros, assim como com as aplicações e seu próprio banco de dados, em tempo real. Assim sendo, todos eles devem possuir uma interface, e é nesse ponto onde se dá a importância de uma API. Uma API REST proporciona um modelo lógico e simplificado para construir interfaces entre os microsserviços. Por esse motivo, as APIs devem ser baseadas em mensagens, para não afetar a interoperabilidade dos serviços quando ocorrer mudanças ou atualizações, e dirigidos à hipermídia, significando que também podem ser enviadas descrições de possíveis ações, além de, apenas, dados, dessa forma maximizando o baixo acoplamento [32] [33].

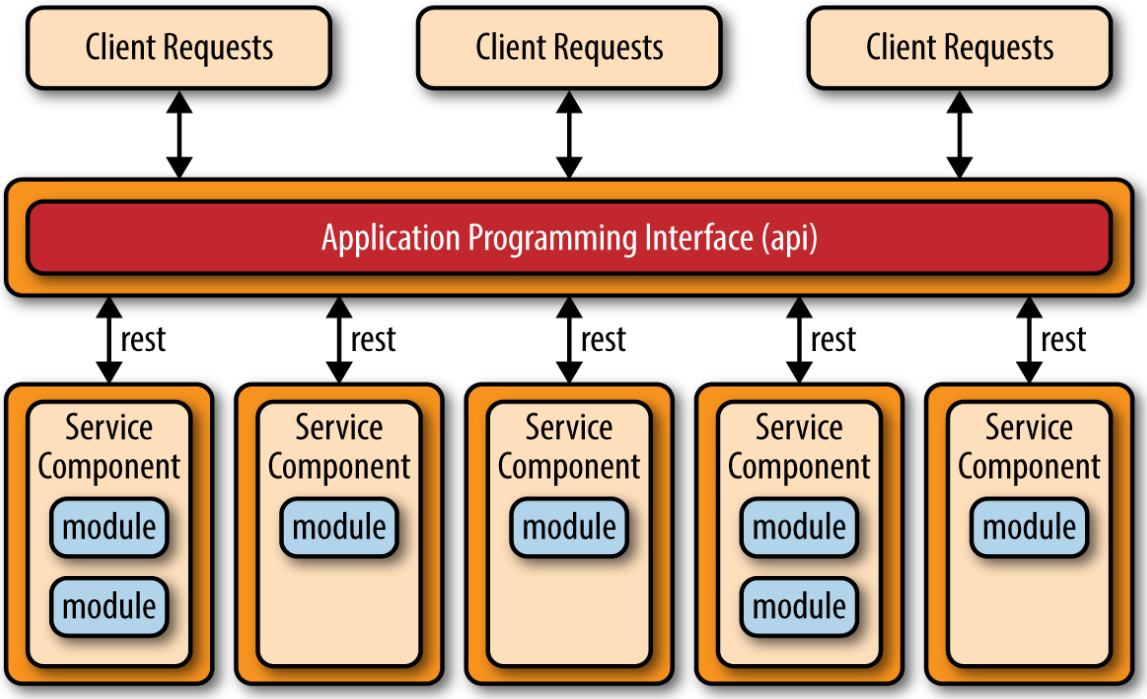

Figura 8: Topologia baseada em API REST

Fonte: Software Architecture Patterns [25]

4.1.1.1 API Gateway

Para essa topologia e para a próxima apresentada (sub-seção 4.1.3) é interessante apresentar pontos importantes sobre um API *Gateway* e como ele influencia nesse modelo.

Um API *Gateway,* mencionado na sub-seção 3.5, é um dispositivo de rede que atua de forma que as APIs não tenham que interagir diretamente com os clientes. Ele

é ponto central onde estão todas as abstrações das funcionalidades e as administra através de políticas [34]. Dessa forma, todas as requisições dos clientes terão que passar, obrigatoriamente, por eles.

Por estarem conectados a todas as interfaces (também chamadas de APIs) dos microsserviços, esses *Gateways* são usados comumente para fornecer três funções essenciais à arquitetura [35]:

1. **Segurança:** Pelo alto desacoplamento, tendo como consequência uma grande liberdade entre os microsserviços, é importante que grandes aplicações, com diversos módulos, todas as interações com clientes externos sejam feitas através de APIs, assim como na forma de comunicação entre os componentes, pois, para cada elemento, outro componente também é um "cliente externo". E um *gateway* é usado para proteger os pontos de conexão dessas APIs;

2. **Orquestração:** Como já foi referido na seção 3 sobre os microsserviços, eles são desenvolvidos para exercer uma função única, de maneira satisfatória. Portanto, é imprescindível que haja um mecanismo de organização nas requisições às interfaces dos componentes, para evitar o problema de múltiplas chamadas;

3. **Roteamento:** Nesse ponto, o API *Gateway* tem a função de esconder a complexidade de roteamento entre a aplicação-cliente e os microsserviços.

