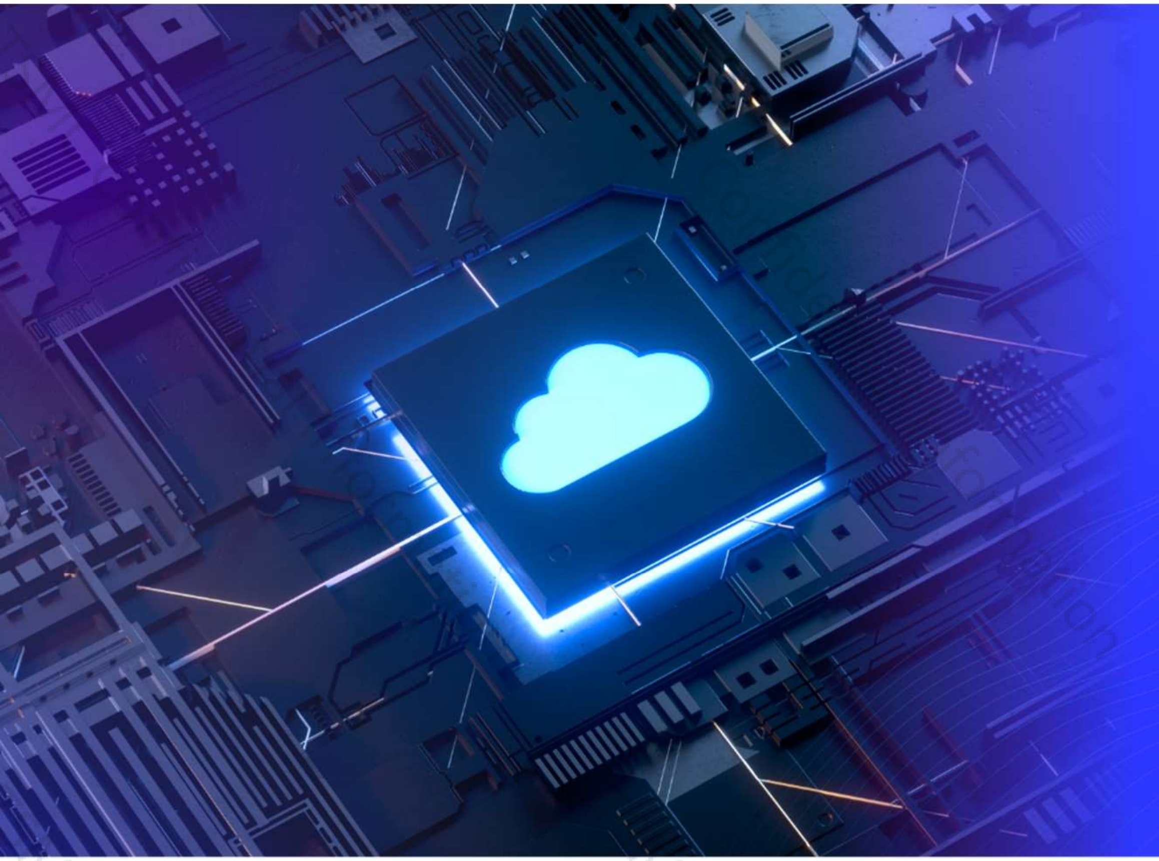
台灣積體電路製造股份有限公司
Taiwan Semiconductor Manufacturing Company, Ltd.



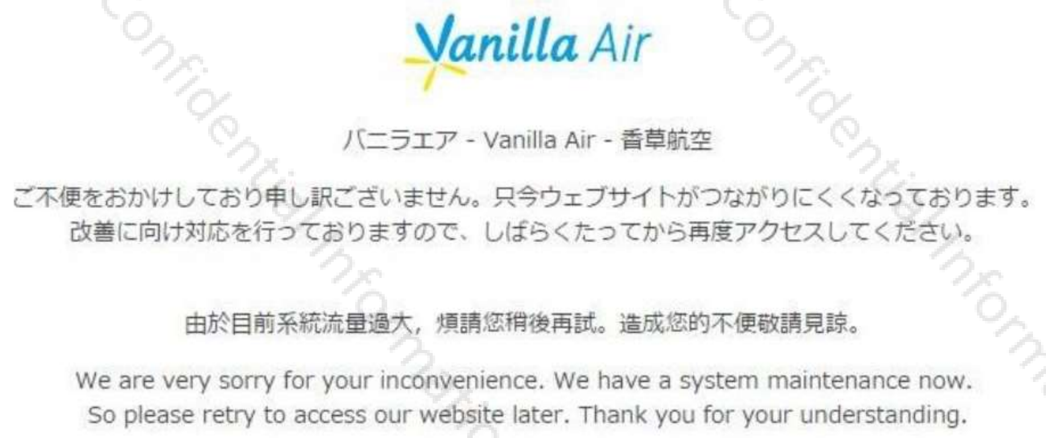
TSMC IT X NCTU CS 課號 5270

CLOUD NATIVE
Development Best Practice

MONOLITHIC TO MICROSERVICES

TSID
IESD | Ricky Chao
April 13, 2022

myfone 購物
PS5
PLAY HAS NO LIMITS
3/11 中午12:00 準時開賣

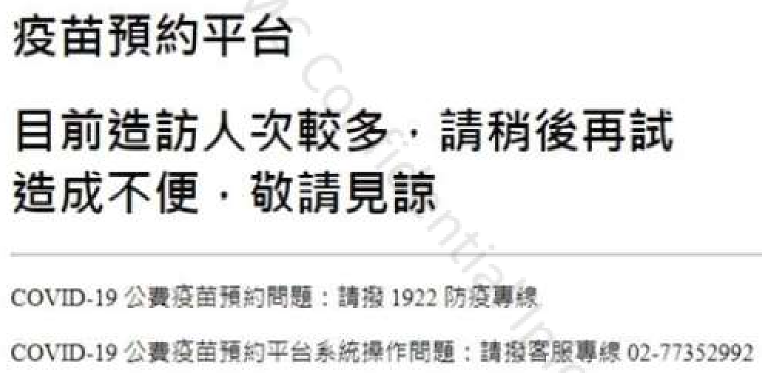


バニラエア - Vanilla Air - 香草航空

ご不便をおかけしており申し訳ございません。只今ウェブサイトがつながりにくくなっております。
改善に向け対応を行っておりますので、しばらくたってから再度アクセスしてください。

由於目前系統流量過大，煩請您稍後再試。造成您的不便敬請見諒。

We are very sorry for your inconvenience. We have a system maintenance now.
So please retry to access our website later. Thank you for your understanding.



疫苗預約平台

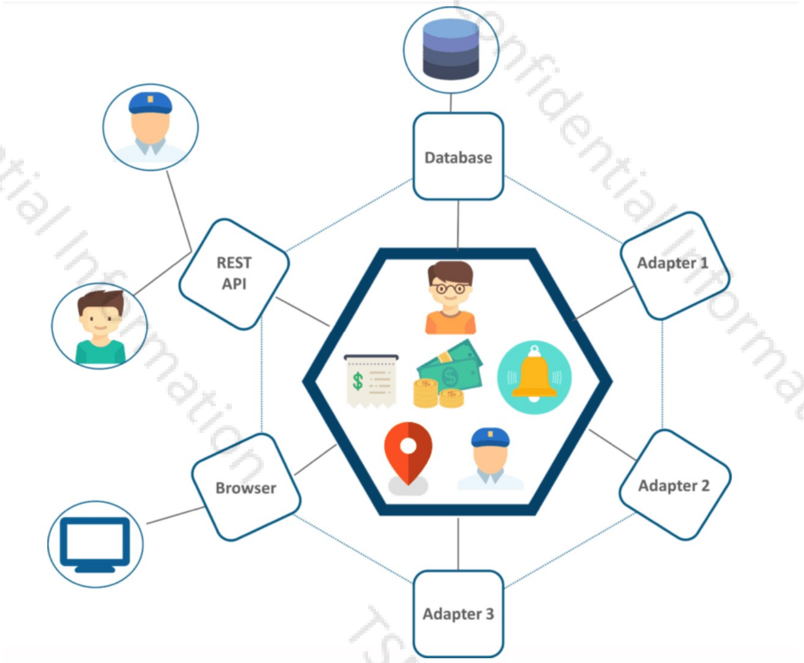
目前造訪人次較多，請稍後再試
造成不便，敬請見諒

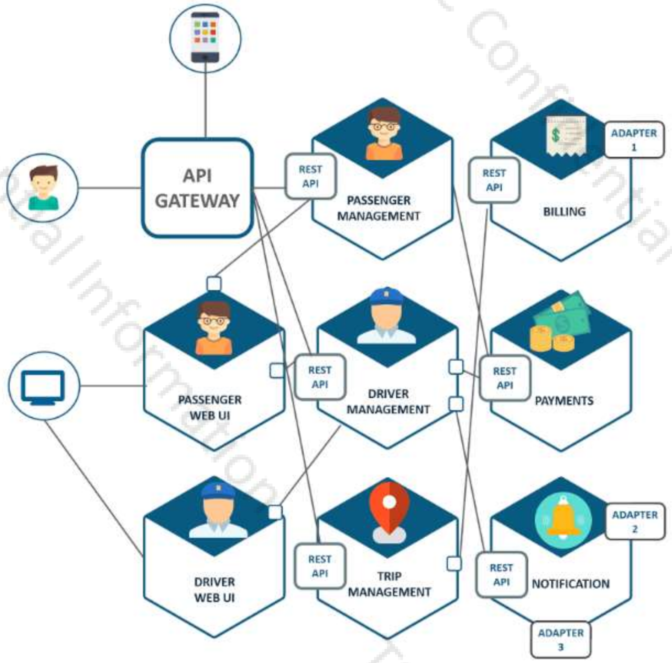
COVID-19 公費疫苗預約問題：請撥 1922 防疫專線

COVID-19 公費疫苗預約平台系統操作問題：請撥客服專線 02-77352992

About Me

- **Education**
  - **NTHU CGLab** (2001 ~ 2003)
- **Work Experience**
  - **Tecent** (2017 ~ 2019)
    - Senior Researcher
  - **ASUS AICS** (2019 ~ 2020)
    - Engineer Manager
  - **Amazon AWS** (2020 ~ 2021)
    - Cloud Architect
  - **TSMC IESD** (2021 ~ present)
    - Technical Manager
- **Contact**
  - **jhchao@tsmc.com**


Database
REST API
Adapter 1
Adapter 2
Adapter 3
Browser


API GATEWAY
REST API
PASSENGER MANAGEMENT
REST API
BILLING
ADAPTER 1
PASSENGER WEB UI
REST API
DRIVER MANAGEMENT
REST API
PAYMENTS
DRIVER WEB UI
REST API
TRIP MANAGEMENT
REST API
NOTIFICATION
ADAPTER 2
ADAPTER 3

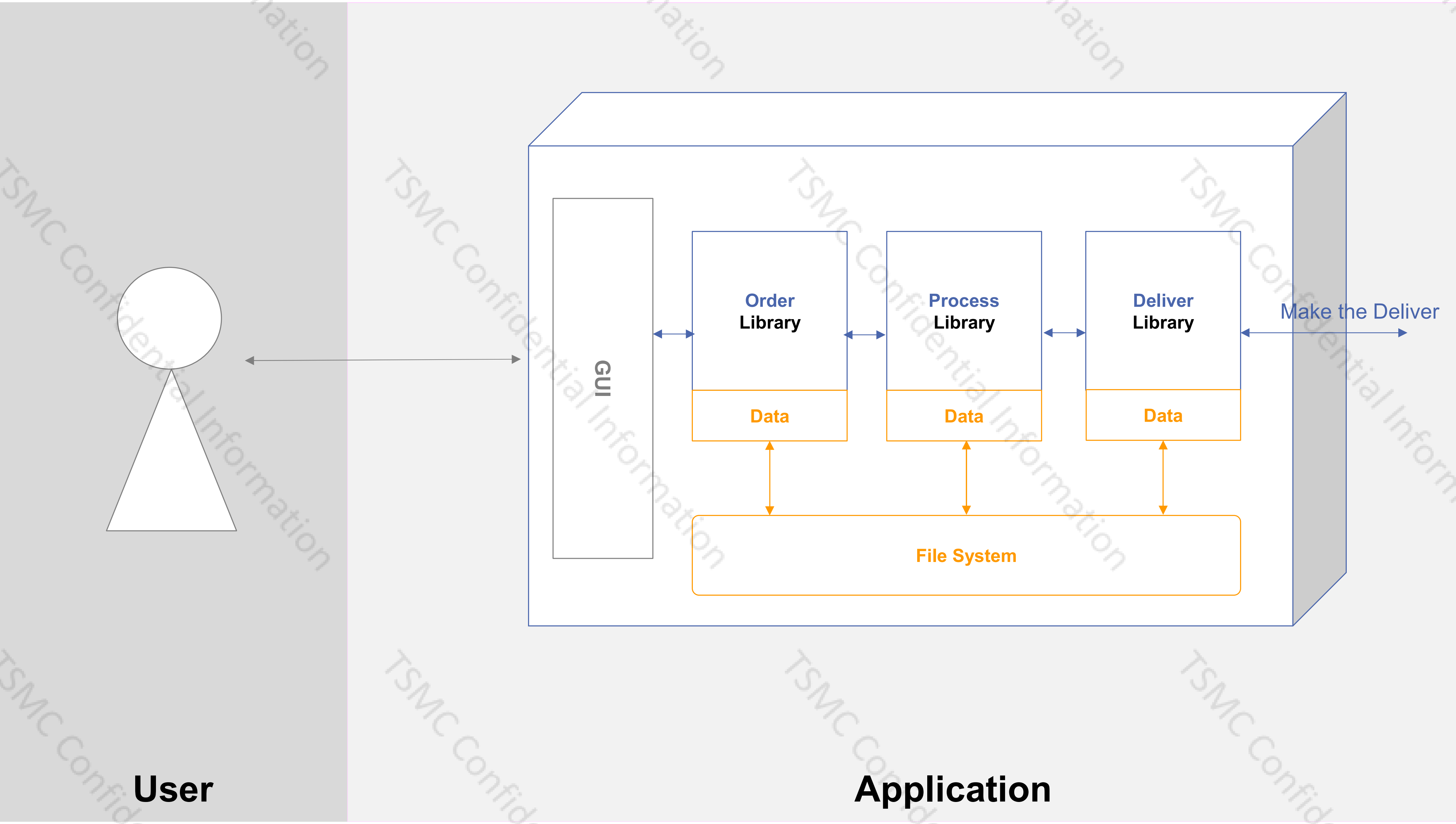
Preparation

- Install nodejs
  - https://nodejs.org/en/
- Download and setup mongodb
  - https://www.mongodb.com/try/download/community
  - https://www.mongodb.com/try/download/compass
- Download and setup nats-server
  - https://github.com/nats-io/nats-server/releases/

```
> brew install nats-streaming-server
```

