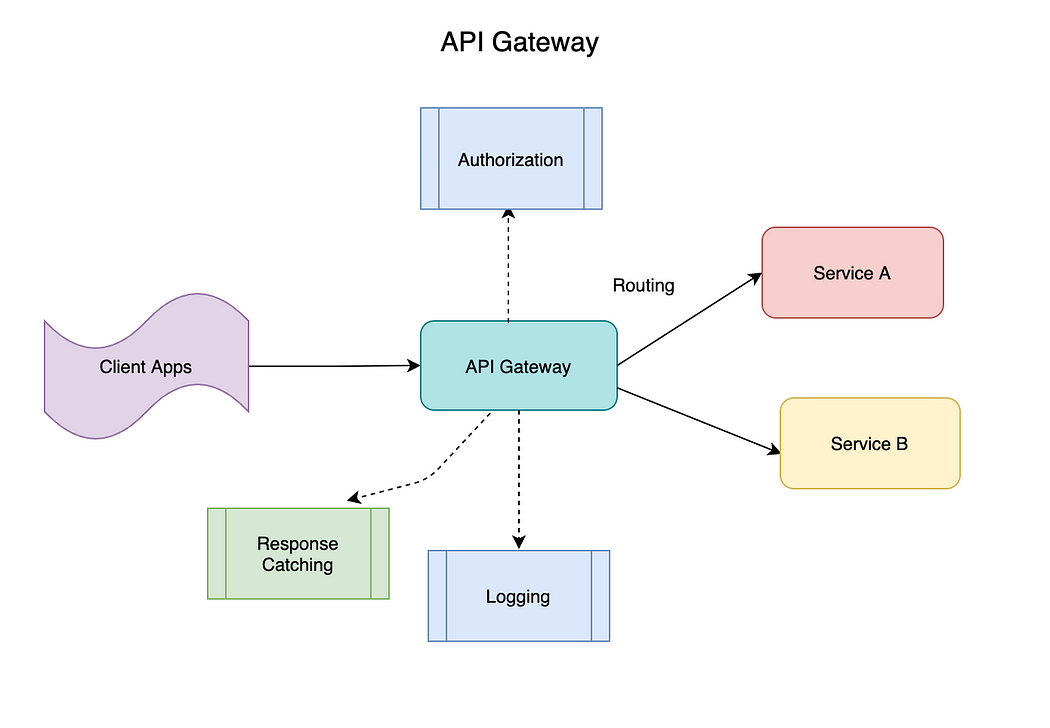
**Einführung in API Gateways**

Ein API Gateway ist ein Management-Tool, das zwischen einem Client und einer Sammlung von Backend-Services sitzt. Es fungiert als Reverse Proxy(ist ein Server, der als Vermittler zwischen einem Client (z.B. einem Webbrowser) und einem oder mehreren Backend-Servern fungiert), um API-Aufrufe entgegenzunehmen, die benötigten Services zu aggregieren und passende Ergebnisse zurückzugeben. Zusätzlich können API Gateways Aufgaben wie Authentifizierung, SSL-Termination, Ratenbegrenzung und Logging übernehmen.



**Vorteile von API Gateways**

1. **Entkopplung der Clients**: Clients müssen interne Endpunkte nicht kennen; nur ein Endpunkt wird exponiert, der verschiedene Serviceanfragen verteilt.
2. **Vereinfachtes Management**: Gemeinsame Anliegen wie Ratenbegrenzung, Logging und Authentifizierung werden zentral verwaltet.
3. **Protokolltransformation**: Ein Gateway kann Client-Anfragen in Service-Anfragen über verschiedene Protokolle konvertieren.

**Spring Cloud Gateway**

Spring Cloud Gateway bietet eine Bibliothek zum Erstellen eines API Gateways auf Basis von Spring WebFlux. Es zielt darauf ab, eine einfache und effektive Methode zum Routing von APIs sowie zur Bereitstellung von Sicherheits-, Monitoring- und Resilienzfunktionen zu bieten.

**Komponenten von Spring Cloud Gateway**

1. **Routes**: Definiert durch eine ID, eine Ziel-URI, eine Sammlung von Prädikaten und Filtern.
2. **Predicates**: Java 8-Funktionsprädikat, um anhand von HTTP-Anfragen zu filtern.
3. **Filters**: Modifizieren Anfragen und Antworten vor/nach dem Senden von Anfragen.

