Computação em Nuvem

Modelos de arquitetura em nuvem

Você sabia que seu material didático é interativo e multimídia? Isso significa que você pode interagir com o conteúdo de diversas formas, a qualquer hora e lugar. Na versão impressa, porém, alguns conteúdos interativos ficam desabilitados. Por essa razão, fique atento: sempre que possível, opte pela versão digital. Bons estudos!

De forma geral, a arquitetura de um sistema é a especificação de quais são os seus componentes e como devem ser conectados. Sabemos que as aplicações em nuvem são sistemas que envolvem vários componentes de software. A comunicação entre esses elementos é feita por meio da Internet, então o desempenho e a segurança dependem de um modelo de arquitetura adequado. Além disso, as aplicações em nuvem devem ser escaláveis para lidar com um volume crescente de dados e clientes, o que torna necessário incluir mecanismos de replicação e balanceamento de carga.

Nesta webaula, vamos conhecer os principais modelos de arquitetura, são eles: a arquitetura multicamadas, a arquitetura de microsserviços e a Computação sem Servidor (*Serverless*). Também vamos introduzir o conceito de *Edge Computing*, um novo paradigma para lidar com problemas de escalabilidade de soluções em nuvem.

Arquitetura multicamadas

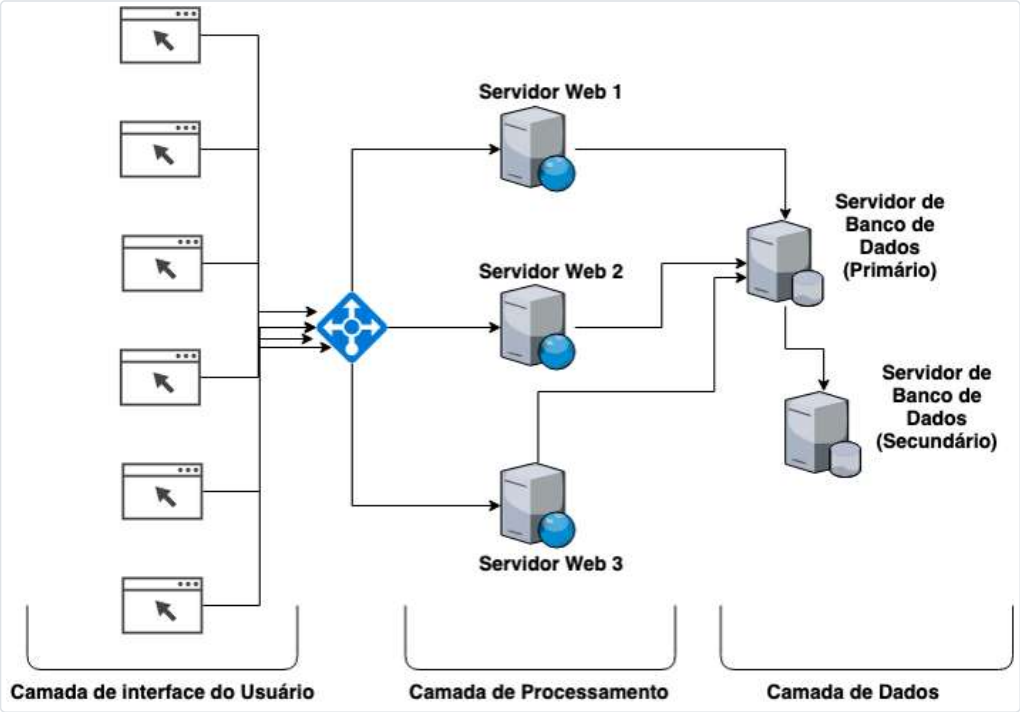
Nesse modelo de arquitetura, a aplicação é dividida em várias camadas, sendo cada uma delas responsável por um conjunto específico de funcionalidades. Os componentes de uma camada podem interagir com os componentes das camadas vizinhas, além de serem executados em servidores diferentes para melhorar o desempenho. Nesse caso, temos uma separação física entre as camadas.

Em geral, as aplicações multicamadas são divididas em três:

- **Camada de interface do usuário** (controle da interação com o usuário).
- **Camada de processamento** (implementação da lógica das principais funcionalidades da aplicação).
- **Camada de dados** (armazenamento persistente de dados).

