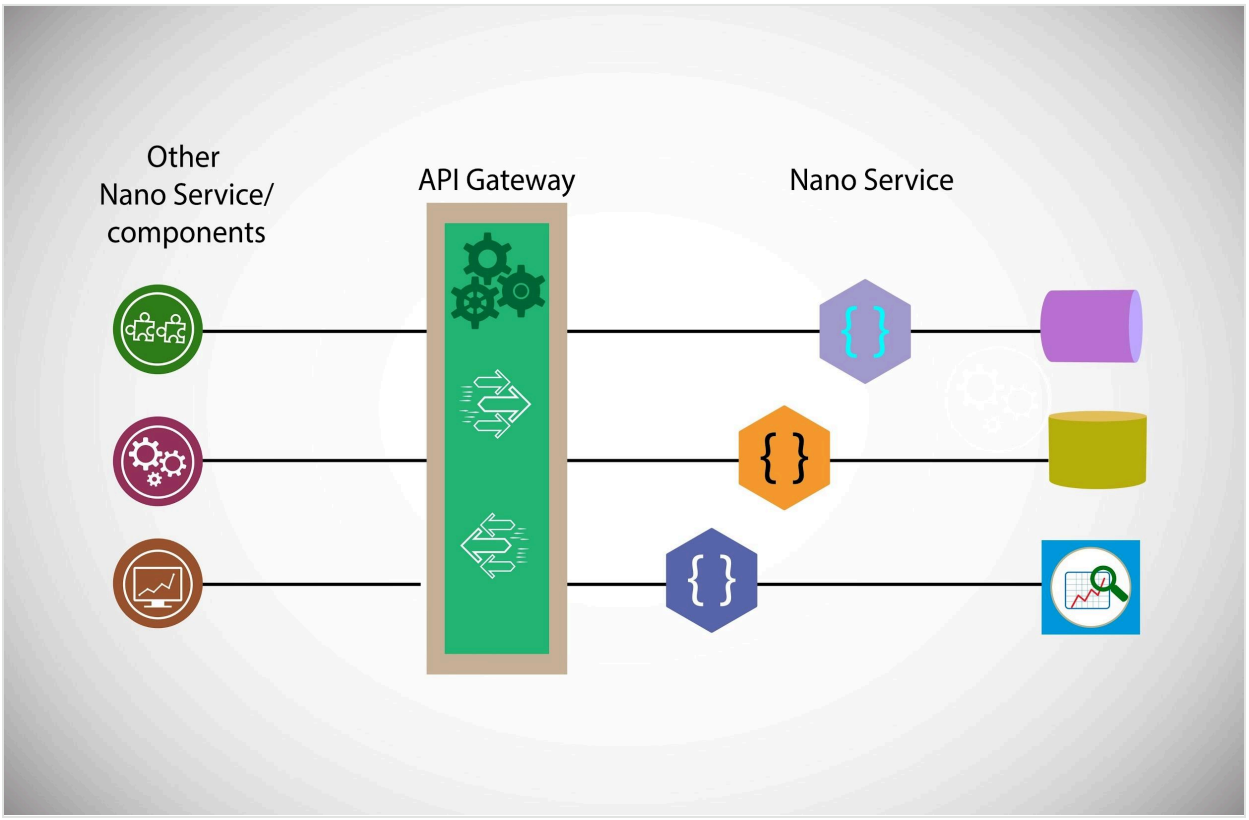
API Gateway & Rate Limiting Study Notes (Day 13–14)

I. The API Gateway: The "System's Front Door"

In a microservices world, you don't want clients talking to 50 different services. You need a single entry point.

1. What is an API Gateway?

Think of the API Gateway as a Nightclub Bouncer. It checks IDs (Authentication), ensures the club isn't too full (Rate Limiting), and tells you which room the music you like is in (Request Routing).



