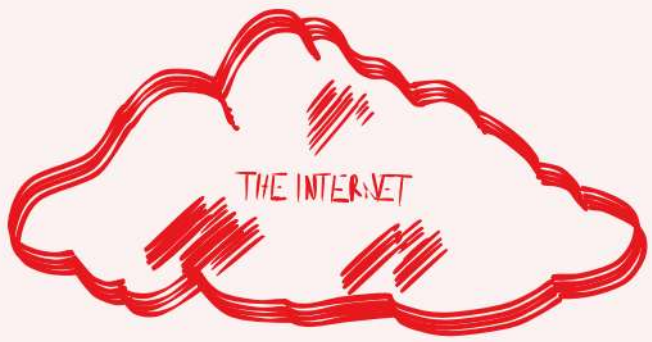
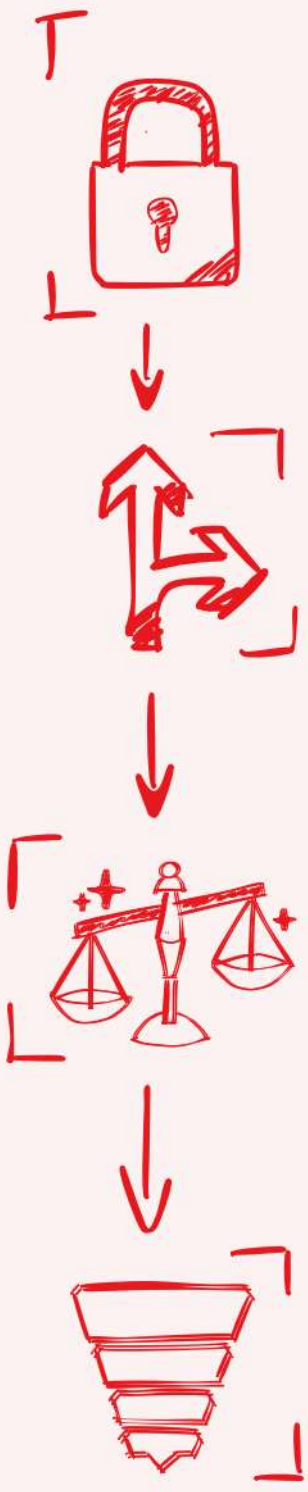
NETFLIX ZUUL
API GATEWAY FRAMEWORK

ZENTRALE



EINGANGSSTELLE



SICHERHEIT

- AUTHENTIFIZIERUNG (USER PRÜFEN)
- AUTORISIERUNG (RECHTE PRÜFEN)
- SSL/TLS VERSCHLÜSSELUNG

ROUTING

- SERVICES SELBSTÄNDIG ERKENNE
- ANFRAGE AN RICHTIGEN SERVICE WEITERLEITEN
- KANN VERSIONIERUNG ÜBERNEHMEN
- ENTKOPPELT FRONTEND - BACKEND

