Microsserviços

Matheus Alec Guimaraes Moura – 11811GIN026

Arquitetura monolítica

O conceito de microsserviços se refere a uma arquitetura para construção de softwares que veio para solucionar problemas encontrados na arquitetura clássica, conhecida como **monolítica.** A arquitetura monolítica é o padrão comumente conhecido, onde todos processos necessários para a execução do software se encontram em um único bloco, ou seja, são acoplados como se fossem um único serviço. Este tipo de arquitetura ainda é muito utilizada hoje em dia e possui suas vantagens, como a facilidade de desenvolver, facilidade de projetar, simplicidade para o deploy(enviar código novo para o servidor da aplicação) e outros. Contudo esta arquitetura possui desvantagens com grandes impactos em seu sistema, em termos de qualidade, integridade e manutenibilidade, logo em um sistema que é dado como um único serviço temos que uma pequena alteração pode afetar o sistema como um todo, momentos de pico em alguma funcionalidade do sistema pode prejudicar o desempenho de todo sistema, sistemas maiores com alta complexidade a realização de manutenções(evolutivas, adaptativas ou corretivas) se tornam cada vez mais custosas e demoradas.

Arquitetura de microsserviços

Devido aos problemas encontrados na arquitetura monolítica é que surge a proposta da arquitetura baseada em microsserviços, aqui o software não se comporta como um bloco único que reúne todos seus serviços em um mesmo local, em microsserviços o sistema será dividido em pedaços(serviços) separados e independentes que em conjunto vão representar o seu sistema. O conceito no geral é simples, mas a sua execução pode ser complexa e feita de inúmeras formas. Destaca-se que para um serviço ser enquadrado como um microsserviço ele deve possuir código fonte e camada de dados(banco de dados) logicamente isolado dos outros serviços de seu sistema. Em casos que o serviço é isolado de outros, mas compartilha um banco de dados com outro serviço do mesmo sistema isso é enquadrado no conceito de micromonólitos. Também é muito comum, encontrar um microsserviço que “junta” diversos outros microsserviços com a finalidade de executar uma determinada função.

Hospedagem dos serviços

Como a arquitetura pretende construir de modo descentralizado os serviços que compõem um software, ou dividir um software já existente em pedaços menores, temos que ter em mente

onde esse “pedaço” do sistema será alocado e executado. Essa etapa pode ocorrer de diversos modos, assim você pode utilizar de um servidor local, máquinas virtuais locais ou utilização de cloud computing.

Comunicação entre os serviços

API

Sem este conceito, a implementação de microsserviços seria impossível, como o sistema é distribuído em módulos, existe uma necessidade desses módulos se comunicarem entre si, e com não estão hospedados em um mesmo local lógico, essa comunicação geralmente é feita usando serviços de API(Interface de programação de aplicação). API estabelece uma interface que permite a comunicação entre sistemas de modo externo ou via web ou de modo interno. Imagine uma API como uma ponte que por meio dela um sistema pode se comunicar com outro e utilizar de seus serviços (sem saber o que está sendo executado por dentro do sistema, com acesso apenas a saída desejada). A comunicação via API pode ocorrer utilizando diversas linguagens como, XML e JSON .

Exemplo: Um sistema de ecommerce precisa de um meio de realizar o pagamento dos produtos colocados no carrinho, com pouco tempo para desenvolver essa funcionalidade do zero, a equipe de desenvolvimento opta por utilizar uma API de pagamento oferecida pelo PagSeguro. Os desenvolvedores não sabem como o sistema de pagamento do PagSeguro funciona por dentro(código fonte), mas basta apenas estudar a documentação da API e executar os comandos corretos dentro do sistema do ecommerce para acessar a funcionalidade de pagamento da API.

