Academic Year: 2021/22

THE INDUSTRY OF THE WEB SERVICES

Focus on SOA and Microservices

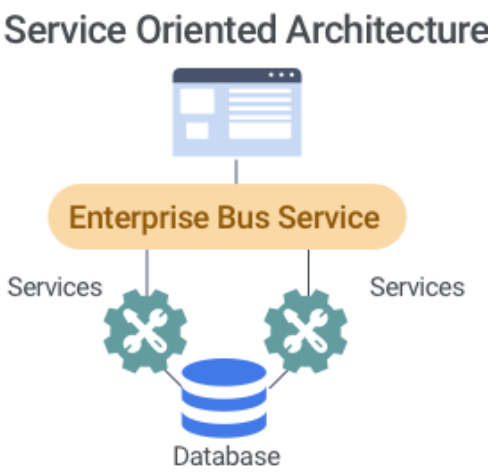
Made by:
Alejandro Fernández Pérez

INDEX:

1. Explanation of the SOA concept.

o What is SOA?

SOA is a Service Oriented Architecture whose paradigm is to organize and to use distributed capabilities, which may be under control of different ownership domains. All of them, based on a Software Architecture, which is the fundamental organization of a system, divided by different components, related them mutually. This Architecture has a huge role in the environment and in the design and evolution of its Principles: Loose Coupling and Service Contract, Abstraction, Reusability, Composability, Autonomy, Optimization, and Discoverability.



Picture 1: SOA System Graphical Example

o Characteristics of SOA.

1. To think in Services, rather than in Objects.
2. To select the Services that they are going to be discovered.
3. To choose its own Mechanism of Communication.
4. To consider a lot the Security Mechanism.
5. To consider the reusability.
6. SOA is not a Web Service, but it can be implemented by a Web Service.

o Types of SOA components.

There are four types of components in the Service Oriented Architecture: The Front End, the Service Repository (it is a database, whose function is to listen servers, which use UDDI Language defined over XML. And also, it has an extensible standard, which defines some required information about available interfaces and argument types, and it can provide extra information), the Service Bus, and the Service, which is a mechanism to enable the access to a prescribed interface and to consistent capabilities with restrictions and different policies, and which can be divided into three components: the Contract, the Interface, and the Implementation, in turn divided into the Business Logic, and the Data.

To focus more on the SOA components, they can be emphasized among two types: the Services, and the Connections. The first one are functions or business processings, which are well-defined, self-contained, and does not depend on the context or the state of other services, for example: the Weather Forecast Service. The second one are the links, which are connected as self-contained distributed services mutually, for example: the Connection between HTTP ⇔ SOAP.

o Important challenges of SOA.

The Service Oriented Architecture has to complete three challenges, if it wants to continue developing its product:

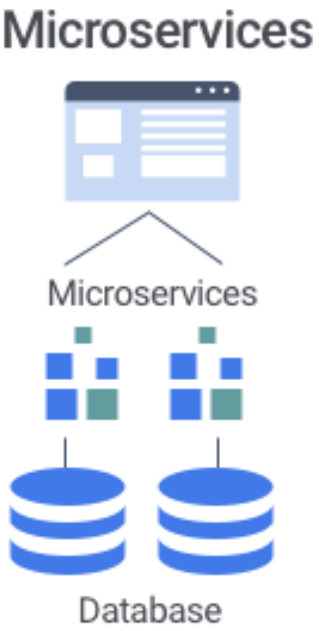
- First Challenge: Managing Services, because if there is not enough attention on the governance of those Services, the performance of them will have many problems, and the reliability will decrease.
- Second Challenge: Providing proper security for the different roles.
- Third Challenge: Ensuring the inter-operability of the Services.

2. Microservices as SOA and with APP Scaling.

A Microservice is a Service Oriented Architecture of the pattern, wherein applications are built as a collection of different smallest service units and independents. I mean, it is focused on the decomposing of an application into single-function modules with well-defined interfaces. These modules can be independently deployed and operated by small teams that own the entire lifecycle of the service.