4.1.2 Topologia Baseada em Aplicações REST

Esse modelo difere do anterior no quesito de que, contrário ao uso de uma API REST, as requisições dos clientes são recebidas por aplicações *web*. Como é ilustrada na Figura 9 à frente, a interface entre os clientes e os serviços é feita por aplicações *web* que, remotamente, acessam os serviços através de uma interface REST. Interfaces REST são interfaces que se comportam de acordo com o modelo, de forma assíncrona, sem estados definidos e sem se preocupar com detalhes de desenvolvimento dos componentes [10]. Outra diferença desse padrão para o modelo da seção 4.1 é no tamanho dos componentes, nessa topologia, os componentes tendem a serem maiores, pois podem apresentar mais funções do negócio. Esse padrão é mais comum em aplicações de pequeno e médio porte, com menos complexidade.

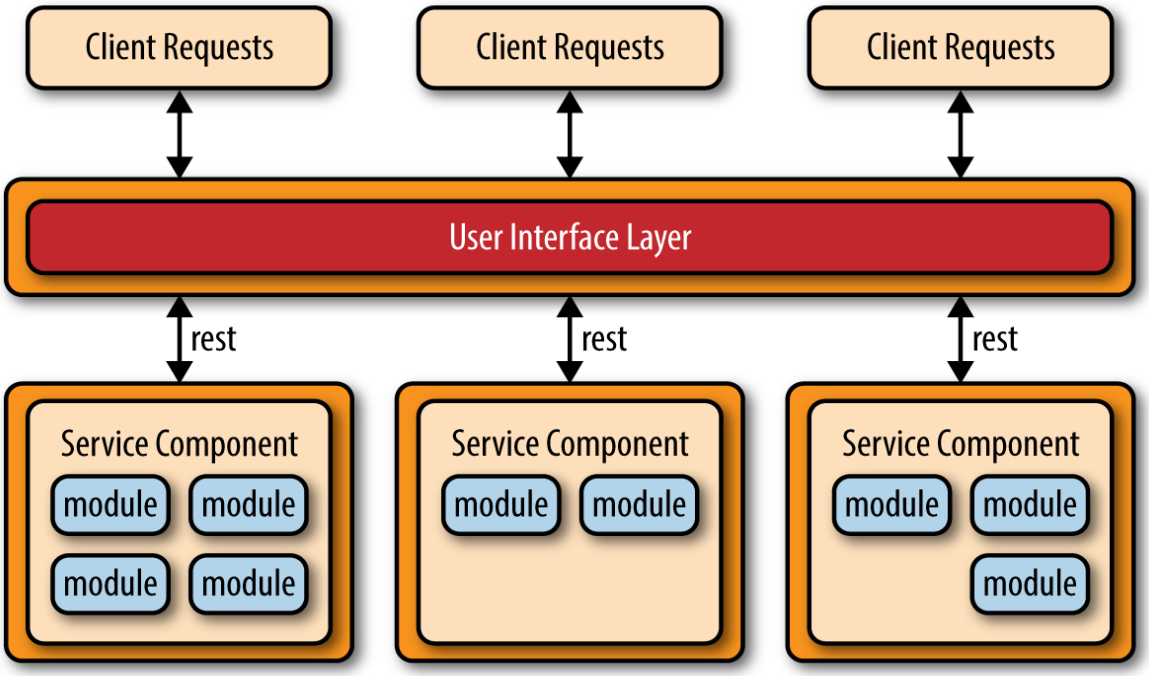

Figura 9: Topologia baseada em Aplicações REST

Fonte: Software Architecture Patterns [25]

4.1.3 Topologia Baseada em Mensagens Centralizadas

Outra topografia comum de microsserviços é o modelo baseado em mensagens centralizadas (Figura 10). Esse formato é semelhante ao apresentado anteriormente, porém há uma troca no acesso à camada de interface. Nesse padrão, o acesso é feito através de um corretor de mensagens centralizado, no lugar de uma aplicação *web*. O corretor não executa nenhuma função de orquestração ou roteamento, ele é só um meio de acesso remoto aos componentes. Esse modelo pode ser usado em casos onde seja necessário um controle mais sofisticado na camada de transporte entre a interface e os componentes.

Como benefícios sobre a topologia baseada em aplicações REST, é dada a importância aos mecanismos de fila de mensagens, assim como na assincronia entre elas, no monitoramento, no tratamento de erros e na escalabilidade.

