cache

Motivation

Der Cache soll vorrangig die Microservices entlasten, indem er die Response zwischenspeichert und der Registry für eine gewisse Zeit zur Verfügung stellt. Insbesondere der Service Weather ist davon betroffen, da dieser eine API von OpenWeatherMap nutzt, welches pro Tag 4000 Wettervorhersagen treffen kann, sonst wird dieser Kostenpflichtig bzw. kann dann keine Requests mehr entgegen nehmen.

Free	Startup **40 USD** / month	Developer **180 USD** / month	Professional **470 USD** / month	Enterprise **2.000 USD** / month
60 calls/minute **1,000,000 calls**/month	**600 calls**/minute **10,000,000 calls**/month	**3,000 calls**/minute **100,000,000 calls**/month	**30,000 calls**/minute **1,000,000,000 calls**/month	**200,000 calls**/minute **5,000,000,000 calls**/month
Current Weather Minute Forecast 1 hour∗ Hourly Forecast 2 days∗ Daily Forecast 7 days∗ Historical weather 5 days∗ Climatic Forecast 30 days Bulk Download	Current Weather Minute Forecast 1 hour∗∗ Hourly Forecast 2 days∗∗ Daily Forecast 16 days Historical weather 5 days∗∗ Climatic Forecast 30 days Bulk Download	Current Weather Minute Forecast 1 hour Hourly Forecast 4 days Daily Forecast 16 days Historical weather 5 days Climatic Forecast 30 days Bulk Download	Current Weather Minute Forecast 1 hour Hourly Forecast 4 days Daily Forecast 16 days Historical weather 5 days Climatic Forecast 30 days Bulk Download	Current Weather Minute forecast 1 hour Hourly Forecast 4 days Daily Forecast 16 days Historical weather 5 days Climatic Forecast 30 days Bulk Download
Basic weather maps Historical maps	Basic weather maps Historical maps	Advanced weather maps Historical maps	Advanced weather maps Historical maps	Advanced weather maps Historical maps
Weather triggers	Weather triggers	Weather triggers	Weather triggers	Weather triggers
Weather widgets	Weather widgets	Weather widgets	Weather widgets	Weather widgets
Uptime 95%	Uptime 95%	Uptime 99.5%	Uptime 99.5%	Uptime 99.9%

∗ - 1,000 API calls per day by using One Call API
∗∗ - 2,000 API calls per day by using One Call API

Requirements

Die Registry soll Requests von dem Deconcentrator entgegennehmen und überprüfen, ob diese Request innerhalb einer fest definierten Zeit bereits eine Response erhalten hat. Ist dies der Fall guckt die Registry in den Cache, um sich die dort Zwischengespeicherte Response zu holen und diese an

den Sender der Request zu leiten. Dabei “ersetzt” der Cache den angesprochenen Microservice.

Functional

- /CAF100/ The system must ...
- /CAF100/ The system must ...
- /CAF100/ The system must ...
- /CAF200/ The cache must ...
- /CAF200/ The cache must ...
- /CAF200/ The cache must ...

Non Functional

- /CANF100/ The system must ...
- /CANF100/ The system must ...
- /CANF100/ The system must ...

- /CANF200/ The cache must ...
- /CANF200/ The cache must ...
- /CANF200/ The cache must ...

User Stories

- /CAUS100/ Als Betreiber möchte ich Anfragen die das selbe Ergebnis erzeugen abfangen und damit die Microservices entlasten.
- /CAUS101/ Als Betreiber möchte ich die Anfragen an die verschiedenen APIs reduzieren um nicht in ein teureres Preispaket zu fallen.

Use Cases (?)

Technologies

Für Node.js existieren mehrere Caching Lösungen. Bei den ersten recherchen fielen die npm packages “memory-cache” und “node-cache” auf. Da “memory-cache” seit drei Jahren kein Update bekommen hat, haben wir uns letzten endes für “node-cache” entschieden.