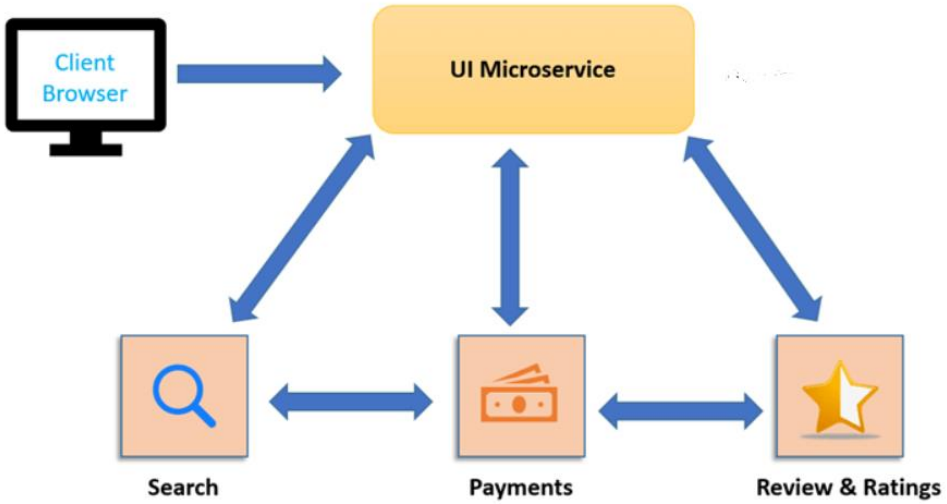
Additionally, the prefix "micro" is referred to the sizing of a microservice which must be manageable by a single development team (5 to 10 developers). In this methodology, big applications are divided into the smallest independent units.

But to be more precise, a Microservice Architecture is an architectural development style, which allows building an application as a collection of small autonomous services, developed for a business domain.



Picture 2: Microservices Graphical Example.

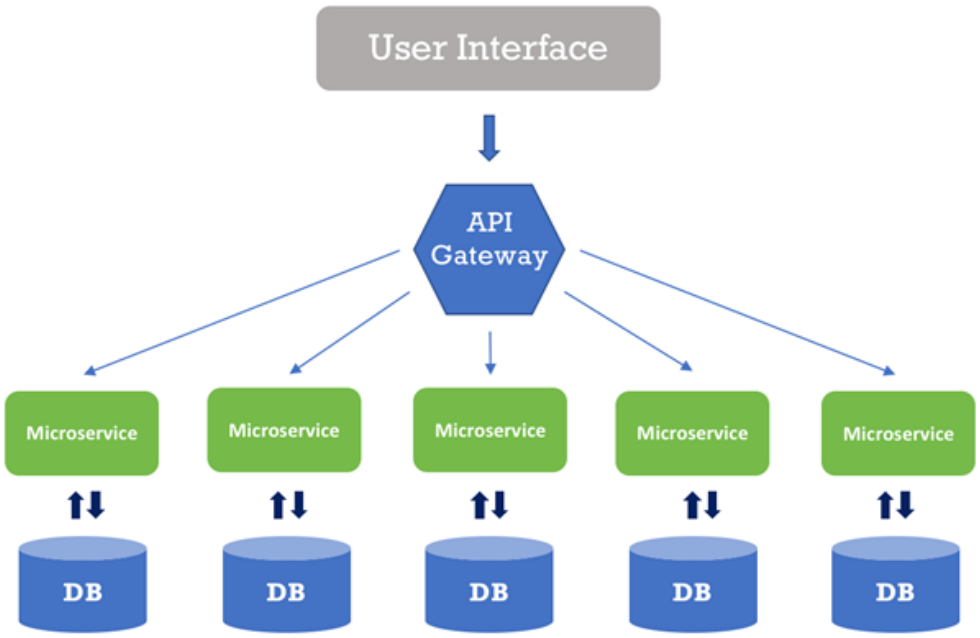
An example of Microservice Architecture could be an e-commerce application, where each Microservice is focused on a single business capability: Search, Rating and Review, and Payment. Each one has its server and they are communicated mutually, I mean, distributed individual modules which are communicated between them. This is because the communication between Microservices is a stateless communication, where each pair of requests and responses is independent, with an effortlessly communication, and with an independent data store.



