Contents

[1. Inleiding 2](#_Toc125997591)

[2. Verbetering 2](#_Toc125997592)

[3. Realisatie 3](#_Toc125997593)

[3.1. Relationele databank 3](#_Toc125997594)

[3.2. Document georiënteerde databank 4](#_Toc125997595)

[4. Besluit 6](#_Toc125997596)

# Inleiding

De opdracht bestaat uit de verbetering van een bestaande Araani-applicatie: de Araani Analytics Benchmarking. Deze applicatie toont in een webbrowser aan de hand van videobestanden en bijbehorende metadatabestanden de prestaties van de rook- en vlamdetectie algoritmes in grafiekvorm. Alle bestanden worden bewaard op de harde schijf van een lokale machine. De structuur van de locatie van de bestanden en een aantal bijbehorende gegevens staan genoteerd in een excel-bestand dat zich ook op de harde schijf bevindt.

De originele versie van Araani Analytics Benchmarking is gebouwd als een docker-image op de lokale machine. Bij opstart van de service leest de applicatie de inhoud van het excel-bestand in het werkgeheugen. Eenmaal opgestart wordt via de local host de web-GUI geactiveerd en worden een aantal parameters ingesteld om de gewenste data uit de excel-tabel te filteren en de prestaties te berekenen.

# Verbetering

De originele versie van Araani Analytics Benchmarking loopt weliswaar al in een docker omgeving maar is monolithisch van architectuur. De verbetering bestaat er dus uit om deze monolithische architectuur om te bouwen naar een microarchitectuur die uit meerdere met elkaar communicerende services bestaat. In een eerste stap zullen de gegevens die Araani Analytics Benchmarking nodig heeft voor de berekeningen van de prestaties niet meer uit een excel-tabel maar uit een databank opgehaald worden. In figuur 1 zien we de gegevens in de kolommen van de excel-tabel die naar een databank zullen verplaatst worden.

Text

Description automatically generated

Figuur : gegevens voor de Araani Analytics Benchmarking die naar een databank zullen verplaatst worden.

De videoPath-kolom bevat het pad naar de metadatabestanden over de video met uitgevoerde detectietest op brandontwikkeling. De library-kolom toont voor welk algoritme de test gedaan werd, bv. voor Smoke of voor Flame. De category-kolom verwijst naar het type test dat uitgevoerd werd. De laatste vier kolommen bevatten gegevens over de aangebrachte annotaties op de video.

# Realisatie

## Relationele databank

In een eerste aanpak opteren we voor een relationele databank met volgende tabellen:

* **categories** met velden:
  + id: int, auto increment, primary key
  + description: varchar(100) ,
* **libraries** met velden:
  + id: int, auto increment, primary key,
  + description: varchar(50),
* **videos** met velden
  + id: int, auto increment), primary key
  + path: varchar(255),
  + category\_id: int, foreign key naar **categories**(id)
  + library\_id: int, foreign key naar **libraries**(id)
  + annotation: text.

Het annotation-veld bevat de gegevens van de kolommen gtFlame, gtSmoke, v1.0 en v2.0 uit de excel-tabel in JSON-formaat, bv.:

{

"groundTruth": { "gtFlame": false, "gtSmoke": true},

"version": { "v1.0": false, "v2.0": true }

}

De docker-applicatie bestaat uit volgende images:

* een databank image met videos-, categories- en libraries-tabel: **mariadb**,
* een api image voor het beheer van de gegevens in de tabellen: **araani/video\_api**,
* een applicatie image die de Araani Analytics Benchmarking code bevat: **araani/aab**,
* een image voor het aanmaken en beheer van de databank: **adminer**.

De tabellen in mariadb worden met de adminer service aangemaakt en indien nodig gewijzigd.

Via docker-compose.yml worden drie services in het netwerk opgestart:

* videodb,
* video\_api,
* dashboard.

Het opstarten van de adminer service gebeurt afzonderlijk met het docker-compose.admin bestand.

* De videodb service host de mariadb databank. Deze service is enkel bereikbaar binnen het docker-netwerk op poort 3306.
* De video\_api service zet een verbinding op met de videodb service en host een uvicorn webserver die de FastAPI-applicatie (app) runt. Deze webserver luistert op poort 8000 naar requests naar de databank. De endpoints van de FastAPI-applicatie zijn bereikbaar van buiten het docker-netwerk op poort 8000 en zijn met Swagger gedocumenteerd in JSON formaat op localhost:8000/docs. Met deze documentatiepagina kunnen gegevens in de databank opgevraagd, toegevoegd, gewijzigd of verwijderd worden.

