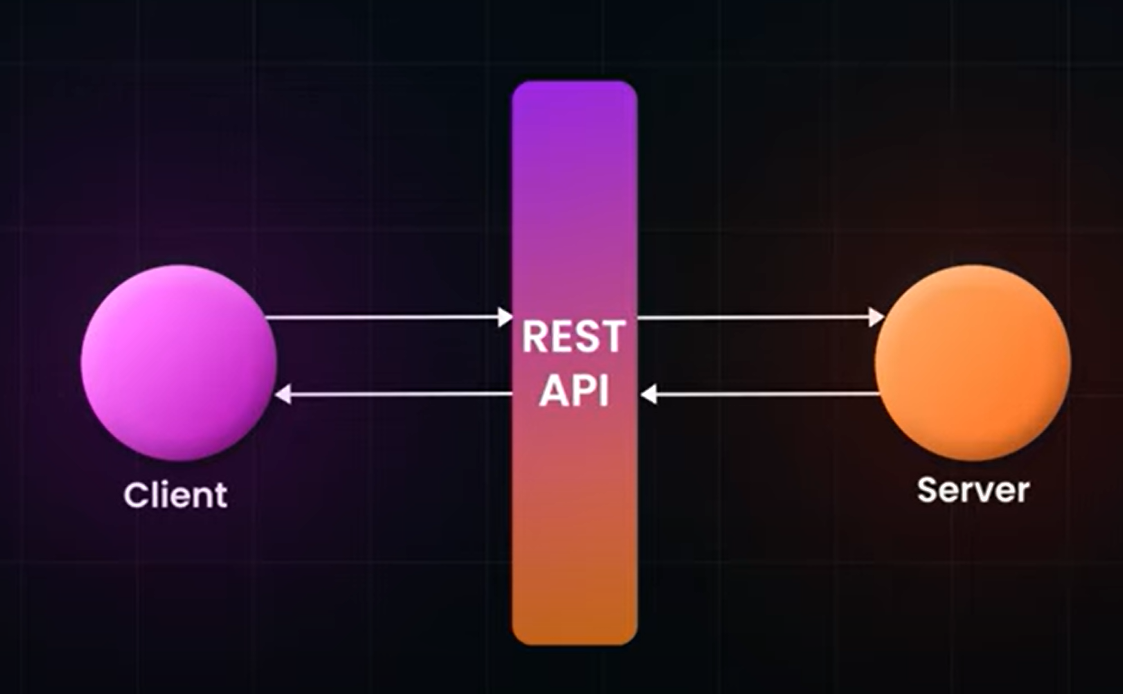
