Internet of Things advanced

Projectbundel

IT Factory

Groep 6
Vince Druyts, Dries Janssen, Gijs Verreydt,
Thibaud Wagemans, Siebe van Rompay

Academiejaar 2020-2021

Campus Geel, Kleinhoefstraat 4, BE-2440 Geel

INHOUDSTAFEL

INHOUDSTAFEL 3

1 INLEIDING 4

2 PLAN VAN AANPAK 5

2.1 Aanleiding en achtergrond 5
2.2 Verwacht resultaat 5
2.3 Business case en doelgroepen 6
2.4 Planning 6
2.5 Informatie en rapportering 7
2.6 Projectteam 7

3 PROOF OF CONCEPT 8

3.1 GPS module 8
3.2 Ultrasoon sensor 11
3.3 Database 13
3.4 Visualisatie 15
3.5 Alert 18

4 USECASEDIAGRAM 21

5 KOSTENANALYSE 23

6 BESLUIT 24

1 INLEIDING

Wij, groep 6, hebben een project uitgewerkt en gerealiseerd voor het bedrijf Coeck NV. Coeck NV is een familiale onderneming die een breed assortiment kwaliteitsproducten produceert. Zij distribueren ook ruwbouw en afwerkingsmaterialen. Zij hadden het probleem dat hun ophaalrondes niet optimaal waren en dat A-frames eventueel konden verdwijnen. Bovendien was er ook geen overzicht van alle informatie over de A-Frames.

Met deze informatie zijn wij aan de slag gegaan om een systeem uit te werken dat de locatie van de A-frames toont en ook of deze geladen zijn of niet. De locaties worden simpelweg getoond op een kaart met daarbij de info of het A-frame geladen is.
Met onze resultaten van dit project willen we Coeck NV een simpele en efficiënte oplossing aanbieden om hun huidige manier van A-frames te beveiligen en op te halen.

2 PLAN VAN AANPAK

2.1 Aanleiding en achtergrond

- Betonfabriek Coeck NV is een familiale onderneming uit Niel opgericht in 1929. Het bedrijf is toe aan 4 generaties ervaring in de vervaardiging van betonproducten.
    - De onderneming produceert een breed assortiment kwaliteitsproducten waarin beton het hoofdbestanddeel vormt. Anderzijds distribueren ze een ruim aanbod van ruwbouw- en afwerkingsmaterialen.

- Processen/onderdelen

    - Identificatie van de verschillende A-Frames (serienummer)
    - Lokalisatie A-frame
    - Aantal A-frames in bezit
    - Ophaalrondes optimaliseren
    - Administratie waarborg
    - Detectie bij diefstal

- Nadelen/inefficiëntie

    - Ophaalrondes
    - Bezit aantal frames niet 100% gekend
    - A-Frame is duur (+- €600) – diefstal?

2.2 Verwacht resultaat

- Het nut van onze implementatie zou zijn dat Coeck NV volledig overzicht heeft van waar al hun A-frames staan en kan nagaan waar ze overal geweest zijn, dit om te voorkomen dat ze gestolen worden of dat ze te lang op een werf staan terwijl ze leeg zijn.

- Dit project is voor Coeck NV, maar dit project kan nog voor meerdere bedrijven handig zijn. Coeck NV is niet het enige bedrijf met waardevolle spullen die blijven staan op een werf en gestolen kunnen worden.

- Het systeem moet er uiteindelijk voor zorgen dat Coeck NV altijd moet weten waar het A-Frame zich bevindt door de locatie te visualiseren. Als de locatie veranderd moet het systeem een alarmering sturen. Daarnaast moet Coeck NV ook kunnen zien of het A-Frame geladen is met tegels en als de één (of meerdere) A-Frames klaar voor ophaal zijn moet ons systeem ook een alarmering sturen met “Klaar voor ophaal”.

2.3 Business case en doelgroepen

- Het nut voor Coeck NV is dat ze een mogelijkheid hebben om op elke moment de locatie van hun A-Frame te kunnen bekijken, dit ook om diefstal tegen te gaan en gewoon om een mooi overzicht te creëren van welke A-Frame op welke locatie te vinden is (aan de hand van serienummer).
  Daarnaast willen ze ook de mogelijkheid om de status van het A-Frame te controleren om te zien of deze al dan niet klaar is om op te halen.
  Dit alles wilt Coeck NV zien als iets automatisch (automatisch alarmeringen versturen).