Exercise

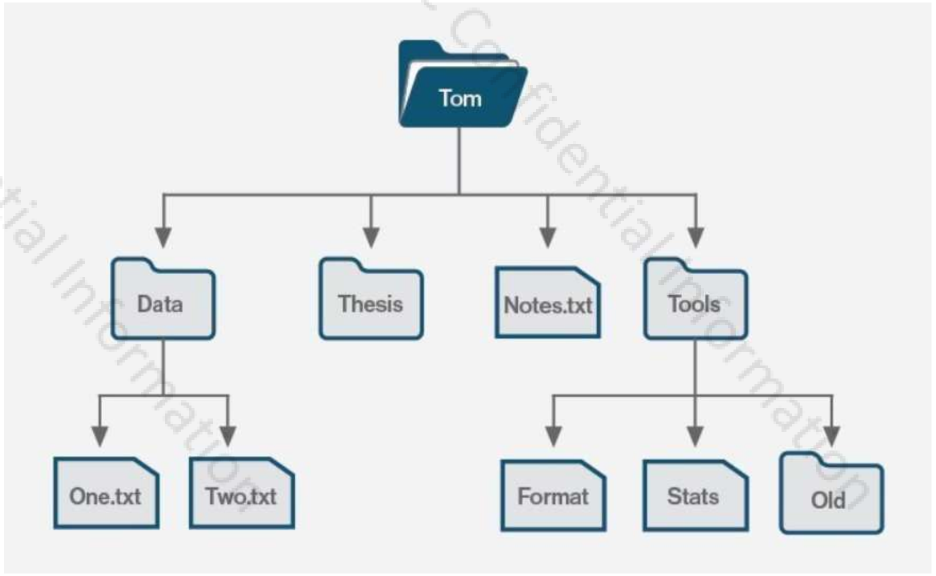
```
node .\exercise_0\index.js
```


GUI
Order Library
Process Library
Deliver Library
Data
Data
Data
File System
Make the Deliver
User
Application

Exercise

```
node .\exercise_1\index.js
```

Problem…



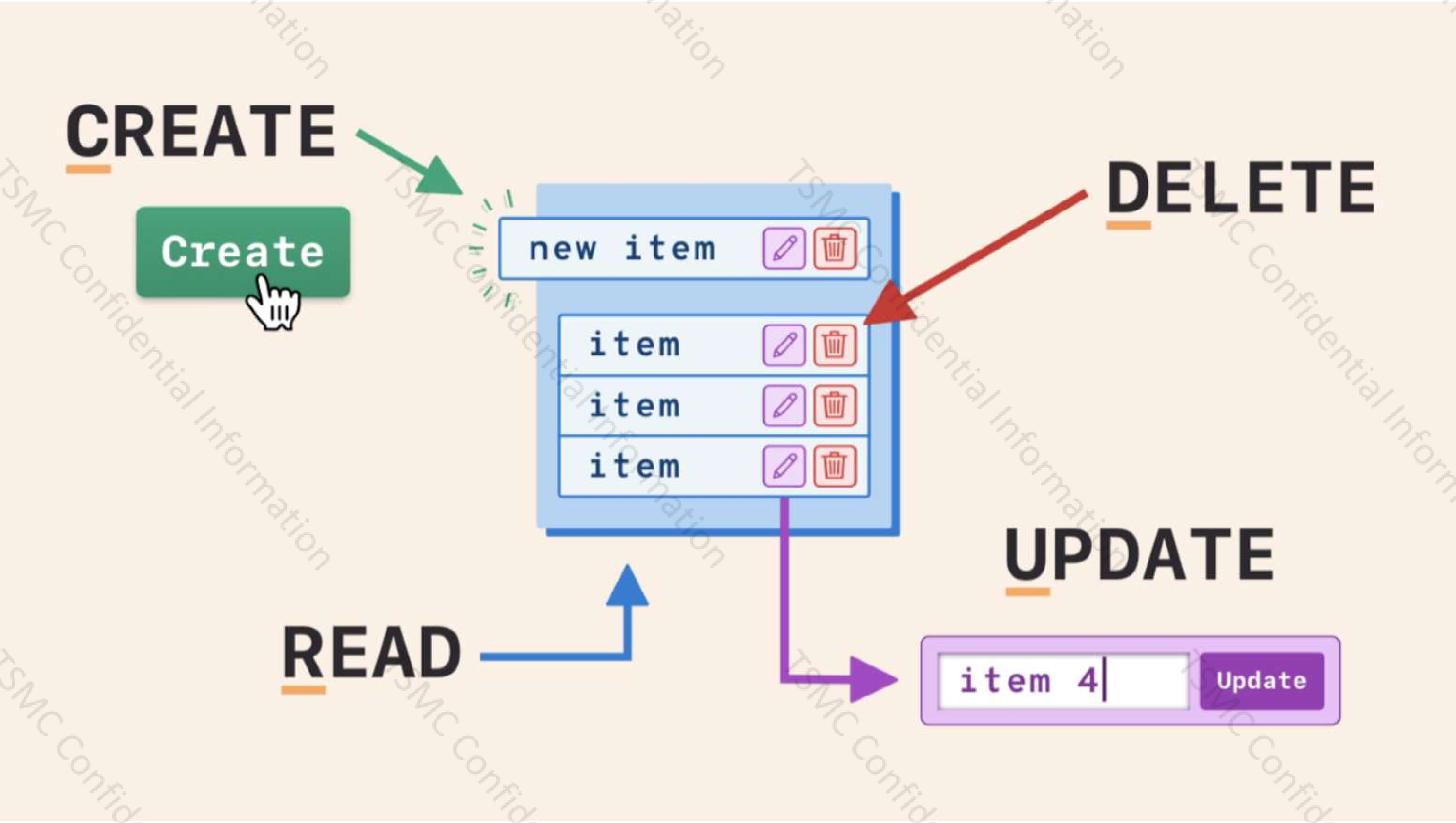
- **Non Efficiency**
  - Storing and retrieving of data can’ t be done efficiently
- **Inconsistency Issue**
  - Data inconsistency is higher in the file system
- **Security Issue**
  - File system offers lesser security
- **Rollback Issue**
  - File system doesn’ t have a crach recovery mechanism
- **Query Issue**
  - There is no efficient query processing

SQL vs NoSQL

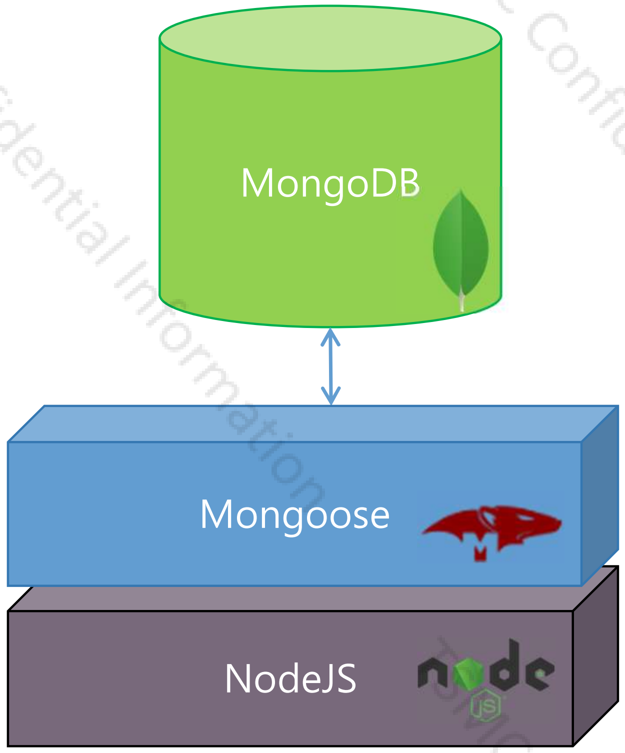
SQL	NoSQL
Relational Database	Non-relational, Distributed Database
Vertically Scalable	Horizontally Scalable
Table based Database	Document based, Key-Value Pair
Pre-Defined Schema	Dynamic Schema
Not Suitable for Hierarchical Data Storage	Best Suitable for Hierarchical Data Storage
Can be used for Complex Queries	Not Good for Complex Queries

id	name	password
1	Ricky	abc123
2	Kevin	x4y5z6

```
[
  {
    id: 1,
    user: 'Ricky',
    password: 'abc123',
  },
  {
    id: 2,
    user: 'Kevin',
    password: 'x4y5z6',
  },
]
```

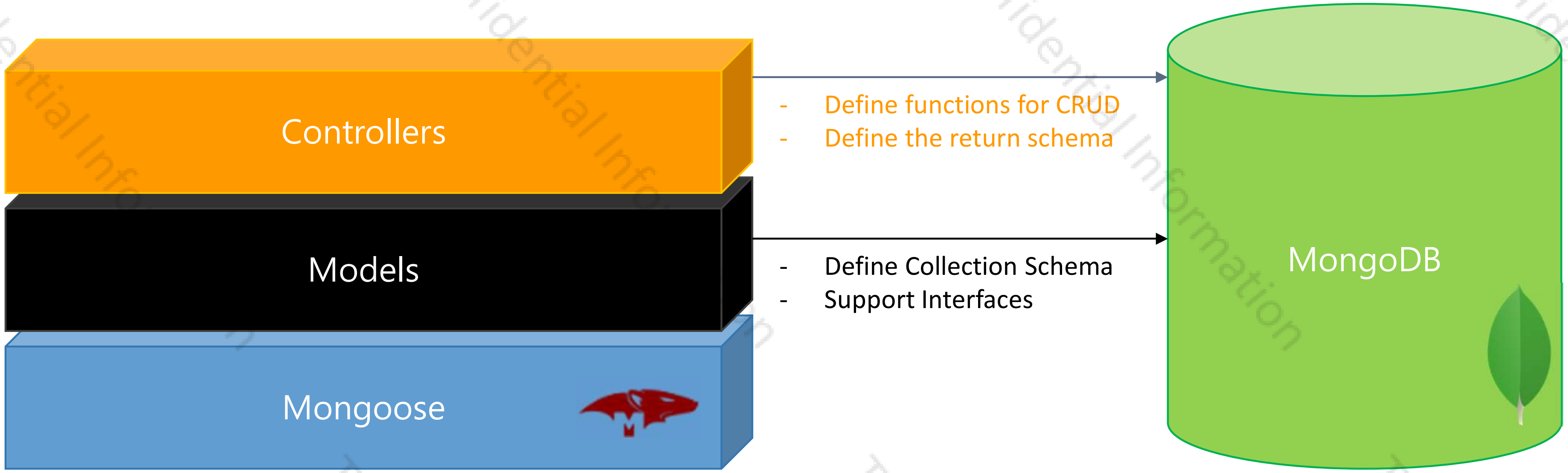

CREATE
Create
new item
DELETE
item
item
item
UPDATE
READ
item 4
Update

Mongoose as Object Data Model



- ODM (Object Data Model)
- High level API interface
- Javascript

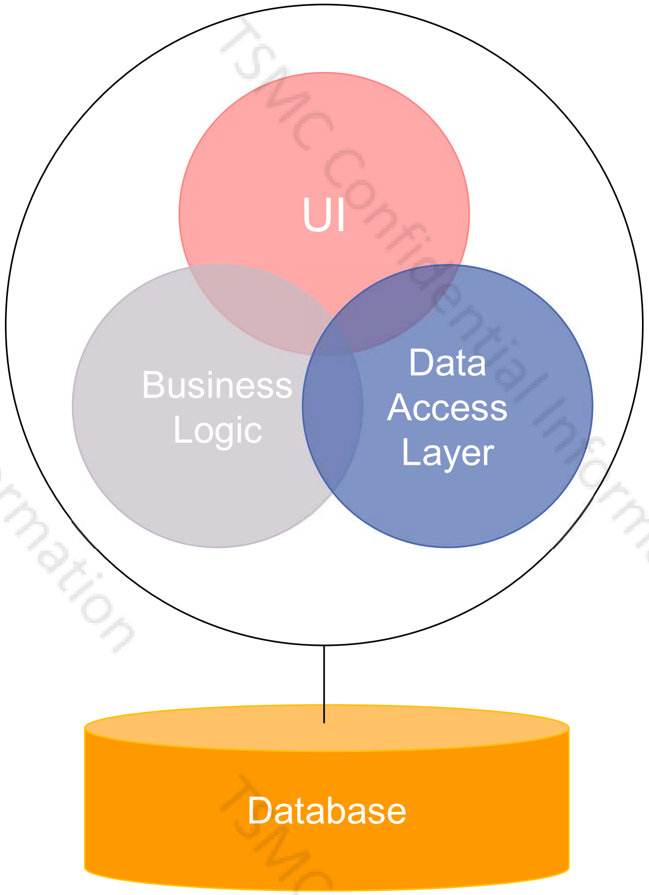
Mongoose CRUD Hierarchy


Controllers
Models
Mongoose
MongoDB
- Define functions for CRUD
- Define the return schema
- Define Collection Schema
- Support Interfaces

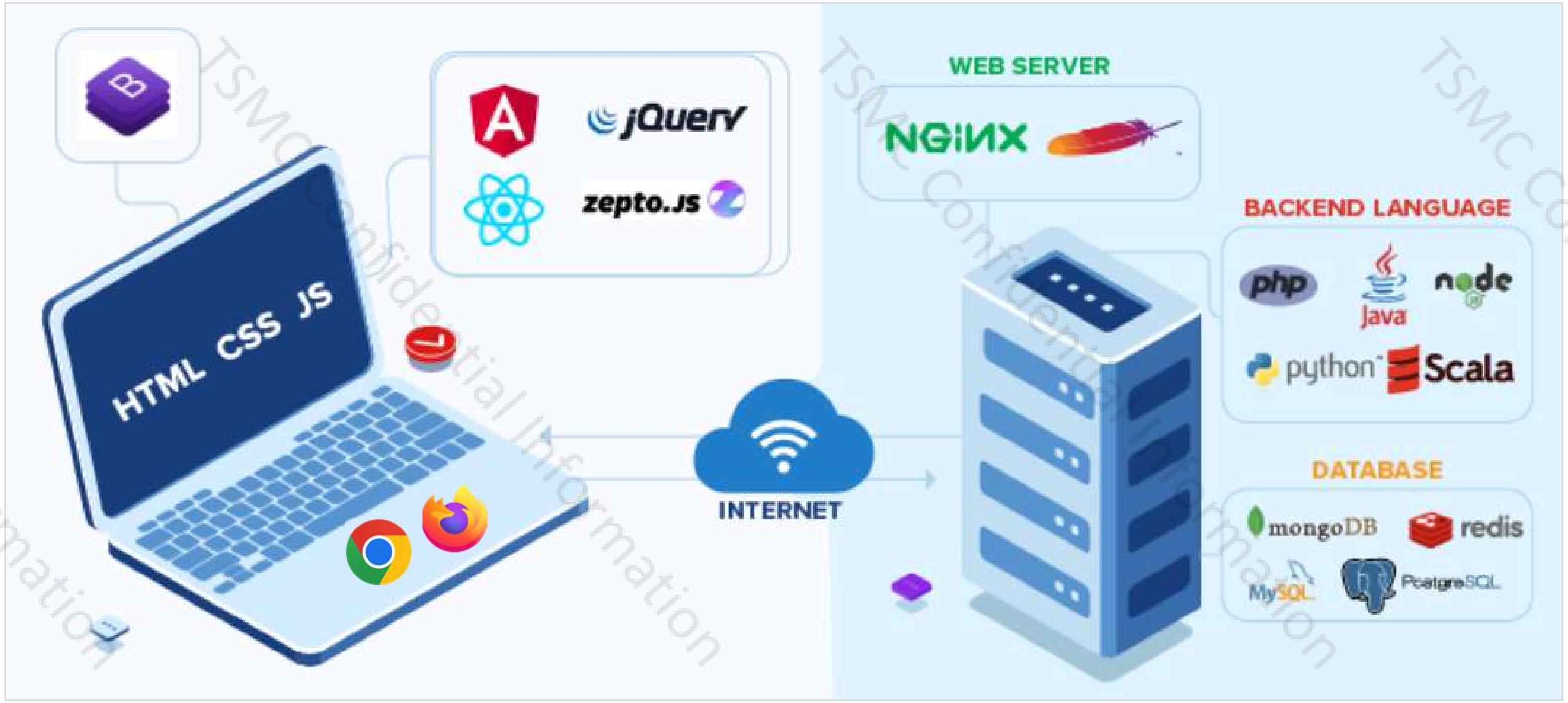
Exercise

```
node .\exercise_2\crud.js

node .\exercise_2\index.js
```

