O artigo nos traz o conceito de “arquitetura de microsserviços”, que é usado para caracterizar uma forma de projeto de software onde o foco é a descentralização e desacoplamento de dados e os serviços são implantados independentemente, onde cada um executa o próprio processo e se comunicam através de recursos leves. Esses serviços podem ser implantados de forma independente, e há um núcleo mínimo de centralização.

Nos últimos anos tem sido cada vez mais frequente o uso de microsserviços por parte dos desenvolvedores, e a sua utilização tem sido muito benéfica (do ponto de vista dos autores), e se tornou um estilo padronizado para a construção de soluções. Parte dessa aderência à essa nova arquitetura, se deve ao descontentamento com a arquitetura monolítica.

Um sistema monolítico é construído como se constituisse uma única unidade e tem a lógica pautada em executar a solicitação em um único processo. Com o passar do tempo, a manutenção desse tipo de *software* se torna muito trabalhosa por conter muito acoplamento; Ou seja, se uma parte do sistema apresentar um problema, afeta outras partes dentro do mesmo módulo (as alterações nesse sistema exigem a construção e implantação de uma nova versão da aplicação no que diz respeito ao servidor), ou seja, o monólito tem que ser refeito. Assim, o que poderia ser simples, acaba se tornando complexo. Essa questão, juntamente da popularidade de aplicações implantadas em nuvem, tem criado um certo descontentamento com o modelo monolítico.

A arquitetura de microsserviços foi criada para atuar como uma solução para essas frustrações, sendo feita de pequenos serviços individuais, que trabalham em conjunto para o funcionamento da aplicação. Os microsserviços podem ser escalados de maneira individual (em contraposição ao monolito, onde toda a aplicação deve ser escalada, consumindo mais recursos) e os módulos de cada serviços são firmes e bem definidos, podendo ser escritos em diferentes linguagens e até mesmo gerenciados por equipes diferentes, o que aumenta a sua popularidade (Amazon, Netflix e The Guardian são alguns exemplos de empresas pioneiras na implementação desse modelo).

Uma coisa interessante sobre esse tipo de arquitetura, é que não existe um modelo definido para replicar essa arquitetura. Na verdade, sua aplicação pode ser identificada através de alguns fatores, como por exemplo: ‘Componentização via serviços’, ‘Organização de acordo com capacidades empresariais’, ‘Produtos, não Projetos’, ‘Endpoints Inteligentes’, ‘Gerenciamento Descentralizado’, Infraestrutura Automatizada’, ‘Projetar para Falhas’ e ‘Design Evolucionário.

Nas arquiteturas de microsserviços, a componentização é um conceito que ocorre principalmente por meio de serviços em vez de apenas bibliotecas (dentro do processo, chamadas por função). A desvantagem, é o custo maior das chamadas remotas, a necessidade de APIs mais granulares e a dificuldade maior em alocar responsabilidades entre componentes que cruzam processos.

Ao utilizar os **Endpoints inteligentes**, os microsserviços buscam pelo desacoplamento e pela coesão, para que tenham sua própria lógica de domínio e recebam uma solicitação, apliquem a lógica e produzam uma resposta usando APIs Rest.

Nesse tipo de arquitetura, a **organização de acordo com capacidades empresariais** é o foco da abordagem das interfaces. Obtemos equipes multifuncionais que abrangem desde experiência do usuário até gerenciamento de projetos, onde cada equipe pode ser por exemplo, responsável por desenvolver e operar um produto, como é o caso da “Compare the Market”. Vale reforçar, que esse tipo de modularização também pode ser aplicado na arquitetura monolítica, embora não seja comum.

**“Produtos, e não Projetos"** é o modelo mais adotado pelos defensores da arquitetura de microsserviços. Essa ideia autoexplicativa, parte da premissa de que um projeto é um software que é considerado concluído assim que é entregue, e a equipe que o fez é consequentemente dissolvida. Já o produto, exige mais responsabilidade e comprometimento, já que uma equipe de desenvolvedores é responsável pelo software em produção e aumenta o contato com seus usuários.

Através do **Gerenciamento Descentralizado de Dados,** os microsserviços permitem que cada serviço gerencie seu próprio banco de dados, sejam instâncias diferentes da mesma tecnologia de banco de dados ou sistemas de banco de dados totalmente diferentes. Infraestrutura Automatizada é um recurso bastante utilizado pelas equipes que desenvolvem softwares, para automatizar testes e garantir a confiabilidade do software, no gerenciamento de microsserviços em produção.

As arquiteturas de microsserviços compreendem que é importante realizar a **projeção imaginando as falhas de serviço** que podem ocorrer, por indisponibilidade de servidor, e diversas outras causas. Nesse cenário, o sistema deve agir de forma inteligente, como um disjuntor, que é desativado assim que percebe um problema que pode acometer os outros fios (nesse caso, o restante da aplicação). Dessa maneira, caso haja uma falha, é importante ser capaz de detectá-la rapidamente e, se possível, restaurá-la automaticamente.

O conceito de ‘**Design Evolucionário**’, apresentado pelos microsserviços, é como os profissionais da área caracterizam toda a decomposição de serviços, por exemplo, os princípios pelos quais a aplicação deve ser dividida, e que o controle de mudanças, não significa necessariamente menos mudanças, mas sim mudanças que fazem a diferença, e tornam o software mais eficiente.

Embora os microsserviços sejam fortemente recomendados pelos autores, é importante reforçar que não existe nenhum tipo de bala de prata. Os microsserviços também apresentam desafios, como dimensionamento, distribuição e gerenciamento. Assim como qualquer arquitetura, existem prós e contras, e para que sua implementação seja bem-sucedida, é importante que faça sentido em todo o contexto do sistema a ser criado.