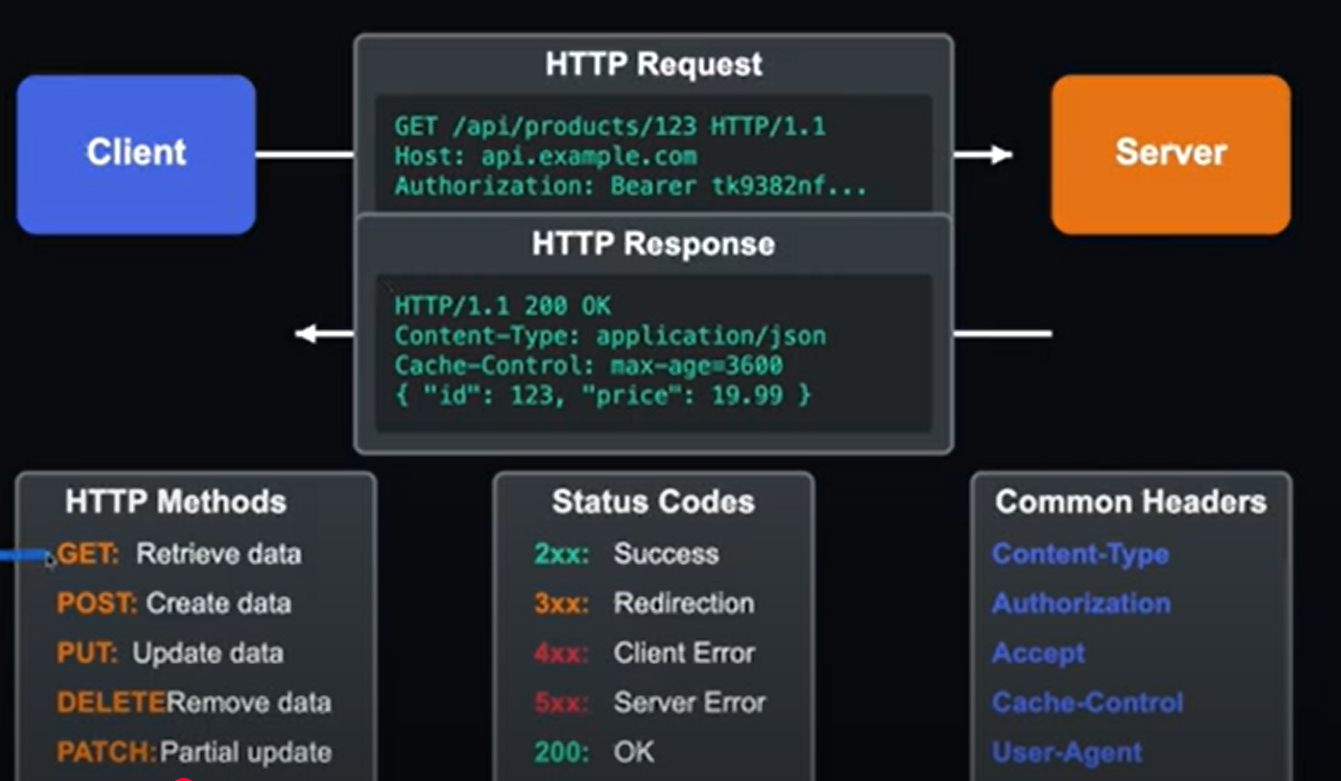
**1. What is REST?**