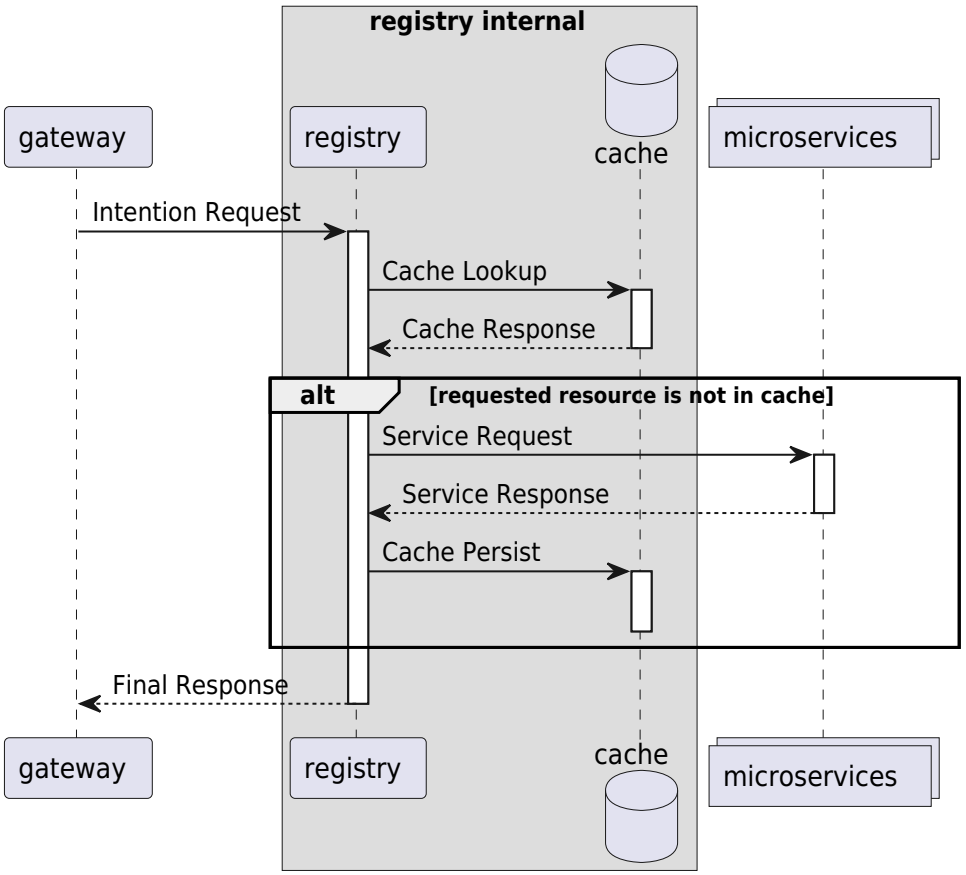
“node-cache” ist eine simple Caching Lösung, die nach dem Key-Value prinzip funktioniert. Der Funktionsumfang besteht dabei aus den Methoden “set”, “get” und “delete”, wobei die Methode “set” einem zusätzlich erlaubt noch einen Timeout (“ttl” bzw. “time to live” genannt) zu übergeben. Ist der Timeout überschritten, wird der Eintrag automatisch aus dem Cache gelöscht. Der Nachteil dieser Lösung ist, dass nur eine Millionen einträge pro Cache Instanz eingetragen werden können. Da aber gleich viel in den Cache eingetragen wird, wie die Anzahl angebotener Funktionen aller Microservices, wird dieser Nachteil nicht eintreffen.

Integration

 # An welcher Stelle wird der Cache in das System eingebaut? (Registry)

 # Wie wird sie eingebaut?

 # Wie sieht das Endprodukt aus?



Abnahmekriterien

 # Was musst der Cache / die Registry können, damit wir das Projekt als erfolgreich definieren können?

Resultierende Aufgaben

 # Was müssen wir also tun?

 # Auflistung der Tickets die entstehen...

 # Vielleicht lieber direkt bei GitHub