Problem…



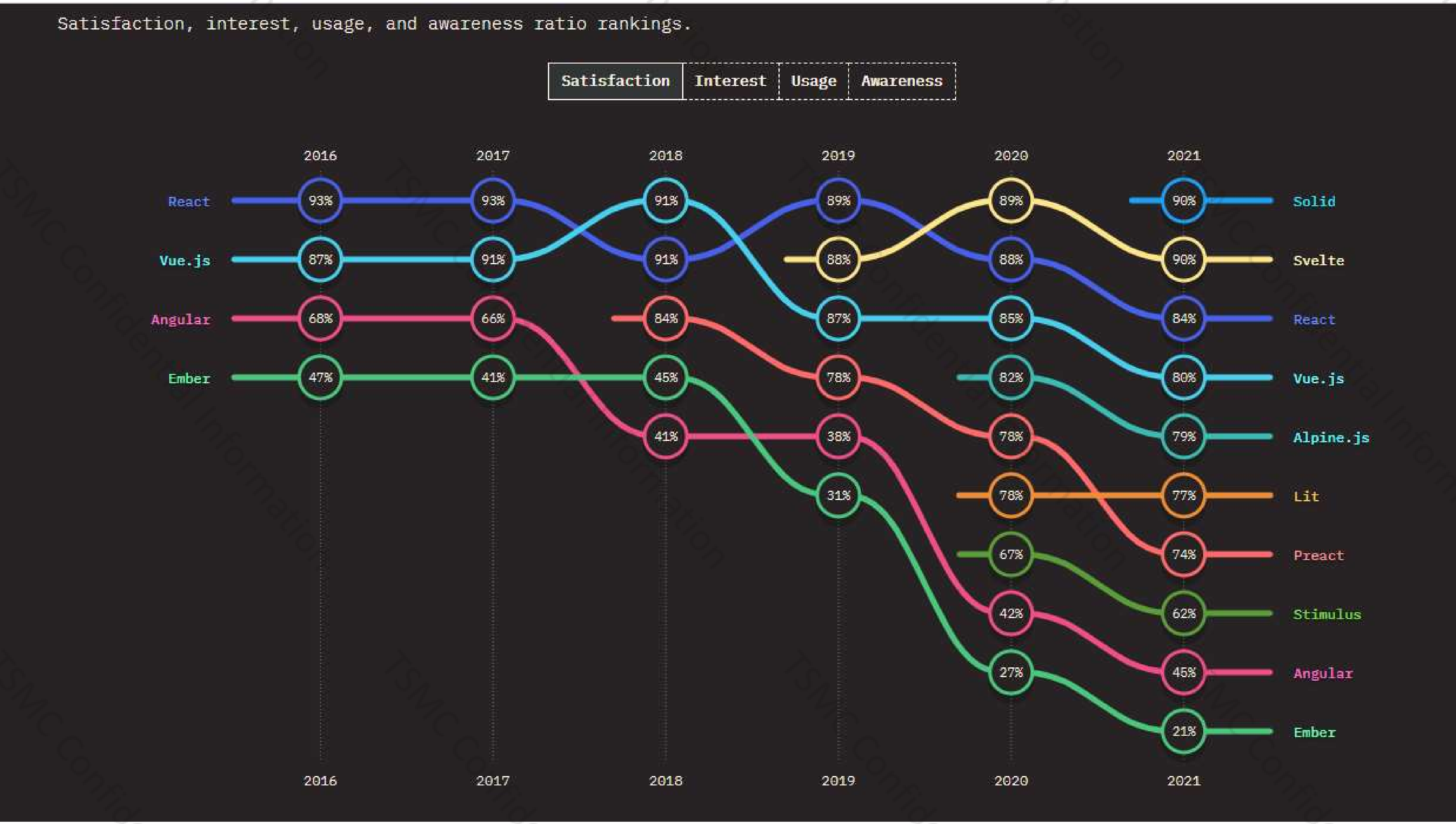
- **Scalability**
  - Back-end ramped up faster as growing
- **Resource Optimization**
  - Server do all the work (request parsing , database fetching, html generating)
- **Upgradation**
  - Whole service need to be upgraded
- **Switch Frameworks**
  - Enabled new technology (React…etc)?
- **Deployment**
  - Need to wait other features for integration



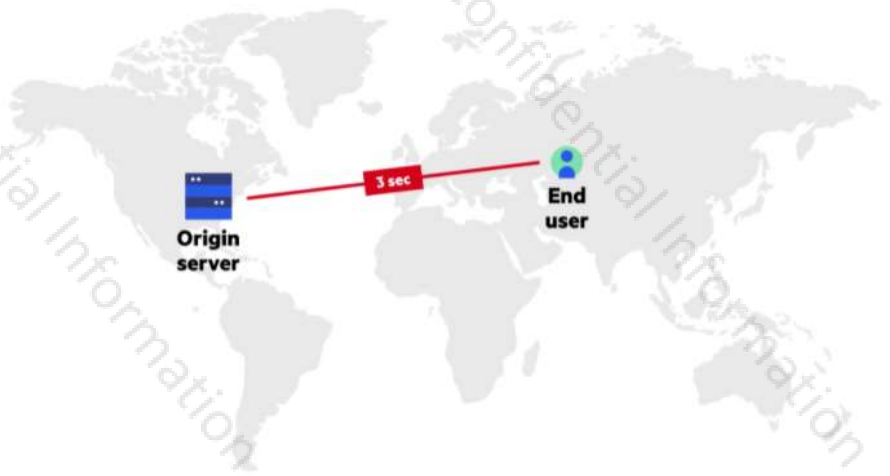
Frontend

Backend

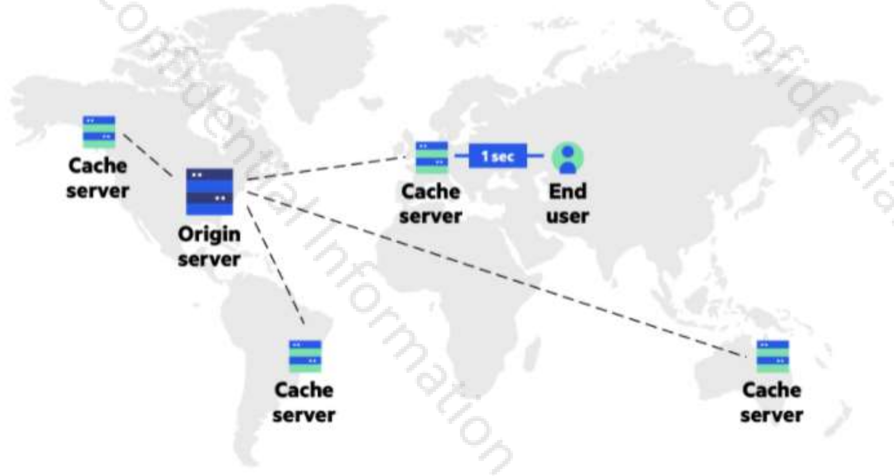
Satisfaction, interest, usage, and awareness ratio rankings.

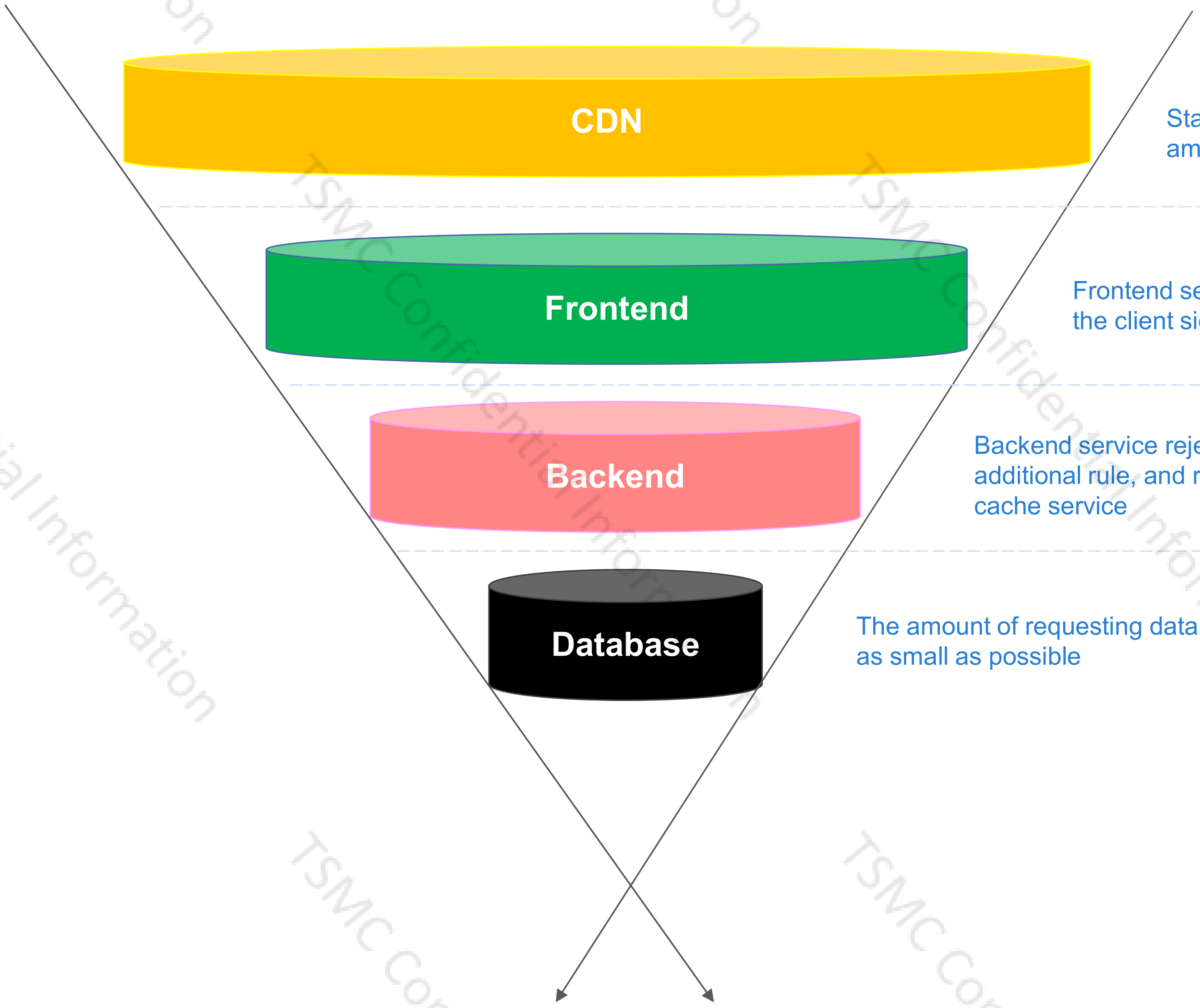


Without CDN



With CDN



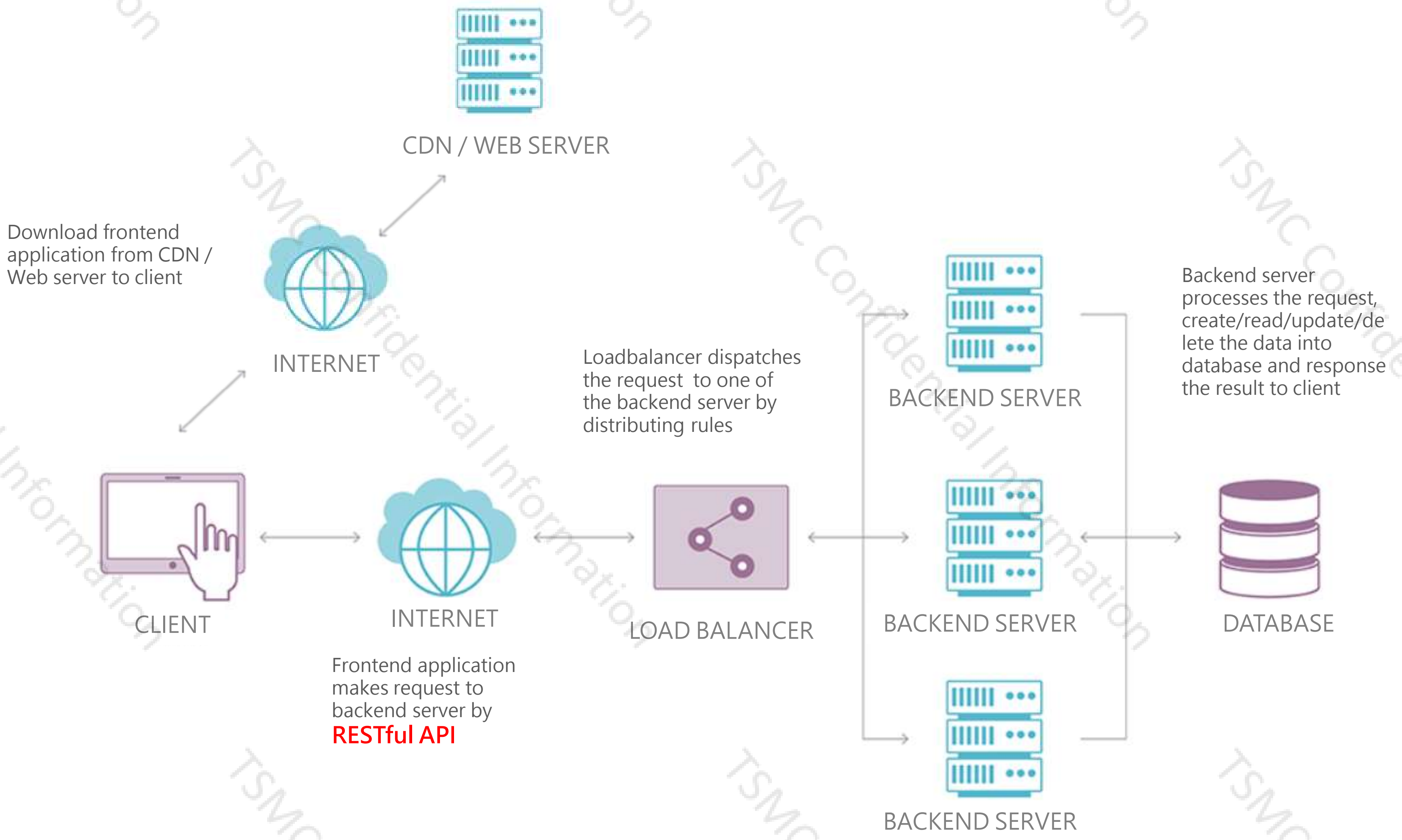


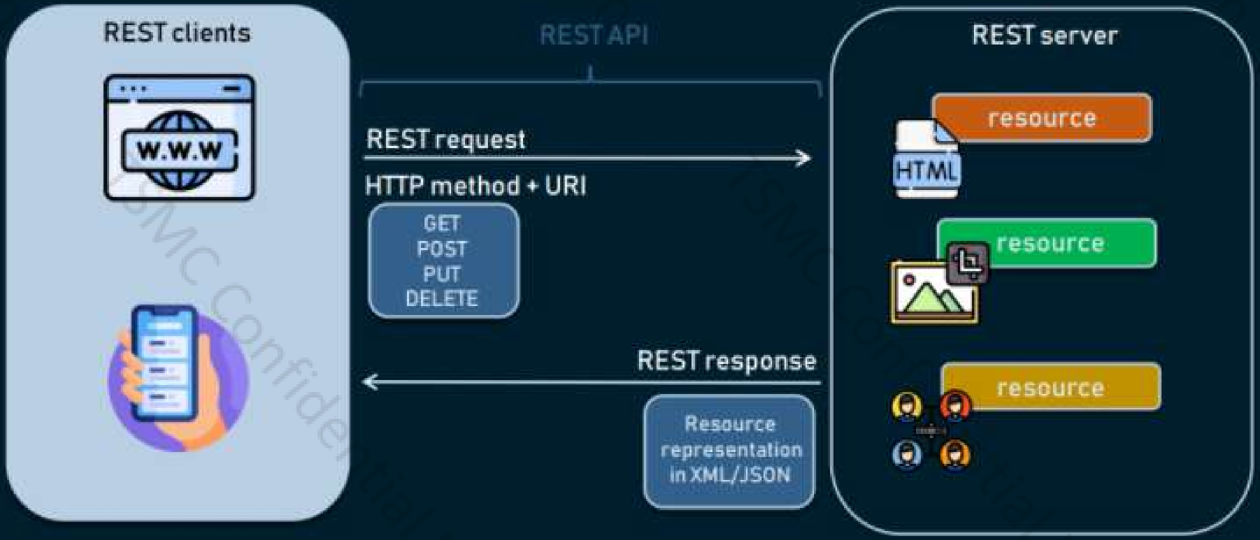
Static data is stored in CDN to decrease the amount of request in frontend server