REST stands for Representational Data Transfer. It’s a set of simple rules how data transfer over the web. It uses HTTP methods like Get, Post, put, delete.

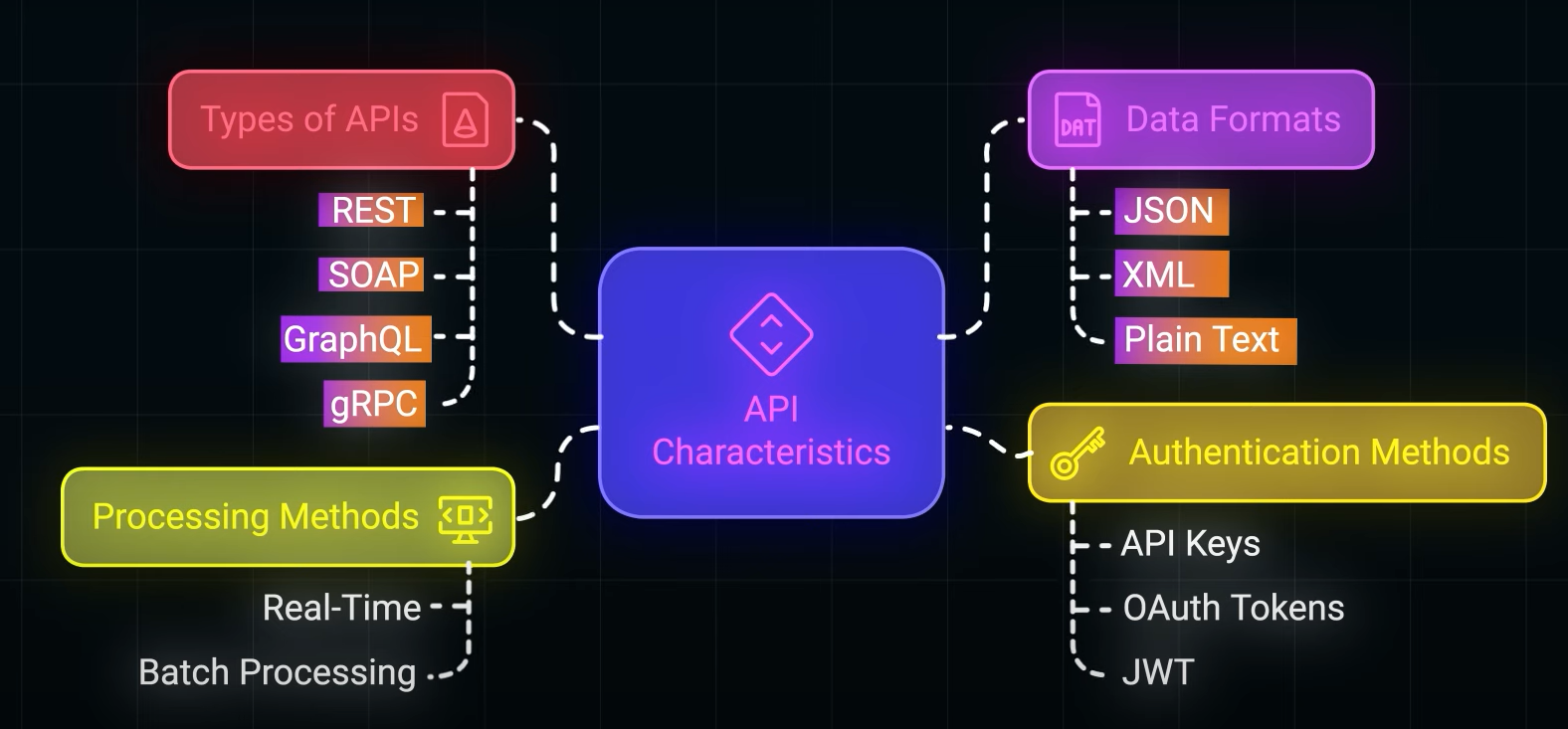
**2. What is REST API?**

Rest API act as a mediator between Client and Server to seamless data transfer.