- Het project is bestemd voor medewerkers van de administratie van Coeck NV zodat zij de locatie kunnen kennen op elke moment. Daarnaast zullen ook de systeembeheerders een gebruiksvriendelijke applicatie moeten hebben.

- Daarnaast hebben we als passieve gebruiker ook nog de klanten van Coeck NV als doelgroep. Het aspect dat de klant een alarmering ontvangt wanneer er iets veranderd aan het A-Frame en op de hoogte moet gebracht worden van zijn bestelling en de bijhorende details.
  Het is altijd handig als je als klant ook weet waar je bestelling zich bevindt, want de klant heeft namelijk ook een zekere verantwoordelijkheid! Ook de administratieve mederwerkers van Coeck NV en de systeembeheerder zijn mogelijke doelgroepen.

2.4 Planning

- Allereerst moeten we de GPS-module en de ultrasoon-sensor werkende zien te krijgen dat deze correcte data doorsturen. Daarna moeten we een werkende database zien te verkrijgen zodat onze data correct kan worden opgeslagen. Als laatste stap moeten we de database zien uit te lezen en zorgen dat de data op een gebruiksvriendelijke manier wordt weergegeven (easy to use).

- Als taakverdeling hebben we er voor gekozen om altijd afzonderlijk iemand voor de GPS, ultrasoon sensor, database en visualisatie (hulp van anderen was beschikbaar indien nodig) in te zetten.

- Er was ook een stricte timing gemaakt om elke week onze vooruitgang te bespreken in teamverband.

2.5 Informatie en rapportering

- We hebben eerst een infosessie gekregen over het project voor Coeck NV. Daarna hebben we online gesprekken gehouden met onze opdrachtgever zodat we onduidelijkheden konden oplossen.

- Ook tijdens deze gesprekken hebben wij onze vooruitgang gedeeld, zodat de opdrachtgever het project kon opvolgen.

2.6 Projectteam

Vince heeft de rol van materiaalverantwoordelijke gekregen in het team en heeft daarnaast de GPS module werkende gekregen plus bijhorende code.

Dries heeft de ultrasone sensor werkende gekregen plus bijhorende code en heeft de usecasediagram opgesteld.

Thibaud en Siebe hebben zich gebogen over de database en hoe we de data gaan ontvangen.

Gijs heeft zich gebogen over de visualisatie aan de hand van Grafana.

Als er zich problemen vormden hebben we als team geholpen bij de rest.

3 PROOF OF CONCEPT

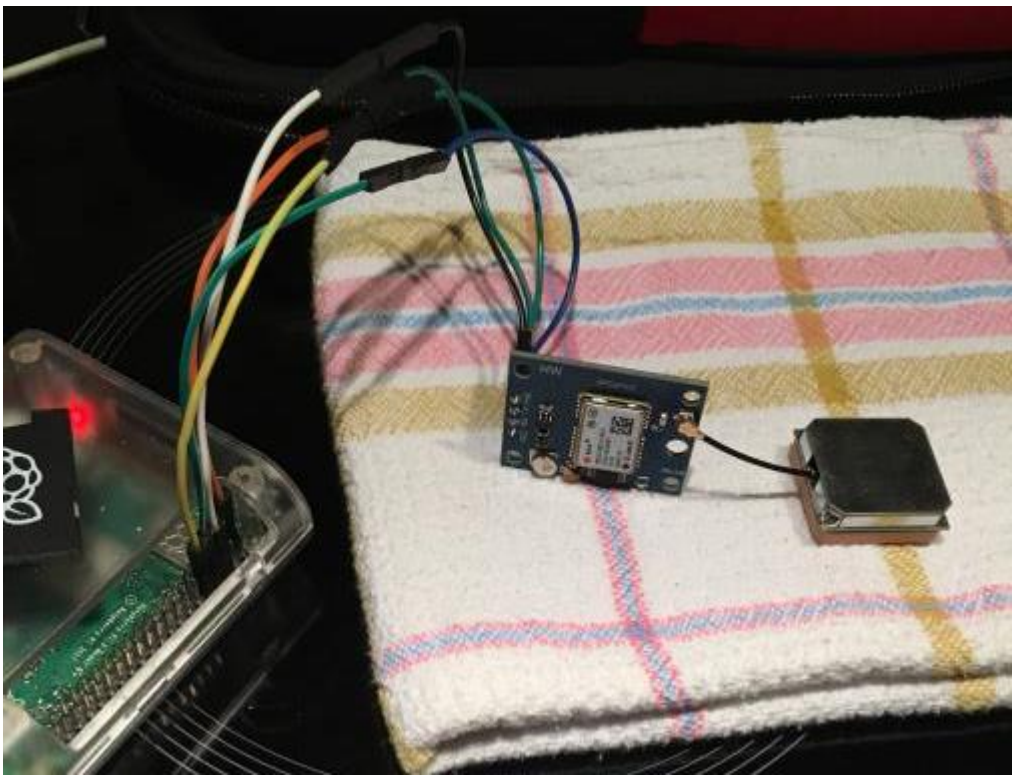
Na het concept en de details gehoord te hebben van betonfabriek Coeck, zijn wij begonnen aan een Proof of Concept om dit kleinschalig uit te voeren. In deze sectie ziet u een verduidelijking van de gebruikte technologieën.

