**Day 3**

Create the folder as sequelize-product-api

cd sequelize-product-api

npm init -y package.json file

npm install express pg sequelize pg-hstore

express 🡪 to create rest api

pg -🡪 to connect db

sequelize🡪 provide model with function to do operation on table.

pg-hstore 🡪 model serialization and deserialization taken care by this module.

Put and patch method is use to update existing records

Put : it is use to update object all property : pname and price using id

Patch : it use to update few property or partial object. like using pid update pname or price etc.

Mkdir product\_end\_point\_mvc\_ sequelize

cd productend\_point\_using\_mvc\_style

npm install express pg sequelize pg-hstore

config -🡪 folder

db.config.js which contains database connection.

model 🡪 Folder

ProductModel.js

Which provide schema information like Product with all variable names and their data types.

Repository 🡪folder

ProductRepository.js

Which contains pure database related code.

Controller 🡪 folder

ProductController.js

Contains req and res information this function interact with repository to do operation on table.

Routes 🡪 :

ProductRoute.js

Which provide all http method information as well as sub path and base upon sub path it will call controller specific methods.

App.js or index.js or main.js

Load all modules and enable any middleware if need and provide main path for respective route file and provide port number with listener.

Product

PID (PK) PName Price

Orders

OId(PK), OrderDate, Pid(FK)

Micro Service : According to micro service, that service is responsible to do small task. Like product, orders, customer, login etc. these service can be created using same language or different language using same db or different db. It can be deploy independently using different port numbers.

LoginModule customer module product module

Model

Repository

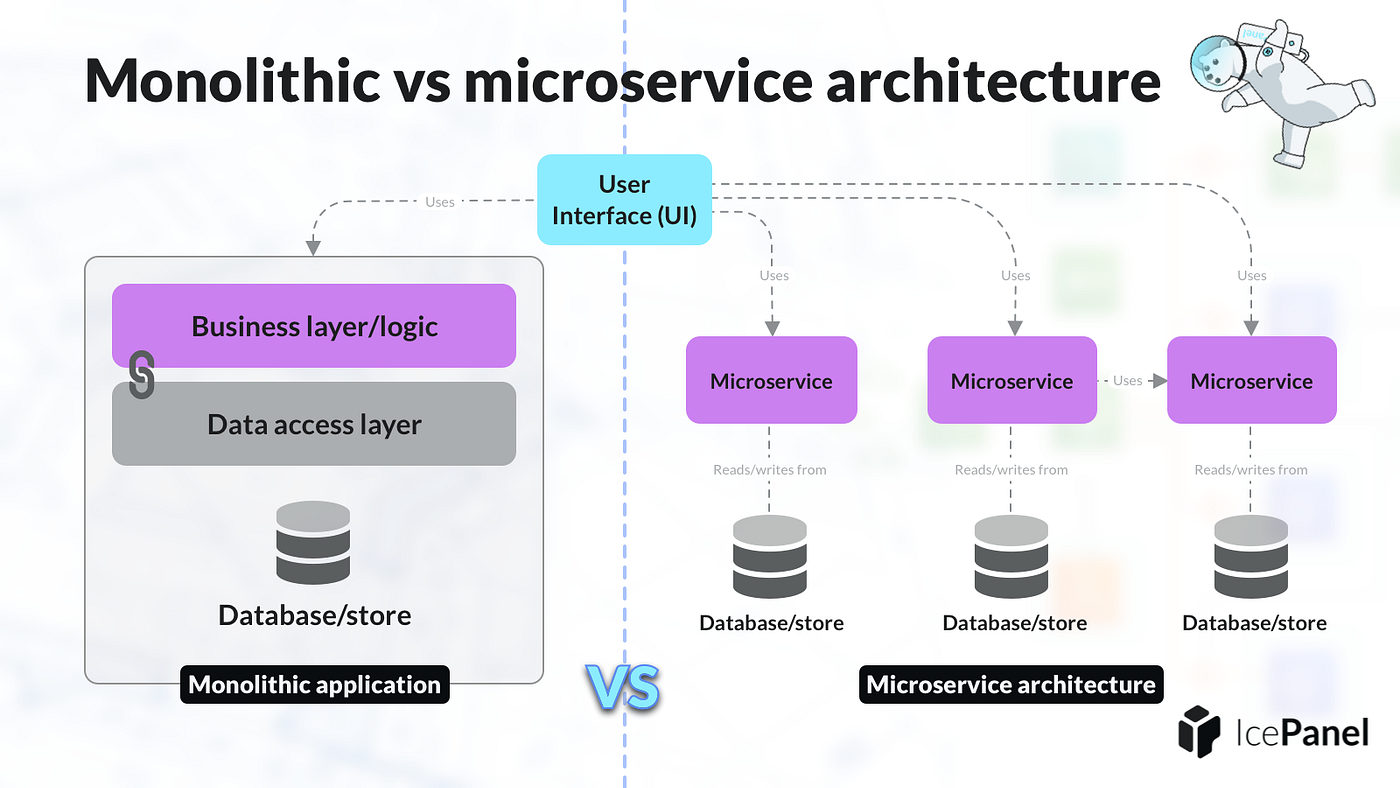
Controller

Router

Payment module feedback module

GitHub, which help us to create shared repository.

CI and CD tools Jenkin



**Axios react JS**

**Fetch JS**

**HttpClient Angular**

**RestTemplate : Spring boot**

**FeignClient : Spring boot**

**WebClient : Spring boot**

**1st we need to use Eureka Server to deploy more than one micro service created using same or different language.**

**Spring boot provide us Eureka Server which help to deploy more than one micro service created using different languages.**

**2nd consule also third party module which help us to deploy more than one micro service develop using same or different language.**

**Please refer below URL to download consule**

[**https://developer.hashicorp.com/consul/install?ajs\_aid=a1152d2e-05b5-4163-926f-3fac8ac0e3ab&product\_intent=consul**](https://developer.hashicorp.com/consul/install?ajs_aid=a1152d2e-05b5-4163-926f-3fac8ac0e3ab&product_intent=consul)

**please extract downloaded folder**

**the open the terminal or command prompt**

**consul agent -dev**

[**http://localhost:8500/ui**](http://localhost:8500/ui)

**account micro service 9191**

mkdir account-micro-service

cd account-micro-service

npm init -y

npm install express consul

**it hold more than one account details in array or from db.**

**Accno, name, emailid, amount**

**Using emailid get the amount**

**This end point we can test using 9191**

**Using db as postgres**

**gpay micro service 9292**

mkdir gpay-micro-service

cd gpay-micro-service

npm init -y

npm install express consul axios

**it hold gpay id and emailid**

**if we pass gpay id and using gpaid get emailid using this emailid we want to check the balance.**

**Using db as mysql**

**Node js provided pre defined core modules ie net modules which help to communicate between more than one node or machine using TCP/UDP protocol.**