1 Monolithic Architecture

Before going into Microservices and the Microservice architecture, the Monolithic architecture approach must be explained first. The Monolithic architecture approach was till recently the preferred design option for software. In a Monolithic application all different components and functions of the business logic are combined into one indivisible program[1]. Generally these components are the user interface, business rules and data access. While individual components might be developed separately, they remain tightly coupled[3] and any change completed in any of them requires the whole program to be rebuild and redeployed[4]. More often than not development in one component requires functional changes in multiple other, adding on the development cost, complicating the build and testing process and inducing delays in deployment. A single bug in any one component can potentially halt the entire application's operation and create a nightmarish situation for on-call engineers trying to figure out the root cause and usually resulting in multiple unrelated to the issue teams joining in till root cause analysis is complete. Additionally Monolithic applications usually have large codebases, which can be cumbersome when implementing changes and difficult to manage over time[3]. Another major issue with Monolithic applications is scalability. Usually different components have conflicting resource requirements but because of the unified design all requirements must be handled together making scaling up the application impossible vertically, only allowing horizontal scaling through multiple copies. Scaling horizontally is very resource consuming and restricted. Finally Monolithic design allows for little to no flexibility for incorporating newer, state of the art technologies, slowly resulting in legacy applications that have to be completely redesigned and reconstructed when performance degrades. Despite the many drawbacks of Monolithic architecture, it is still favored for certain applications because of some core benefits. The most important one is performance. In most cases Monolithic applications outperform their modular counterparts[3]. Also initial design and implementation are easier since individual components are usually clearly defined at later stages. Additionally a single codebase and unified build and deployment process simplifies configuration management, testing and monitoring[3]. It is clear that the Monolithic architecture approach works well for smaller applications and helps to get things up and running faster. Furthermore when development complexity and deployment time come second to performance a Monolithic application usually has the edge over a modular approach.

2 Microservice Architecture

Microservices and the Microservice Architecture are the last few years one of the most popular design option for software applications. In the Microservice Architecture the application is structured as a collection of independent services, called Microservices. Each Microservice corresponds to a different part of the

business logic, executing a well defined unique process[1][2]. These Microservices utilize lightweight communication mechanisms, such as API interfaces, so that they can operate in unison and achieve the same final results as a Monolithic application. Microservices can be build and tested separately and be deployed and scaled independently.

3 IoT device Simulators

There's little doubt that the Internet of Things is here to stay. IoT reshaped the way humans and machines interact with the environment and is now strongly influencing most industries. In recent years more and more everyday devices are coming equipped with all sorts of sensors and internet connectivity abilities, generating big amounts of data. Along with IoT enabled devices and sensors, there's a lot of development done on IoT applications in order to make use of all the generated data and better utilize the devices. Same as all other applications, IoT apps need to be thoroughly tested before deployment to production. One way for this to be done is to create an actual IoT network of devices, specific to the application, to generate all the data and test the application. It is not hard to notice the issues with this implementation. First of all, it is very expensive to create an actual IoT network of devices, especially when there's a need to test on large ecosystems. Furthermore, it is very common for an application early in development to be redesigned and changed often and in such case IoT ecosystem used for testing will have to be redesigned and changed as well. This adds to the development cost and introduces delays to the development process. Even if an IoT network of devices is already in place for deployment reasons, testing an application using the live network can introduce security risks. Instead of using actual data from real IoT devices, synthetic data can be generated, simulating real IoT networks.

References

[1] Konrad Gos and Wojciech Zabierowski. The comparison of microservice and monolithic architecture. pages 150–153, 04 2020.

[2] Martin Fowler James Lewis. Microservices: a definition of this new architectural term. https://martinfowler.com/articles/microservices.html, 2014.

[3] Ivy Wigmore Rahul Awati. What is monolithic architecture? https://www.techtarget.com/whatis/definition/monolithic-architecture, 2022.

[4] Freddy Tapia, Miguel Ángel Mora, Walter Fuertes, Hernán Aules, Edwin Flores, and Theofilos Toulkeridis. From monolithic systems to microservices: A comparative study of performance. *Applied Sciences*, 10(17), 2020.