3.1 GPS module

Als onderdeel van onze Proof of Concept hebben wij gebruik gemaakt van een GPS-module (in dit geval NEO-6M) en deze aangesloten op de Raspberry Pi.

Deze module hebben wij gebruikt om GPS data te verkrijgen via een aangekoppelde satelliet. Deze component zou uiteindelijk op een robuuste manier op het A-Frame gemonteerd worden. De GPS module zal voortdurend geografische coördinaten blijven opvragen. Op die manier weten we op elke mogelijke moment waar dit A-Frame zich bevindt.

Later zullen we deze data gebruiken om allerlei dingen mee te realiseren. Hier zie je hoe de aansluiting er zou kunnen uitzien.



Na de aansluiting zijn we begonnen met het schrijven van de Python code die ons zal helpen om ons doel te realiseren.

Deze Python code zal er simpelweg voor zorgen dat de opgevangen coördinaten geprint worden (en nog wat bijhorende informatie over de locatie), zodat we met deze coördinaten in een later stadium verder kunnen werken. Nu zullen we de belangrijkste onderdelen van de Python code kort bespreken.

In onderstaande code definiëren we enkele variabelen zoals bijvoorbeeld “ser” waar we een tty terminal linken aan een seriële poort (GPIO x) en enkele anderen die we later zullen gebruiken.

```
# De tty aangeven waarop de seriele data binnenkomt
ser = serial.Serial ('/dev/ttyAMA0', 9600, timeout=1)
gpgga_info = '$GPGGA,'
GPGGA_buffer = 0
NMEA_buff = 0
GPGGA_data_available = ''
longi = ''
lat = ''
```

In dit deeltje komt weliswaar een van de belangrijkste onderdelen van het script, de PubNub configuratie. Wij hebben gebruik gemaakt van PubNub om onze data door te sturen over een centrale channel (in ons geval “frameData”). Het belangrijkste onderdeel is om de subscribe- en publish key juist te configureren.

```
# Pubnub configuratie
pnconfig = PNConfiguration()
pnconfig.subscribe_key = ""
pnconfig.publish_key = ""
pubnub = PubNub(pnconfig)
pnconfig.uuid = "theClientUUID"
```

Hier zeggen we dat het script moet blijven runnen zolang de seriële poort open is. We gaan ook de seriële poort even sluiten en weer openen en dan lezen met het “ser.read(200)” commando.

```
while ser.is_open:
    # ser.flush()
    ser.close()
    ser.open()
    received_data = (str)(ser.read(200))
    GPGGA_data_available = received_data.find(gpgga_info)
    if (GPGGA_data_available>0):
```

