4/3/2024

Architecture Research

[Document subtitle]

Geerman,Genelle G.N.J.
FONTYS ICT

Introduction

System architectures are used by every developer to create a standard for their programs. These architectures are designed for a dedicated purpose, and I want to understand more about these architectures and why they were designed.

The purpose of the project is to host a tournament for quantum chess. The game itself is made by a different person but the tournament management is handled by our client. In essence, the project is a management system. Our client has also mentioned that they want to expand the idea into other games in the future.

Certain requirements need to be met for this project to work. These are going to be how we measure our results and how in-line the options are to our needs.

The non-functional requirements are as follows:

Scalability: With the plan to add more in the future, we need an architecture that allows for smooth integration by scaling the project size.

Security: With more users comes more sensitive data. We need to maintain security as much as possible.

Complexity: With more features comes more complexity. We want to minimize the complexity as much as possible when it comes to the development cycle.

Maintainability: This goes hand in hand with complexity. The system must be maintainable throughout its life cycle. To achieve this, we will have to investigate how quickly we can fix issues when they arise.

To maintain the scope of this research, I have limited the architecture to these 5:

- Client-Server
- Service-Oriented
- Event-driven Architecture
- Serverless
- Microservices

Client-server architecture

In client-server architecture, the client requests the server, and the server responds by providing a resource or service. This architecture is widely used in web applications, where the web browser acts as a client and communicates with a web server hosting the application logic and data.

Client-server architecture abstracts the services it provides to allow the client to not be concerned about performance. A client communicates with the server using an Application Programming Interface (API) which handles authorization and enforcing a specific format, allowing for parsing and cross-platform data exchange (Dustdar, 2005).

Service-Oriented Architecture (SOA)

SOA is an architectural style that focuses on discrete services instead of a monolithic design (SOA Source Book, 2021). SOA facilitates modular design, allowing components to be developed, deployed, and updated independently. It enhances operations by compartmentalizing services to allow for faster communications by standardizing the communication, making it easier to integrate with other systems. Services can be reused across different applications, reducing development time and cost.

SOA is not cost-effective as the time it takes to communicate between each service could mean that the application runs slower and uses more processing power, which could lead to an increase in cost. Additionally, testing does not follow a single paradigm. Each service can have a different requirement or testing framework as tests are more data-driven than traditional testing (WSO2, 2014).

Event-driven architecture

"An event (business or system) may signify a problem or impending problem, an opportunity, a threshold, or a deviation" (Michelson, 2006). Whenever an event happens, event-driven architecture dictates that all interested parties should receive and handle the process, while the invoker of the event should only be concerned about the event happening.

Serverless Architecture

Serverless architecture allows developers to build and run applications without managing the underlying infrastructure. Code is executed in stateless compute containers that are triggered by events. Serverless makes use of products such as docker and Kubernetes to ship and run on any platform. The name serverless itself is a misnomer as a server is required, however, the name originates from the product owner using a hosting service such as Amazon Web Services (AWS). Serverless eliminates the need for infrastructure as this is handled by the service which reduces operational overhead and cost, in turn enabling automatic scaling and pricing based on usage. This optimization of resources and costs balances out the disadvantages of microservices which is why they are often used together (Miller, 2015).

Serverless architectures may introduce vendor lock-in (heavy reliance on a third party and high difficulty of change), and limited control over the underlying infrastructure, potentially affecting performance and customization.

Microservices

A microservice is a small application that can be deployed independently, scaled, and tested independently and that has a single responsibility (Thönes, 2015). Microservices promote agility and scalability by allowing teams to develop, deploy, and scale services independently. They enable faster time-to-market by facilitating continuous delivery and deployment practices (Chen, 2018). Independent development of each service allows for the isolation of issues when problems arise.

Microservices introduce complexity in terms of service discovery, communication, and data consistency. Managing many services requires robust planning and automated deployment tools. (Calandra, 2021). However, microservices are incredibly scalable, which is one of its primary appeals.

Conclusion

In conclusion, the types of architecture I investigated were types of architectural types that define large service models and how they behave. Client-server architecture states that a client and a server communicate often using an API. This way it allows for data sharing and using it on every machine. SOA states that, instead of a monolithic approach, we split the project into smaller services which allows for reuse and easier integration with other existing services (through API calls e.g., HTTP calls) at a minor cost of performance and cost. Service-oriented is defined by splitting the product into smaller, contained services to allow for easier change and integration. While Event-driven architecture prioritizes performance and efficiency by having interested parties listen and respond. Serverless relies on the product owner hosting their services on a third-party platform, reducing cost and maintenance as well as handling architecture.

Microservices is an implementation of Service Oriented Architecture that uses HTTP as its way to communicate with other services. Often used in conjunction with serverless, microservices are highly scalable and maintainable while its cost and performance issues can be solved by integrating serverless.

These architectures help with the development of software by deploying patterns to define structures that facilitate faster development through commonalities and processes that have been tested rigorously and are adopted by industries that consider them standards.

References

Chen, Lianping. (2018). Microservices: Architecting for Continuous Delivery and DevOps. 10.1109/ICSA.2018.00013.

Dustdar, S.; Schreiner, W. (2005). "A survey on web services composition" (PDF). International Journal of Web and Grid Services. 1: 1. CiteSeerX 10.1.1.139.4827. doi:10.1504/IJWGS.2005.007545

Thönes, J. (2015). Microservices. IEEE Software, 32(1), 116-116.

SOA Source Book - What is SOA? (2021) https://collaboration.opengroup.org/projects/soa-book/pages.php?action=show&ggid=1314

Stine, M., & Melo, F. (2014). Developing Microservices for PaaS with Spring and Cloud Foundry. InfoQ. https://www.infoq.com/presentations/microservices-pass-spring-cloud-foundry/

WSO2. (2014). https://wso2.com/library/articles/2014/04/how-to-efficiently-test-service-oriented-architecture/

Michelson, B. (2006). Event-Driven Architecture Overview. https://doi.org/10.1571/bda2-2-06cc

Miller, R. (24 Nov 2015). "AWS Lambda Makes Serverless Applications A Reality"