LOAD BALANCING

- VERTEILUNG AUF MEHRERE INSTANZEN
- DYNAMISCHE ANPASSUNG
- HOHE VERFÜGBARKEIT & SKALIERUNG

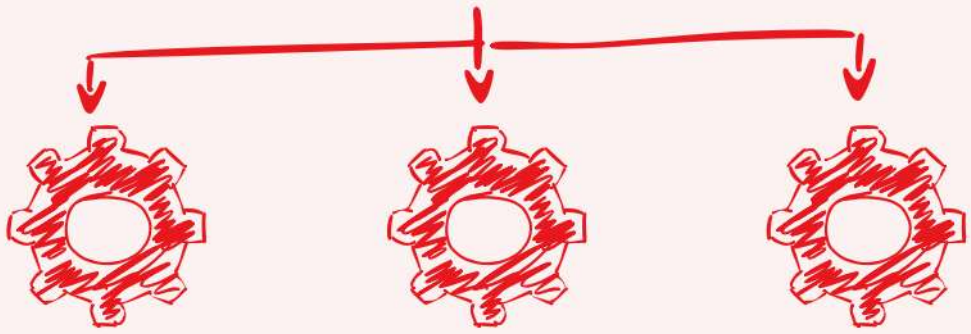
FILTER

- HEADER PRÜFEN UND ANPASSEN
- REQUESTS BLOCKIEREN ODER UMSCHREIBEN
- REGELN FÜR LOGGING & POLICIES

MONITORING

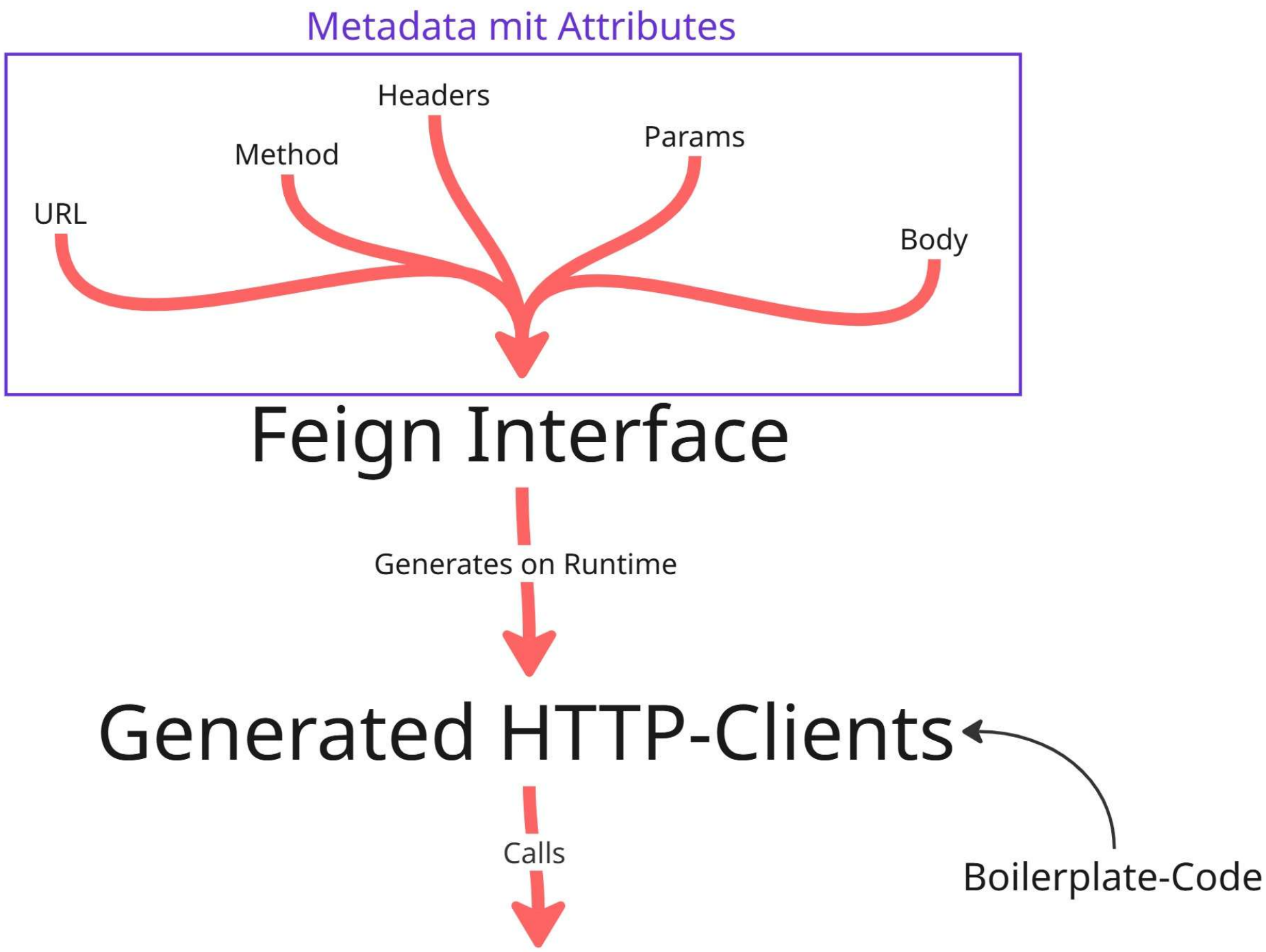
- TRAFFIC & LATENZ MESSEN
- FEHLER UND AUSFÄLLE ERKENNEN
- SICHERHEITS- UND NUTZUNGSSTATISTIKEN

SERVICES



ANDRÉS, CHAIMAA, JANNIS

FeigN



In Java, **Feign** is a **declarative HTTP client** developed by Netflix and widely used with **Spring Cloud**. It simplifies the process of making HTTP requests to REST APIs by letting you write Java interfaces annotated with metadata, instead of writing boilerplate networking code manually.

