# Data caching

## Huidige stand van zaken

Voor het verminderen van API requests en het eventueel op voorhand kunnen opslaan van bepaalde berekeningen, zijn wij begonnen aan het cachen van data. Dit is grotendeels geslaagd, maar er ontbreekt hierbij een cruciaal onderdeel. Dit is de initiële cache. Er worden al 6-7 jaar metingen opgeslagen en dit komt neer op miljoenen metingen. Het opslaan van de metingen op de manier hoe we nu de cache updaten zou ruim een maand duren.

Wat nu dus de stand van zaken is, is dat de feature in principe werkt maar door het ontbreken van de initiële cache, kan deze nog niet gebruikt worden. Zodoende staat deze feature dus los in een branche genaamd: *feature/data-caching*. In de desbetreffende branch vind je een bestand genaamd *features/meetjestad/MeetJeStadCachingService.java*, waarin alles omtrent het cachen staat.

Voor het zo accuraat mogelijk kunnen houden van de gegevens, hebben wij ervoor gekozen om de locatiegegevens en de daadwerkelijke meetresultaten apart op te slaan. Locatiegegevens zijn belangrijk voor het kunnen ophalen van de laatst bekende locatie, gezien er ook wel eens foutieve locaties vanuit de API binnenkomen. Wanneer dit het geval is, zal dus de laatst bekende locatie ook gekoppeld worden aan de meting. Verder is bekend dat er in de toekomst wellicht ook fijnstof en andere zaken gemeten zullen worden. Zodoende hebben we dus besloten hier nu al rekening mee te houden door bijvoorbeeld de luchtvochtigheid en temperatuur apart op te slaan met een link naar de originele meting id en daarbij een meting type ook op te slaan.

## Vervolg

In principe is het enige wat er nog moet gebeuren is een oplossing vinden voor het opslaan van de initiële cache.

Het probleem waar wij tegen aanliepen hiermee, is dat data soms afhankelijk is van andere data. Wij controleren bijvoorbeeld of een meetstation verplaatst is, en wanneer dit het geval is updaten wij de locatie in de database. Gezien het belangrijk was om locatiegeschiedenis bij te houden (voor het weten van de laatst bekende locatie), houden wij dit in een aparte tabel bij. Wanneer dit dus asynchroon zou gebeuren, kan dit ervoor zorgen dat de locatie niet klopt met de meting.

Bovenstaande situatie zorgt ervoor dat dit niet asynchroon kan gebeuren, en hierbij zijn er meerdere afhankelijkheden waarbij dit slechts een voorbeeld is. Een mogelijke oplossing zou zijn om de ID’s op voorhand te berekenen en in een keer een grote bulk aan data opslaan.

Wat goed is om te weten is dat de daadwerkelijke data ophalen niet lang duurt, het is echt het verwerken van de data waar de bottleneck ligt.

Als er verder gewerkt wordt met data caching moet het volgende meegenomen worden:

1. Het zou mooi meegenomen zijn om naast data caching, de directe API-calls naar MeetJeStad in de code te houden als backup. Dus als de cache faalt, dat er automatisch overgestapt wordt naar MeetJeStadAPI calls. Dit is GEEN EIS, eerder een suggestie die verder besproken moet worden met de stakeholders.
2. Het filteren van de data die gecached wordt, gebeurt op het moment op de verkeerde plek in het proces. Namelijk wanneer de data wordt opgehaald van de externe API om toe te voegen aan de cache. Dit klopt niet met de meest recente eisen van de stakeholders. Dit moet dus aangepast worden zodat de foutieve metingen ook gecached worden, en dan dus het filter alleen voor wijken en de overzichten gebruiken.
3. De tijdslimiet voor metingen wordt momenteel verkeerd toegepast in de data caching branch, deze moet verbeterd worden. Zie het hoofdstuk Tijdsbestek metingen in “Ontwerpkeuzes”.