Na figura a seguir, temos o exemplo de uma aplicação Web baseada na arquitetura em camadas.

Exemplo de uma aplicação com arquitetura em camadas e componentes replicados



Fonte: elaborada pelo autor.

Nesse caso, o navegador web executa a **camada de interface** com o usuário, já na **camada de processamento**, o servidor web responde às requisições de acordo com as regras de negócio. Na **camada de dados**, um Sistema Gerenciador de Banco de Dados administra o armazenamento das informações do sistema.

> Em um ambiente de nuvem, podemos usar diversos serviços para hospedar uma aplicação com essa arquitetura. Por exemplo, podemos alocar máquinas virtuais ou contêineres para executar um servidor web (na camada de processamento) e usar serviços de Bancos de Dados em Nuvem para persistências das informações (na camada de dados).

Os provedores de nuvem também oferecem serviços para replicação dinâmica dos componentes. No exemplo ilustrado na figura, há três réplicas do servidor web e duas réplicas do servidor de banco de dados. Esse tipo de replicação pode ser automatizado em ambientes de nuvem. Além disso, os provedores oferecem mecanismos para balanceamento de carga. A mesma figura também demonstra o balanceamento de carga das requisições dos clientes entre as réplicas do servidor Web. Se houver falha em um dos servidores, as demais réplicas continuam a atender as requisições. Além disso, a carga de trabalho é dividida entre as réplicas, o que resulta em menor tempo de resposta aos clientes. Com isso, a aplicação dividida em camadas melhora o grau de desempenho e confiabilidade, mesmo que a arquitetura ainda seja cliente-servidor. Os provedores de Computação em Nuvem permitem a alocação ou liberação automática de réplicas de acordo com as variações no volume de requisições dos clientes.

Novas arquiteturas de serviços web

Em geral, as principais funcionalidades de uma aplicação são implementadas na camada de processamento, por exemplo, na forma de um Serviço Web (*Web Service*). Esse serviço é denominado **monolítico** quando todos os seus módulos funcionais estão implementados em um único componente de software. Se a aplicação possui muitas funcionalidades, o serviço monolítico pode apresentar alguns problemas como:

- Ocupação de muito espaço de memória.
- Dificuldade de implementar correções e de evolução do software.
- Alto custo de replicação, etc.

Diante desses problemas, novas arquiteturas de serviços web foram propostas para melhorar o desempenho e a escalabilidade, assim como facilitar a replicação. Essas abordagens se baseiam na ideia de dividir o serviço web monolítico em diversos componentes de software independentes. Entre as principais abordagens, podemos citar a **arquitetura de microsserviços** (DRAGONI *et al.*, 2017) e a **arquitetura *Serverless*** (Computação sem Servidor) (BALDINI *et al.*, 2017), cujo principal exemplo é o modelo Função como Serviço (FaaS – Function as a Service). A figura a seguir ilustra a evolução desses modelos, que se reflete no grau de modularização da aplicação.

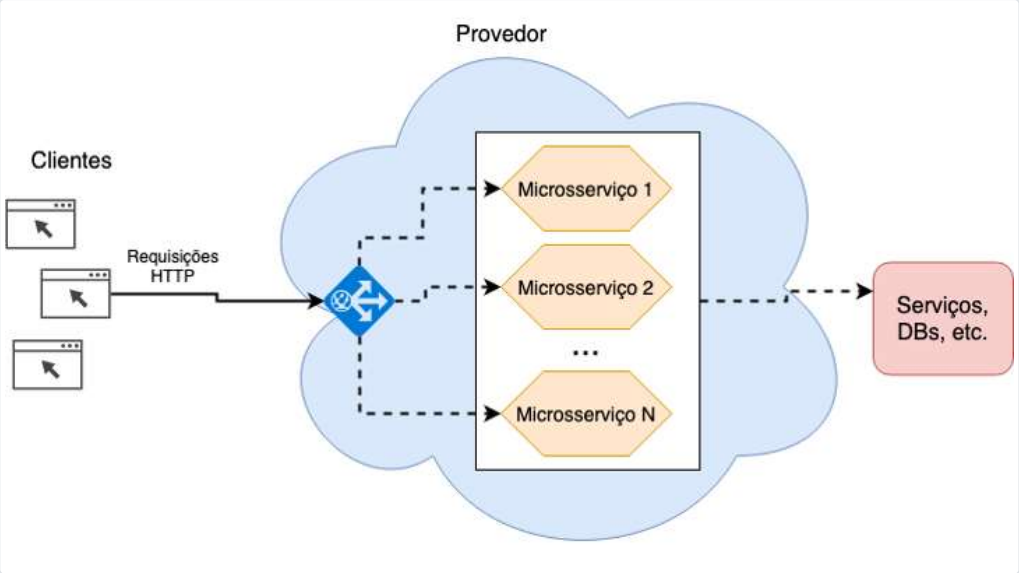
Evolução dos padrões de arquitetura de software como serviço