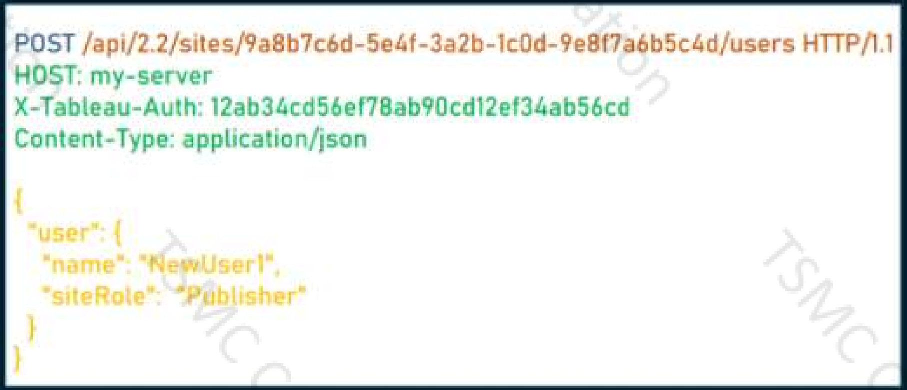
Frontend service reject the illegal request in the client side

Backend service reject the illegal request additional rule, and respond some result from cache service

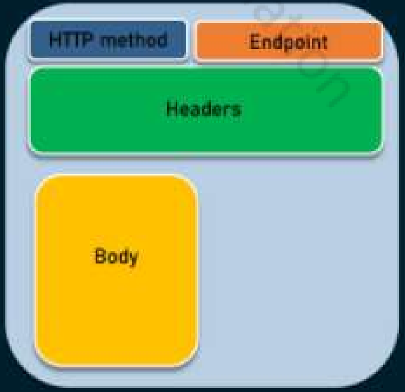
The amount of requesting data is maintained as small as possible


CDN / WEB SERVER
Download frontend application from CDN / Web server to client
INTERNET
CLIENT
INTERNET
Frontend application makes request to backend server by RESTful API
Loadbalancer dispatches the request to one of the backend server by distributing rules
LOAD BALANCER
BACKEND SERVER
BACKEND SERVER
BACKEND SERVER
Backend server processes the request, create/read/update/delete the data into database and response the result to client
DATABASE





```
POST /api/2.2/sites/9a8b7c6d-5e4f-3a2b-1c0d-9e8f7a6b5c4d/users HTTP/1.1
HOST: my-server
X-Tableau-Auth: 12ab34cd56ef78ab90cd12ef34ab56cd
Content-Type: application/json

{
  "user": {
    "name": "NewUser1",
    "siteRole": "Publisher"
  }
}
```



- Accept and Respond with JSON
- Use Nouns Instead of Verbs in Endpoint Paths
- Use Logical Nesting on Endpoints
- Handle Errors Gracefully and Return Standard Error Codes
- Allow Filtering, Sorting, and Pagination
- Maintain Good Security Practices
- Versioning our APIs
- Document our APIs

Accept and Respond with JSON

```
{
  "username": "jhchao",
  "password": "abc123"
}
```

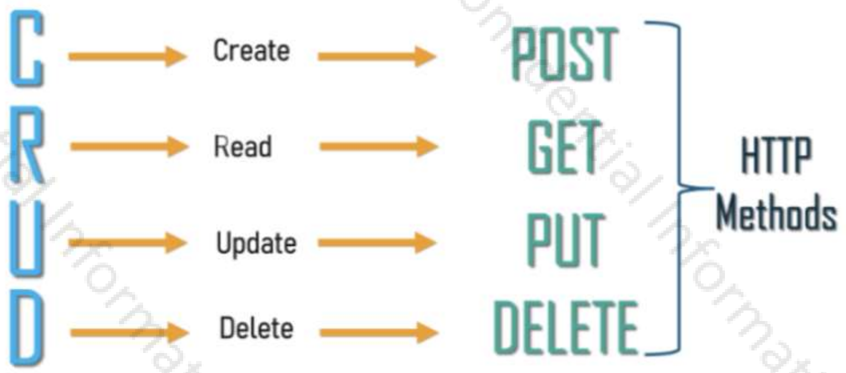
Request

```
{
  "ok": true,
  "message": "login success",
  "id": "1"
}
```

Response

- **JSON** is the standard for transferring data, almost every networked technology can use it
  - Content-Type: application/json

Use Nouns Instead of Verbs in Endpoint Paths



- The action should be indicated by the HTTP request method
  - **GET** retrieves resources
  - **POST** submits new data to the server
  - **PUT** updates existing data
  - **DELETE** removes data
  - It maps **CRUD** operations

Use Logical Nesting on Endpoints

```
app.get('/orders/:orderId/delivers', (req, res) => {
  const { orderId } = req.params;
  const delivers = [];

  // code to get delivers by orderId
  // ...
  //

  res.json(delivers);
});
```

- Endpoints can be designed with grouping that contain associated information

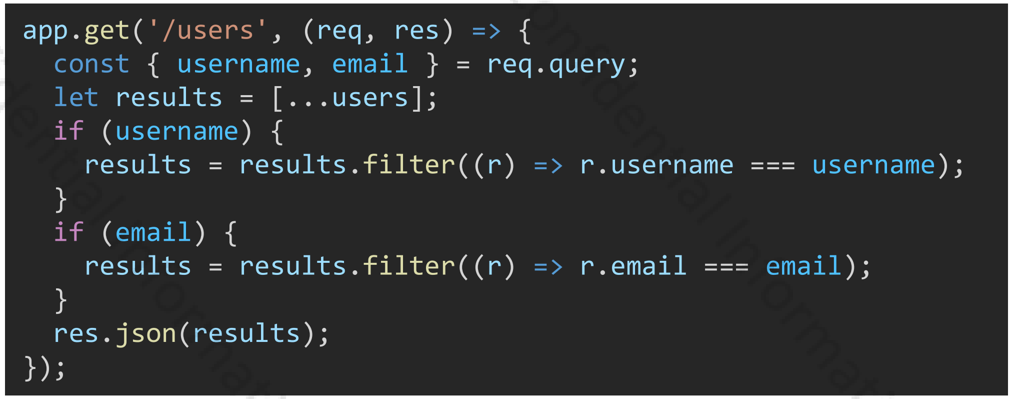
Handle Errors Gracefully and Return Standard Error Codes

```
app.post('/users', (req, res) => {
  const { email } = req.body;
  const userExists = users.find((u) => u.email === email);
  if (userExists) {
    return res.status(400).json({
      error: 'User already exists’
    });
  }
  res.json(req.body);
});
```

- Common error HTTP status code:
  - 400 Bad Request
  - 401 Unauthorized
  - 403 Forbidden
  - 404 Not Found
  - 500 Internal server error
  - 502 Bad Gateway
  - 503 Service Unav

- Provide error information
  - Message
  - Sub Error Code
  - Reference help links

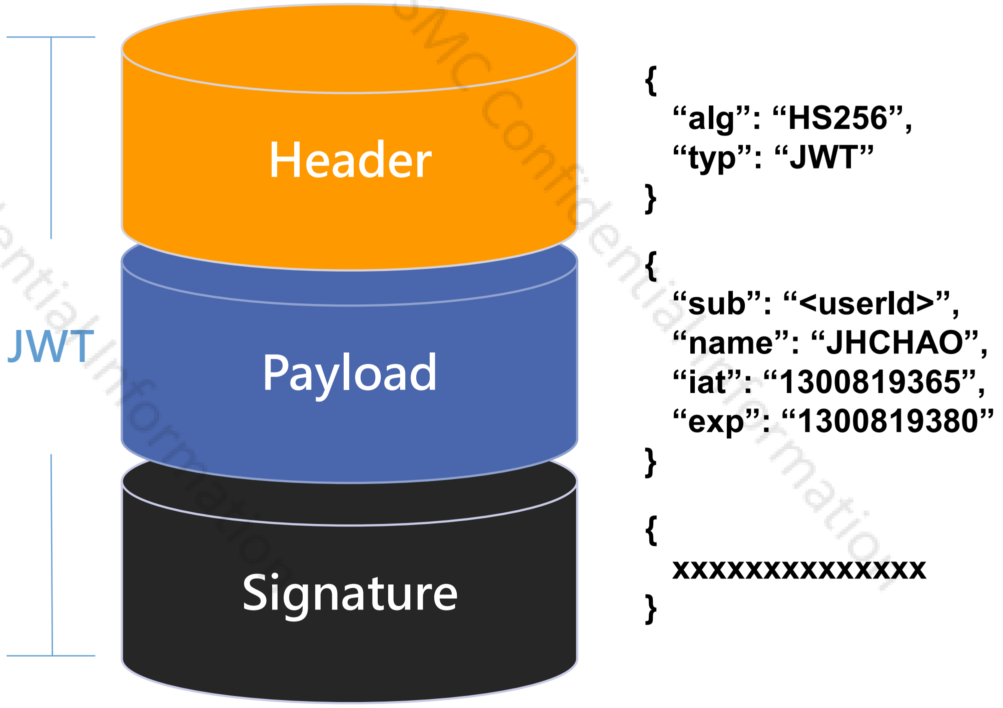
Allow Filtering, Sorting, and Pagination



```
app.get('/users', (req, res) => {
  const { username, email } = req.query;
  let results = [...users];
  if (username) {
    results = results.filter((r) => r.username === username);
  }
  if (email) {
    results = results.filter((r) => r.email === email);
  }
  res.json(results);
});
```

- Filtering and pagination both increase performance by reducing the usage of server resources
  - Decrease the overloading of transferring data

Maintain Good Security Practices(1)

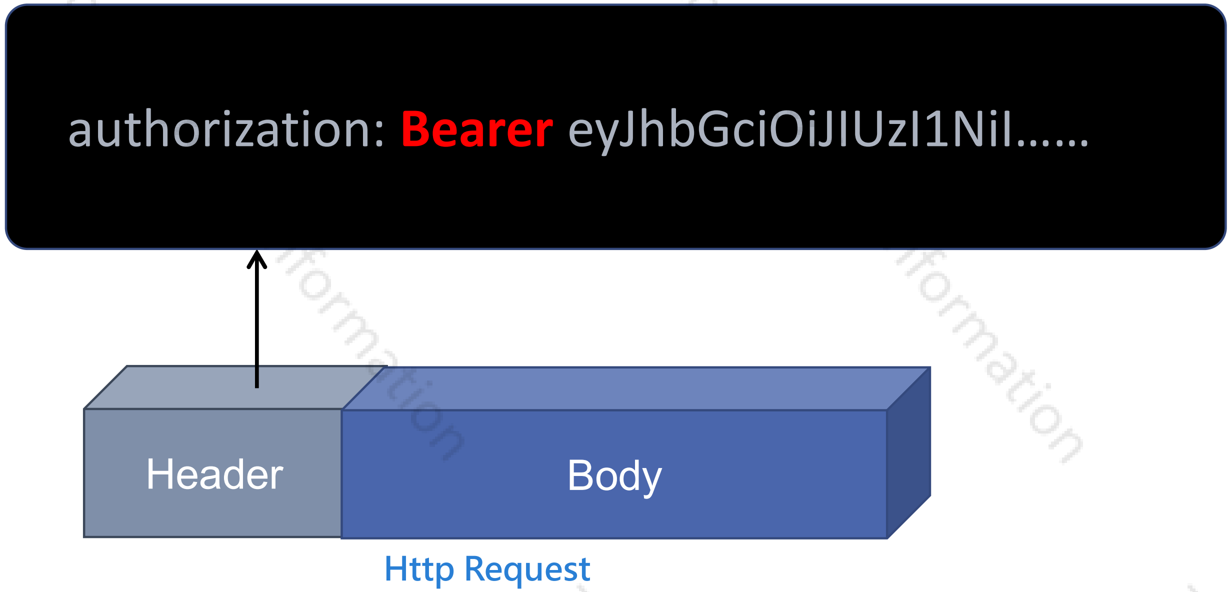


```
{
  "alg": "HS256",
  "typ": "JWT"
}
{
  "sub": "<userId>",
  "name": "JHCHAO",
  "iat": "1300819365",
  "exp": "1300819380"
}
{
  xxxxxxxxxxxxxx
}
```

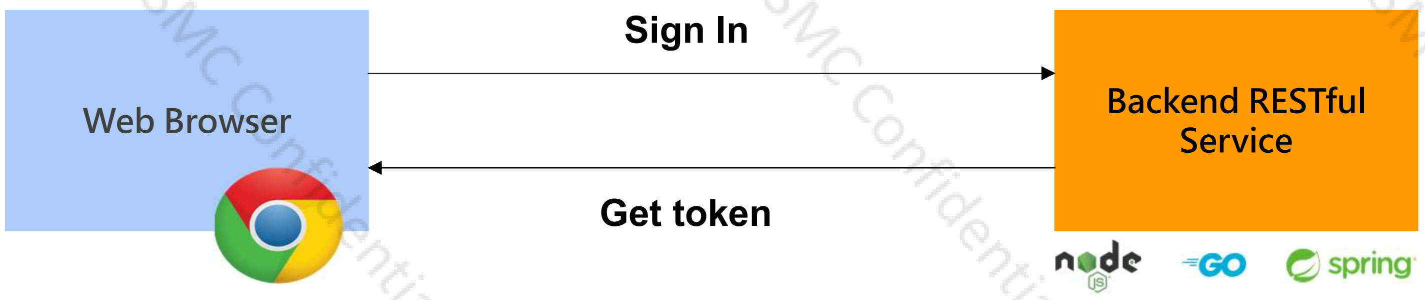
base64(Header).base64(Payload).base64(Signature)