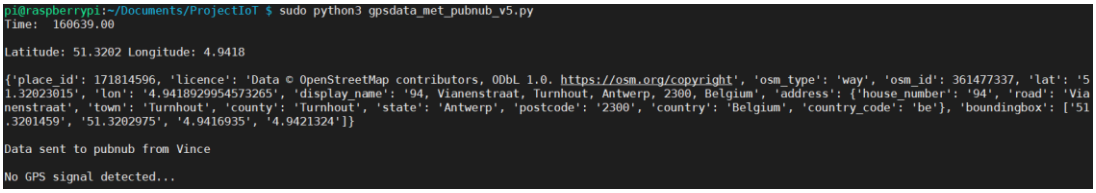
Dit stukje zegt dat we onze latitude en longitude gaan opvullen en gaan printen.

```
lat = (float)(nmea_latitude_new.lstrip('0').rstrip("'"))
lat = convert_to_degrees(lat)
longi = (float)(nmea_longitude_new.lstrip('0').rstrip("'"))
longi = convert_to_degrees(longi)
print ('Latitude:', lat,'Longitude:', longi,'\n')
```

Last but not least gaan we doen aan “reverse geocoding”, wat wil zeggen dat we van ruwe coördinaten naar straatnaam, stad, … gaan om af te printen (enkel voor een visueel beeld te krijgen van waar we zitten). Om te eindigen gaan we dan publishen naar het PubNub channel “frameData”.

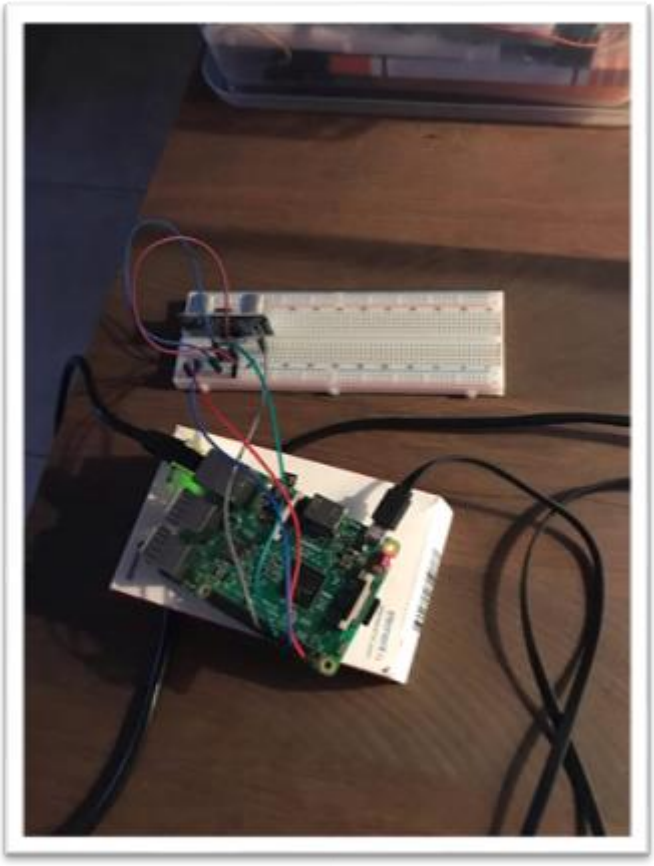
```
# Reverse geocoding
app = Nominatim(user_agent="project")

# Adresinfo opvragen
address = get_address_by_location(lat, longi)
# Print de data
print(address)

message = [lat, longi]
pubnub.publish().channel("frameData").message(message).sync()
```

Op deze foto kan je zien dat de tijd, latitude en longitude worden geprint en de bijhorende straatnaam, stad, … door reverse geocoding. De PubNub publish werkt zoals we willen en wanneer de GPS module even geen signaal krijgt, krijgen we de melding “No GPS signal detected”.



3.2 Ultrasoon sensor

Eén van de vereisten van dit project was om gebruik te maken van een ultrasoon sensor. Deze sensor wordt in de praktijk gemonteerd op een A-frame. Bij een correcte configuratie van deze ultrasoon sensor is het mogelijk om te zien of een A-frame geladen is of juist niet geladen is.
Dit is zeer handig om te weten aangezien je hier uiteindelijk ook alerts/meldingen kan aan gaan koppelen.



Bovenstaande afbeelding toont de opstelling van de sensor die werd gebruikt voor onze Proof of Concept. Een A-frame kan ofwel geladen zijn wanneer de ultrasoon detecteert dat er bijvoorbeeld nog tegels voor de sensor staan ofwel niet geladen wanneer de sensor een afstand buiten zijn ingestelde waarde meet.

