Microservice Architecture using ASP.NET Core

Bhavana R, Rakshitha S G, Rajashree Shettar, Minal Moharir

Abstract: Microservice architecture is a variation of the structured form of service-oriented architecture that arranges an application as a series of loosely coupled services. This is a self-contained process that takes advantage of specific and unique business capabilities. The Microservice architecture approach provides many advantages in terms of scalability and flexibility. Microservices can be built by either dotnet framework solution (such as asp.net and asp.net web Api) or use solutions based on a core Microsoft dotnet framework (such as asp.net core). This paper presents an overview of Microservice architecture along with its advantages and disadvantages. We have also presented why ASP.NET core can be used instead of ASP.NET to build Microservices.

Keywords: Microservice Architecture, ASP.NET core Scalability, Services, Flexibility.

I. INTRODUCTION

The core of Software Engineering Solutions was the capacity to modify a system as a result of changing minimum cost criteria [1]. Few examples would be targeting high cohesion, low coupling, modularization. Microservices are software components where development and deployment independence are a primary concern [2]. The concept of loose coupling is key to the Microservices theory. If systems are built using independent services, better flexibility can be accomplished [3].

Microservice Architecture constructs an application as a series of small independent services. When there is a complex application designed using this architectural model, it can be broken down to a small number of services. Services in Microservice architecture are mostly processes that interact across network utilizing technology-agnostic protocols like HTTP to achieve a purpose [4]. Each of the services is individually deployable. They are decentralized and designed with automated process and released. Depending to what works most, services can be implemented using various programming languages.

Services are liable for the preservation of their own data. The details of the internal implementation of each services are secret from the other services. Services can have various libraries, various technologies or the same framework. The team is generally accountable for the whole life cycle of the Microservice [5].

ASP.NET Framework is a classical and popular application development platform and framework. It enables running and deployment of various web applications like web services, web application, windows GUI and command Line. It contains various libraries with broad functionalities. The main modules of this platform are Base Class Library (BCL), Application Framework Library and Common Language Runtime (CLR).

By deciding to use Microsoft platform to create a microservice system, we can either opt to use a dotnet framework solution such as asp.net and asp.net web Api or use solutions based on a core Microsoft dotnet framework such as asp.net core. It has lots of modern approaches for building web applications with cross-platform capability where single code is made to run on different platforms. It supports vast production and standard infrastructure.

II. ADVANTAGES OF MICROSERVICE ARCHITECTURE

- Flexibility is improved by breaking a system into smaller components. Provided there exist a license, it is easier to engage with open source, when an application is loosely coupled. System must be decomposed into smaller loosely coupled components in order to incorporate the functionalities needed into the system from open source components [3].
- Since the services are separate, only the necessary parts of the application which need to be scaled up can be allocated the resources that is needed [6]. Efficient use of resources is the result. Sections of the system that require more computational power may be assigned the required resources without scale-up of the whole network [7].
- Functionality can be published faster. There is no need to wait until the entire system is published. The priority for every company is to bring about rapid improvements in the production. It is easier to manage changes when an application is broken down into a greater number of smaller components.
- Focusing on security is easier whenever it is required. Appropriate resources can be allocated to less sensitive services requiring less security. More protection is needed for more sensitive services. And these services could be placed in more protective zones [3].
- The best and most suitable tools could be used to build each service. Parts of the system can be moved to the cloud [9]. If any of the components need specialized competencies to manage, it can be put on the cloud, which is decided by the company.
Microservice Architecture using ASP.NET Core

- Redundancy is typically believed to be avoided. It is a common way to increase the resilience. Microservices will help to more effectively apply this concept [3].

II. MONOLITHIC VS. MICROSERVICE ARCHITECTURE

In monolithic application, it is unable to run modules independently. Any modification to a monolithic module needs a reboot of the entire application [3]. It may be difficult to understand and modify a monolithic application. It is particularly true as the application gets bigger [4].

- Adding a new developer or replacing a leaving team member is difficult.

