ISABEL MERCADO & CLAUDIA ALVES
COOKING MADE SIMPLE
"MY FAVORITE COOKBOOK OF THE YEAR"
OLIVIA WILSON

The Knowledge Hub Universities | partnered with Coventry University

KH6003CEM
Web API Development

Report

Maged Saqr

202000687

Table of Contents

Introduction

Designing a web application for sharing and finding recipes requires study and consideration of which architecture to use while implementing the application. This report discusses at three different architectures to design it: RESTful APIs, Microservices, and Monolithic. Each architecture has its own advantages and disadvantages when it comes to performance, scalability, structure, and complexity. This paper outlines the basic differences between the three architectures in terms of characteristics, theoretical implementation, and comparisons in graphs and numbers. It also discusses which architecture was chosen between the three and why it was chosen.

Architectures Comparison

Any web application can be created using Monolithic, Microservices, or REST architectures. Monolithic architecture is an architectural style that focuses on designing the application functionalities as a single unit. In this architectural style, all components are developed, scaled, and deployed as a single unit. This means that the application should be written with one tech stack. Having a monolithic architecture means that the application will most likely be large, complex, and has to be tested after every change is committed [5]. One bug in any component can bring the entire application system down.

On the other hand, Microservices architecture is an architectural style used for designing applications in which the application is broken down into smaller or micro applications or services. The application is divided into multiple microservices based on the application's functionalities, in which each service is responsible for a single functionality or a job. Each microservice is deployed independently and has its own version [4]. The main benefit of microservice architecture is that if one service fails, it does not affect the other services and the rest of the application will function normally without being down to fix the failed service.

Another architecture is the REST architecture, which is an architectural style used to design web applications. It's a client-server architecture based on stateless communications between client and server in which the receiver retains no session information. REST architecture divides its components into resources in which each resource is identified using URIs. These resources are performed on using HTTP methods like GET, POST, PUT, and DELETE. Pros of REST architecture are simplicity, scalability, and loose coupling between client and server in which clients only need to know the URIs and HTTP method in order to interact with the server's resources. Cons of REST are complexity and inconsistency in which RESTful services rely on status codes and HTTP methods [3]. Another downside of this architecture is the under and over fetching of data as the shape of incoming data is always unknown.

Rest API	**Microservices**	**Monolithic**
Consists of interfaces	Consists of components	Consists of a single unit
Large in size	Small in size	Large and complex
Development time separate from service implementation	Fast development	Slow development
Not scalable	Easy to scale	Difficult to scale
Good Performance	Latency in performance due to network calls	Faster in performance due to no network calls

Table 1. Architectures Comparison

Architectures Implementation (Theoretical)

Monolithic Architecture:

- The recipe web application will be developed, scaled, and deployed as a single unit.
- All components, including user management, recipe handling, and ingredient management will be tightly coupled within the same database and tech stack.
- Any testing or adding changes will be very complex to implement once the application is deployed. Any modification will need to be extensively tested to make sure the system remains functional. Any failure in any component might bring the whole system down.
- Monolithic architecture is simpler and easier than microservices and RESTful architectures as there is only one database and unit to handle [6].

Microservices Architecture:

- The recipe tool will be broken down into small independent services, in which each one is responsible for a specific functionality such as:
    - user management
    - recipe handling
    - ingredient management
- Each microservice will be deployed independently. This provides scalability, maintenance, and fault isolation. Which means that if one microservice fails, it doesn't affect or bring down the whole system.
- One downside of managing multiple microservices is the complexity in terms of communication between services, deployment, and monitoring [6].

REST Architecture:

- REST architecture can be utilized within both monolithic and microservices architectures.
- Resources such as users, recipes, and ingredients will be shown using URIs. Clients will interact with these resources using standard HTTP methods like:
  - GET: GET methods are used by clients and servers to retrieve recipe data from a server.
  - POST: POST methods are used by clients and servers to create new recipes.
  - PUT: PUT methods are used by the server to update an existing recipe or ingredient on the server.
  - DELETE: DELETE methods are used by the server to delete an existing resource like recipe details or ingredients on the server.
- RESTful services provide simplicity, scalability, and loose coupling between the client and server.
- One downside in designing RESTful APIs is that it requires careful consideration of resource identification and response formats in order to avoid under or over-fetching of data.

