ONLINE SHOPPING WEBSITE USING MICROSERVICES ARCHITECTURE

[1]J. Amita ,[2]A. Shubham, [3]K. Sayali, [4]S. Saurabh, [5]P. Krishna, [6]S. Shantanu

AUTHORS PROFILE

Mrs. Amita Jajoo, Dept. of Information Technology, D. Y. Patil College of Engineering, Pune.
Shubham Aher, B.E. D. Y. Patil College of Engineering, Pune.
Sayali Kulkarni, B.E. D. Y. Patil College of Engineering, Pune.
Saurabh Singal, B.E. D. Y. Patil College of Engineering, Pune.
Krishna Patil, B.E. D. Y. Patil College of Engineering, Pune.
Shantanu Shamsundar, B.E. D. Y. Patil College of Engineering, Pune.

Abstract

Our main aim is to implement Microservices Design in an Online Shopping Web Application. Now, our application can offer numerous services viz. Database Service, adding things to Cart, widespread necessities, search choice etc. This routing information of these services will be stored in the Service registry of the application. We have thought of implementing the following design which consists of Web UI, API Gateway, Service Registry and the services that our application can provide. These services can have their own database as well as they can be designed (if needed) using different technologies. This is what Microservices architecture offers while building an application. Starting with the Web User Interface, it interacts with the user showing what services this application can provide. As the user demands a particular service, the Web User Interface sends HTTP requests to the API Gateway of the application. This API Gateway is the base of an API Management Solution. It can be considered as a point of entry into the system and helps to interact between multiple services as well as APIs. The most necessary role the API gateway plays, is guaranteeing reliable processing of every API call. It also provides the feature to style API specifications and manage all of them centrally. This API gateway will then hit the service registry. It consists of metadata of the services of the system like the network locations of instances of the services. It includes the instances of various services which our application can offer. All the services will interact with the Service Registry without even knowing the URL if they can just give a name for each service. Now using this registry, we can route to the different services easily as it contains routing information of every service and also the services are going to be provided to the user. All the services are going to be registered as a client on the service registry and the registry will act like a server. This registry will also need to implement load balancing so that it will hit the service which is free when a load is generated on the server.

Keywords: *API Gateway, Microservices, Web-Application, Service-Registry.*

1. INTRODUCTION:

The plot of microservices is that it is easier to create and maintain some kind of applications when they are broken into smaller, compassable pieces or items that work along. Each element is continuously developed and severally maintained and then application is purely the sum of its constituent components. Microservices are different to a traditional, “monolithic” application which develops the application in one block-like structure. Applications devise as a group of standard elements easier to know, test and most importantly easier to maintain over the life of the applications. It allows organizations to achieve scalability, be able to simplify complex architecture of applications and improve the time it takes to get working enhancements to production. This approach has been proven to be superior and preferred for giant enterprise applications which are developed by groups of geographically and culturally various developers. Microservices are trending, getting a lot of attention through blogs, articles, and discussions on social media and conference presentations. At the same time, there are some people in the software industry who claim microservices as nothing new. Naysayers state that the idea is rebranding of Software Oriented Architecture (SOA).

However, despite both the hype and the skepticism, the microservices design pattern has key advantages – particularly when it comes to enabling the agile development and delivery of complicated enterprises. We will learn more about the approach and compare it with the Monolithic architecture . This series can describe the various components of a microservices architecture.

2. PROBLEMS IN MONOLITHIC ARCHITECTURE

Monoliths are designed as a single unit and hence are responsible for each possible functionality like handling of the HTTP requests, executing domain logic, database operations, interactions with the browser client and so on.

Because of such a big and single building block, if we need to make small changes in the system then we will have to build and deploy the whole application. Building and Deploying isn't the only disadvantage –simply think about scaling. You have to run numerous instances of the monolith, even if you know that problem is in only one element.

3. PROPOSED SYSTEM

Microservice architecture is a service-oriented architecture in which software applications are divided into small units or groups and these groups are loosely coupled with each other and opposite to this, monolithic software applications are built as large single units. So monolithic applications are complex to operate.
A microservice is a group of small services so we can build every service independently from the other, or we can use different programming languages for different services and run on their own. Basically, each service tries to satisfy the specific user requirement or business functionality that you're working on.