Het is zo geconfigureerd dat de sensor gaat zeggen dat een A-frame geladen is wanneer de afstand kleiner is dan **100cm,** dit is flexibel ingesteld dus kan je altijd nog veranderen.

```
# Control if A-frame has a load

hasLoadCheck = hasLoad

if distance >= 100:
        hasLoad = False
        #message = "Load: " + str(hasLoad)
        status = "Niet geladen"
else:
        hasLoad = True
        #message = "Load: " + str(hasLoad)
        status = "Geladen"

print(str(frameSerialNumber) + " - distance: %.1f cm" % distance + "    -  " + str(status))

if hasLoadCheck != hasLoad:
        distance1 = str(distance)
        Sensor = [status,distance1]
        print(Sensor)
        pubnub.publish().channel("frameData").message(Sensor).sync()
        print("Data sent to pubnub channel.")

time.sleep(10)
```

Wanneer de sensor een waarde meet die groter of gelijk is aan 100 cm gaan we er vanuit dat de A-frame niet geladen meer is aangezien het object dat voor de sensor staat, is weggehaald.

In bovenstaande foto kan je ook zien hoe we de variabelen distance en status gaan doorsturen over een pubnub channel genaamd “framedata”. Deze pubnub publish gebeurt wel alleen als de toestand van de sensor veranderd. Dus wanneer de status van false naar true of van true naar false gaat.

Via Pubnub kan je aan de hand van een publish key gemakkelijk data versturen naar een bepaald channel. In onze Proof of Concept gaan de ultrasoon sensor en de GPS beide hun data versturen aan de hand van dezelfde publish key en channel naam.
Onze PI die we gebruiken voor de database kan aan de hand van de corresponderende subscribe key aan deze data geraken en wegschrijven in de database

```
pi@raspberrypi:~/Documents/Project_IOT $ sudo python sensorPublishWorks.py
fwGT2CQw - distance: 16.2 cm    -  Geladen
fwGT2CQw - distance: 207.0 cm    -  Niet geladen
['Niet geladen', '206.999480724']
Data sent to pubnub channel.
fwGT2CQw - distance: 206.3 cm    -  Niet geladen
fwGT2CQw - distance: 151.8 cm    -  Niet geladen
fwGT2CQw - distance: 7.1 cm    -  Geladen
['Geladen', '7.06558227539']
Data sent to pubnub channel.
fwGT2CQw - distance: 28.2 cm    -  Geladen
fwGT2CQw - distance: 41.9 cm    -  Geladen
^C
Turning off ultrasonic distance detection...

pi@raspberrypi:~/Documents/Project_IOT $
```

Bovenstaande foto toont de werking van dit pubnub channel aan. Deze gaat gebruikt worden om sensordata door te sturen, waarbij de nodige variabelen distance en status worden doorgestuurd over het channel ‘framedata’.

3.3 Database

In ons project maken wij gebruik van een MariaDB database. Deze gebruiken wij om alle data over de A-frames ordelijk op te slaan. Hiermee kan je snel het hele overzicht van A-frames bekijken. Ook gebruiken we de database om de visualisatie mogelijk te maken.
We stellen de tabel van de database als volgt in. (3.3.1) Hierdoor kan de database het FrameID opslaan samen met de serienummer van het A-Frame, de locatie van het A-frame in longitude en latitude, of het frame al dan niet geladen is en de bijhorende datum en tijd.

#	Naam	Type
1	**FrameID**	int(11)
2	**SerieNummer**	text
3	**isGeladen**	tinyint(1)
4	**Longitude**	text
5	**Latitude**	text
6	**Datum**	date