Comparison in Graphs and Numbers

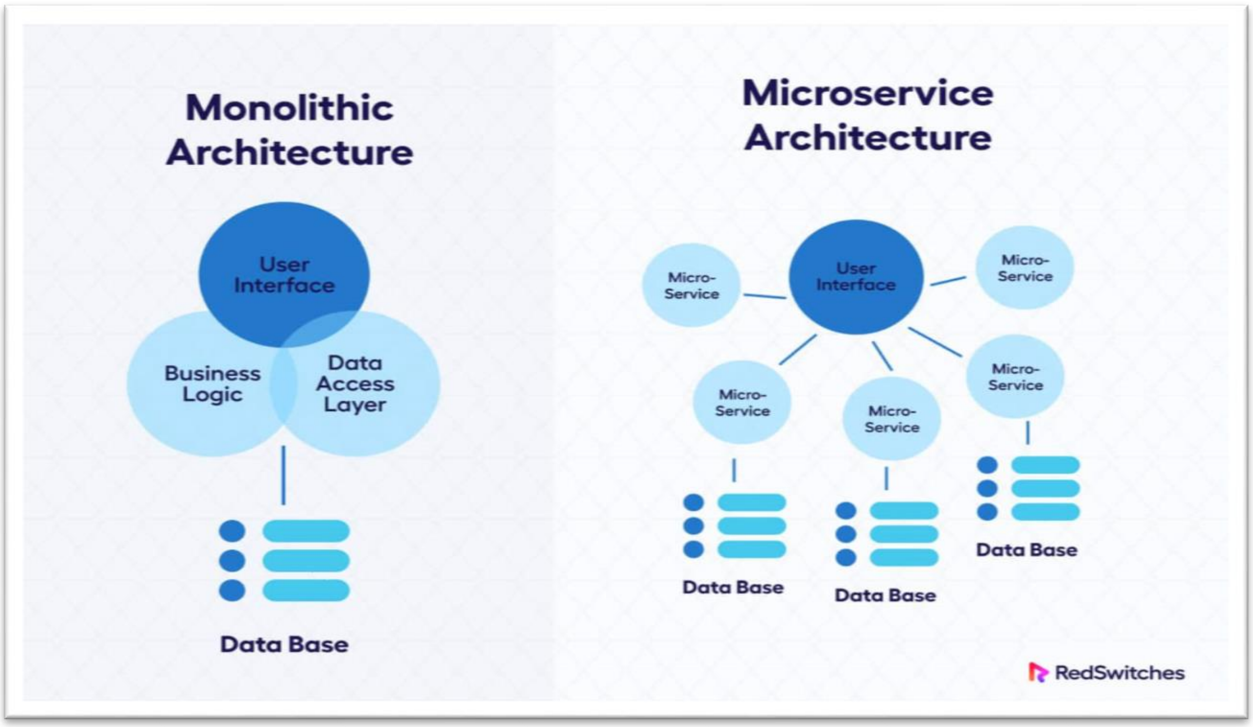

Figure 1. Monolithic vs Microservices [1]