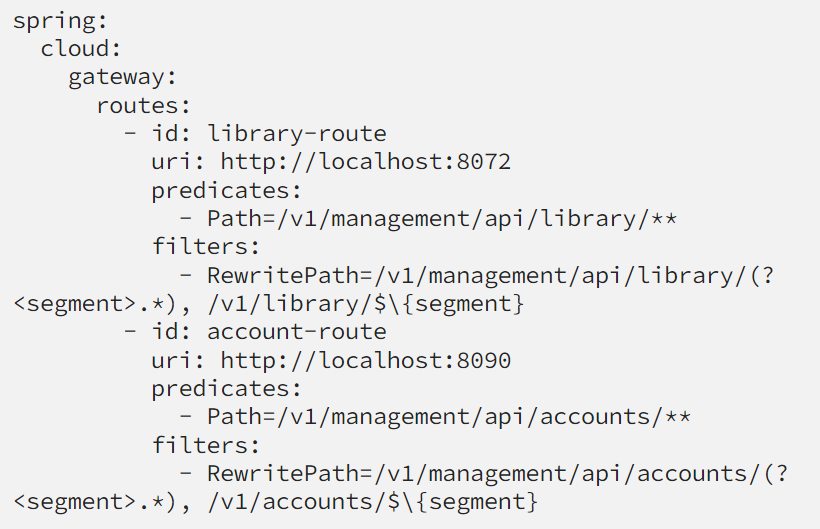
**Workflow**

Das Gateway empfängt eine Anfrage und prüft über das Gateway Handler Mapping, ob die Anfrage eine Route trifft. Dann wird sie an den Gateway Web Handler weitergeleitet, der eine Filterkette auf die Anfrage anwendet. Filterlogik kann vor oder nach der Proxy-Anfrage angewendet werden.

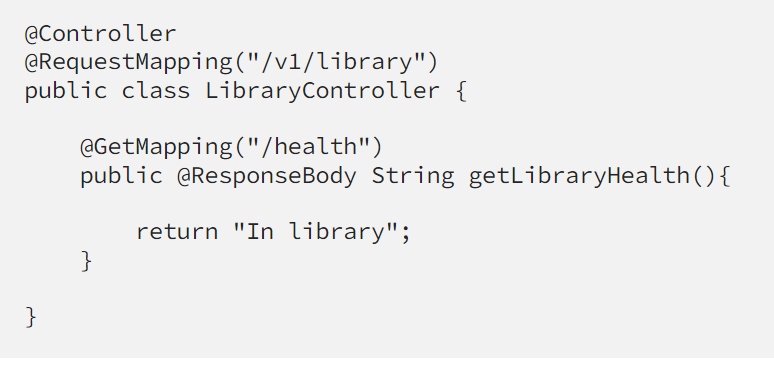
**Beispiel für Code**

Hier ist ein Code-Beispiel für eine Konfiguration des Spring Cloud Gateways mit zwei Microservices (Bibliothek und Konten):

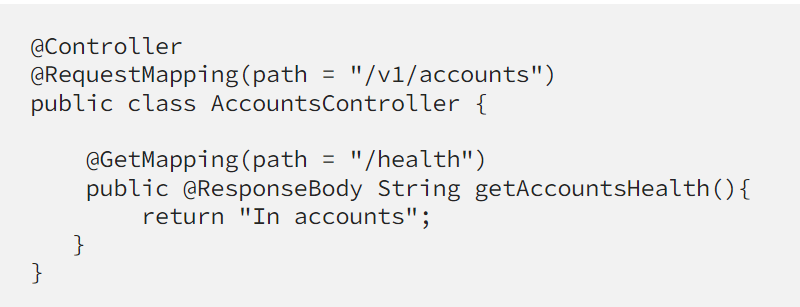
**application.yml:**



**LibraryController:**



**AccountsController:**



Diese Konfiguration ermöglicht das Routing von Anfragen über das API Gateway zu den entsprechenden Microservices.

**Nachteile von API Gateways**

* Zusätzlicher Netzwerk-Hop kann die Performance beeinträchtigen.
* Kann zum Single Point of Failure werden.
* Verwaltung und Hinzufügen von Routen kann komplex sein.

**Zusammenfassung:** Wir haben die Vorteile von API Gateways, die Funktionen des Spring Cloud Gateway, den Workflow und ein Beispiel für die Erstellung eines API Gateways besprochen.