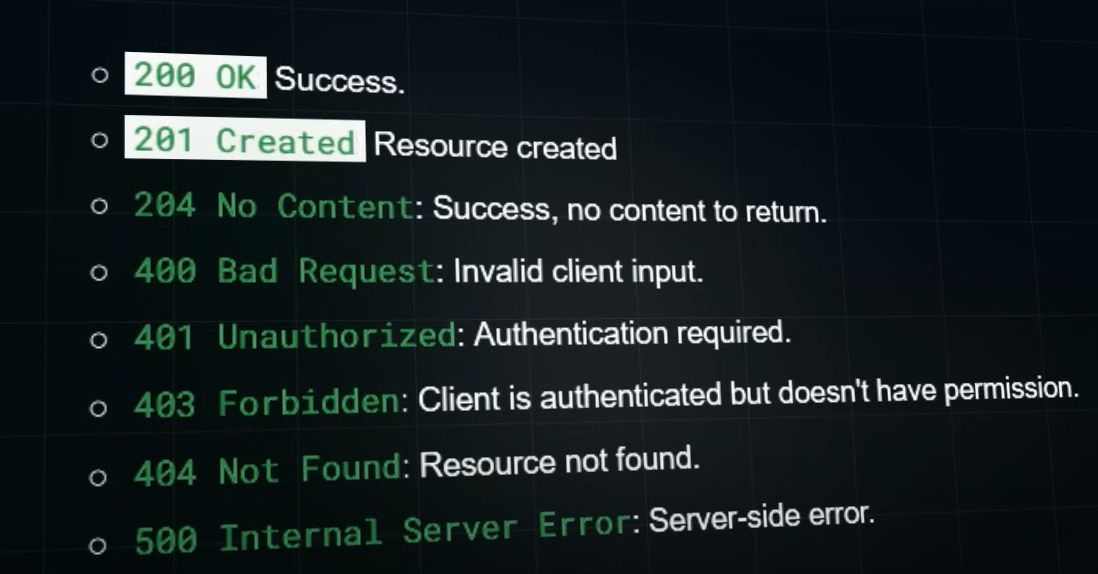




**3. Types of API Characteristics.**



**4. HTTP Status Codes**



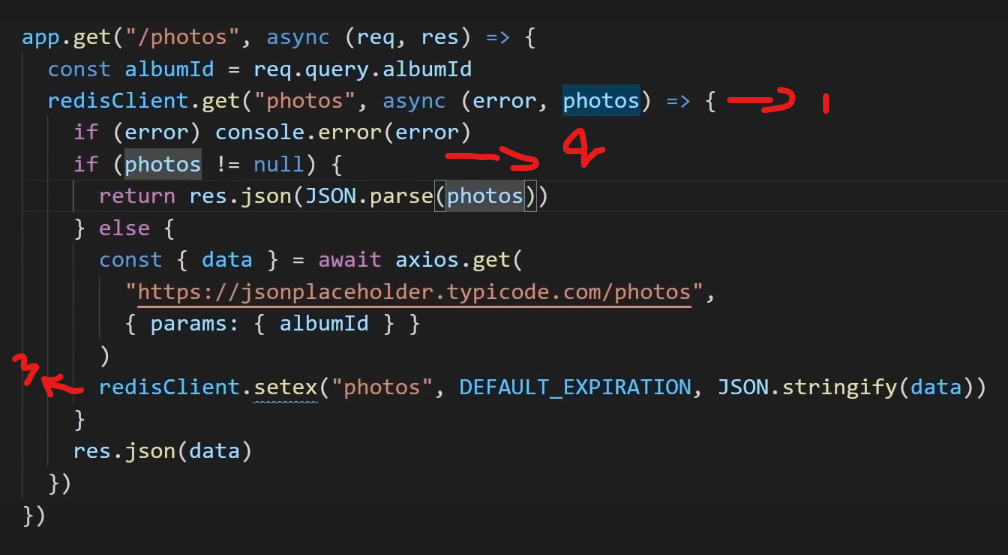
**5. Versioning**

It’s very important to maintain versioning of API methods, there are 2 ways to handle it

1. URI versioning – simply gives /v1/customers, /v2/customers
2. Header Versioning – we can handle through route attribute via headers by providing headers=”Accept-Version=v1/v2”

**6. Redis Cache**

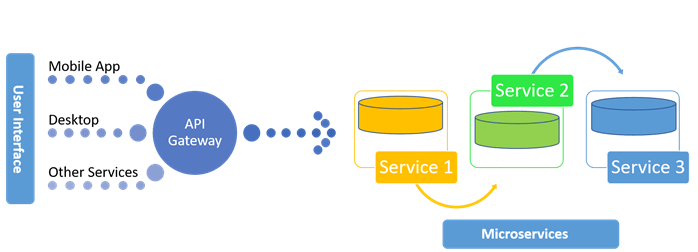
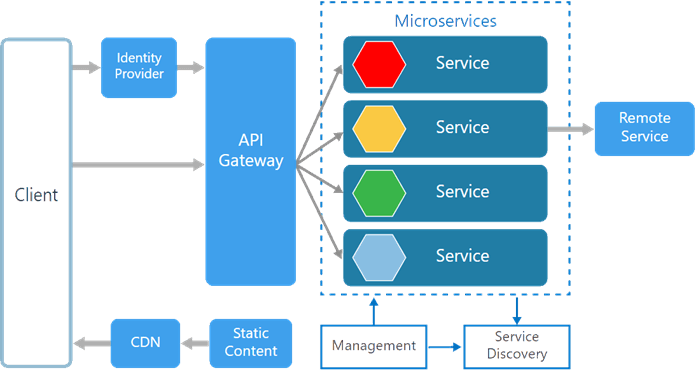
It helps to fetch data from cache rather to get it from DB for every request, to reduce time.



1. Trying to get data from Redis Cache
2. Check whether data is present in Redis cache or not
3. Set data into Redis Cache.

