**MicroServices**

**Content**

1. **Defination**
2. **Evolution**
3. **MicroServices Architecture**
4. **Built an API**
5. **Messaging**
6. **Service Registery**
7. **Activities Services**
8. **Identity Service**
9. **Json Web Token**
10. **API Gateway**
11. **System Testing**
12. **Dockerizing and deploying the Application**
13. **Setting Up development Environment**
14. **MICROSERVICES File Structrure**

**Before MicroServices:**Which is **Monolithic Architecture.**

**Disadvantages:**



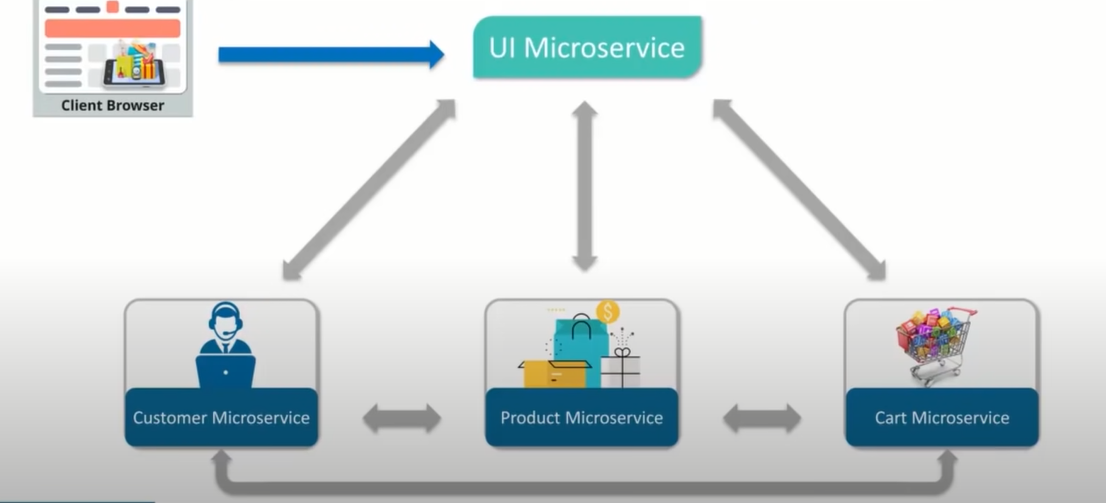
Evolution Of MicroServices:



They don’t share data structures but communicated threw API’s.

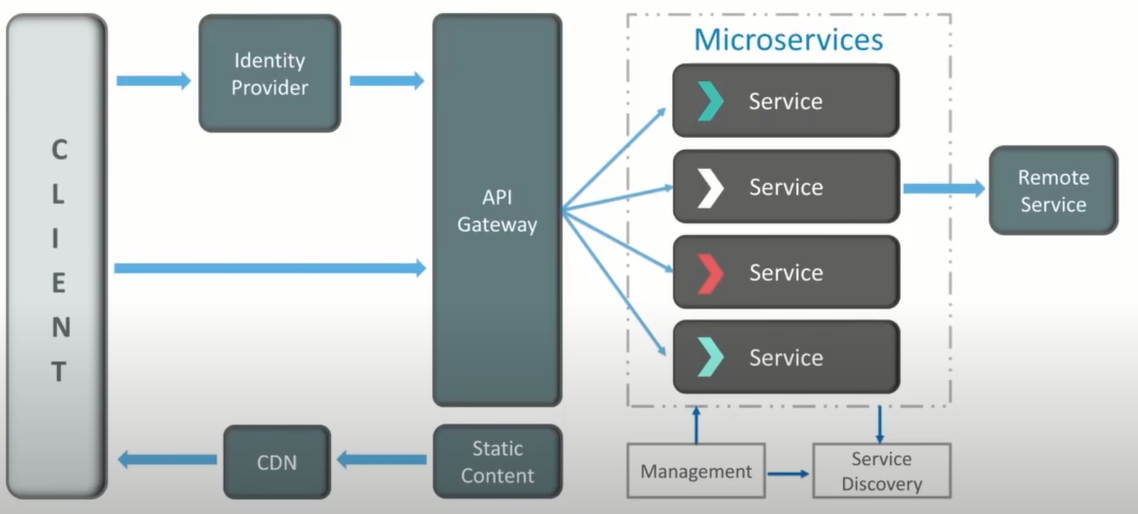


**Structure Of MicroServices**:With e-commerce example.



* Compenents divided into seperate modules.
* State-less
* Data is federated(has own data-model & data.)
* Multiple instances for one MicroServices

**Architecture:**



Microservice is :

* Strong
* Independent
* Lossly -coupled
* Code -based(individually)
* May not be same technology,library or stack

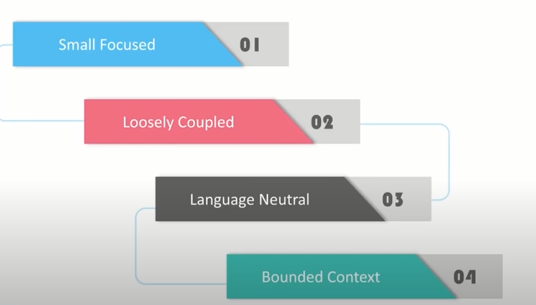
**Management Component:**

Placing services from nodes.Service lookup to find end point for a service.