Picture 3: Microservice Architecture of an E-commerce Application.

o Features of the Microservices.

- Their APPs are mostly dedicated to databases or another types of storage for necessary services.
- Use complex APIs: Application Programming Interfaces, which is defined as a set of definitions and protocols that are used to develop and integrate the application software, allowing communication between two of them, through a set of rules.
- Are focused on decoupling.
- Are allowed to do systematic changes, creating a new service.
- Have a strong emphasis on DevOps (Program of Development and Operations), and Continuous Delivery.
- Consist Full-stack in nature.
- Use lightweight protocols like HTTP, REST, or Thrift APIs.
- Are designed for Host Services, which can work independently.
- Is not include the component sharing.
- Each service has its independent data storage.
- Are better for small and web-based applications.
- Have a quick and easy implementation.
- Are based on the bounded context for coupling.
- Have an elaborated and a straightforward Messaging System.
- Based on a Highly Scalable Architecture.
- More focused on people's collaboration and freedom of choice.
- Their modules are the loosely coupled.
- Their Management of the project can also be modularized.
- Have a meager cost of scalability.
- Use multiple technologies as multiple features in an application.
- Their service is ideal for evolutionary systems, where can't anticipate the types of devices that may access to the application.



Picture 4: Microservices in an API example.

3. Conclusion about handling data in each service separately.

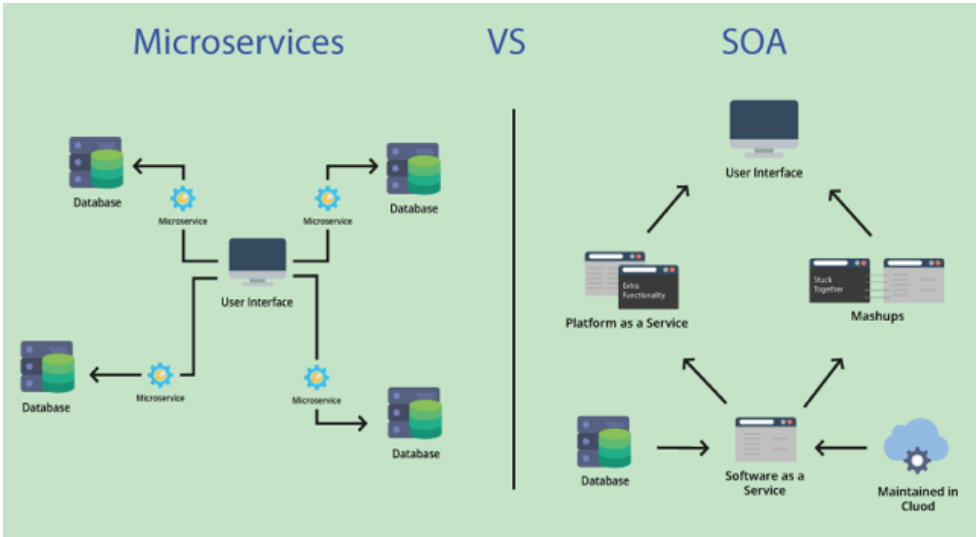
As everything, all technological or computer architecture has its benefits or advantages and its disadvantages. To focus, firstly, in SOA, its service can be characterized into different negative and positive points of view. The benefits are: to have an easy edition and update of whatever service. Can access to the service data from the same directory every time, due to services have the same directory structure. Can communicate among them by a common language in an independent platform. To be easy to debug and test the independent services, because the services are of small size over the full-fledged application. To reuse the service of an existing System, or to build alternately the new system. To adjust or modify the different external environments. Can develop applications without replacing existing applications. To offer reliable applications for testing and debugging the independent services as compared to a large number of code. To reduce the cost by adding value to the core investments from leveraging existing assets, and by building faster the new systems for less money with the aim of reducing the expenses of integration, having more flexibility, and achieving the inter-operability as great value to long term. To increase the employee productivity, hereafter the existing skills, and a duplicate functionality. To build services for partnerships, based on standards, interactions, and integration according to what is needed. To be more agility, by the progress of the applications, and the long duration of them in the time; by the division of the Back End, replaced over time; by the emphasis on core-competencies; by the support of the incremental implementation approach; by the outsourcing of the services until a new business model. To have scalable services, hereafter scalable systems and with progress, which can scale up to mobile devices, and down to large systems and across organizations. To manage complex systems, without requiring centralized services, and making a progress to the high level users. And to allow the flexibility if there is loose coupling.