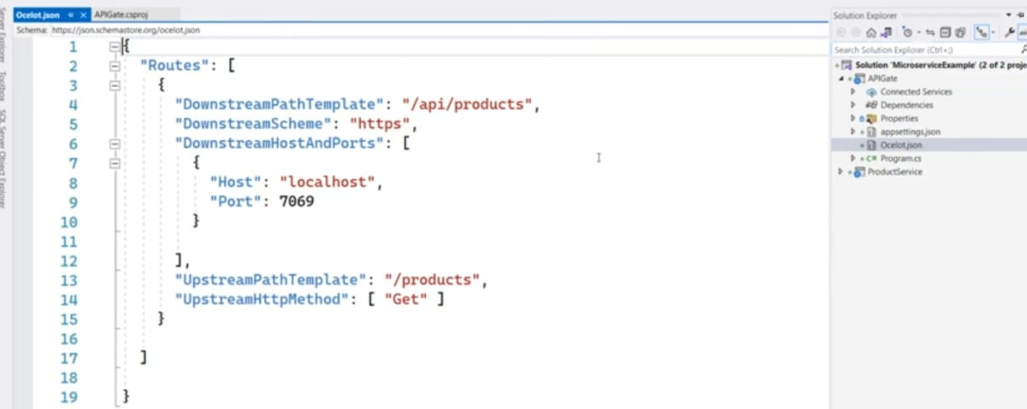
**6. Microservices**

Microservices is an architectural style where an application is built as a **collection of small, independent services that communicate over lightweight protocols like HTTP** or messaging queues. Each service focuses on a specific business capability, ensuring loose coupling and high cohesion.



API Gateway:

To communicate multiple service in microservices we need to interact using API gateway, in which you can install Ocelot package and create an Ocelot.json file and create Routes with Downstream and upstream details.



And configure this with in program.cs file.

A screenshot of a computer program

AI-generated content may be incorrect.

**7. Key Features of Microservices Architecture?**

* **Decentralized governance:** Independent teams for development.
* **Componentization:** Each service is a component.
* **Flexibility in technology:** Services can use different tech stacks.
* **Scalability:** Services can scale independently.
* **Resilience:** Faults are isolated.

**8. How Microservices communicate with each other?**

IPC stands for Inter Process Communication. It deals how microservices communicate each other. There are 2 ways, microservice communication one is Synchronous, and another one is Asynchronous.

Synchronous – HTTP/REST API & gRPC, immediate response

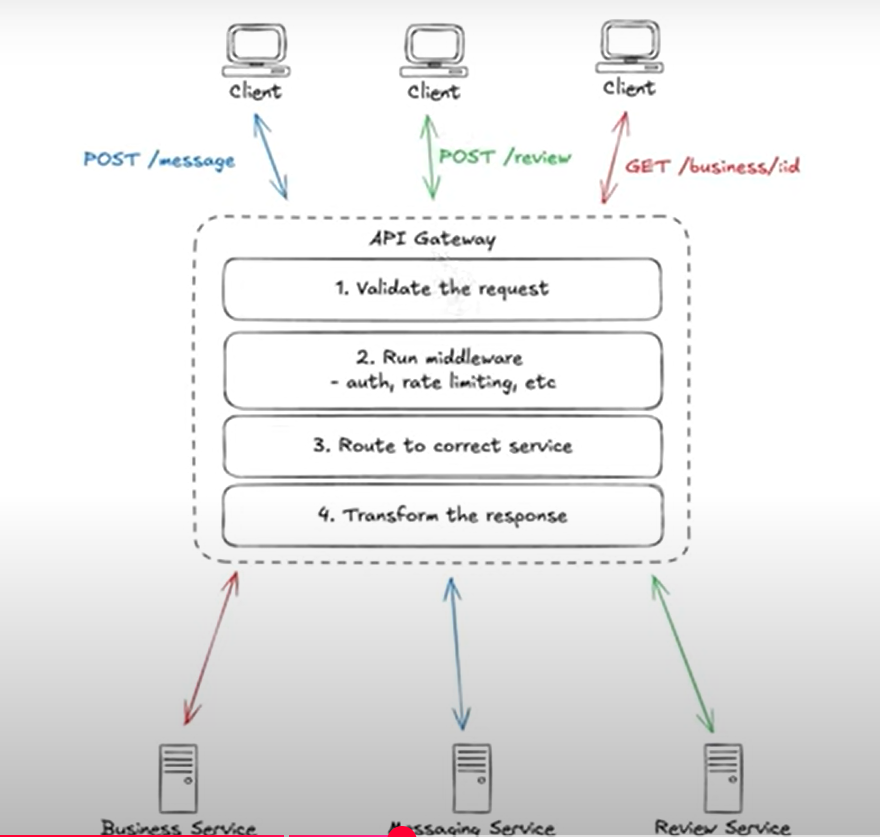
Asynchronous – AVRO & KAFKA, response not immediate



**9. What is the role of API Gate way in Microservices?**

API Gateways handle:

* Routing requests to appropriate services.
* Load balancing.
* Authentication and authorization.
* Caching and monitoring.



