Merce Bauza

6 Followers · About Follow

Monolithic vs SOA vs microservices

Merce Bauza Apr 9, 2019 · 2 min read

Recently I have joined a team that want to migrate their software architecture from monolithic to microservices.

I am going to dive into those terms to understand the differences between them and when to use which.

What are components?

In order to understand the types of architecture, I will briefly explain what the ambiguous term “components” refers to in this blog post.

A **component** is a unit of software that is independently replaceable and upgradeable.

What is a monolithic architecture?

The first type is **monolithic architecture**, where all the software components are built into a **single piece**, assembled together and tightly packaged.

The disadvantage of the monolithic architecture is that all the components are interconnected and interdependent, leading to a failing system if one of the components fails.

And also all the components must be present in order to execute or compile the app, failing to do so if one of them is not present or it fails.

It is commonly used when the web app is simple (and it will remain simple in the future) and the developer team is small.

What is SOA?

SOA stands for **Service-oriented architecture** and it refers to an architecture using a **collection of services**.

The services have different responsibilities but they still communicate with each other and are interconnected.

The communication can involve either simple data passing or two or more services coordinating some activity.

This type of architecture is more common when the web app has to be scalable, involves a bit more functionality and it is a bit more complex.

What are microservices?

The last type, (but not least) the **microservices architecture** takes the same approach of using a collection of services as the SOA and it takes it a bit further, where the services can be developed, deployed, and maintained independently without being interconnected and interdependent.

The main difference between the SOA and the microservices is that the services can communicate with other services through simple APIs, leaving the **services loosely coupled** between each other.

And another difference is that the services are **responsible of only a single thing**, where is SOA might have different responsibilities.

- Each service is a separate codebase, which can be managed by a small development team.
- Services can be deployed independently. A team can update an existing service without rebuilding and redeploying the entire application.
- Services are responsible for persisting their own data or external state. This differs from the traditional model, where a separate data layer handles data persistence.
- Services communicate with each other by using well-defined APIs. Internal implementation details of each service are hidden from other services.
- Services don't need to share the same technology stack, libraries, or frameworks.

Microservices


Download on the
App Store


GET IT ON
Google Play