Fonte: elaborada pelo autor.

Modelo de microsserviços

No modelo de microsserviços, cada um dos módulos funcionais de uma aplicação monolítica vira um serviço menor e especializado, denominado microsserviço. Cada mircosserviço é um componente de software independente. Assim, ele pode ser executado como um serviço web em um contêiner ou máquina virtual. A figura a seguir exemplifica a arquitetura de microsserviços. Nesse caso, uma aplicação é modularizada em N microsserviços independentes, que podem se comunicar com serviços remotos, como outra aplicação, com sistemas legados ou com um banco de dados (DB – *Database*). A mesma figura também ilustra o papel de um *API Gateway*, que é um componente responsável por redirecionar as requisições dos clientes para o microsserviço apropriado de acordo com a funcionalidade requisitada.

Exemplo de aplicação com arquitetura de microsserviços



Fonte: elaborada pelo autor.

A arquitetura de microsserviços é adequada para aplicações complexas, por exemplo, uma aplicação para consolidação de transações financeiras com cartão de crédito, pois envolve várias etapas e entidades, como lojas, bancos, adquirentes, etc. Nesse caso, a aplicação pode ser modularizada em vários microsserviços específicos que lidam com cada parte do problema.

Como discutido em (DRAGONI *et al.*, 2017), duas características principais definem um microsserviço. Entenda-as a seguir.

Alto grau de coesão das funcionalidades

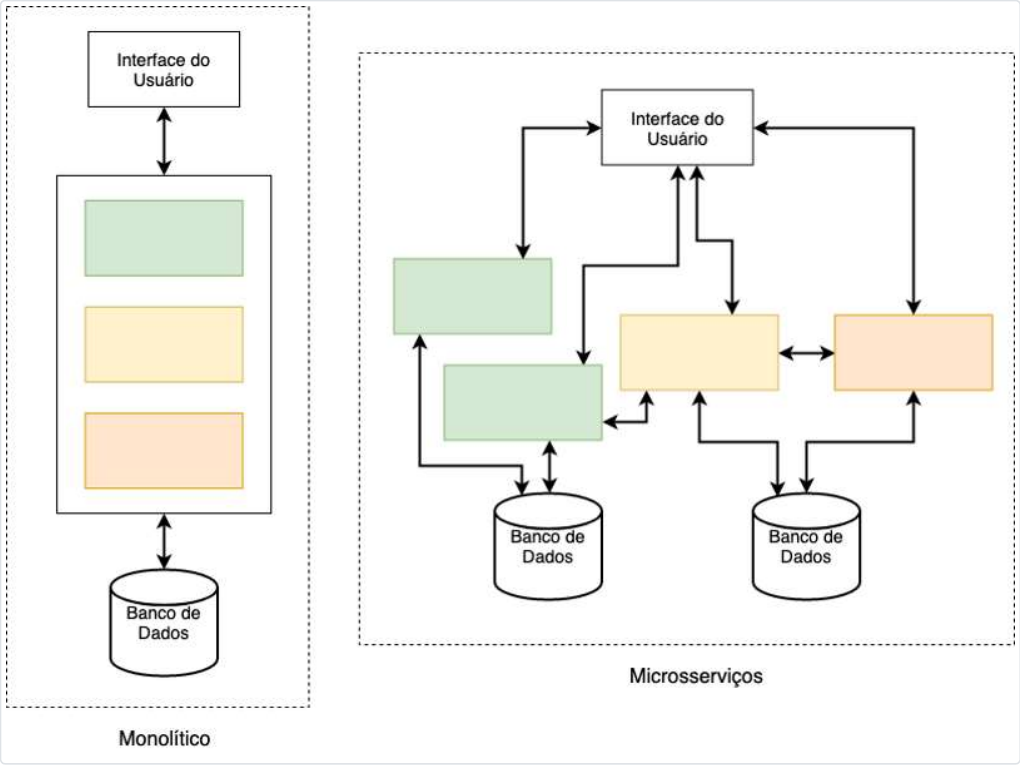
A coesão significa que um microsserviço implementa um conjunto de funcionalidades relacionadas, que dependem umas das outras.