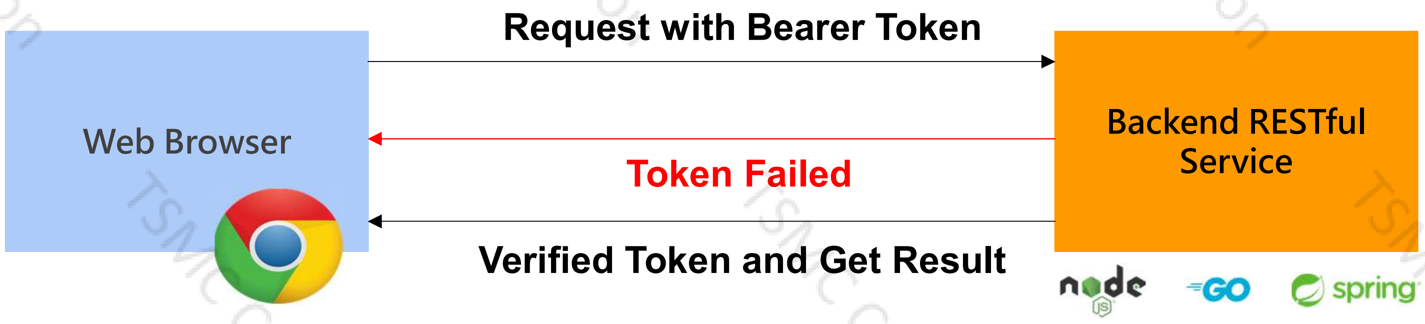
- Using HTTPS(SSL/TLS) for security is a must
- Leverage API key or token for authentication and authorization
  - JWT
  - API Key

Maintain Good Security Practices(2)



Maintain Good Security Practices(3)



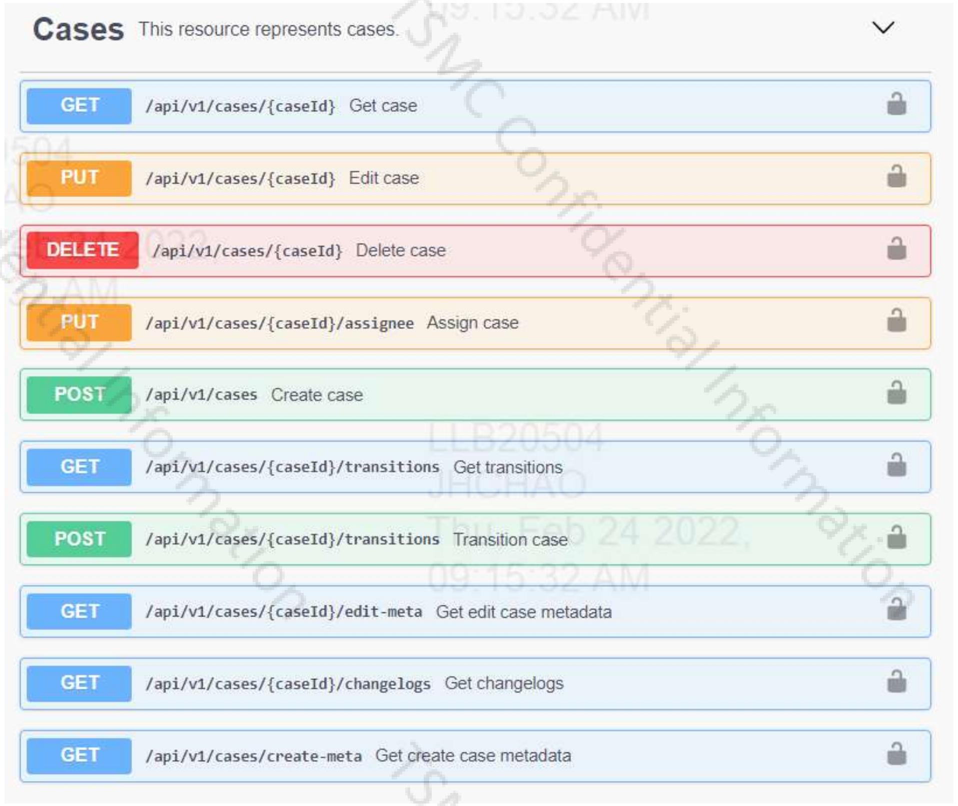


Versioning our APIs

```
app.get('api/v1/orders', (req, res) => {
  const orders = [];
  // code to get orders
  // …
  //

  res.json(orders);
});

app.get('api/v2/orders', (req, res) => {
  const orders = [];
  // different code to get orders
  // …
  //

  res.json(orders);
});
```

- APIs only need to be up-versioned when a breaking change is made
  - a change in the format of the response data for one or more calls
  - a change in the request or response type (i.e. changing an integer to a float)
  - removing any part of the API.
- Versioning is usually done with `/v1/`, `/v2/`, …etc
  - we won't break third party apps that use our APIs

Document our APIs

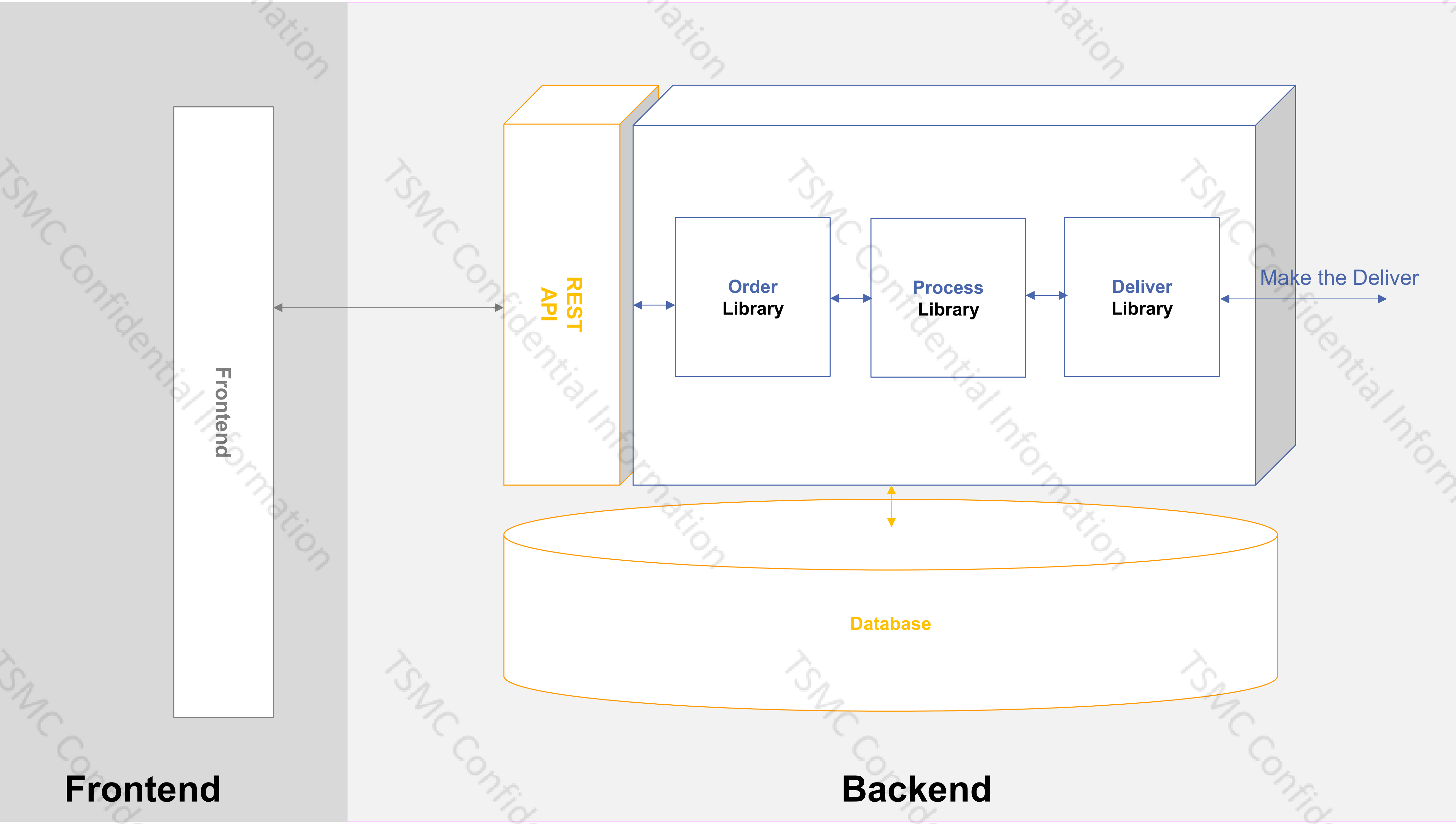


Cases This resource represents cases.

GET	`/api/v1/cases/{caseId}`	Get case
PUT	`/api/v1/cases/{caseId}`	Edit case
DELETE	`/api/v1/cases/{caseId}`	Delete case
PUT	`/api/v1/cases/{caseId}/assignee`	Assign case
POST	`/api/v1/cases`	Create case
GET	`/api/v1/cases/{caseId}/transitions`	Get transitions
POST	`/api/v1/cases/{caseId}/transitions`	Transition case
GET	`/api/v1/cases/{caseId}/edit-meta`	Get edit case metadata
GET	`/api/v1/cases/{caseId}/changelogs`	Get changelogs
GET	`/api/v1/cases/create-meta`	Get create case metadata

https://swagger.io/docs/specification/about

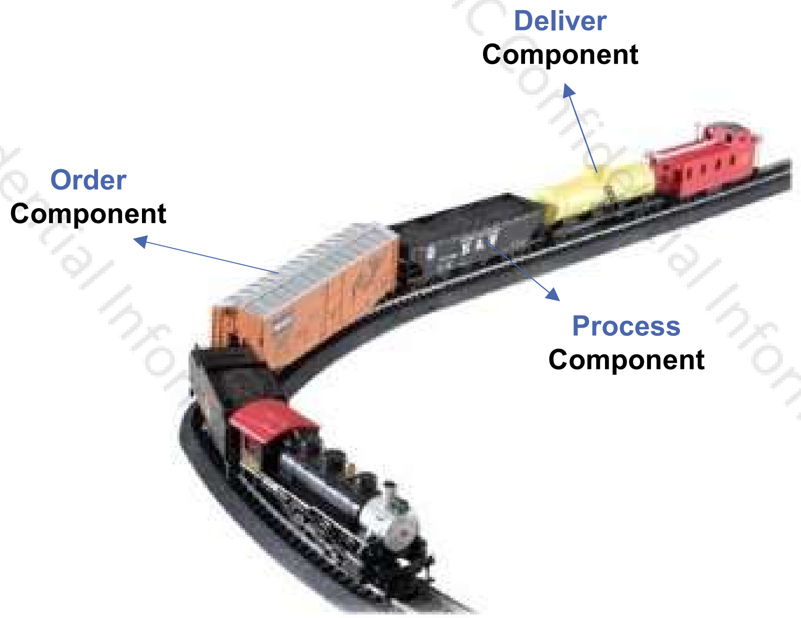
- Provide the documents for the usage of APIs
  - Along with the API service as the document endpoint
  - Swagger


Frontend
REST API
Order Library
Process Library
Deliver Library
Make the Deliver
Database
Frontend
Backend

Exercise

```
node .\exercise_3\service.js

node .\exercise_3\index.js

Node .\exercise_3\jwt.js
```

Problem…

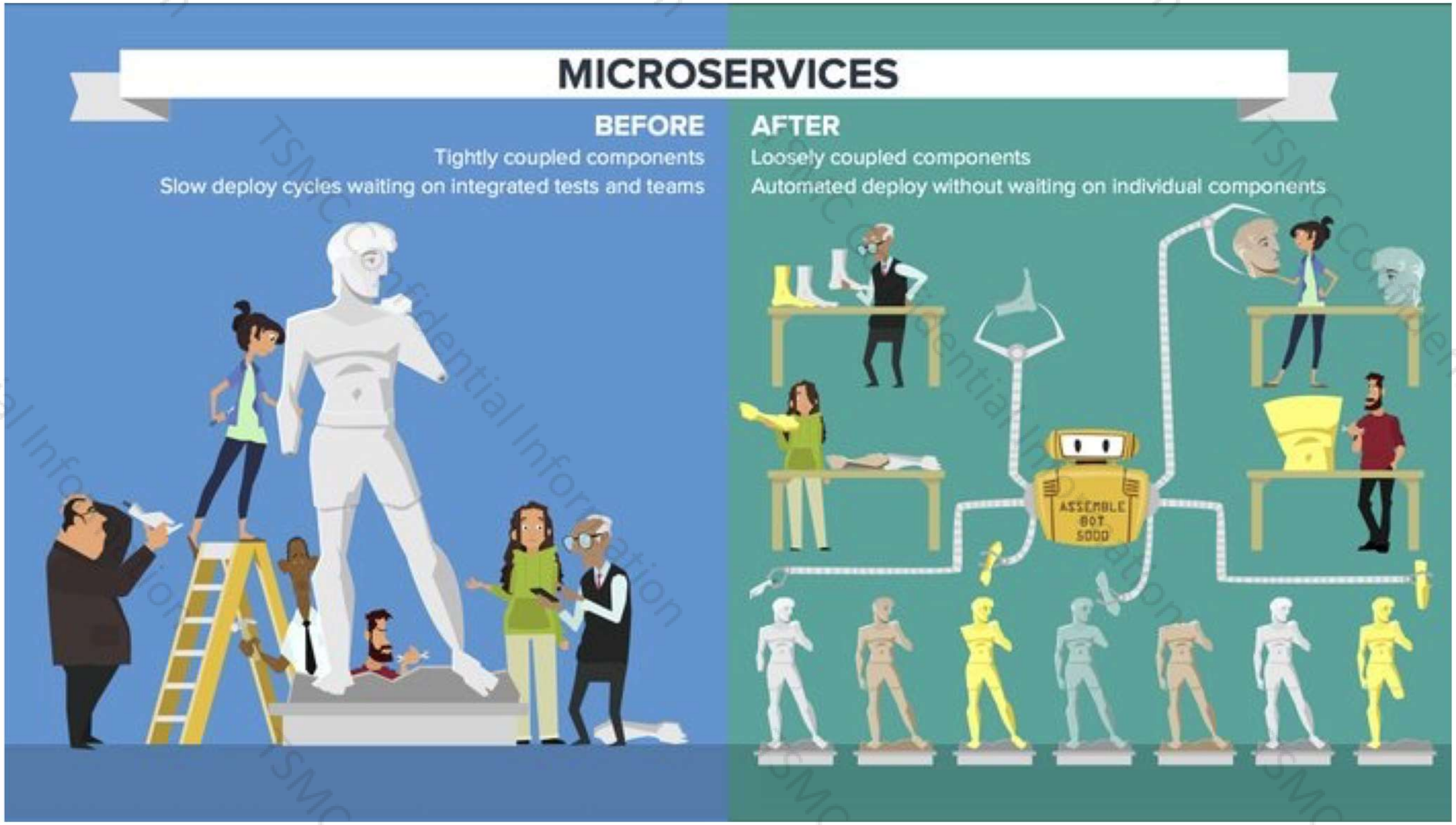


- **Large Codebase**
  - All the module of codes are located in a huge single repository
- **No Clear Ownership**
  - Team member sometimes needs to develop features on 2-3 modules in the same time
- **Long Deployment Cycles**
  - The deployment needs to wait the completion of all features for some specific version
- **Scaling with Undifferentiated**
  - The scaling size is on whole application, not on the specific services