3.3.1: Tabel opstelling

In ons programma importen we eerst alle belangrijke modules en stellen we de PubNub configuraties in. We verbinden dan met onze database en zetten ook de cursor voor in onze database klaar. (3.3.2)
Daarna stellen we een class in om de verkregen data via PubNub van de sensor en gps op te slaan in onze database. Deze class slaat elke verkregen waarde van de sensor en gps op in de juiste plaats in de database samen met de tijd. (3.3.3)
Het laatste deel van de code zorgt ervoor dat er continu naar inkomende waarde van de sensor en gps word geluisterd. Die dan kan ingelezen worden in de database. (3.3.4)

```
import os
import time
import sys
import datetime
import pubnub
import smtplib
from email.mime.text import MIMEText

import MySQLdb

from pubnub.enums import PNStatusCategory
from pubnub.callbacks import SubscribeCallback
from pubnub.pnconfiguration import PNConfiguration
from pubnub.pubnub import PubNub

# pubnub publish/subscribe keys
pnconfig = PNConfiguration()
pnconfig.publish_key = "pub-c-5cdfe9d4-4827-4d33-9604-a3ace2729f12"
pnconfig.subscribe_key = "sub-c-ecefcaa8-29bc-11eb-ae78-c6faad964e01"
pubnub = PubNub(pnconfig)

CHANNEL = "frameData"

database = MySQLdb.connect(host="localhost", user="pi", passwd="raspberry",db="IoTProject")
cursor = database.cursor()
```

3.3.2: Code Database, Belangrijke imports en configuraties

```
class MySubscribeCallbackITF(SubscribeCallback):

    def message (self, pubnub, message):

        SerieNummer = "000001"
        Longitude= ""
        Latitude = ""
        isGeladen = ""
        Tijdstip = datetime.datetime.now()

        if "Geladen" in message.message or "Niet geladen" in message.message:

            isGeladen = message.message[0]
            Afstand = float(message.message[1])

            print(isGeladen)
            print(Afstand)

            if "Niet geladen" in isGeladen:
                #email prep
                message = "Frame met serienummer:" + SerieNummer + " leeg en klaar om opgehaald te worden."
                msg = MIMEText(message)
                msg['Subject'] = 'Frame is leeg'
                msg ['From'] = 'grafanaalertitfg6@gmail.com'
                msg['To'] = 'VerreydtG@gmail.com'

                #send email
                username = 'grafanaalertitfg6@gmail.com'
                password = 'IoTProject'
                server = smtplib.SMTP('smtp.gmail.com:587')
                server.starttls()
                server.login(username,password)
                server.sendmail(msg['From'], msg['To'], msg.as_string())
                #server.quit()
                print("Status Alert Gestuurd")

            cursor.execute("INSERT INTO SENSOR(SerieNummer,isGeladen,Afstand,Tijdstip) VALUES(%s, %s, %s, %s)",(SerieNummer,isGeladen,Afstand,Tijdstip))
            database.commit()

        else:

            Latitude = message.message[0].lstrip("u")
            Longitude = message.message[1].lstrip("u")

            print(Latitude)
            print(Longitude)
            print(Latitude_temp)
            print(Longitude_temp)

            cursor.execute("INSERT INTO GPS(SerieNummer,Longitude,Latitude,Tijdstip) VALUES(%s, %s, %s, %s)",(SerieNummer,Longitude,Latitude,Tijdstip))
            database.commit()

            Latitude_temp = float(Latitude)
            Longitude_temp = float(Longitude)

    def presence(self, pubnub, event):
            print("[PRESENCE: {}]".format(event.event))
            print("uuid: {}, channel: {}".format(event.uuid, event.channel))

    def status(self, pubnub, event):
            if event.category == PNStatusCategory.PNConnectedCategory:
                    print("[STATUS: PNConnectedCategory]")
                    print("connected to channels: " + CHANNEL)
```