Capacidade de responder requisições

A capacidade de responder requisições implica que os microsserviços, de fato, se comportam como um servidor, ou seja, provêm algum serviço. Eles podem responder requisições de clientes ou requisições de outros microsserviços.

A arquitetura de microsserviços representa aplicações distribuídas que são formadas pela composição de microsserviços independentes. Esse conceito é ilustrado na figura a seguir. Nesse caso, temos um serviço monolítico formado por três módulos funcionais. Em uma arquitetura de microsserviços, cada um desses módulos poderia ser implementado como um (micro) serviço independente. Como mostra a figura, apenas alguns dos microsserviços podem ser replicados e cada um deles pode ter seu próprio banco de dados.

Comparação entre serviço monolítico e microsserviços

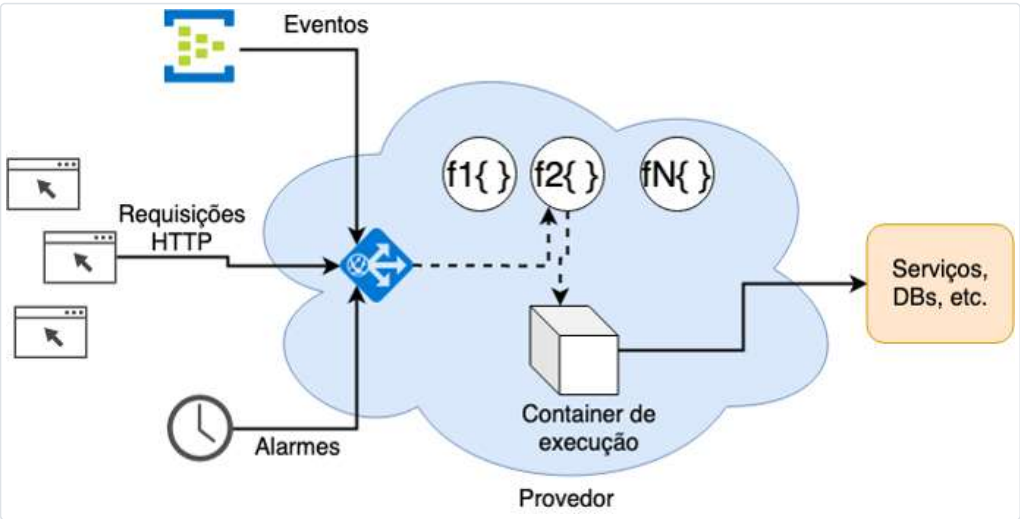


Fonte: elaborada pelo autor.

Computação sem Servidor (*Serverless*)

No modelo Computação sem Servidor (*Serverless*), o nível de granularidade é ainda maior que na arquitetura de microsserviços. O objetivo é explorar o fato de que cada microsserviço pode ainda ser dividido em funções específicas e cada função pode ser invocada de forma independente por um software cliente. No ambiente de nuvem, esse modelo pode ser implementado na forma de **Função como Serviço (*FaaS – Function as a Service*)**. O nome "Computação sem Servidor" remete à ideia de que, neste modelo, o cliente não precisa alocar servidores para a execução de uma aplicação. Isso é feito dinamicamente pelo provedor quando alguma função da aplicação é invocada. Assim, a tarifação no modelo *Serverless* é feita de acordo com o tempo de execução das funções e não pela alocação de recursos (BALDINI *et al.*, 2017). A figura a seguir mostra as principais características dessa arquitetura.