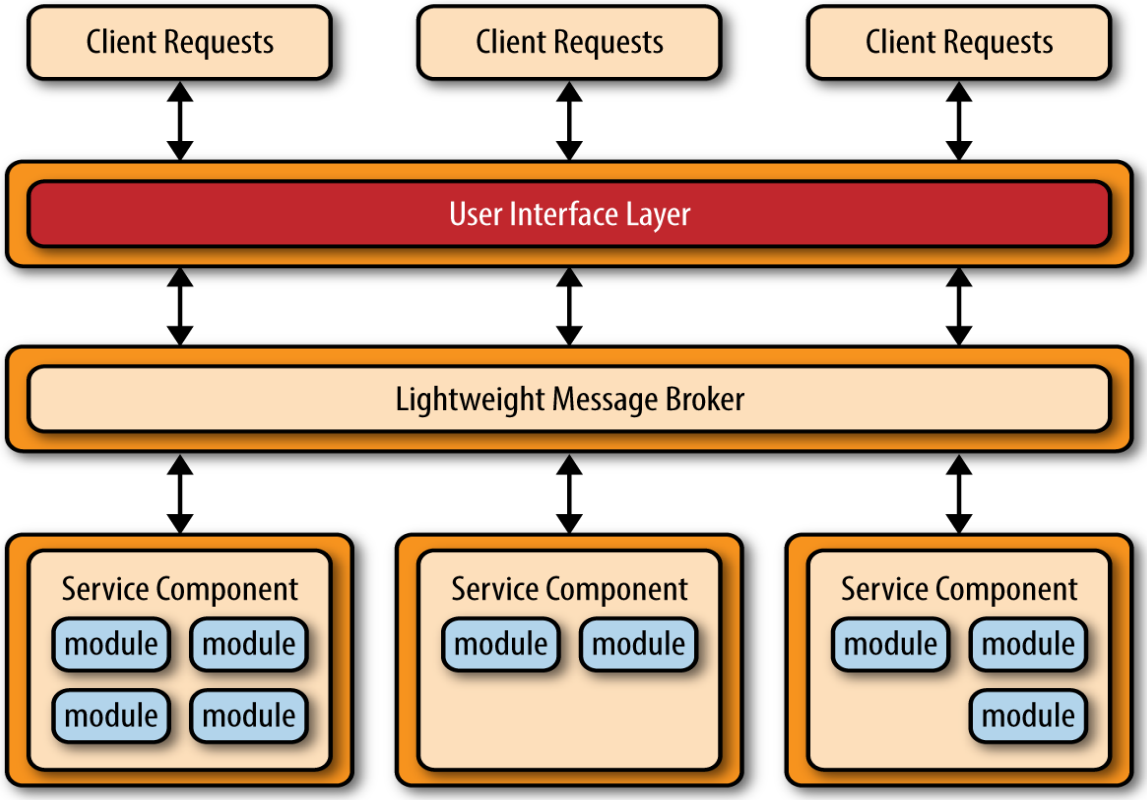

Figura 10: Topologia baseada em Mensagens Centralizadas

Fonte: Software Architecture Patterns [25]

4.2 Considerações e Análise

Devido à maioria das funções de um sistema baseado na arquitetura de microsserviços estar contida em módulos menores, esse modelo pode resolver diversos benefícios de facilidade de desenvolvimento, escalabilidade, robustez e suporte, encontrados em aplicações monolíticas ou em aplicações baseadas em SOA. Outros benefícios, assim como já foi citado na seção 3, são a possibilidade entrega de funções de forma contínua e a possibilidade de executar testes, fazer e reagir a mudanças em componentes isolados, sem afetar o sistema todo.

Todavia, por ter a mesma natureza distribuída da Arquitetura Baseada em Eventos (não tratada neste trabalho), ela compartilha dos mesmos problemas de manutenção, disponibilidade, autenticação, criação de contratos e governança dos componentes.

5 I CONCLUSÕES

O investimento em informatização do negócio por parte das empresas tem avançado cada vez mais em decorrência da crescente dependência de tecnologia para solucionar conflitos e pelo fácil acesso à internet. O uso de novas tecnologias tem contribuído para que as aplicações possam executar funções, de maneira a resolver problemas, de forma mais fácil e eficiente, e, dessa forma, direta ou indiretamente, gerar receita no negócio.

As APIs surgem com o objetivo de integrar diversas plataformas e equipamentos ao sistema responsável por solver as questões, e fazer com que desenvolvedores terceiros criem produtos associados ao seu serviço. Através do uso das APIs, a comunicação é feita sem que haja intervenção dos usuários, onde esses podem ser pessoas ou outros sistemas ou serviços.