- **API ou web service**: É muito comum encontrar pessoas se referindo a API como web services, o que acontece é que o web service se refere a um serviço que é acessado via web(algo que a API se propõe a implementar), logo podemos dizer que o web service é um tipo de API, porém o recíproco não é verdadeiro;
- **APIs(REST, SOAP):** O conceito de API é de certo modo extenso e pode ser explorado e aplicado de muitas formas, duas arquiteturas de comunicações via API muito utilizadas são o REST e o SOAP, veja este pequeno artigo que explica ambos conceitos:
    - https://www.redhat.com/pt-br/topics/integration/whats-the-difference-between-soap-rest;

Mensageria

O padrão da comunicação entre microsserviços pode ser na utilização de serviços de mensageria, onde os serviços do sistema não se comunicam diretamente, assim os serviços do sistema ficam constantemente alimentando um servidor de mensageria que recolhe as informações dos diversos microsserviços e as armazena em um único local. Desse modo temos uma comunicação assíncrona entre microsserviços, permitindo reúso de informações, oferecendo alta escalabilidade e reduções no tempo para obtenção de informações.

Vantagens e desvantagens

O uso deste tipo de arquitetura para construção de softwares possui como **vantagem**:

- **Agilidade:** Com o sistema dividido em pedaços menores, as equipes podem ser divididas de modo a ficarem responsável por apenas um ou mais serviços, o que permite menores equipes que consequentemente são mais fáceis de gerir;
- **Alta escalabilidade:** O uso dos microsserviços se adapta de modo íntegro e rápido diante de demandas crescentes;
- **Reúso:** Com funcionalidades(serviços) divididos, podemos ter esse serviço sendo utilizado diversas vezes por outros serviços, sem a necessidade da repetição de código(algo comum em monólitos);
- **Manutenibilidade:** Manutenções são mais fáceis, e mais rápidas;

O uso deste tipo de arquitetura para construção de softwares possui como **desvantagem**:

- **Complexidade de implementação:** Um sistema monolítico, dependendo se seu tamanho já se torna complexo, imagine um sistema que é composto por diversos pedaços(serviços), isso exige do responsável pela análise e projeto do sistema uma divisão perfeita que seja capaz de entregar tudo que o sistema precisa para o seu funcionamento e sempre pensando na escalabilidade e outros pontos importantes na qualidade do sistema.
- **Alto uso de tráfego:** Geralmente os microsserviços utilizam a rede para se comunicar, e isso pode gerar uma sobrecarga em algum dos microsserviços. Atualmente temos soluções para este tipo de problema que realizam o “load balance”;
- **Custo em infraestrutura:** O custo com a infraestrutura neste tipo de arquitetura é um ponto a se destacar, dado que essa separação de serviços pode exigir contratações de servidores em nuvem, compra de diversos servidores locais ou um custo maior com um servidor que hospedará diversas máquinas virtuais;
- **Replicação:** A “base” dos projetos pode ser a mesma, assim pode existir um pequeno padrão de código que se repete dentro de mais de um serviço;

União de arquiteturas

Também podemos utilizar uma arquitetura mista, onde implementa-se ambos conceitos, por exemplo, em um sistema de arquitetura monolítica uma funcionalidade está consumindo muito recurso de hardware e prejudica todo funcionamento do sistema, uma solução pode ser isolar essa

funcionalidade(em outro ser vidor) e colocá-la como um serviço para o seu sistema, assim melhorando o desempenho do sistema como um todo.

Referências

- https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api;
- https://www.redhat.com/pt-br/topics/integration/whats-the-difference-between-soap-rest;
- https://www.zappts.com/blog/arquitetura-monolitica-e-microsservicos/;
- https://www.guj.com.br/t/o-que-sao-arquiteturas-monoliticas/385610/11;
- https://cursos.alura.com.br/forum/topico-qual-a-diferenca-entre-web-services-e-api-55330
- https://becode.com.br/o-que-e-api-rest-e-restful/;
- https://becode.com.br/o-que-e-api-rest-e-restful/;
- https://blog.geekhunter.com.br/arquitetura-de-microsservicos-x-arquitetura-monolitica/;
- https://www.redhat.com/pt-br/topics/integration/what-is-apache-kafka;
- Vídeo com excelentes explicações sobre microsserviços e conceitos importantes:
    - https://www.youtube.com/watch?v=jSnLOoGjQ80;