Visão geral da arquitetura *Serverless*



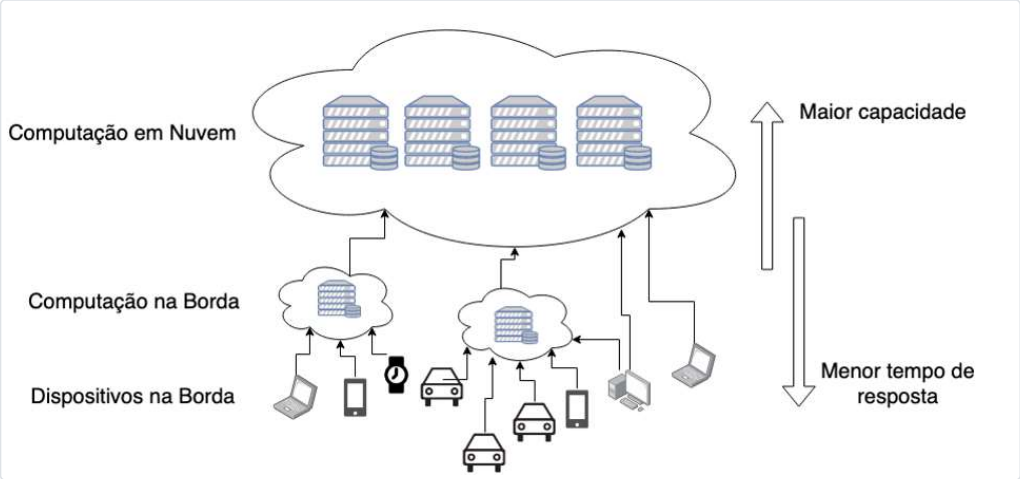
Fonte: elaborada pelo autor.

Cada funcionalidade específica da aplicação precisa ser implementada como uma função isolada que pode receber parâmetros e retornar algum resultado. O código das funções de uma aplicação (f1 ... fN) são enviados para o provedor. A execução de uma função pode ser disparada por diversas formas, como uma requisição web, eventos em outros componentes (como uma inserção em um banco de dados), ou por um alarme (o que permite agendar a execução de uma função). As requisições são enviadas para um gateway, que identifica a função apropriada e aloca os recursos necessários para executá-la (um contêiner, por exemplo). A execução da função pode resultar em resposta a um cliente ou em acesso a outros serviços ou bancos de dados, por exemplo.

Edge Computing

Nas aplicações de Big Data e Internet das Coisas, um grande volume de dados precisa ser transmitido até o provedor para ser processado e, depois, eventuais respostas necessitam ser enviadas até os dispositivos finais. Essas questões motivaram a consolidação do paradigma de Computação nas Bordas (*Edge Computing*), cuja ideia principal é mover o processamento dos dados para a borda da rede (SHI *et al.*, 2016). Uma visão geral da *Edge Computing* é apresentada na figura a seguir. A execução de serviços na borda da rede pode beneficiar algumas soluções, por exemplo, aplicações de mobilidade urbana, que precisam de respostas rápidas para análise de dados do trânsito em uma determinada região.

Visão geral da arquitetura de *Edge Computing*



Fonte: elaborada pelo autor.

O objetivo desse modelo não é substituir a Computação em Nuvem. Na verdade, são abordagens complementares, pois as aplicações que exigem maior capacidade de armazenamento e processamento de dados continuam a ser executadas de forma centralizada no provedor de Computação em Nuvem. Um bom exemplo disso é um sistema que utiliza Aprendizado de Máquina para identificar fraudes em compras com cartão de crédito. A seguir, são descritas algumas importantes aplicações práticas que podem se beneficiar das vantagens do modelo descentralizado da Computação em Bordas:

- Redes de sensores
- Cache de dados
- Edifícios inteligentes (*smart buildings*)

Nesta webaula, você aprendeu muitos conceitos relacionados com a arquitetura de aplicações em nuvem. Além disso, viu como os diferentes modelos podem beneficiar ou impactar negativamente a escalabilidade ou o desempenho dos serviços.