Deliver
Component

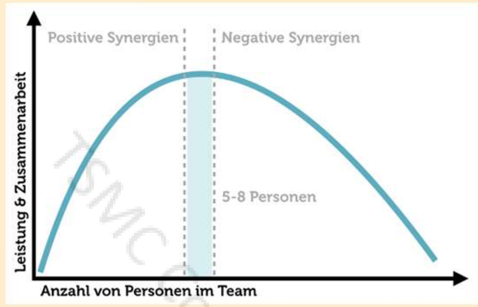
Order
Component

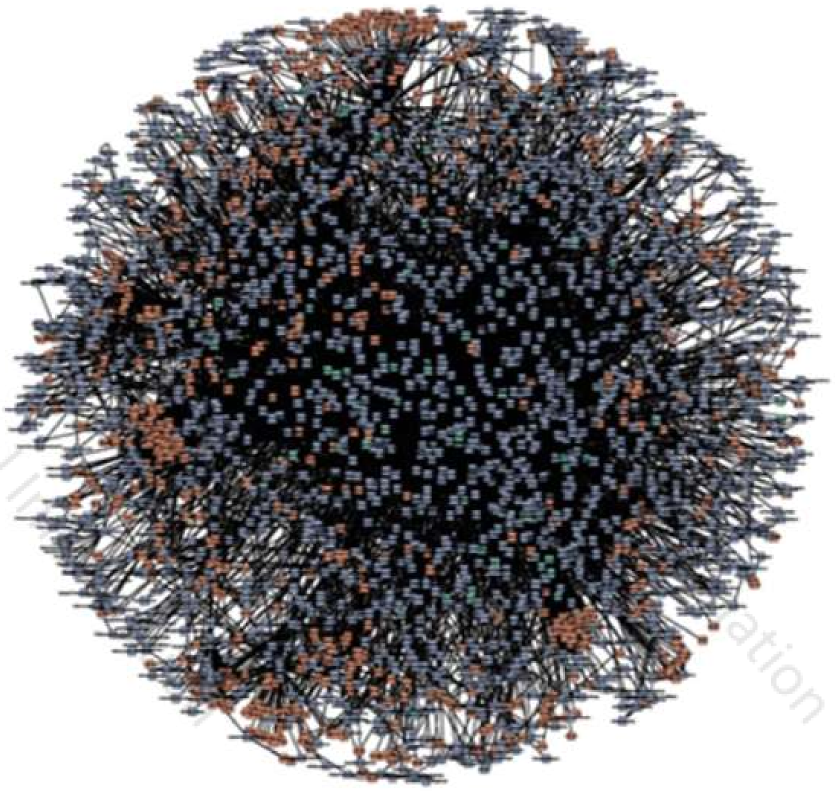
Process
Component


MICROSERVICES
BEFORE
Tightly coupled components
Slow deploy cycles waiting on integrated tests and teams
AFTER
Loosely coupled components
Automated deploy without waiting on individual components
ASSEMBLE BOT 5000

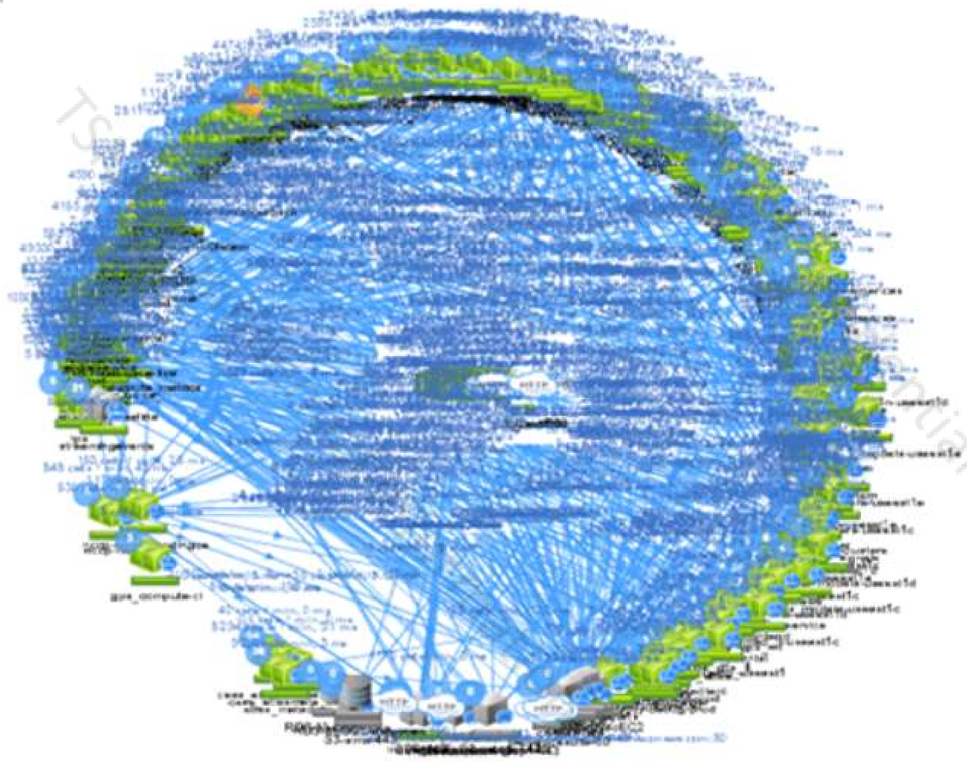
amazon

Two Pizza Rule = Better Productivity

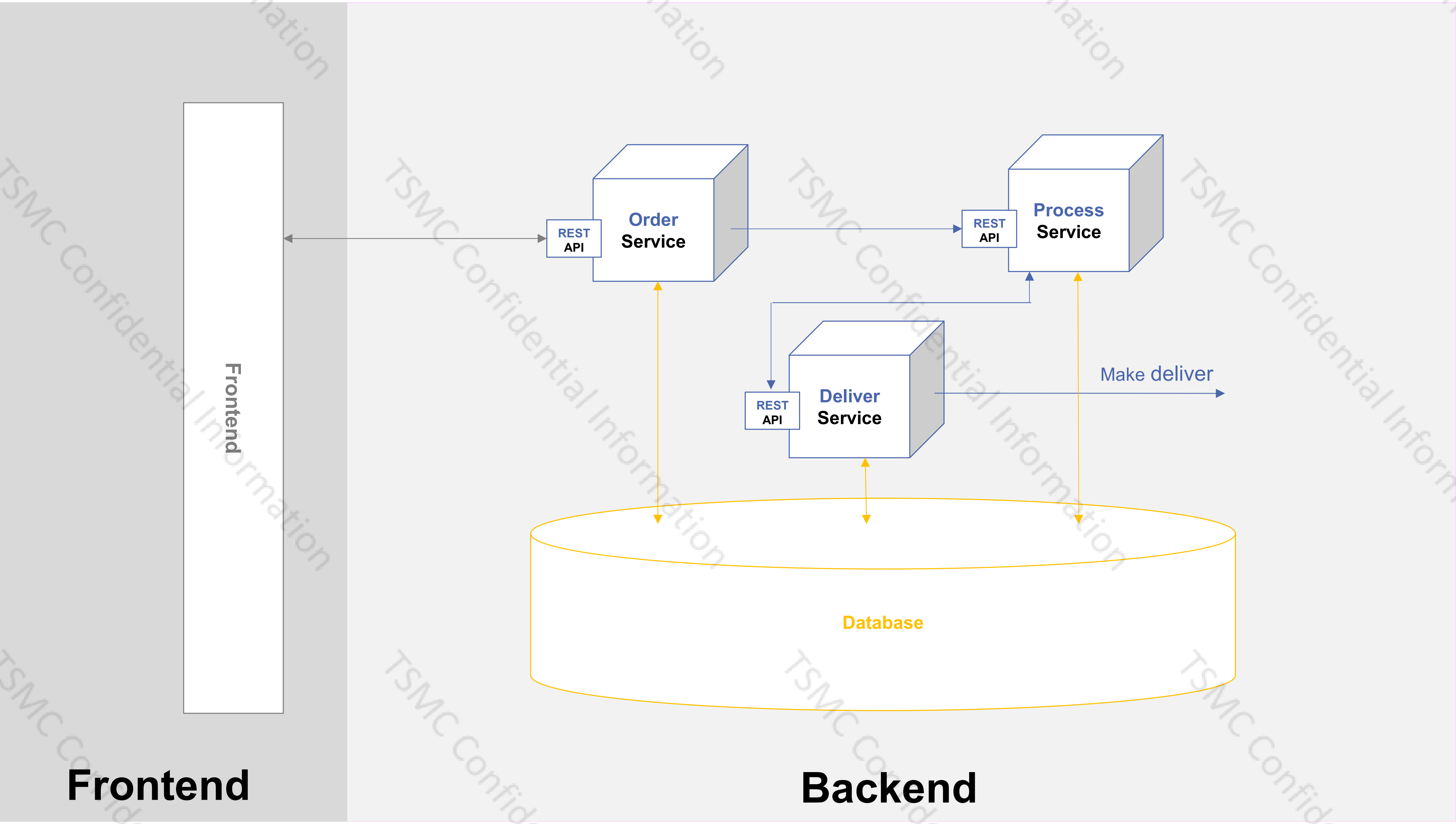




amazon.com



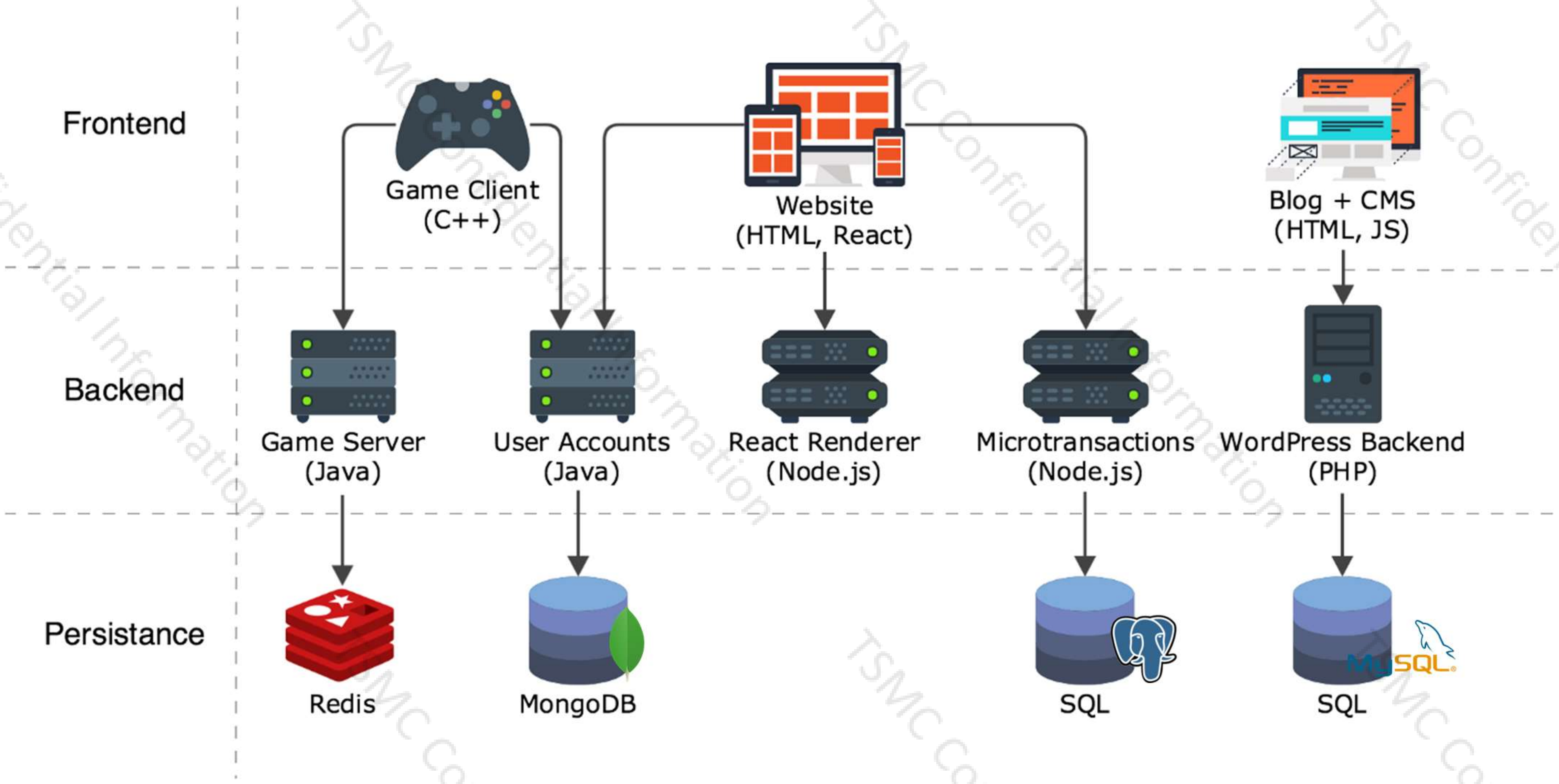
NETFLIX

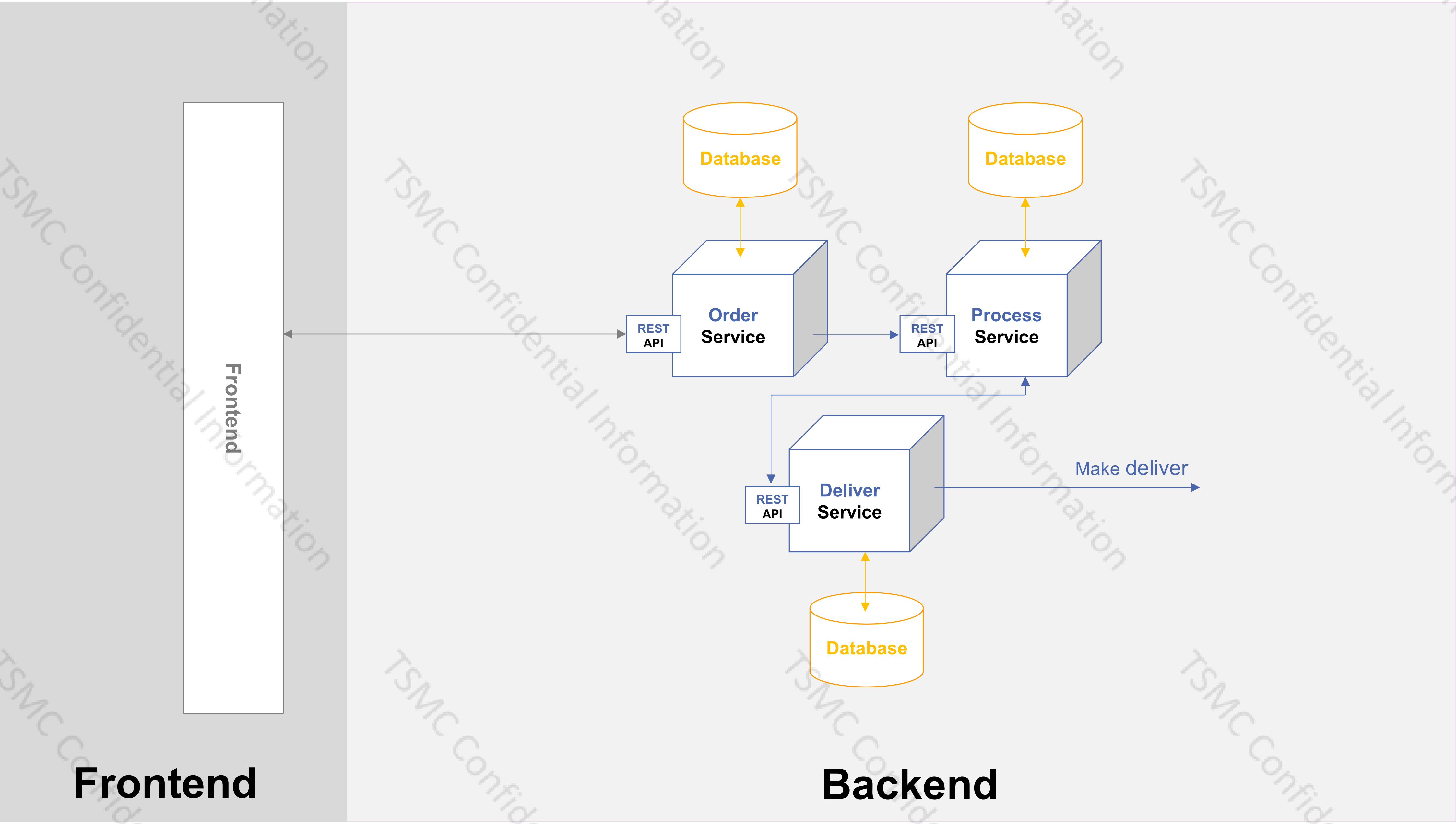

Frontend
REST API
Order Service
REST API
Process Service
REST API
Deliver Service
Make deliver
Database
Frontend
Backend

