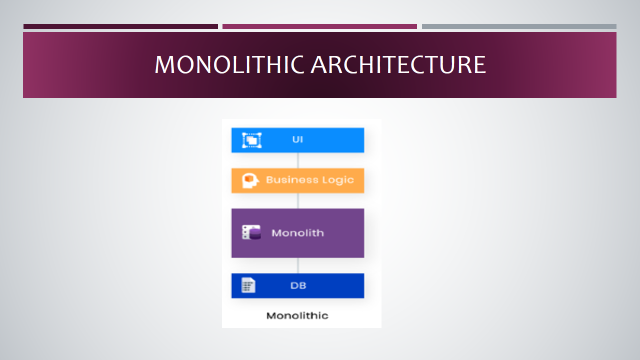


Microservices is an architecture of splitting tightly packed application into smaller or chunks of applications.

Microservices has taken the application development industry to the next level .it has left a great impact on organizations by putting together many major benefits.

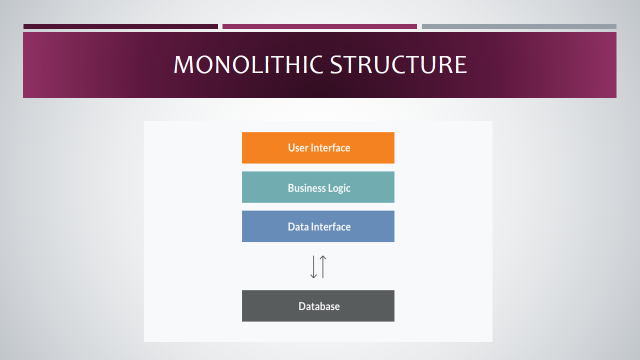
But before a deep dive into microservices. Let’s have a look on

* What was the approach before microservices?
* What were the drawbacks of other software architecture that made microservices rising these days?



**Monolithic Approach, the previous trend**

Traditionally, monolithic approach was in trend before microservices. Monolith means assembled everything in one piece. The monolithic architecture defines a single-tiered software framework in which various modules from a single platform are merged into one system.



**Components of Monolithic Application**

Components can be:

* Authorization — responsible for authorizing a user
* Presentation — responsible for handling HTTP requests and responding with either HTML or JSON/XML (for web services APIs).
* Business logic — the application’s business logic.
* Database layer — data access objects responsible for accessing the database.
* Application integration — integration with other services (e.g. via messaging or REST API). Or integration with any other Data sources.
* Notification module — responsible for sending email notifications whenever needed.

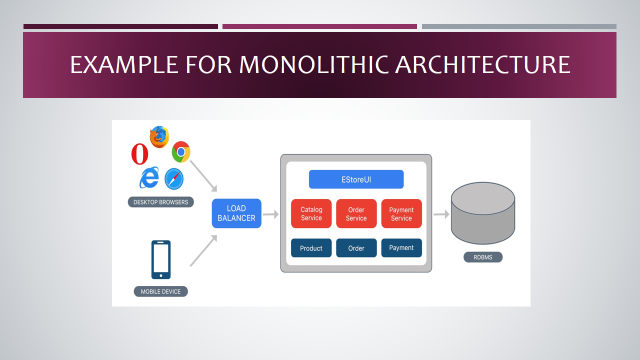
**Example for Monolithic Approach**

Consider an example of an Ecommerce program, which authorize buyers, places an order, checks inventory of goods, authorizes payment and ships ordered products. The application consists of many components

including e-Store Customer User Interface (Store Web View) along

with some backend resources to test inventory, approve and

 charge payments and shipping orders for goods.



**Merits of Monolithic Application**

* Easy to develop — It's much simpler to go with Monolithic

Architecture at the beginning of a project.

* Simple to check — You can implement end-to-end testing,

for example by simply launching the application and testing the

UI with Selenium.

* Simple to deploy —The packed program must be copied to server.

Easy to scale horizontally —with several copies running behind a

load balancer.

**De-Merits Of Monolithic Application**

* Maintenance — If the application is too large and complex to fully

understand, it is difficult to make changes quickly and correctly.

* The application size will slow down startup time.
* After every update you need to redeploy the entire program.
* When separate modules have conflicting resource specifications,

monolithic systems can also be difficult to scale.