Otherwise, the negative points are: to have the service in a homogenous IT environment. To have the service with tight coupling, rather than loose coupling. To have a service where the things do not change. To have a service where the real-time performance is critical. To validate all inputs before they are sent to the service. To be a costly service in terms of human resources, development, and technology. To need to send and receive messages and information frequently on some Web Services for reaching a million requests per day easily. To have a high investment cost. And to not be suitable for GUI (Graphical User Interface) applications, even with a heavy data exchange.

Finally, to focus on the Microservices, according to the Web Services, they can be characterized into different negative and positive points of view. The benefits are: to be an easy architecture pattern for developers. To have a faster Integrated Development Environment (IDE) with the aim that the developers are more productive and faster. To have a faster web container for speeding up the process of implementations and developments. To allow to the development team to implement, to develop, and to scale its service independently of all of the other teams. To provide inter-operability between some Software APPs and some Online Platforms (the first ones are runned on the second ones). To count with Open Standard and Protocols, based and texted on data formats. To work through many common firewall security measures without any change in their filtering rules. To know how a client can communicate with others. To be responsible of knowing how a server is found and bound. And to be responsible that the client sends a common sense request, and the server replies with common sense.

Otherwise, there are negative points, that they are: to not allow that the data center controls the decisions of the clients. To not assist on the implementation of naming and discovery in the scalable cluster-style services. To be developed for building monolithic applications, and without any support for developing distributed applications. To test is more difficult. To implement compulsorily by the developers the inter-service communication mechanism. If there is to implement cases with multiple services, is necessary the coordination between teams. Microservices are costly, because is always necessary to maintain all servers for the different business tasks.
So, to conclude, SOA is an ideal architecture method for large and complex business applications, because is most suited for environments, which require the integration with many diverse applications. However, workflow-based applications, which have a well-defined processing flow, are challenging to implement with the help of SOA architecture patterns. Therefore, this is the explanation of small applications are not ideal for SOA, because do not require of middleware messaging components. On the other hand, the Microservices are well suited for smaller and well partitioned web-based Systems.

To resume, while Microservices are more focused on decoupling, SOA is focused on application service reusability. While Microservices are built to perform a single business task, SOA is built to perform numerous business tasks. While Microservices are full-stack, SOA is monolithic in nature. While Microservices have services with their own data storage, SOA involves sharing data storage between services. While Microservices are designed to Host Services which can work independently, SOA is designed to share resources across services. While Microservices have strong emphasis on DevOps and Continuous Delivery, SOA is not focused on them. While Microservices are highly scalable architecture, SOA is a less scalable architecture.



Picture 5: Differences between SOA & Microservices in Web Services.

4. References or Links.

https://www.xataka.com/basics/api-que-sirve => To know about what an API is.

https://www.guru99.com/microservices-vs-soa.html#9 => To know about the Microservices, according to the Web Services nowadays.

PDF and PowerPoints of the Subject XML & Web Services of the University of Novi Sad – FTN => To know about the SOA and some advantages and disadvantages among this and Microservices.

Edited images by Adobe Photoshop => To aim better pictures in the Project.

https://www.techtarget.com/searchapparchitecture/tip/Microservices-vs-SOA-Choose-the-right-app-architecture => To know more about the Microservices pattern.

https://scoutapm.com/blog/soa-vs-microservices => To know about the difference between SOA and Microservices.