Why use Feign?
Cleaner, declarative API calls.
Easy integration with microservices architectures.
Works well with **load balancing, retries, fallbacks, and service discovery** in Spring Cloud.

Netflix Eureka Discovery Service

Framework & Einsatz

Was ist Eureka?

- Open-Source Framework von Netflix
- Dient als Service Discovery (Dienstfindung)
- Besteht aus Eureka-Server (Registry) und Eureka-Clients (Dienste)
- Nutzt Heartbeats zur Registrierung und Verfügbarkeit

Eureka in der Praxis

- Ermöglicht es, dass sich Microservices selbst anmelden und gefunden werden
- Hilft bei Lastverteilung (Load Balancing)
- Erhöht Fehlertoleranz
- Spart manuelles Konfigurieren von IP-Adressen

Eureka = Telefonbuch für Microservices

Beschreiben Sie was OAuth ist und wie OAuth funktioniert.

Was ist OAuth?

- Autorisierungsstandart
- Ermöglicht Anwendungen Zugriff auf **geschützte Daten** ohne Login/Passwort.
- Der User kann entscheiden, welche Daten er preisgeben will. Für diese wird ein **Token** erstellt.

Beispiel

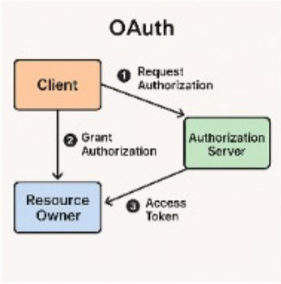
1. Login mit Google
- App schickt User zu Google um sich einloggen
- Google fragt nach erlaubnis der App einen OAuth token zu generieren
- User stimmt zu
- Google generiert einen Token für die App
- Login in der App erfolgreich, ohne das sie je das Passwort sah

Wie funktioniert OAuth?

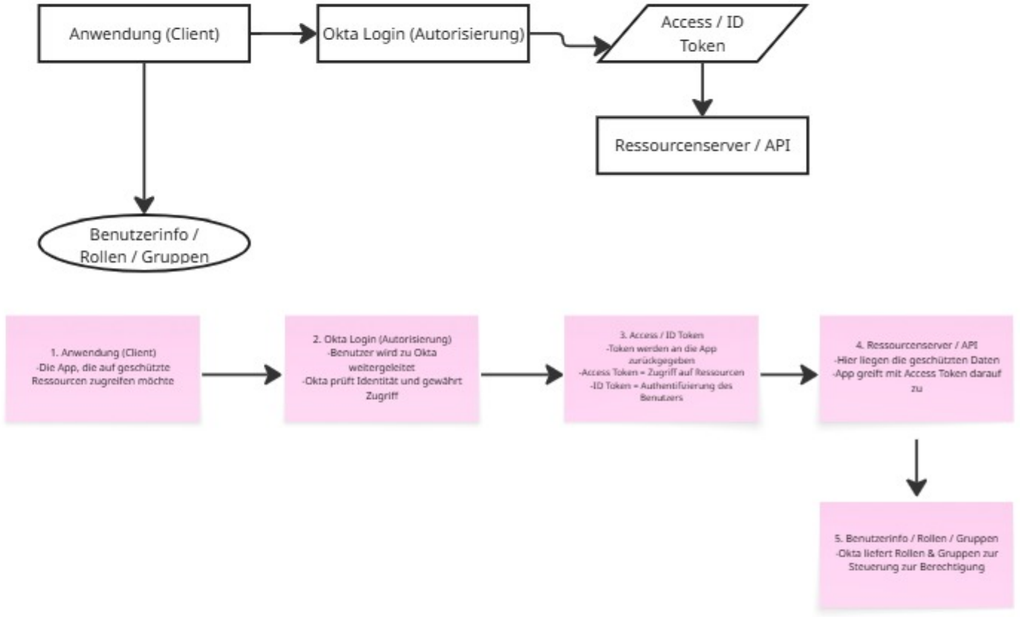
1. Ein Benutzer öffnet eine App, die Daten braucht. z.B. Kalender
2. Die App schickt den Benutzer zum **Login-Server.** z.B. Google
3. Dort meldet sich der Benutzer an und erlaubt den Zugriff auf die Daten.
4. Die App bekommt einen **Code** zurück.
5. Mit diesem Code holt sich die App ein **Access Token**.
6. Mit dem Token darf die App die gebrauchten Daten vom Server holen.