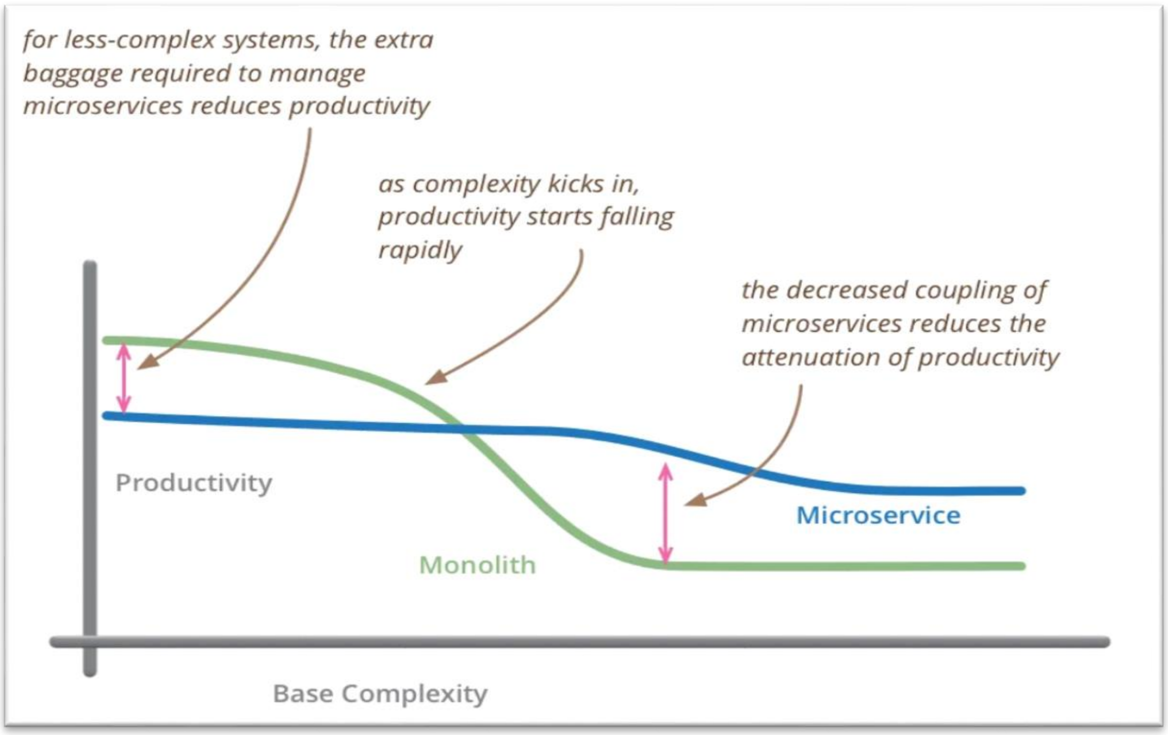

Figure 2. Productivity Graph Monolithic vs Microservices [2]

This graph above (Figure 3) shows that the productivity of monolithic architecture is high at first compared to microservices, but it drastically decreases below microservices when complexity kicks.

Planned Architecture

I plan to design my web application using microservices architecture as its flexibility, scalability enables services for recipe sharing and discovery to be simpler and more applicable than any other architecture. This architecture will divide the application into microservices that will include a user service that will handle authentication and user's profile management, a recipe service that will handle all the recipes and its details, an ingredient service that will handle ingredients details and shopping lists, a recommendation service that will handle user recommendations and personalized suggestions, and a database service that will handle data management and storage. REST architecture will also be applied to this application in which the architecture's components or microservices will be scaled and deployed by communicating through RESTful APIs. also, a RESTful API gateway service will handle client requests and authentication in order to create a discovery tool for recipe sharing that guarantees flexibility and isolation of errors.

Conclusion

In conclusion, this report has discussed three different architectures for designing a web application for recipe sharing and discovery API tool. The three architectures, monolithic, microservices, and RESTful, each had unique advantages and disadvantages. Monolithic architecture provides simplicity but lacks scalability and maintainability, microservices architecture provides scalability and fault isolation but increases complexity in communication and deployment, while RESTful architecture enables loose coupling and flexibility but also requires careful design to avoid data inefficiencies. After comparing the three architectures, I have chosen to implement the web application using a microservices architecture combined with a RESTful API. This combination is considered suitable for my web application design as it provides scalability, flexibility, maintainability, and fault isolation.

References

[1] J. J, “Monolith vs microservices: Comparison of 4 key areas,” RedSwitches, https://www.redswitches.com/blog/monolith-vs-microservices/ (accessed Mar. 12, 2024).

[2] P. Chamsomboon, “Monolithic vs Microservice, what do you choose?,” DEV Community, https://dev.to/panupongdeve/monolithic-vs-microservice-what-do-you-choose-5bkf (accessed Mar. 12, 2024).

[3] C. Roca, “REST API: What it is, how it works, advantages and disadvantages,” ThePower Business School, https://www.thepowermba.com/en/blog/rest-api-what-it-is (accessed Mar. 12, 2024).

[4] C. A. Team, “Advantages and disadvantages of microservices architecture,” Cloud Academy, https://cloudacademy.com/blog/microservices-architecture-challenge-advantage-drawback/ (accessed Mar. 12, 2024).

[5] O. Dushenin, “Monolithic architecture. advantages and disadvantages,” Medium, https://datamify.medium.com/monolithic-architecture-advantages-and-disadvantages-e71a603eec89 (accessed Mar. 12, 2024).

[6] “Monolithic vs Microservices Architecture: Pros and Cons,” OpenLegacy, https://www.openlegacy.com/blog/monolithic-application (accessed Mar. 12, 2024).