Voor de video-requests zijn twee APIrouter-objecten aangemaakt omdat twee get-requests met dezelfde definitie, namelijk get\_one() and get\_many(), in hetzelfde router-object als gevolg heeft dat altijd de eerste definitie aangeroepen wordt. Hetzelfde geldt voor de twee put-requests update\_path() en update\_annotation().

Bewerkingen op de databankgegevens kunnen ook gedaan worden met de Strawberry GraphQL bibliotheek op localhost:8000/grapql. Door gebruik te maken van deze querytaal kan het aantal endpoints verminderd worden omdat in de query kan aangevinkt worden wat moet opgevraagd worden uit de databank.

Verder heeft deze service de poetry tool aan boord waardoor de app ook kan gedraaid worden buiten de docker-omgeving. Daarvoor moet wel de externe poort van de videodb service terug opengesteld worden.

* De dashbord service zet een verbinding op met de videodb maar maakt geen gebruik van een webserver om gegevens uit de databank te op te halen. Deze service zal geen gegevens wijzigen maar is louter een dataconsumer die via de repository classes de nodige queries naar de databank stuurt.

## Document georiënteerde databank

De metadata in het annotatieveld van de **videos** tabel zijn in JSON-formaat. Dit formaat laat ons toe om deze gegevens vloeiend verder uit te breiden, al of niet in geneste vorm. Om te ondervinden op welke manier het gemakkelijkst met gegevens in JSON-formaat kan gewerkt worden, is in een tweede aanpak geëxperimenteerd met de document georiënteerde mongdb databank.

Het docker-applicatie is daartoe uitgebreid met volgende images:

* Een databank image met videos-, categories- en libraries-tabel: **mongo:6.0**,
* Een image voor het beheer van de gegevens in de databank: **mongo-express:1.0.0**.

Via docker-compose.yml wordt samen met de vorige drie services de **mongodb** service opgestart. Het opstarten van de **mongo-express** service gebeurt samen met de **admirer** service via het docker-compose.admin bestand.

Omdat in de repository classes de queries voor een relationele databank verschillen t.o.v. deze voor een NoSQL databank, wordt in het .env-bestand met de omgevingsvariabele MONGO\_DATABASE\_ON beslist met welke databank het docker-netwerk aan de slag gaat:

* MONGO\_DATABASE\_ON=ON 🡪 mongodb wordt gebruikt
* MONGO\_DATABASE\_ON=OFF 🡪 mariadb wordt gebruikt

Het annotation-veld in het VideoViewModel heeft een string als datatype. Daarom moet een extra omzetting gebeuren voor het afhandelen van een videodocument uit de mongdb in de Swagger docs:

* Bij het opvragen van een video-document uit de databank moet het annotation-veld van het document eerst geserialiseerd worden naar een JSON-formatted string vooraleer een VideoViewModel aan te maken (json.dumps):

A screenshot of a computer

Description automatically generated with medium confidence

* Na het aanmaken van een document vanuit een Video-object moet vooraleer het document in te voegen in de databank het annotation-veld gedeserialiseerd worden naar een python object (json.loads):

A screenshot of a computer

Description automatically generated with medium confidence

# Besluit

* Na het afwegen van de voor-en nadelen zal beslist worden om de applicatie verder te ontwikkelen met mariadb of mongodb. Mongo-express leent er zich gemakkelijker toe om gegevens in JSON-formaat te bewerken. Nadeel is dat het moeilijker is om de data op een correcte manier op te slaan en weer te geven. Na de gemaakte keuze zullen alle gegevens uit het excel-tabel in de databank geplaatst worden.
* Een eerste stap in de verdere ontwikkeling zal zijn het aanpassen van de code in de dashboard service zodat de gegevens die nodig zijn voor het berekenen van de algoritmeprestaties niet langer uit het excel-bestand maar uit de databank komen. Daarvoor is in github al een project aangemaakt.
* In een volgende stap zal de docker-applicatie niet langer op een lokale machine lopen maar op een cloudcomputingplatform, bv. Azure. Daarbij aansluitend zullen ook alle videobestanden en metadatabestanden in de cloud gestockeerd worden.
* Tenslotte wordt ook gedacht aan een service met web GUI die het mogelijk zal maken om de gegevens in de databank op een gebruiksvriendelijke manier te beheren.
* Repository op github:

[BartDewancker/tools\_aabMicro: aab as microservices architecture (github.com)](https://github.com/BartDewancker/tools_aabMicro)