Wichtige Punkte

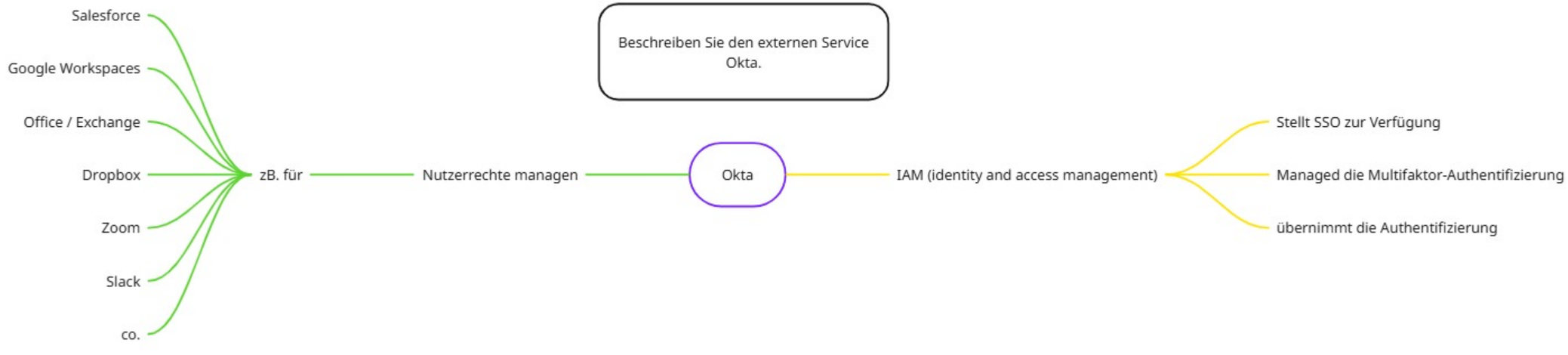
- Das Passwort bleibt **immer beim Login-Server**, die App sieht es nie.
- Der Benutzer entscheidet, welche seiner Daten freigegeben werden.
- Tokens sind nur für kurze Zeit gültig. Das macht sie sicherer
- Mit einem **Refresh Token** kann die App ein neues Access Token bekommen, ohne dass der Benutzer sich wieder einloggen muss.



Beschreiben Sie wie der Service Okta in der Anwendung genutzt wird.



Beschreiben Sie den externen Service Okta.



Meier, Weber, Beere

04.09.2025

Spring Cloud

Was ist das Spring Cloud Framework?

Spring Cloud bietet Entwicklern Tools, mit denen sie schnell einige der gängigen Muster in verteilten Systemen erstellen können (z. B. Konfigurationsmanagement, Service Discovery, Circuit Breaker, intelligentes Routing, Micro-Proxy, Control Bus).

