Data caching

## Inleiding

Met data caching wordt het lokaal opslaan van de data uit de externe MeetJeStad API bedoeld. Deze functionaliteit is momenteel gedeeltelijk gerealiseerd. In dit bestand wordt toegelicht wat data cashing toevoegt aan MB Ontdekt, wat de huidige status is en wat de vervolgstappen zijn.

## Doel

Er zijn meerdere redenen om data caching toe te passen in MB ontdekt, hier worden de belangrijkste benoemd:

Ontbrekende locatiegegevens van metingen kunnen met data caching makkelijk aangevuld worden. Er kan dan simpelweg via één query aan de database gevraagd worden wat de laatste bekende locatie is. Zie ontwerpkeuzes voor een omschrijving van dit probleem.

Data caching vermindert het aantal requests dat gestuurd wordt naar de API van MeetJeStad, wat de benodigde rescources aan hun kant verminderd.

Met data caching kunnen bepaalde (intensieve) berekeningen opgeslagen worden, bijvoorbeeld in welke wijk een meting valt en wat de gemiddelde wijktemperatuur is op een bepaald moment. Zo hoeven deze niet steeds opnieuw te worden gedaan, wat de performance verbeterd.

Door de data te cashen, wordt er een eigen backup gecreëerd. Waardoor de applicatie minder afhankelijk wordt van een continue beschikbaarheid van MeetJeStad.

Er worden nieuwe features mogelijk. Zo kan er met data caching makkelijker gewerkt worden met accounts en kunnen er meldingen gestuurd worden wanneer meetstations over een langere periode geen data meer verstuurd hebben.

## Huidige stand van zaken

Het cachen van data 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 weg filteren van de foutieve data gebeurt momenteel op de verkeerde plek, namelijk tijdens het opslaan in de cache. Voor verduidelijking waar wanneer filtering toegepast moet worden, zie het bestand Ontwerpkeuzes.
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”.