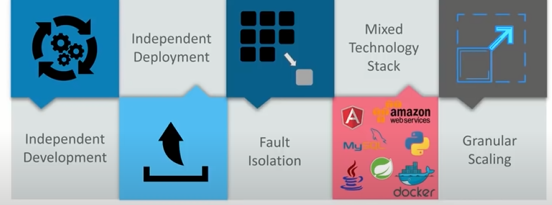
**API GATEWAY :**

Forwarding call to back-end.Agregate the results from various services and return the response.

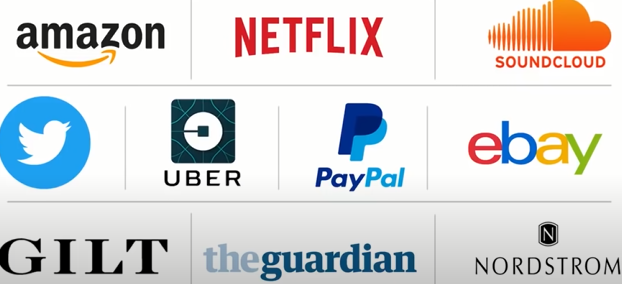
**Features of Microservice:**



**Adavntages Of MicroServices:**



**Companies Using MIcroServices:**



**Build an API:**

* Model(model->dto)
* Repositories(repository-> Irepository)
* Business(business->Ibusiness)
* Controller
* MappingConfig.cs
* startUp.cs

**Structure:**

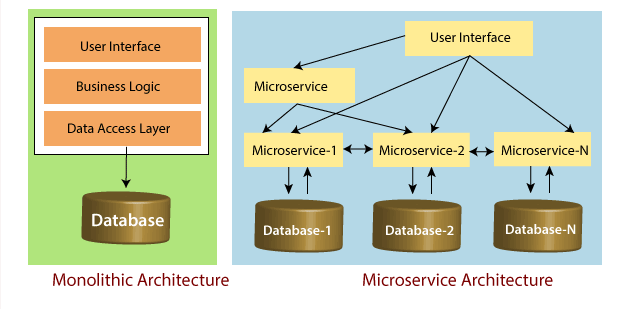
M

|  |  |
| --- | --- |
| **Repositories** | **Business** |
| **1.DapperContext** | **1.repositories/Methods** |
| **2.async method with DTO** | **2.async method with model** |
| 1. **wait.resp.respName**   **Mapper** | 1. **createConn**   **Await <model>(cond)**  **return** |

**REPOSITORIES:**

**BUSINESS LOGIC:**

**CONTORLLERS:**



**MICROSERVICES File Structrure:**

API GateWay:

* Tarffic routing
* Expose unified End Point
* API Composition
* Caching
* Logging
* Authentication
* Authorization
* Load balancing
* Service Discovery

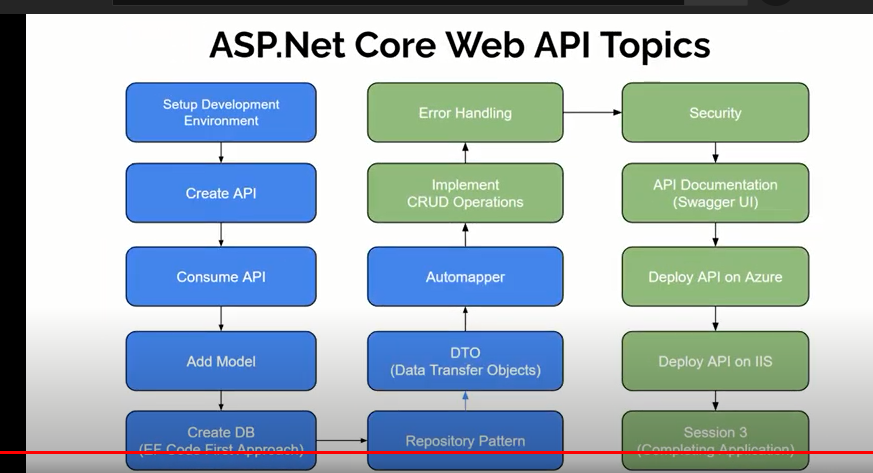
Types:

1. Azure API GateWay:
2. Ocelot API GateWay:(No dependency)

Ocelot API GateWay:

**Setting Up development Environment:**

* Visual studio Coummunity 2019
* Sql Server Express edition 2017
* Visual Studio Code (for IDE)



Dictionary:

DistributedCache

SetStringAsync

GetListAsync