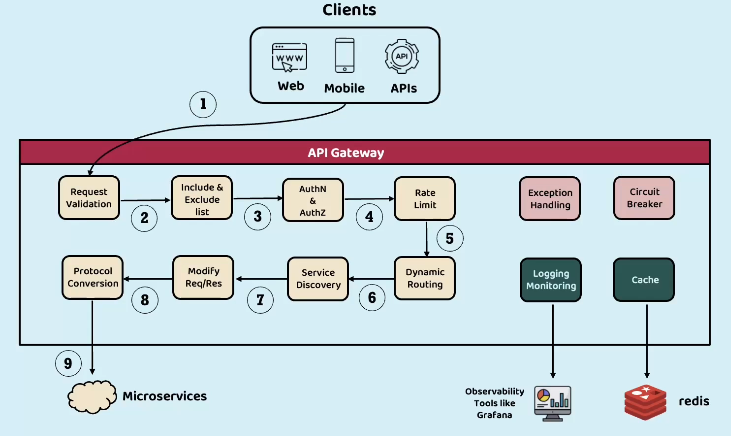
Gateway, Routing, and Cross-Cutting Concerns in Microservices

# Introduction

An API gateway (or so called an edge server) acts as a single entry point to the microservices network. It’s basically a reverse proxy that intercepts user requests, routes them to appropriate back-end services and performs common tasks (cross-cutting concerns) like authentication, logging, statistics, rate limiting, connection to a billing system to monetize the APIs, etc.

With centralizing all these common tasks, you won’t have to implement these logics across each service and provide consistency in terms of these logics across your services. Using a common library used across all services instead of having it implemented in a single-entry point may cost you impact analysis and integration tests when you have some changes to make sure everything works fine in case of changing security policies for example. (?)

It will help you make your system more fault tolerant and resilient by implementing a retry mechanism and timeouts in service calls. It also helps you prevent cascading failures when calling downstream services using circuit breakers for example. (?)



* You can have request validation to check for empty headers, or query parameter or checking if the payload of a request adheres to the specified JSON schema for example.
* You can have black lists of some IPs that are not allowed to send requests
* You can also cache some responses from calls to certain endpoints for like 5 minutes

And serve the upcoming requests from the cache instead of making a new request.

* You can switch to HTTP from HTTPS for making calls to internal services
* You can implement rate limiting for each IP for example

# Spring Cloud Gateway