Os Microsserviços, por outro lado, aparecem como alternativa para um modelo de arquitetura muito comum atualmente, a arquitetura em camadas. Por ter natureza distribuída, esse modelo propõe a quebra da estrutura monolítica em partes independentes, provendo maior desacoplamento entre os módulos, os quais passam a ser responsáveis por, na maioria das vezes, apenas uma função, porém sem perder a comunicação entre eles.

O presente trabalho buscou fazer uma breve análise relacionada ao uso simultâneo das APIs e dos Microsserviços a partir de uma Arquitetura Baseada em Microsserviços, assim como apresentar seus impactos nos negócios. Foi possível observar que, na economia moderna, o uso dessas tecnologias se tornou um ponto forte e um diferencial para obter sucesso em um negócio. Associando os pontos fortes das duas tecnologias, as aplicações se tornam mais valiosas, tanto economicamente (para os provedores e afiliados, gerando receita e benefícios ao sistema) quanto na eficiência na resolução de problemas (e neste aspecto, é favorável para os três players integrantes da economia: o fornecedor, o parceiro e usuário final). Pelo baixo acoplamento e pela grande facilidade de integração, a combinação dessas duas tecnologias possibilita entregas contínuas de funcionalidades com maior aptidão. Esse conjunto de atributos viabiliza o constante crescimento do sistema, a partir dessas entregas, e traz valor a todas as partes interessadas no negócio.

REFERÊNCIAS

[1] JACOBSON, Daniel; BRAIL, Greg; WOODS, Dan. *APIs: A Strategy Guide*. 1. ed. Estados Unidos da América: O'Reilly Media, 2012. 136 p.

[2] API *Evangelist History of APIs*. Disponível em: <https://history.apievangelist.com>. Acesso em: 07 ago. 2017.

[3] UMA BREVE HISTÓRIA DAS APIS COM A PLUGA. Disponível em: <https://mundoapi.com.br/materias/uma-breve-historia-das-apis-com-a-pluga/>. Acesso em: 07 ago. 2017

[4] *A brief history of microservices*. Disponível em: <http://blog.leanix.net/en/a-brief-history-of-microservices>. Acesso em: 08 ago. 2017

[5] *What are Microservices*?. Disponível em: <https://martinfowler.com/microservices/#what>. Acesso em: 16 dez. 2017

[6] Exemplos de APIs que você usa todo dia e não sabe. Disponível em: <https://sensedia.com/blog/apis/exemplos-de-apis/>. Acesso em: 16 dez. 2017

[7] Microservice Architecture. Disponível em: <http://microservices.io/articles/whoisusingmicroservices.

html>. Acesso em: 16 dez. 2017

[8] *API Directory Eclipses 17,000 as API Economy Continues Surge*. Disponível em: <https://www.programmableweb.com/news/programmableweb-api-directory-eclipses-17000-api-economy-continues-surge/research/2017/03/13>. Acesso em: 08 ago. 2017

[9] *What are APIs? (The Technology Perspective)*. Disponível em: <https://medium.com/@robert.broeckelmann/what-are-apis-the-technology-perspective-ca7e33d383c1>. Acesso em: 11 ago. 2017

[10] FERREIRA, Cleber de F.; MOTA, Roberto Dias. COMPARANDO APLICAÇÃO WEB SERVICE REST E SOAP. S.l. Disponível em: <http://web.unipar.br/~seinpar/2014/artigos/pos/Cleber_de_F_Ferreira_Roberto_Dias_Mota%20(1).pdf>. Acesso em: 12 ago. 2017

[11] *Integration architecture: Comparing web APIs with service-oriented architecture and enterprise application integration*. Disponível em: <https://www.ibm.com/developerworks/websphere/library/techarticles/1503_clark/1305_clark.html>. Acesso em: 12 ago. 2017

[12] O que é API?. Disponível em: <https://www.tecmundo.com.br/programacao/1807-o-que-e-api-.htm>. Acesso em: 12 ago. 2017

[13] NIJIM, Sharif; PAGANO, Brian. *APIs For Dummies: Apigee Special Edition*. Estados Unidos da América: John Wiley & Sons, 2014. 44 p.

[14] DE, Brajesh. API *Management: An Architect's Guide to Developing and Managing APIs for Your Organization*. 1. ed. **Índia**: Apress, 2017. 195 p.