3.3.3: Code Database, Class om de data die binnegehaald door PubNub te verwerken

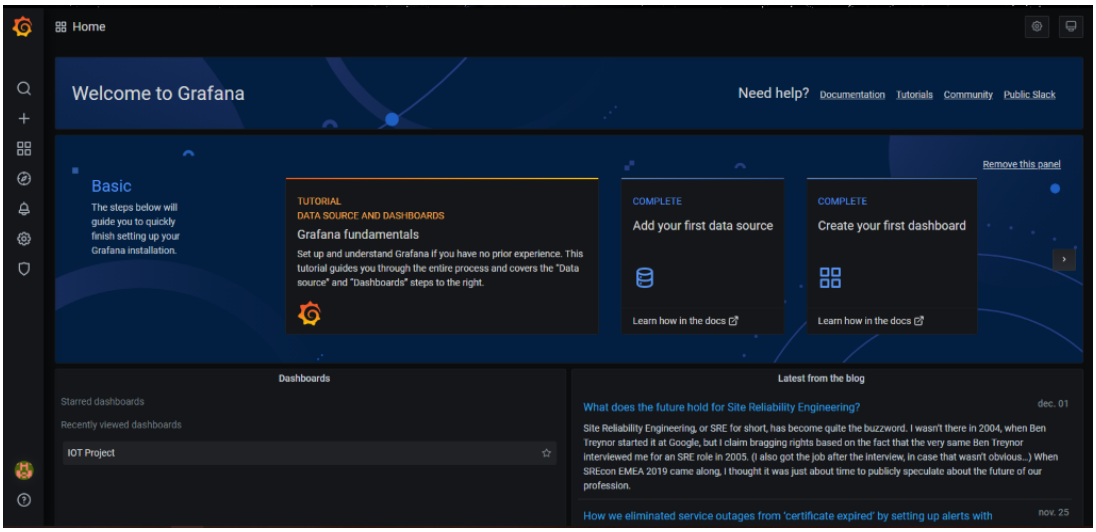
```
90
91  print('Listening...')
92
93  pubnub.add_listener(MySubscribeCallbackITF())
94  pubnub.subscribe().channels(CHANNEL).execute()
95
96
97
```

3.3.4: Code Database, Gegevens binnen halen via PubNub channel

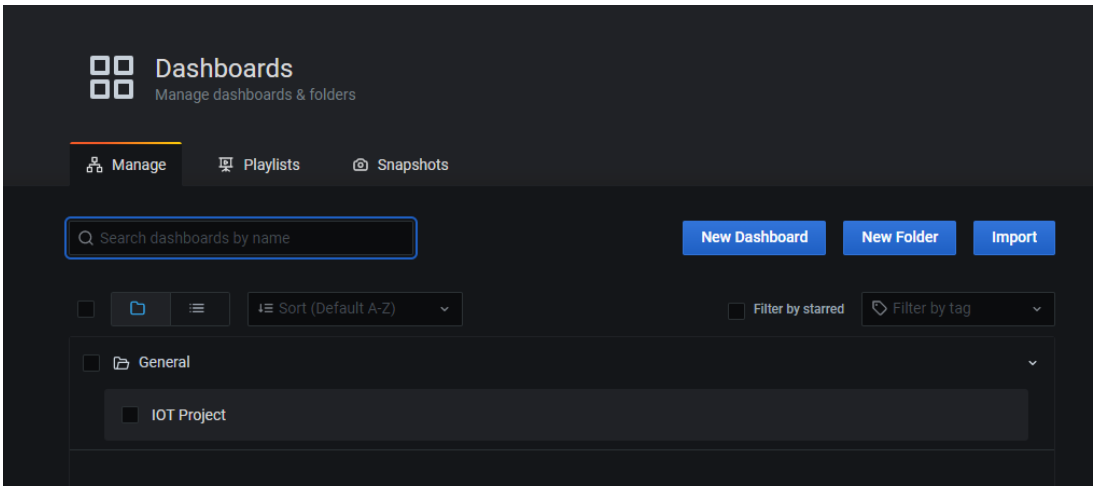
3.4 Visualisatie

Voor onze visualisatie hebben wij gekozen om een Grafana dashboard te gebruiken. Grafana is in staat om met een heel breed aanbod van databases te connecteren en om de data op verschillende manieren te verwerken en te visualiseren. Er is ook vrij veel documentatie over de verschillende toepassingen en hoe men deze kan gebruiken. Ook zijn er community forums waar gebruikers problemen of zelf gemaakte dashboards kunnen delen.