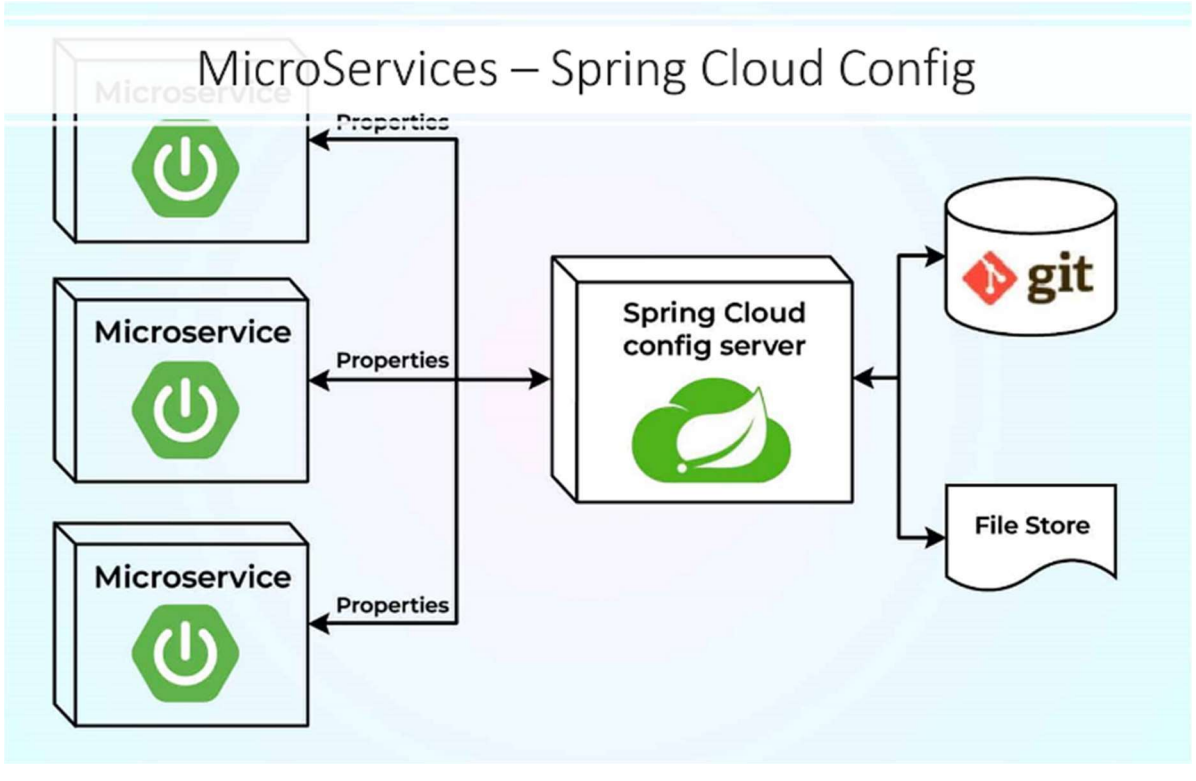
Die Koordination verteilter Systeme führt zu Boilerplate Mustern, und mit Spring Cloud können Entwickler schnell Dienste und Anwendungen erstellen, die diese Muster implementieren.

Was hat es mit Microservices zu tun?

Spring Cloud sorgt in der Microservice Architektur dafür, dass die Services miteinander kommunizieren, sich selbst finden und skalierbar laufen können. Es bietet die nötige Infrastruktur, damit Microservices als ein stabiles Gesamtsystem funktionieren.

Welche Tools sind verfügbar in der Spring Cloud?

- Distributed/versioned configuration
- Service registration and discovery
- Routing
- Service-to-service calls
- Load balancing
- Circuit Breakers
- Distributed messaging
- Short lived microservices (tasks)
- Consumer-driven and producer-driven contract testing



HYSTRIX

DEFEND YOUR APP

Was ist Hystrix?

- Java-Framework von Netflix
- Für Ausfallsicherheit in verteilten Systemen
- Nutzt das Circuit Breaker-Prinzip (wie eine Sicherung)
- Verhindert, dass Fehler einen ganzen Service lahmlegen

Vorteile

- Isoliert Zugangspunkte zu entfernten Systemen
- Stoppt Kaskadenausfälle
- Ermöglicht Resilienz.

Aufgabe in der Anwendung

Ohne Hystrix:
- Service-Aufruf hängt/ist langsam –> blockiert weitere Anfragen
- Risiko von Kettenreaktionen und Time-outs

Mit Hystrix:
- Früherkennung von Fehlern/Langsamkeit
- Circuit Breaker öffnet –> problematische Aufrufe werden gestoppt
- Fallback liefert Ersatzantwort (z.B. "später versuchen")
- Rest des Systems bleibt verfügbar/stabil