Exercise

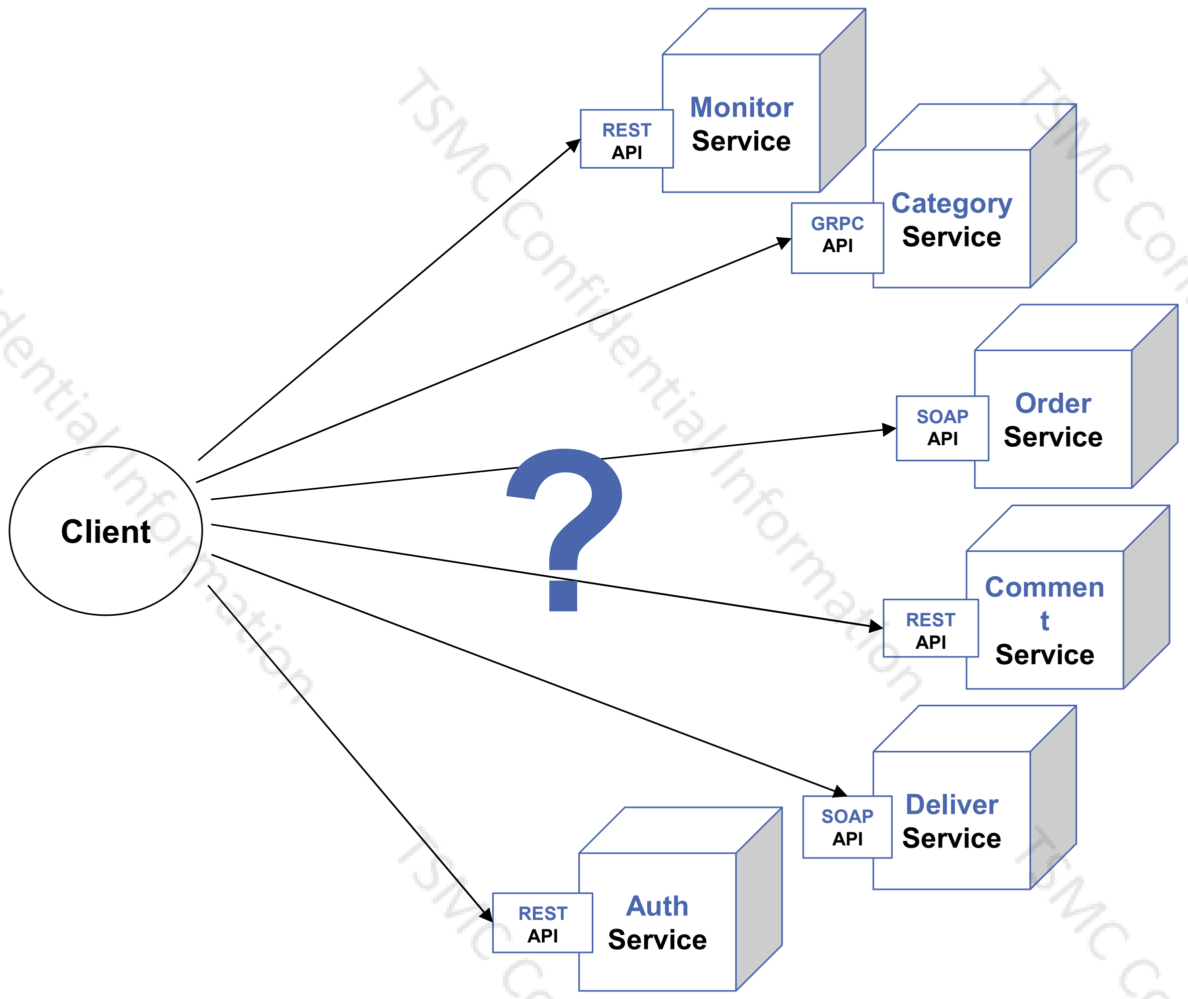
```
node .\exercise_4\service.js

node .\exercise_4\index.js
```

Decentralized Governance

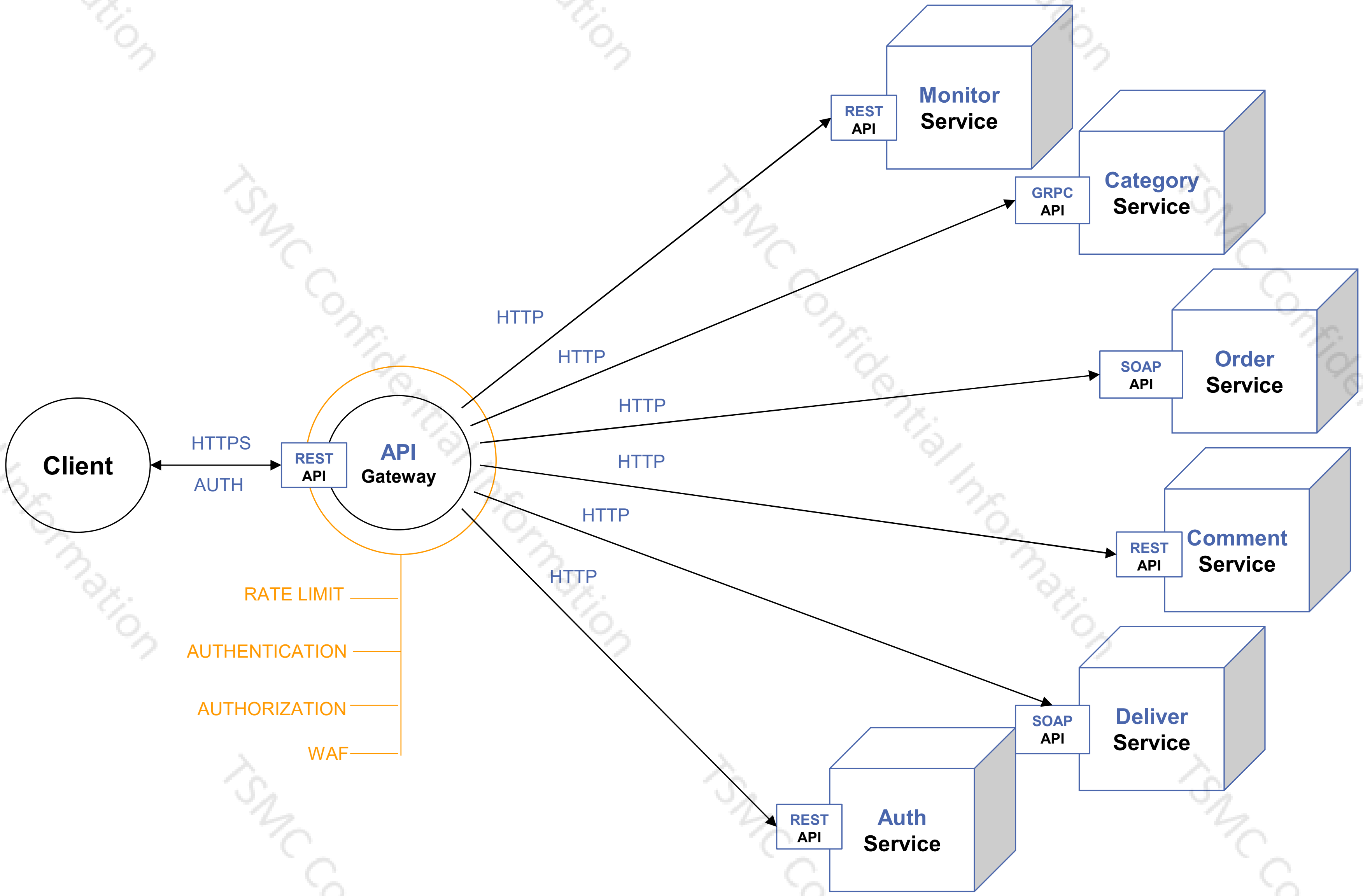



Frontend
Database
Database
Order Service
REST API
Process Service
REST API
Deliver Service
REST API
Make deliver
Database
Frontend
Backend

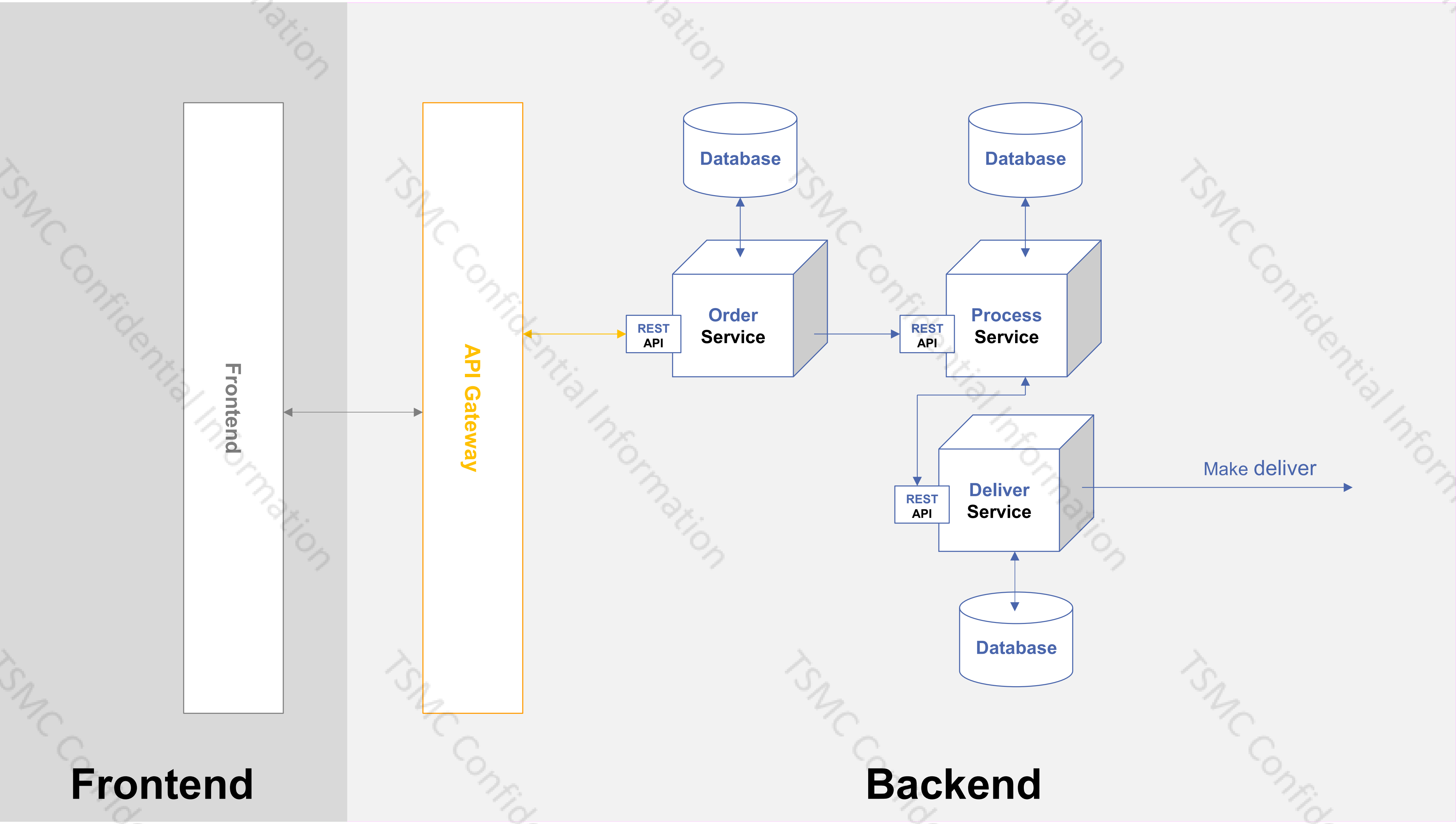
Problem…



- **Decouple**
  - Client needs to config all endpoints of required services
- **Security**
  - Directly access microservice
- **Complexity**
  - Analytics, WFA, Authorization … etc
- **Protocol**
  - Client needs to access service with different protocols



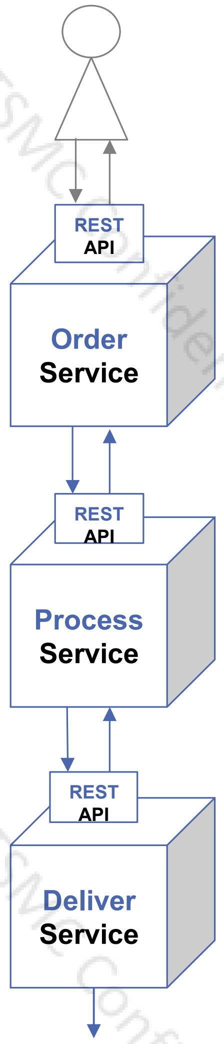
Client
HTTPS
AUTH
REST API
API Gateway
RATE LIMIT
AUTHENTICATION
AUTHORIZATION
WAF
HTTP
HTTP
HTTP
HTTP
HTTP
HTTP
REST API
Monitor Service
GRPC API
Category Service
SOAP API
Order Service
REST API
Comment Service
SOAP API
Deliver Service
REST API
Auth Service


Frontend
API Gateway
Database
Database
Order Service
REST API
Process Service
REST API
Deliver Service
REST API
Make deliver
Database
Frontend
Backend