Large code base reduces profitability. Quality of code will be declined. Original modularity will be lost. Monolithic application prohibits independent work by the developers. Development efforts and redeployment efforts must be coordinated by whole team [4]. If any one of the components has to be updated, the entire application has to be redeployed. Monolithic application puts obstacles in the way of frequent updates [4].

In Monolithic application, scalability is typically a concern. Monolithic architecture can only be scaled in one-dimension [4]. Sometimes the entire application will not have to be scaled up. Only the necessary subset of modules has to be scaled up [3].

III. DISADVANTAGES OF MICROSERVICE ARCHITECTURE

- Though each service can be very simple, as a result of communication between multiple components, there is an increase in complexity. Monolithic systems are simpler when compared to distributed systems. Running services are more complex to manage compared to monolithic systems [3].

- The database will be partitioned across different microservices. Few business transactions that could easily be implemented in monolithic systems are difficult in the microservice system [6]. Implementing a database query involving several joins will in certain cases be a challenge. It is a difficult task to manage the consistency of databases [3].

- Whenever a computation is made outside the boundaries of the module, the request must pass across the network. The result is communication overhead. Microservices approach would result in slower services [3].

- It may not be an easy task to know what other services need without hard-coding certain requirements. In contract testing, the ingoing attributes and outgoing attributes in each case are tested for conformity with the intended attributes [3]. In order to conform with the contract, each Microservice’s development team will check the communication with other services.

- Microservices construct a series of small independent services that do small amounts of work. The focus here is not only on what those services do, but also on communications between them. Any single communications between the services is a possible place for something to go wrong [3].

- A significant amount of calls to the backend Microservices are very costly in terms of battery use from a mobile development viewpoint. A version of the application cannot be built for mobile devices only since all calls will ultimately go to the backend servers.

- It is not easy to share code. Functionality across different services has to be replicated in order to share common utility code between services.

IV. CHALLENGES

Implementation of Microservices is not simple. One challenge for Microservices is to determine when to incorporate functions within a single service and when to split them into separate services [3]. The transition to Microservices needs a change in development methodology. The agile methodology will suit the Microservices approach.

The service may run on a different computer or on a different platform. Chances are that something could go wrong. It is necessary for these systems to have a mechanism for seeing which service causes a bottle neck.

Dispersed services, and more open ports contribute to a wider surface of attack. When each service has its own database, there is more scope for attacks related to the database [3]. Even though the Microservices approach provides substantial advantages, a Microservice architecture needs extra machinery that could impose substantial costs. It is beneficial to be integrated with the cloud to improve the economy of Microservices [7].

Technology stack is another challenge. It is very difficult to change with Monolithic architecture. So, all elements within the application must stick to the initially chosen technologies.

Prior technologies, such as web services, have also faced challenges in discovery, granularity and security [10]. The name Microservices could be misleading as all the services in the Micro service architecture need not be micro [11].

V. ASP.NET FRAMEWORK DRAWBACKS

Web applications built using ASP.NET framework was stable over a time and necessarily targeted to the Windows platform. So, these applications will be platform dependent and prevents it’s use on other operating systems. This feature will be the major drawback of ASP.NET [12]. Due to this many web application developments today are not built using ASP.NET because nowadays web applications need to be platform independent and able to be run on different platforms in order to meet the Customer requirements.

Microsoft decided to build a new framework which support platform independent and features which are limited in ASP.Net such as Limited object Relational Support, slower run time etc [13]. Asp.Net Core provides built-in framework for two most popular frameworks Angular and React. It also provides the in-built support for Dependency injection. Despite of this feature, we no longer need to use third-party libraries such as Auto Factor[12]. Asp.Net Core uses the kernel web server and provides the technological stack that will re-evaluate which technology will be suitable for new project development and it would be helpful for the developers to decide on new technology.
VI. PROPOSED METHODOLOGY

The proposed methodology is ASP.NET Core. The ASP.NET Core is a new and light weight framework compared to legacy ASP.NET. It provides scope to hosting microservices in applications as the core layer. It has different layer architecture such as 3-tier architecture, N-tier architecture etc. It can modify the infrastructure in order to support almost all the technologies [17]. It concentrates on separation of concern like, it separates the UI logic from the Business logic and provides single responsibility to keep away from tight coupling and make the application robust. It contains system.assembly which will have libraries related to web forms. It Describes architecture into various layers as main part of the platform, which helps the developer to build a solution more compatible and without having to reinvent the capabilities needed for the solutions, this basically will save the time of developer.