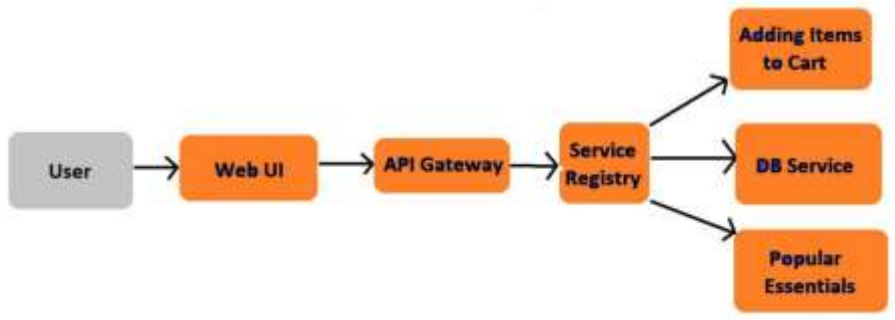

Figure 1. Micro-service architecture

We have implemented the above architecture which consists of Web UI, API Gateway, Service Registry and the services our application will provide. These services will have their own database as well as they can be designed using different technologies. This is what Microservices design offers well as they can be designed using different technologies. This is what Microservices design offers while building an application.

4. METHODOLOGY

4.1 Web UI:

Web user interface is a page that is used by the users to interact with, when a site is fully downloaded on a web browser. A website or any online shopping application site is nothing but a collection of code, but this code is not appropriate for user interaction. In order to ensure that a website is available for guests to use, the code must result in a web interface that users can easily interact with. The data for which the users are looking for, needs to be easily available. While building the application, taking the users into consideration who will be accessing the application, will help in designing a perfect user interface of the application.

4.2 API Gateways:

The API gateways are a layer situated between the client and the services it depends on. An API gateway is basically used for routing purposes. Its flow can be determined as:

● A request is sent to the Gateway by a client.
● The Gateway then processes the request and sends it to the Service.
● Service responds to Gateway.
● Responses that are getting sent to the client are first processed by the Gateway.

4.3 Service Registry:

Communication between the service registry and other elements can be classified into two groups,

● Registration (Communication between the services and the service registry)-
  1. Self-registration
  2. Third-party registration
● Discovery (Communication between the clients and the service registry)-
  1. Client-side discovery
  2. Server-side discovery.

4.4 Database Service:

A monolithic application usually has a single relational database. For the trendy applications, to store and process various forms of information, a relational database is not continuously the simplest alternative. In a microservices architecture, every microservice has its own private data store.

5. CONCLUSION

The microservices architecture is a set of small services. These services are connected with each other. Therefore, this application reduces internal coupling between services and then simplifies their quantifiability and also facilitates the reading of the base of source code.
Monolithic way is not suitable for complex and large applications, on the other hand, Microservices architecture is very useful for large and complex applications. It divides applications into small services. We can easily make changes or find bugs in a single service without interrupting other services.

However, these advantages need bound conditions such as a substantial maturity of the application, a team of qualified developers in distributed applications etc., that they'll be exploited.

6. REFERENCES

[1]. R. Terra, M. T. Valente, M. Viggiato, H. Rocha, E. Figueiredo, "Microservices in Practice: A Survey Study", Conference Paper, 2018.

[2]. P. Jamshidi and C. Pahl, "Microservices: A Systematic Mapping Study", *Proceedings of the 6th International Conference on Cloud Computing and Services Science (CLOSER 2016), Volume 1,* pages 137-146.

[3]. D. P. Jayanto, H. Suryotrisongko and A. Tjahyanto, "Design and Development of Backend Application for Public Complaint Systems Using Microservice Spring Boot", Published by Elsevier B. V., *Peer-review under responsibility of the scientific committee of the 4th Information Systems International Conference 2017.*

[4]. M. Koyuncu, H. Vural and S. Guney, "A Systematic Literature Review on Microservices", Published by Springer International Publishing AG 2017, O. Gervasi et al. (Eds.): ICCSA 2017, Part VI, LNCS 10409, pp. 203–217.

[5]. R. Rizzo, P. Storniolo, A. Messina, M. Tripiciano and A. Urso, "A Systematic Literature Review on Microservices", Published by Springer International Publishing Switzerland 2016, M.E. Renda et al. (Eds.): ITBAM 2016, LNCS 9832, pp. 223–233

.