Exercise

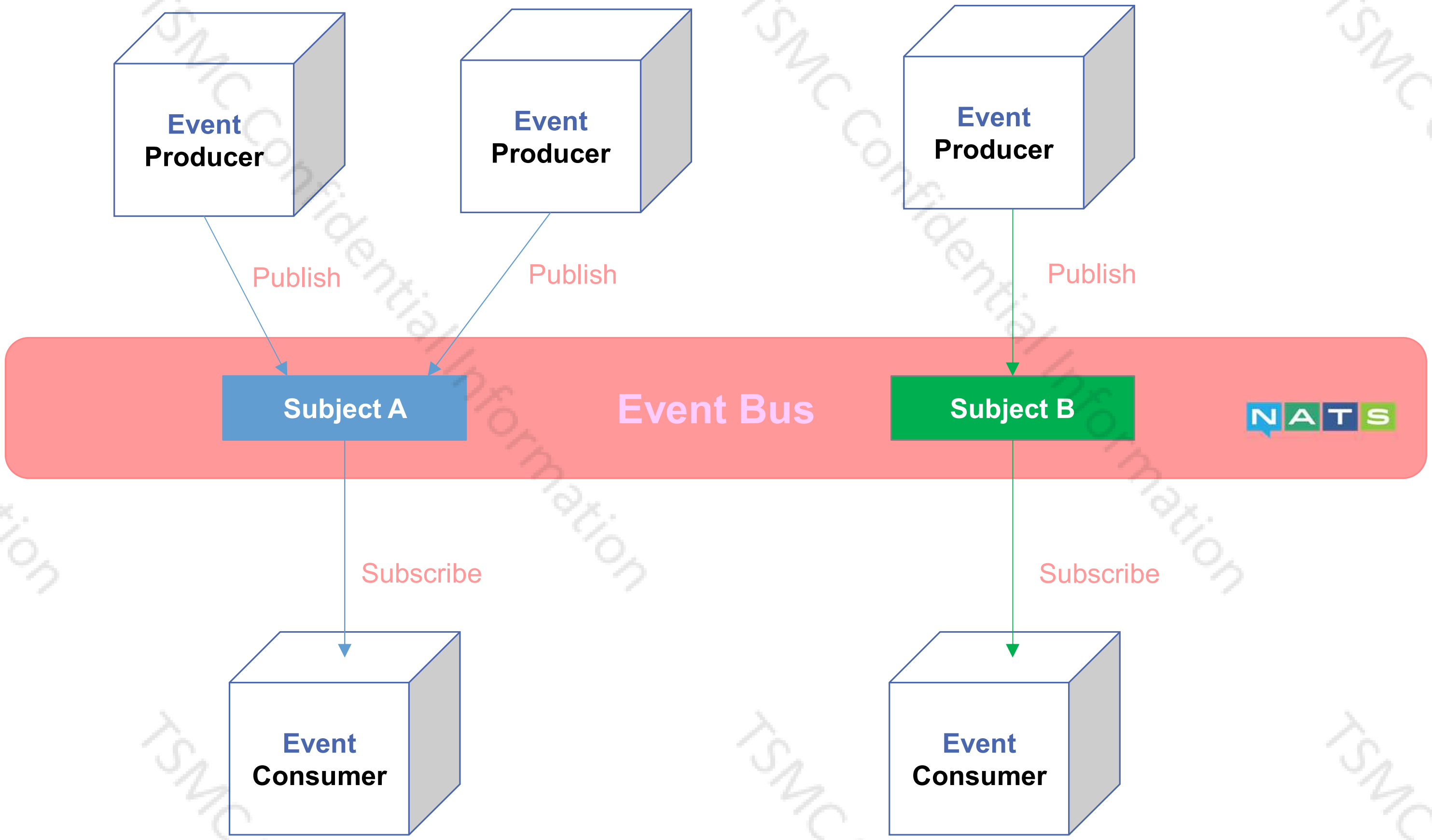
```
node .\exercise_5\service.js

node .\exercise_5\index.js
```

Problem…

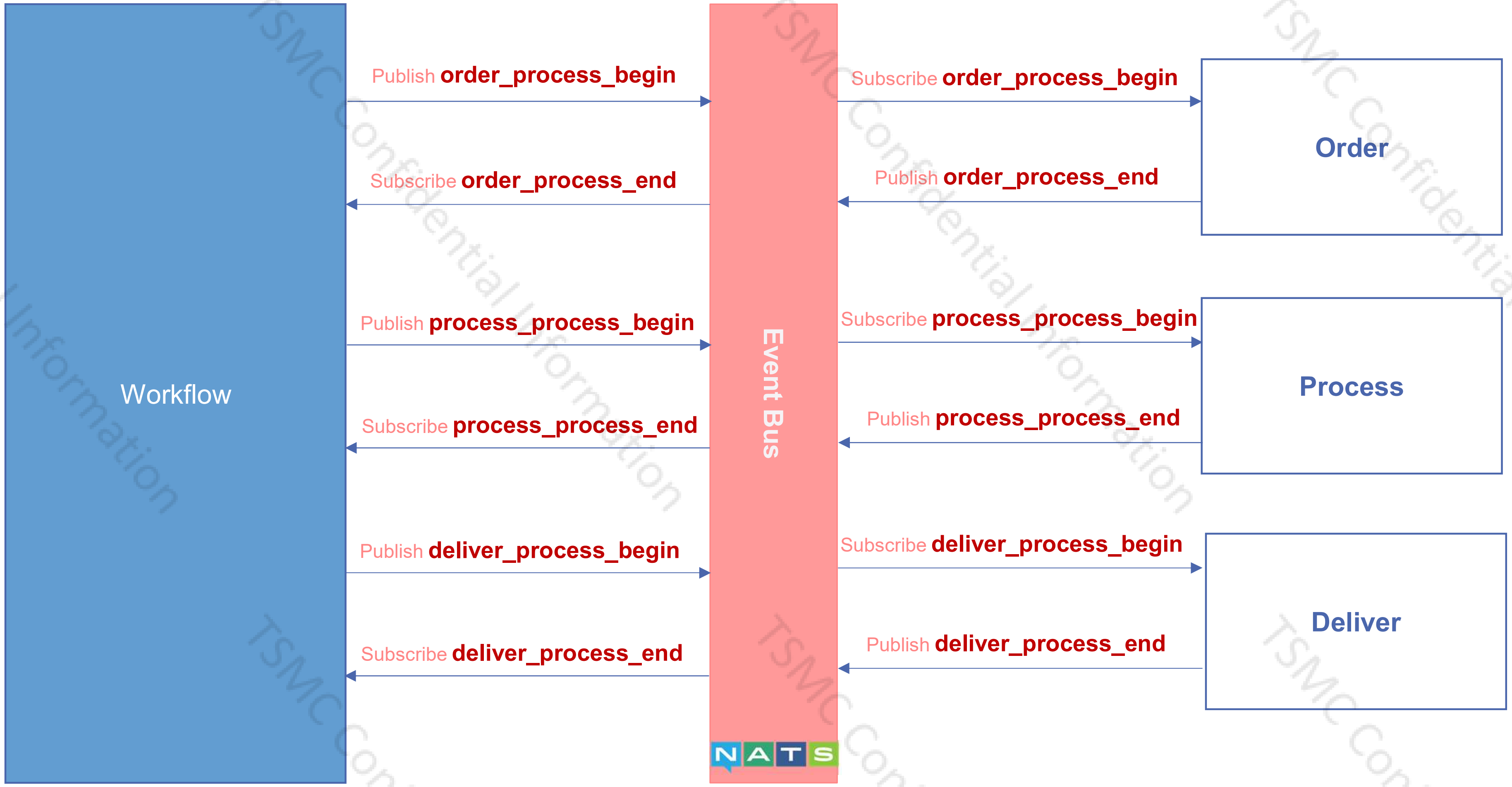


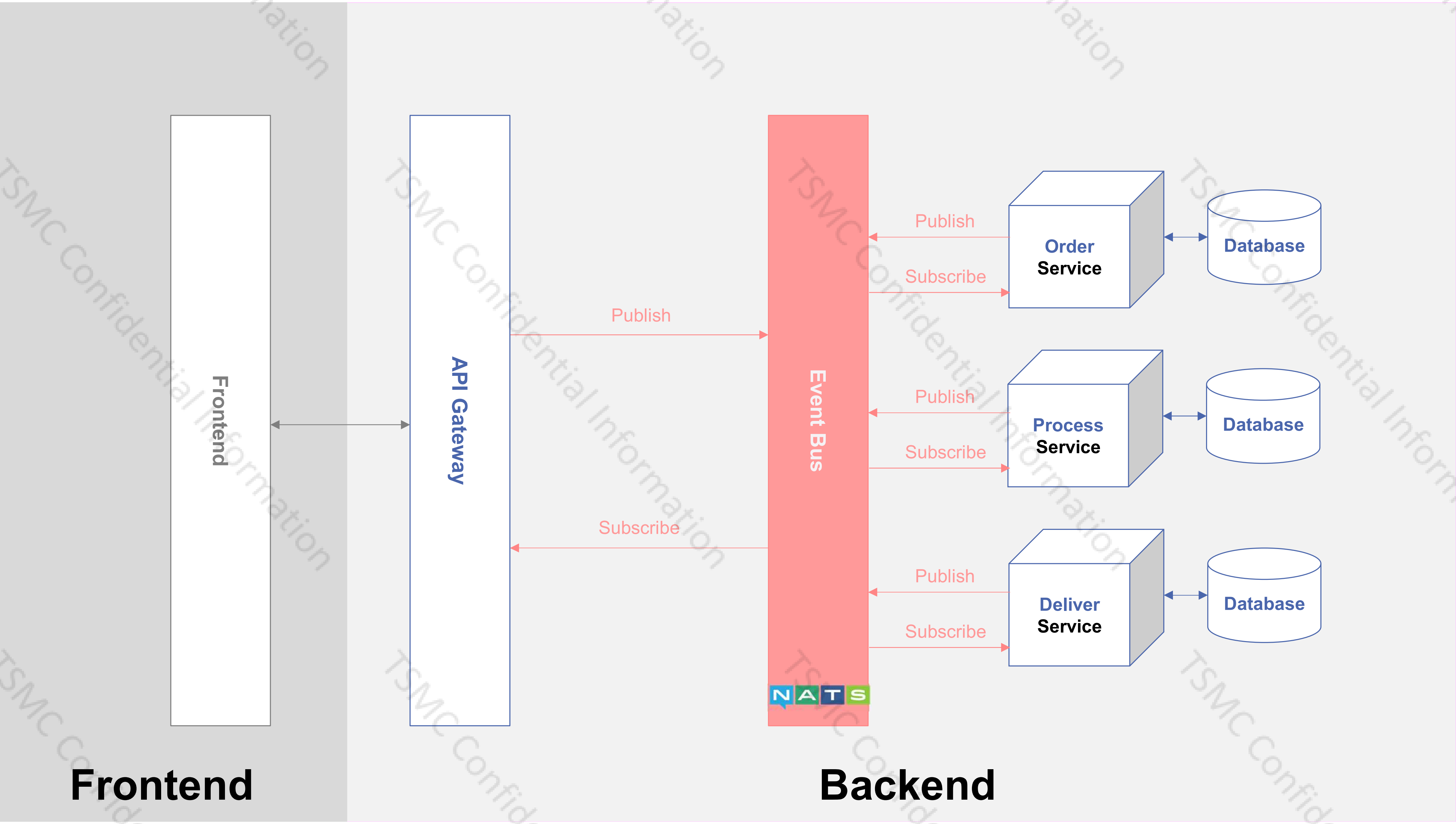
- **Latency**
  - Wait for the completion of whole workflow processing
- **Dependency**
  - Services are coupled by each other
- **Non-Extensibility**
  - Need to figure out the relationships of other services for adding new service
- **Data Serialization**
  - The data of response will be redundantly extended

Event Driven Architecture



SAGA Pattern - Orchestration


Workflow
Publish order_process_begin
Subscribe order_process_end
Publish process_process_begin
Subscribe process_process_end
Publish deliver_process_begin
Subscribe deliver_process_end
Event Bus
NATS
Subscribe order_process_begin
Publish order_process_end
Subscribe process_process_begin
Publish process_process_end
Subscribe deliver_process_begin
Publish deliver_process_end
Order
Process
Deliver


Frontend
API Gateway
Publish
Subscribe
Event Bus
NATS
Publish
Subscribe
Order Service
Database
Publish
Subscribe
Process Service
Database
Publish
Subscribe
Deliver Service
Database
Frontend
Backend

Exercise

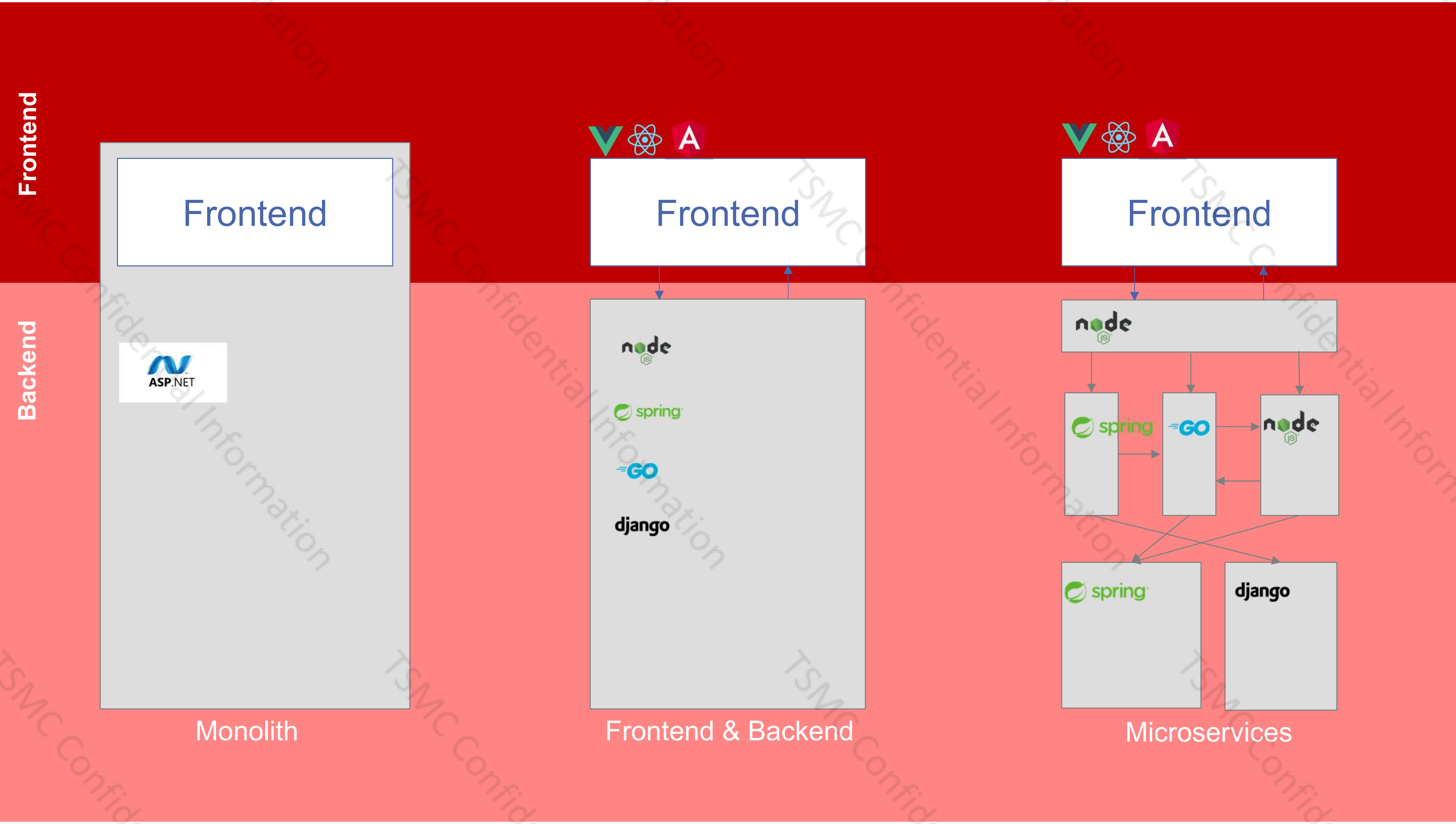
```
node .\exercise_6\service.js

node .\exercise_6\index.js
```

Summary

- Application Revolution
- Data Locality
  - Filesystem to Database
  - SQL & NoSQL
  - CRUD
- Frontend & Backend
  - Language Stack
  - Architecture Design
  - API service design
- Micro Service
  - API Gateway
  - Event Driven Architecture

TSMC IT X NCTU CS
2022 CLOUD NATIVE
Development Best Practice

Backup


Frontend
Backend
Frontend
ASP.NET
Monolith
Frontend
node
spring
GO
django
Frontend & Backend
Frontend
node
spring
GO
node
spring
django
Microservices