[15] *The Why and How of APIs: The Internal API Model*. Disponível em: <https://apigee.com/about/blog/digital-business/why-and-how-apis-internal-api-model>. Acesso em: 13ago. 2017

[16] *Private, Partner or Public: Which API Strategy Is Best For Business*?. Disponível em: <https://www.programmableweb.com/news/private-partner-or-public-which-api-strategy-best-business/2014/02/21>. Acesso em: 13 ago. 2017

[17] *The Why and How of APIs: The Open API Model*. Disponível em: <https://apigee.com/about/blog/technology/why-and-how-apis-open-api-model>. Acesso em: 19 ago. 2017

[18] *Benefits of APIs*. Disponível em: <https://api-all-the-x.18f.gov/pages/benefits_of_apis/>. Acesso em: 26 ago. 2017

[19] *What are the Benefits of APIs*?. Disponível em: <https://www.programmableweb.com/news/what-are-benefits-apis/analysis/2015/12/03>. Acesso em: 26 ago. 2017

[20] *What are some challenges faced when building APIs to expose and integrate systems of large enterprises*?. Disponível em: <https://www.quora.com/What-are-some-challenges-faced-when-building-APIs-to-expose-and-integrate-systems-of-large-enterprises>. Acesso em: 16 dez. 2017

[21] *Application Programming Interface (API): What it is and How it is Utilized in the Transportation Industry*. Disponível em: <http://ltxsolutions.com/application-programming-interface-api-transportation/>. Acesso em: 16 dez. 2017

[22] *Microservices Architectures*. Disponível em: <https://www.happiestminds.com/Insights/microservices/>. Acesso em: 02 set. 2017

[23] GOETSCH, Kelly. *Microservices for Modern Commerce*. 1. ed. Estados Unidos da América: O'Reilly Media, 2016. 68 p.

[24] WOLFF, Eberhard. Microservices: *Flexible Software Architecture*. 1. ed. Estados Unidos da América: Pearson Education, 2017. 395 p.

[25] RICHARDS, Mark. *Software Architecture Patterns*. 3. ed. Estados Unidos da América: O'Reilly Media, 2016. 47 p.

[26] *Microservices*. Disponível em: <https://martinfowler.com/articles/microservices.html>. Acesso em: 10 set. 2017

[27] *Microservices architecture style*. Disponível em: <https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices>. Acesso em: 19 set. 2017

[28] *Microservices & API Gateways, Part 1: Why an API Gateway*?. Disponível em: <https://www.nginx.com/blog/microservices-api-gateways-part-1-why-an-api-gateway/#WhyanAPIGateway>. Acesso em: 16 dez. 2017

[29] *Using an API Gateway in Your Microservices Architecture*. Disponível em: <https://smartbear.com/learn/api-design/api-gateways-in-microservices/>. Acesso em: 16 dez. 2017

[30] *Microsoft Application Architecture Guide*, 2nd Edition. Disponível em: <https://msdn.microsoft.com/en-us/library/dd673617.aspx>. Acesso em: 04 out. 2017

[31] *Understand Event-Driven Software Architecture*. Disponível em: <https://msdn.microsoft.com/en-us/library/dd673617.aspx> Acesso em: 04 out. 2017

[32] *API DESIGN 304: API DESIGN FOR MICROSERVICES*. Disponível em: <http://www.apiacademy.co/resources/api-design-304-api-design-for-microservices/>. Acesso em: 10 out. 2017

[33] *API MANAGEMENT 101: API MANAGEMENT BASICS*. Disponível em: <http://www.apiacademy.co/resources/api-management-101-api-management-basics/>. Acesso em: 10 out. 2017

[34] *API MANAGEMENT 302: USING AN API GATEWAY IN MICROSERVICE ARCHITECTURE*. Disponível em: <http://www.apiacademy.co/resources/api-management-302-using-an-api-gateway-in-microservice-architecture/>. Acesso em: 10 out. 2017

SOBRE O ORGANIZADOR

Marcos William Kaspchak Machado - Professor na Unopar de Ponta Grossa (Paraná). Graduado em Administração - Habilitação Comércio Exterior pela Universidade Estadual de Ponta Grossa. Especializado em Gestão industrial na linha de pesquisa em Produção e Manutenção. Doutorando e Mestre em Engenharia de Produção pela Universidade Tecnológica Federal do Paraná, com linha de pesquisa em Redes de Empresas e Engenharia Organizacional. Possui experiência na área de Administração de Projetos e análise de custos em empresas da região de Ponta Grossa (Paraná). Fundador e consultor da MWM Soluções 3D, especializado na elaboração de estudos de viabilidade de projetos e inovação.

Agência Brasileira do ISBN
ISBN 978-85-7247-202-9