Microservices

R M Shahidul Islam Shahed

What are microservices?

Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are

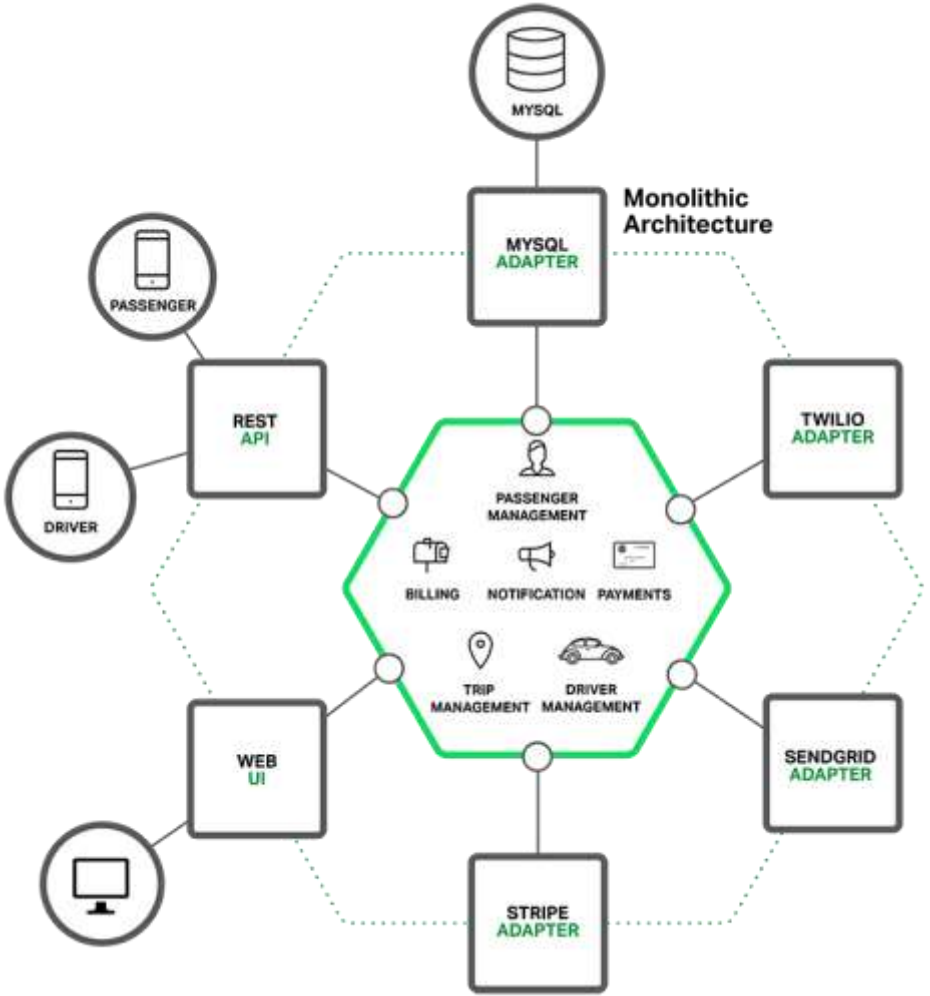
- ★ Highly maintainable and testable
- ★ Loosely coupled
- ★ Independently deployable
- ★ Organized around business capabilities.

What are microservices?