* Reliability — The bug in any module (e.g. memory leak) can potentially

cause the entire process to fail. Moreover, since all the instances of

the application are identical, this bug affects the availability of the entire

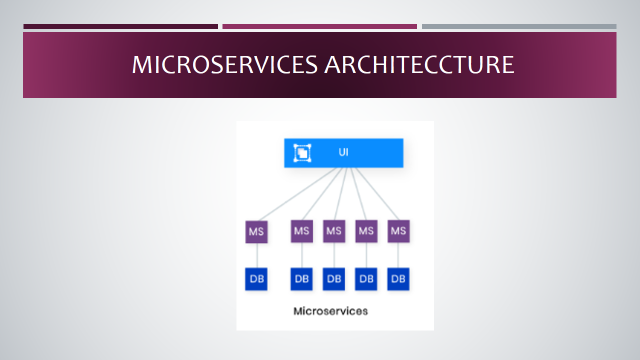
application.

* No matter how easy the initial stages may seem, monolithic applications have difficulty adopting new and advanced technologies. Since

Language or framework changes affect the entire application, efforts

are needed to work closely with the app details, which makes it costly

to consider both time and effort.



MICROSERVICES

Microservices are an approach to the creation of software

in which a broad application is designed as a suite of modular

services (i.e. loosely coupled modules / components).

Each module supports a specific business goal and uses a

simple, well-defined interface to communicate with other

services.

Instead of using a common database as in Monolithic,

Each microservice has its own database. Having a database

per service is necessary if you want to take advantage of

microservices, as it ensures a loose coupling.

Each service has its own database. In addition, a company can use a form of database that is best suited to its needs.

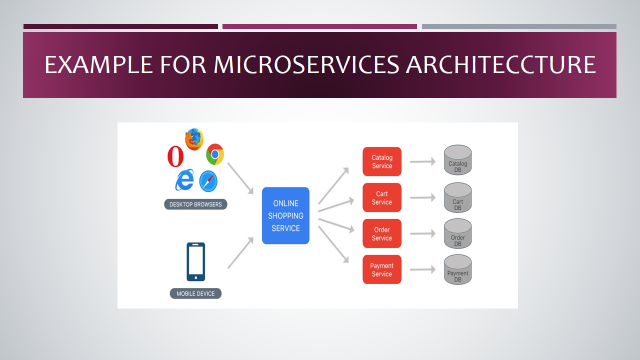
**Example for Microservices Approach**

Consider the same example of an e-commerce program consisting of

many components / modules. Define each component / module as a

separate, loosely coupled unit, depending on the requirement that can

operate with each other on the basis of the scenario.



**Services of Microservices Application**

For a complete application, we can have the following services:

* Authorization Service — Responsible for the authorization of the

Customer

* Request Service — takes a request and processes it.
* Catalog Service — Manage goods and review inventory of items.
* Cart Service — Manage the user cart, this service can use the Catalog service as a data source.
* Payment Services — Manage and approve payments.
* Shipping Service — Products ordered by ships.

**Merits of Microservices Application**

* Microservices Enables the continuous production

and implementation of massive, complex applications.

* Improved testability — Services are smaller and easier to

test.

* Good deployment — Services may be deployed

Independently.

* Each microservice is relatively small.
* Comfortable for a developer to understand.
* The IDE is faster making developers more productive.
* The application starts faster, which makes developers more productive, and speeds up deployments.
* Improved fault isolation. For example, if there is a memory leak in one service then only that service is affected. The other services continue to handle requests. In comparison, one misbehaving component of a monolithic architecture can bring down the entire system.

**De-Merits Of Microservices Application**

* Developers must address the additional complexity of creating a

distributed system.

* Developer tools / IDEs are based on developing monolithic

applications and do not provide specific support for the creation of

distributed applications.

* Testing is more difficult compared to Monolith applications.
* Developers are expected to incorporate an inter service

 communication mechanism. The implementation of cases

involving multiple services requires careful coordination

between the teams.

* Complexity of deployment.
* In production, there is also the operational complexity of

deploying and managing a system of many different types

of services.

**CONCLUSION**

Well, both methods have their merits and demerits, but it depends on

each situation or product / project specifications and which trade- you want. As a monolithic approach best suited for lightweight applications

It is recommended to adopt a monolithic approach first and gradually

shift towards a microservice approach depending on the needs / requirements.