Inhaal opdracht DEV V

Opdracht overzicht

Ontwikkel een open source API, die gebruik maakt van een database. Je maakt een API voor een klant die aan citizen science wil doen. Dit wil zeggen dat een aantal bewoners in Brussel een sensor krijgen die ze thuis kunnen ophangen, en die dus data zal doorsturen naar een centrale API (die jij zal maken). De data moet natuurlijk opgeslagen worden, en opgehaald kunnen worden via een endpoint op een aantal specifieke manieren.

> Dit is een individuele opdracht, samen een paar problemen oplossen is okee, code overnemen of doorsturen niet. Elke vorm van plagiaat zal door de examencommissie afgehandeld worden.

Database tables

Sensors

field	type	default
UUID	uuid	/
name	string	/
measures	String	/
created_at	datetime	default datetime
updated_at	datetime	default datetime

Measurements

field	type	default
UUID	uuid	/
latitude	float	/
longitude	float	/
value	float	/
timestamp	unixtimestamp	/
created_at	datetime	default datetime
updated_at	datetime	default datetime

Nodige endpoints

[POST] /measurement

dit slaat een meting op in de measurements table

Doorgestuurde data:

```
{
    UUID: (aan te maken in de endpoint),
    sensorID: (UUID van de sensor die een meting uitvoerde, verwijst naar de sensors table),
    measuredValue: (float, gemeten waarde van de sensor),
    datetime: (unixtimestamp, tijd van meting),
    latitude: (float, locatie van meting),
    longitude: (float, locatie van meting)
}
```

[POST] /sensor

dit slaat een nieuwe sensor op in de sensors table

Doorgestuurde data:

```
{
    UUID: (aan te maken in de endpoint),
    sensor: (UUID van de sensor die een meting uitvoerde, verwijst naar de sensors table),
    measures: (JSONObject van wat de sensor exact kan meten),
}
```

[GET] /sensors

Geeft alle sensoren weer, met de laatste meting

Verwachte data:

```
[
    {
        sensor: (UUID van de sensor die een meting uitvoerde, verwijst naar de sensors table),
        measures: (String van wat de sensor exact kan meten),
        last: {
            uuid: (UUID van de laatste meting)
            value: (float, laatst gemeten waarde)
            timestamp: (unix timestamp van laatste meting)
    },
    ... (andere sensoren)
]
```

[GET] /sensor/[UUID]

Geeft de laatste 100 metingen weer voor een specifieke sensor

Verwachte data:

```
{
    sensor: (UUID van de sensor die een meting uitvoerde, verwijst naar de sensors table),
    measures: (String van wat de sensor exact kan meten),
    records: (aantal metingen)
    measurements: [
        {
            uuid: (UUID van de laatste meting)
            value: (float, laatst gemeten waarde)
            timestamp: (unix timestamp van laatste meting)
        },
        ... (verdere metingen)
    ]
```

[DELETE] /sensor/[UUID]

Verwijderd de sensor en de metingen hieraan gebonden

Verwacht antwoord:

```
{
    sensor: (UUID van de sensor die een meting uitvoerde)
}
```

Requirements

Tests

- Een volledige end-to-end test
    - Test ook wat of je na verwijdering van een sensor je hier nog data aan toe kan voegen
- Minstens een integration test van elk endpoint
- Een aantal unit tests, waaruit duidelijk blijkt dat je een functie test-driven ontwikkeld hebt.

Git

- Een volledige git flow history, met feature branches
- Duidelijk incrementele veranderingen in de history waaruit af te leiden valt hoe de applicatie tot stand is gekomen.
- Op het einde van de deadline bekijk ik de main branch

Open-source

- Alle belangrijke documenten bevat
    - De readme de endpoints omschrijft, en uitlegt welke data hiernaar gestuurd kan worden
    - De volledige opstart staat omschreven in de readme (.env aanmaken, commando uit te voeren)
    - Een duidelijke contribution guideline, change log, en code of conduct bevat

Algemeen

- Ik verwacht dat de volledige applicatie opstart via docker compose (exclusief aanmaken van .env files)
- Alle code moet volgens conventie geschreven worden
- Duidelijke documentatie
- Juiste variabele namen
- geen restcode die niet meer nodig is

- Ik verwacht dat data persistent is, met andere woorden dat deze ook blijft bestaan indien de service heropgestart wordt in een postgres database
- Indien een sensor niet bestaat, verwacht ik een "400" terug
    - bij verwijderen
    - bij creeren van een nieuwe meting gekoppeld aan de sensor
    - bij ophalen van een sensor
    - Doe dit via een helper functie

Quotering

Deel	waarde
Git flow en commits	3
Open source volledigheid	1
Endpoints	7
Test driven development	5
Conventiegebruik (inline docs, ...)	1
API documentatie	1
Extra functionaliteiten	1
Efficient gebruik van functies en middleware	1

De punten aangegeven in *italic* zijn naargelang de Europese gradingschaal voor uitmuntend en extra werk.

Aangeraden manier van werken

- branch aanmaken en uitchecken
- tests schrijven
- code schrijven om tests te valideren
- code refactoren
- documentatie schrijven (ook readme aanvullen)
- branch mergen (via pullrequest bijvoorbeeld)