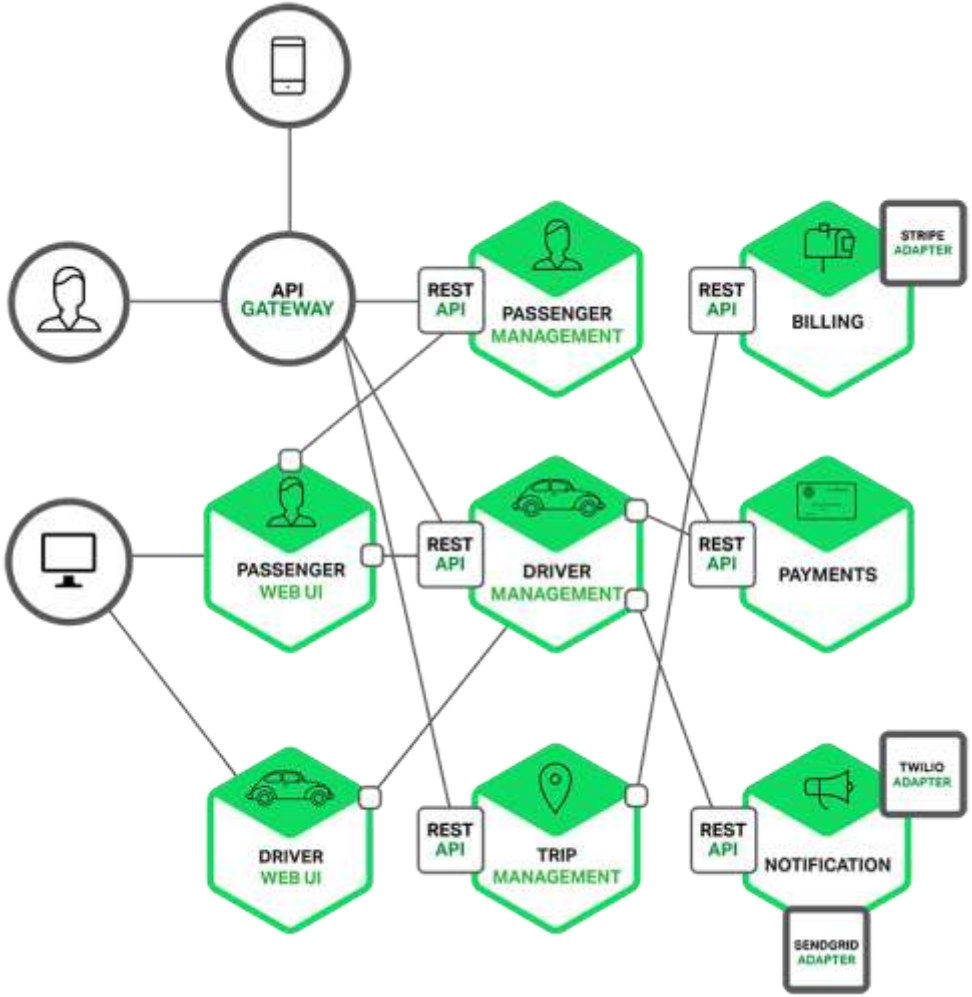
Microservices are a way of breaking large software projects into loosely coupled modules, which communicate with each other through simple Application Programming Interfaces (APIs).

Based on the philosophy of breaking large software projects into smaller, independent, and loosely coupled modules.

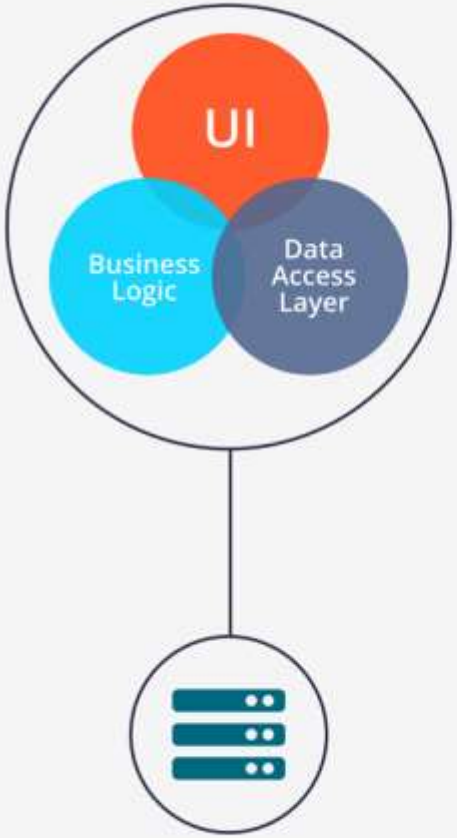
Monolithic Architecture


MYSQL
Monolithic Architecture
MYSQL ADAPTER
PASSENGER
DRIVER
REST API
TWILIO ADAPTER
PASSENGER MANAGEMENT
BILLING
NOTIFICATION
PAYMENTS
TRIP MANAGEMENT
DRIVER MANAGEMENT
WEB UI
SENDGRID ADAPTER
STRIPE ADAPTER