**10. How to handle Fault tolerance?**

* **Retry mechanisms:** Retry failed calls.
* **Circuit breakers:** Using libraries like Hystrix, Resilience4j.
* **Fallbacks:** Provide default responses.
* **Bulkheads:** Isolate resources for critical services.

**11. How do you handle data sharing between microservices?**

* Database per service: Services have their own databases.
* Event-driven communication: Use events to share updates.
* API queries: Services expose read-only APIs.

**12. API First Design**

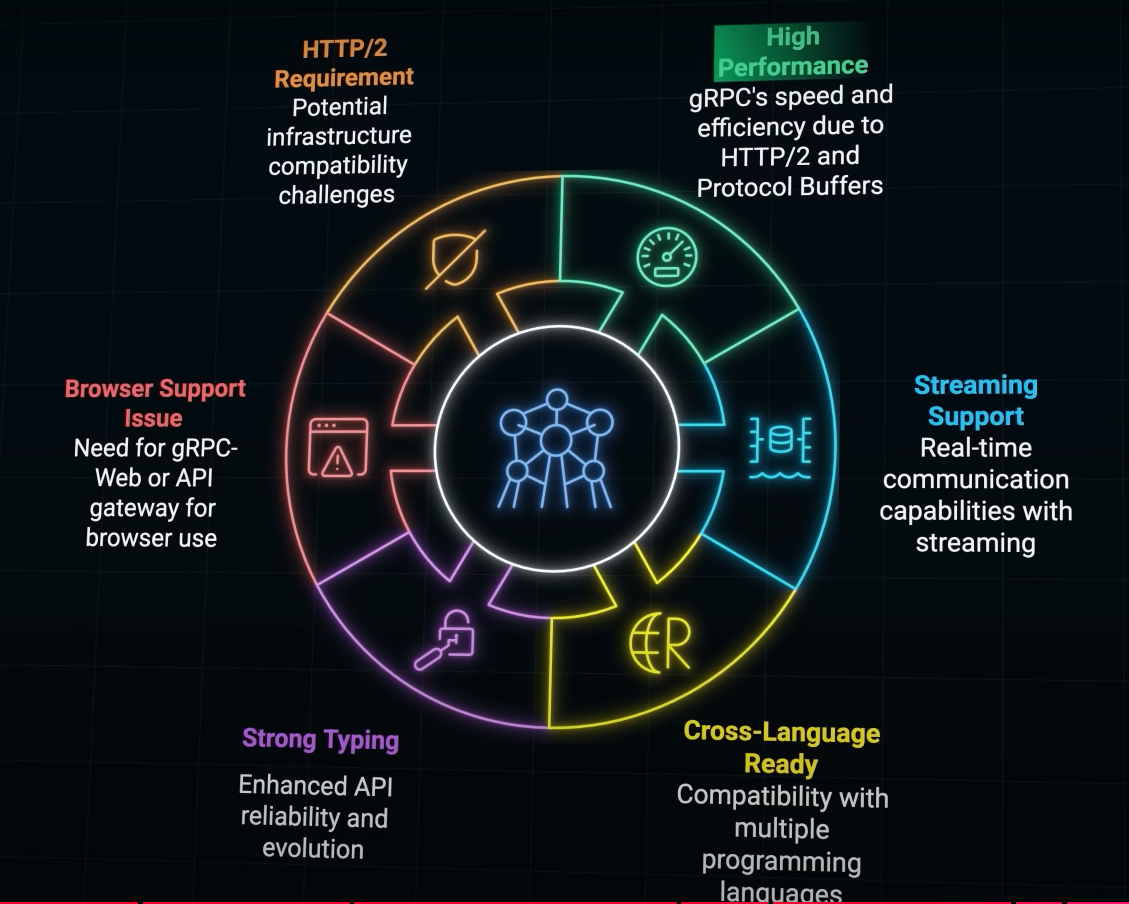
It defines the structure of API and shares with UI developers, and once both UI & Backend developers agree with that, then API implementation taken place.

**13. gRPC**

It is much faster compared with HTTP

Real time communication

Cross language support



**14. Types of API service calls**

Http protocol – Every request we need to do client to server interaction

Web Sockets – Once we made client interaction it will fetch details automatically from server in case fresh data is available in server

Advanced Messaging Queue

gRPC