

# 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<br><b>40 USD</b> / month   | Developer<br><b>180 USD</b> / month  | Professional<br><b>470 USD</b> / month   | Enterprise<br><b>2.000 USD</b> / month   |
|---|--|--|--|--|
| <b>60 calls/minute</b><br><b>1,000,000</b><br><b>calls/month</b>  | <b>600 calls/minute</b><br><b>10,000,000</b><br><b>calls/month</b>   | <b>3,000</b><br><b>calls/minute</b><br><b>100,000,000</b><br><b>calls/month</b>  | <b>30,000 calls/minute</b><br><b>1,000,000,000</b><br><b>calls/month</b>   | <b>200,000</b><br><b>calls/minute</b><br><b>5,000,000,000</b><br><b>calls/month</b>  |
| <a href="#">Current Weather</a><br><a href="#">Minute Forecast 1 hour*</a><br><a href="#">Hourly Forecast 2 days*</a><br><a href="#">Daily Forecast 7 days*</a><br><a href="#">Historical weather 5 days*</a><br>Climatic Forecast 30 days<br>Bulk Download | Current Weather<br>Minute Forecast 1 hour**<br>Hourly Forecast 2 days**<br>Daily Forecast 16 days<br>Historical weather 5 days**<br>Climatic Forecast 30 days<br>Bulk Download | Current Weather<br>Minute Forecast 1 hour<br>Hourly Forecast 4 days<br>Daily Forecast 16 days<br>Historical weather 5 days<br><a href="#">Climatic Forecast 30 days</a><br>Bulk Download | Current Weather<br>Minute Forecast 1 hour<br>Hourly Forecast 4 days<br>Daily Forecast 16 days<br>Historical weather 5 days<br>Climatic Forecast 30 days<br><a href="#">Bulk Download</a> | Current Weather<br>Minute forecast 1 hour<br>Hourly Forecast 4 days<br>Daily Forecast 16 days<br>Historical weather 5 days<br>Climatic Forecast 30 days<br>Bulk Download |
| <a href="#">Basic weather maps</a><br>Historical maps   | Basic weather maps<br>Historical maps  | <a href="#">Advanced weather maps</a><br><a href="#">Historical maps</a>   | Advanced weather maps<br>Historical maps   | Advanced weather maps<br>Historical maps   |
| <a href="#">Weather triggers</a>  | Weather triggers   | Weather triggers   | Weather triggers   | Weather triggers   |
| <a href="#">Weather widgets</a>   | 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

# Was soll der Cache können?

# Muss hier nicht viel sein denke ich...

## 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

*# Brauchen wir hier glaube ich garnicht, bzw. sollte das mit den User Stories für den BeuthBot oder die Registry definiert werden...*

"Als <Rolle> möchte ich <Ziel/Wunsch>, um <Nutzen>"

- /CAUS100/ Als Student möchte ich ...
- /CAUS101/ Als Student möchte ich ...
- /CAUS102/ Als Student möchte ich ...

## 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.

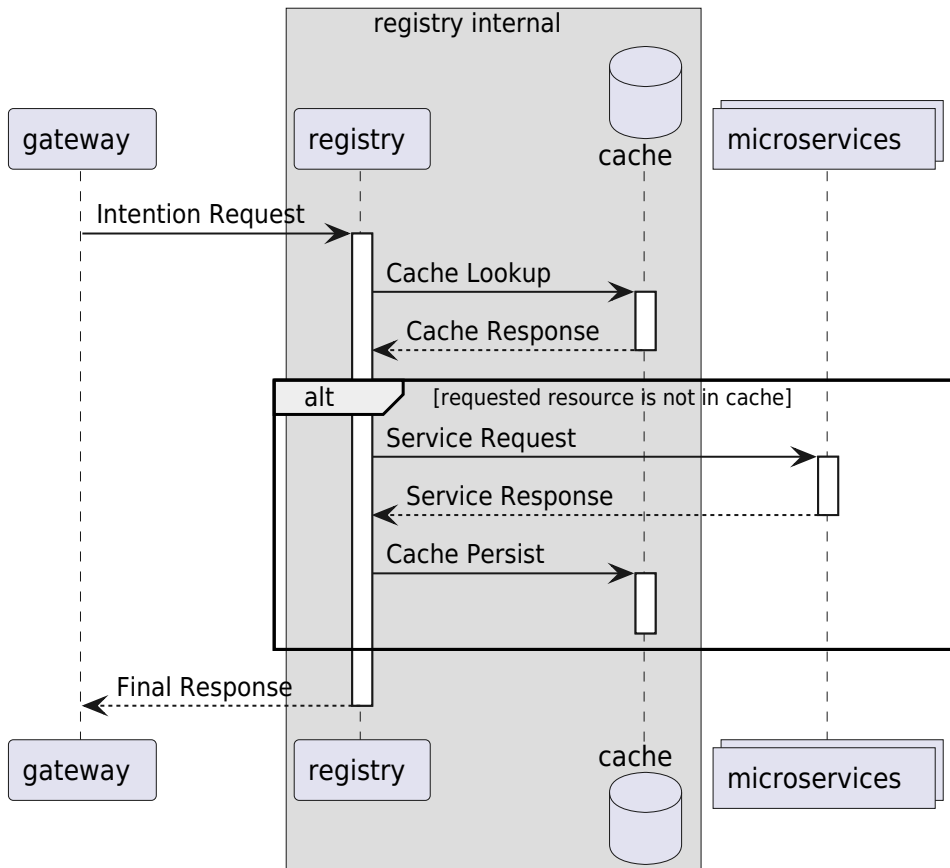
"node-cache" ist eine simple Caching Lösung, die nach dem Key-Value Prinzip funktioniert. Der Funktionsumfang besteht dabei aus den Methoden "set", "get" und "delete", wobei die Methode "set" einem zusätzlich erlaubt noch einen Timeout ("ttl" bzw. "time to live" genannt) zu übergeben. Ist der Timeout 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 muss 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

Nutzungshinweis: Auf dieses vorliegende Schulungs- oder Beratungsdokument (ggf.) erlangt der Mandant vertragsgemäß ein nicht ausschließliches, dauerhaftes, unbeschränktes, unwiderrufliches und nicht übertragbares Nutzungsrecht. Eine hierüber hinausgehende, nicht zuvor durch *datenschutz-maximum* bewilligte Nutzung ist verboten und wird urheberrechtlich verfolgt.