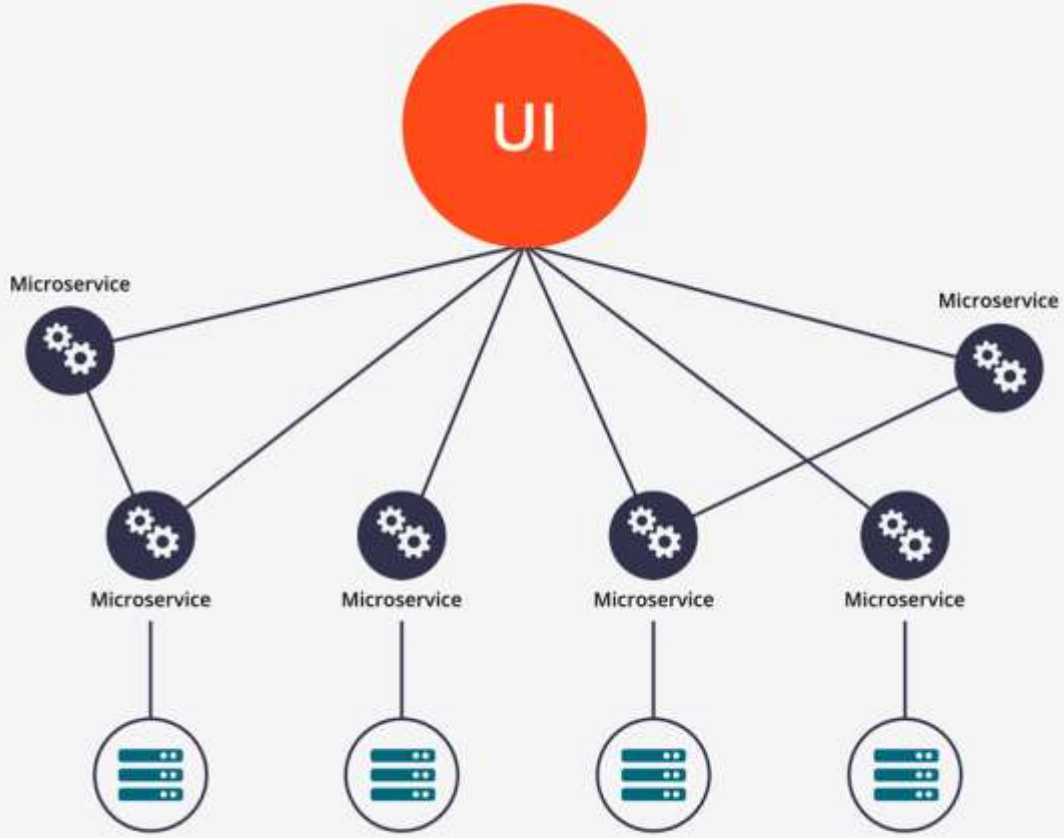
Microservice Architecture


API
GATEWAY
REST
API
PASSENGER
MANAGEMENT
REST
API
BILLING
STRIPE
ADAPTER
PASSENGER
WEB UI
REST
API
DRIVER
MANAGEMENT
REST
API
PAYMENTS
DRIVER
WEB UI
REST
API
TRIP
MANAGEMENT
REST
API
NOTIFICATION
TWILIO
ADAPTER
SENDGRID
ADAPTER

Microservices vs Monolithic



Monolithic Architecture



Microservice Architecture


DEVELOPER
MICROSERVICES
MONOLITHIC

Microservices vs SOA

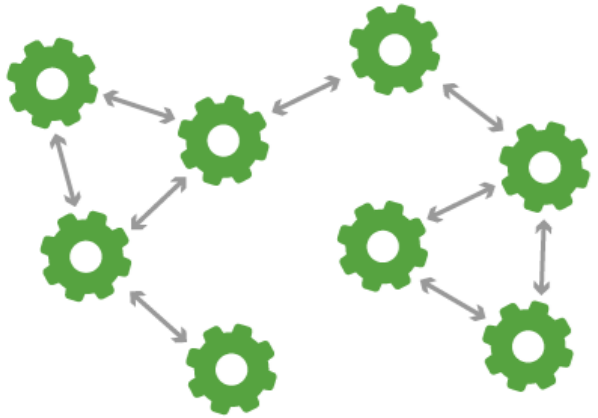
2000's

SERVICE ORIENTED ARCHITECTURE

SOA based applications are compromised of more loosely coupled components that use an Enterprise Services Bus messaging protocol to communicate between themselves.

