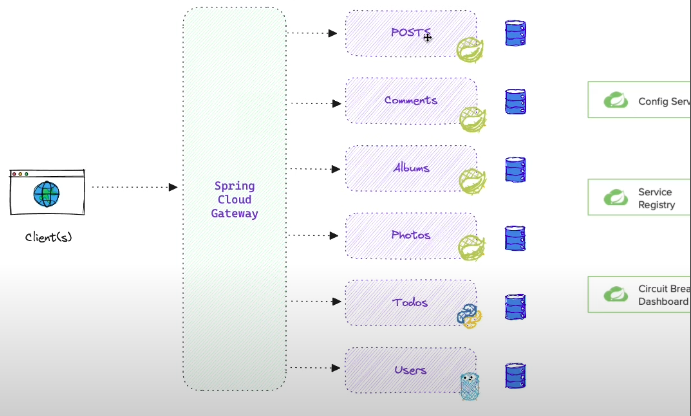
SPRING CLOUD GATEWAY

A gateway in a microservice architecture acts as a single point of entry between clients and a cluster of microservices. It plays a vital role in cross-cutting concerns by handling tasks such as authentication, rate limiting, common business validation, and more.

A white background with blue text

Description automatically generated



Spring Cloud Gateway is built as a Spring Boot application that runs on Netty and utilizes a reactive programming approach.

A white paper with black text

Description automatically generated

A screenshot of a diagram

Description automatically generated

1. **Routes**: Spring Cloud Gateway allows you to define a set of rules called routes that determine how incoming requests should be handled. Routes are configured using a combination of predicates and filters. Predicates are conditions based on request attributes (e.g., path, headers) that determine if a route should be applied. Filters, on the other hand, manipulate the request and response as they pass through the gateway.
2. **Predicates**: When an incoming request arrives, the**Gateway handler** matches the request against the defined routes using predicates. If a match is found, the request is forwarded to the appropriate destination, which can be a specific microservice, a load balancer, or any other backend service.
3. **Filters**: Filters allow you to modify the request and response as it flows through the gateway. Filters can be used for various purposes such as authentication, authorization, rate limiting, logging, request/response transformation, and more. Spring Cloud Gateway provides a rich set of built-in filters and also allows you to create custom filters to cater to specific requirements. These filters can be one URL per se or a global filter.

A screen shot of a computer

Description automatically generated

A black screen with white text

Description automatically generated