Shutterstock

2. Core Responsibilities

- **Authentication & Authorization: Verifying JWTs or API keys before the request ever hits your expensive backend logic.**

- **Request Routing: Mapping `/v1/orders` to the Order Service and `/v1/users` to the User Service.**
- **Protocol Translation: Converting modern HTTP/2 or REST calls into internal gRPC or older SOAP messages.**
- **Centralized Logging: Seeing all traffic in one place for easier debugging.**

Interview Talk Track:

"I implement an API Gateway to decouple the client from my internal microservices. This allows me to handle 'cross-cutting concerns'—like security and rate limiting—in a centralized place. It prevents me from having to rewrite authentication logic for every single service, making the entire system more maintainable and secure."

II. Rate Limiting & Throttling: The Defense

These are the mechanisms used to ensure your system remains available even when under heavy load or attack.

1. Rate Limiting vs. Throttling

Feature	Rate Limiting	Throttling
Action	Rejects excess requests immediately.	Slows down processing (queuing).
Feedback	Returns HTTP 429 (Too Many Requests).	May result in increased latency for the user.
Goal	Protects against abuse/DDoS.	Manages capacity during spikes.

III. Rate Limiting Algorithms

This is a very common technical deep-dive topic in interviews.

- **Token Bucket: A bucket holds N tokens. Each request takes one. Tokens are refilled at a constant rate r. If the bucket is empty, the request is rejected. Allows for small bursts of traffic.**
- **Fixed Window: Resets the counter at the start of every minute/hour.**
    - ***Problem:* If a user sends 100 requests at 10:00:59 and 100 at 10:01:01, they successfully sent 200 requests in 2 seconds, potentially crashing the server.**
- **Sliding Window: A more precise method that looks at the exact timestamp of previous requests to smooth out the traffic flow.**

Interview Talk Track:

"If I need to support bursty traffic—where a user might perform five quick actions followed by silence—I'd choose the Token Bucket algorithm. It's highly efficient and widely used in production because it's easy to implement using a distributed store like Redis."

IV. Distributed Rate Limiting (The Redis Secret)

If you have 10 API Gateways, how do they know if User A has reached their limit?

- **The Problem: Local memory counters only work for that specific server.**
- **The Solution: Use a centralized, fast, in-memory store like Redis.**
- **The Logic: Every time a request comes in, the Gateway asks Redis: *"Has User A hit 100 requests yet?"* Redis increments the count and returns the value atomically.**

Practice Interview Questions (Day 13–14)

1. "If the API Gateway is a 'Single Point of Failure,' how do we fix that?"

- **Answer: We never deploy just one instance. We put the API Gateway behind a high-availability Load Balancer and run multiple instances across different availability zones. If one Gateway instance crashes, the Load Balancer routes traffic to the healthy ones.**

2. "Why is 429 the most important status code for Rate Limiting?"

- **Answer: HTTP 429 (Too Many Requests) tells the client specifically why their request failed. A good client will see this and implement Exponential Backoff (waiting longer and longer before retrying) instead of just spamming the server.**

3. "Where should Rate Limiting live? In the app code or the Gateway?"

- **Answer: Ideally, at the Gateway. If you do it in the app code, the request has already traveled through your network, used up bandwidth, and hit your application server. Doing it at the Gateway (the edge) stops the 'bad' traffic before it costs you money or resources.**

4. "How do you handle 'Noisy Neighbors' in a multi-tenant system?"

- **Answer: We apply Tiered Rate Limiting. We can set different limits based on the user's API Key. A 'Free Tier' user might get 10 requests/min, while a 'Premium' user gets 10,000 requests/min. This ensures one free user can't hog the resources of our paying customers.**

5. "What is the 'Hard' vs 'Soft' limit in Throttling?"

- **Answer: A Hard Limit is a strict cutoff (Rate Limiting). A Soft Limit allows a user to exceed their quota temporarily if the system has extra capacity, but starts slowing them down (Throttling) as the system reaches its maximum load.**