2010's

MICROSERVICES ARCHITECTURE



Microservices are a number of independent application services delivering one single functionality in a loosely connected and self-contained fashion, communicating through light-weight messaging protocols such as HTTP, REST or Thrift API.

SERVICE ORIENTED ARCHITECTURE	MICROSERVICES ARCHITECTURE
Maximizes application service reusability	Focused on decoupling
A systematic change requires modifying the monolith	A systematic change is to create a new service
DevOps and Continuous Delivery are becoming popular, but are not mainstream	Strong focus on DevOps and Continuous Delivery
Focused on business functionality reuse	More importance on the concept of "bounded context"
For communication it uses Enterprise Service Bus (ESB)	For communication uses less elaborate and simple messaging systems
Supports multiple message protocols	Uses lightweight protocols such as HTTP, REST or Thrift APIs
Use of a common platform for all services deployed to it	Application Servers are not really used, it's common to use cloud platforms
Use of containers (such as Docker) is less popular	Use of containers (such as Docker) is less popular
SOA services share the data storage	Each microservice can have an independent data storage
Common governance and standards	Relaxed governance, with greater focus on teams collaboration and freedom of choice

Utilization of MicroServices By

Best Practices

- ★ Find the best microservices architecture
- ★ Outline your microservices
- ★ Domain-Driven Design
- ★ Get everyone onboard
- ★ Utilize RESTful APIs
- ★ Build teams for specific microservices

Best Practices

- ★ Setup server and data storage environment
- ★ Document API’s
- ★ Use the best DevOps toolkit
- ★ Monitoring is key, monitor everything with a tool
- ★ Use automatic security updates

Benefits of Microservices

- ★ Follow the **Single Responsibility Principle**
- ★ **Resilient/Flexible** – failure in one service does not impact other services. If you have monolithic or bulky service errors in one service/module it can impact other modules/functionality.
- ★ **High scalability** – demanding services can be deployed in multiple servers to enhance performance and keep away from other services so that they don't impact other services. Will be difficult to achieve same with single, large monolithic service.

Benefits of Microservices

- ★ **Easy to enhance** – less dependency and easy to change and test
- ★ **Low impact on other services** – being an independent service, this has less chance to impact other services
- ★ **Easy to understand** since they represent the small piece of functionality
- ★ **Ease of deployment**
- ★ **Freedom to choose technology** – allows you to choose technology that is best suited for a particular functionality

Disadvantages of Microservices

- ★ Multiple databases and transaction management can be painful.
- ★ Testing a microservices-based application can be cumbersome.
- ★ Deploying microservices can be complex.
- ★ Inter Service Communication
- ★ Health Monitoring
- ★ Distributed logging
- ★ Cyclic dependencies between services

Reference

1. https://github.com/mfornos/awesome-microservices
2. https://github.com/dotnet-architecture/eShopOnContainers
3. https://microservices.io/
4. https://www.amazon.com/gp/product/1617294543
5. https://www.redhat.com/en/topics/microservices/what-are-microservices
6. https://www.qat.com/15-benefits-microservices/
7. https://dzone.com/articles/microservices-vs-soa-whats-the-difference
8. https://medium.com/@kikchee/microservices-vs-soa-is-there-any-difference-at-all-2a1e3b66e1be
9. https://www.devteam.space/blog/10-best-practices-for-building-a-microservice-architecture/
10. https://www.nginx.com/blog/introduction-to-microservices/
11. https://dotnet.microsoft.com/learn/web/microservices-architecture
12. https://www.c-sharpcorner.com/article/microservice-using-asp-net-core/
13. https://github.com/GoogleCloudPlatform/microservices-demo