![ASP.NET Core Architecture](image)

**Fig - 1: Block Diagram Of ASP.net core architecture**

ASP.NET Core generates essential pre-established N-tier architecture for web application developers, it makes developer to become effortless and spend least time about difficulty of requisite N-tier architecture of applications and it makes developer to concentrate more on business layer logic [14]. No fascination developers may find many web development frameworks that are more friendly design patterns. Developer needs to make the essential tiers by themselves, but they can follow MVC (Model-view-controller) architecture because this architecture is one of the most popular architecture related to web application development. Basically ASP.NET Core accompany single page application (SPA) for the sake of out-tier deployment of project or applications, along it provides supports for traditional view-based model designs and razor pages.

In Single Page Application (SPA), UI and backend API calls will be separated at client and server side. User interface (UI) logic evaluation will be done at client (web browser) side and for processing requests API calls will be made at the server side [15]. Basically, in earlier days angular.js was used to provide all the necessary supports for designing UI (User interface) logic in applications at user side in order to provide the services to the user. Nowadays developer wants even more interactive UI’s which has the single page application feature so that the application will take less time to load. whenever the request is made by the user. So, they are using new UI technologies like React, blazor etc [16]. API calls at the server side is to handle the services requested by the user or client, because to improve the application performance and to satisfy user needs leading to better user experience.

VII. RESULT ANALYSIS

ASP.NET Core is highly modularized, and this design will allow to explicitly customize what framework components are deployed with the application. This makes the application start up faster and requires less memory.

From Fig-2, we can say that ASP.net core is growing faster and is the best choice to build Microservices.

![.NET Framework v/s .NET Core Framework](image)

**Fig - 2: .NET Framework v/s .NET Core Framework**

VIII. CONCLUSION

Microservices would become as large as it requires in order to have an efficient, coherent and reliable function. It is not just the size though; it is about logical cohesion and focus. The key benefit is that the system is divided into smaller parts which can be handled individually. Depending on each case, questions about flexibility, scalability must be answered before switching to Microservices.

The goals must be clear in order to measure benefits. Selected parts of an application could be migrated into Microservices [3]. Design thinking on Microservices could be applied to a monolithic. However, sticking to Monolithic applications, implementing Microservice application include different issues [3]. However, switching to Microservices is not that easy. While weak points of Monolithic are scalability and flexibility, which might not be priority for all applications. ASP.NET Core framework supports integration of modern UI (User-Interface) technologies such as AngularJS, ReactJS etc and it will be useful to create applications which supports single-page approach.

REFERENCES

Microservice Architecture using ASP.NET Core

17. Steve smith, “Overview of ASP.NET CORE MVC” on 2017 online paper.

AUTHORS PROFILE

Bhavana R
UG student
Department of Computer Science and Engineering
RV College of Engineering, Bengaluru, India

Rakshitha S G
UG Student
Department of Computer Science and Engineering
RV College of Engineering, Bengaluru, India

Dr. Rajashree Shettar
Professor and Dean (PG and Research)
Department of Computer Science and Engineering
RV College of Engineering, Bengaluru, India
B.E, M.S, Ph.D
Teaching: 19 years, Industrial: 1 year, Research: 12 years
Publications:
National Conference: 7
International Journals: 43
International Conference: 24

Dr. Minal Moharir
Associate Professor
Department of Computer Science and Engineering
RV College of Engineering, Bengaluru, India
MTech(CNE), Ph.D.(CSE)
Teaching: 14 years
Publications:
National Conferences: 7
International Journals: 15