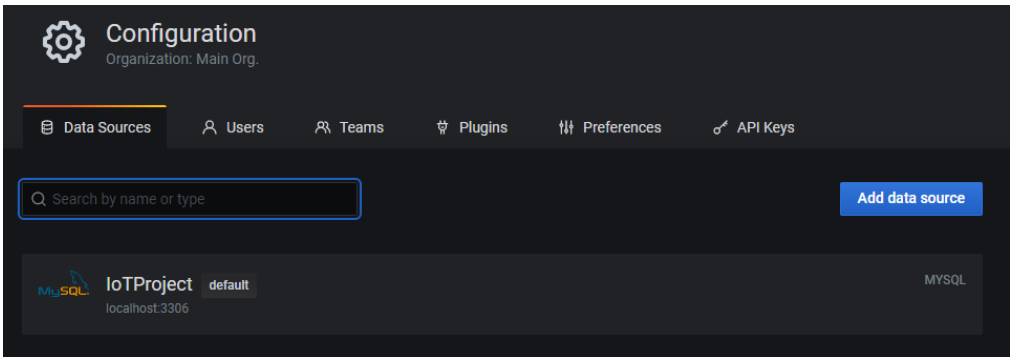
als men Grafana geïnstalleerd heeft en men is ingelogd op de Grafana server pagina komt men op de 'home' pagina.



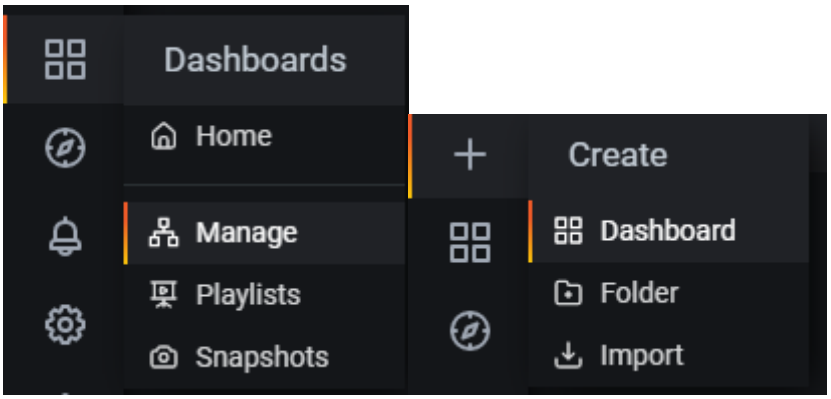
Vanuit hier kan men alle toepassingen gebruiken door te navigeren via de toolbar links. Belangrijkste opties zijn 'dashboards', 'alerting' en 'configuration'.
In dashboards kan men nieuwe dashboards aanmaken en bestaande dashboards aanpassen.



In configuration kan men o.a. users aanmaken, plugins beheren en data sources beheren. Het is belangrijk dat men eerst en vooral een datasource toevoegd vooraleer men een dashboard kan maken.

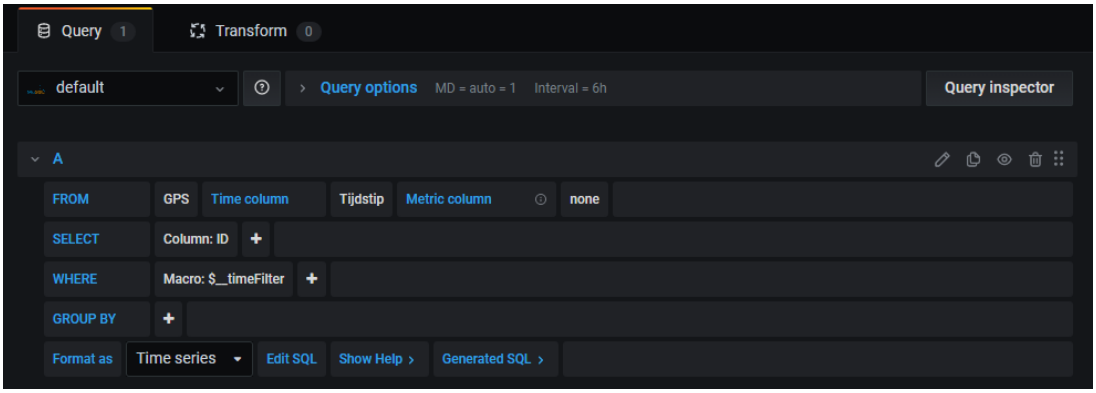


Nadat de datasource is toegevoegd en bereikbaar is (er is een test functie bij het toevoegen van de data source) kan men een dashboard maken. Dit kan door ofwel op het + icoontje te gaan staan en op dashboard te klikken ofwel door naar de manage dashboards pagina te gaan.



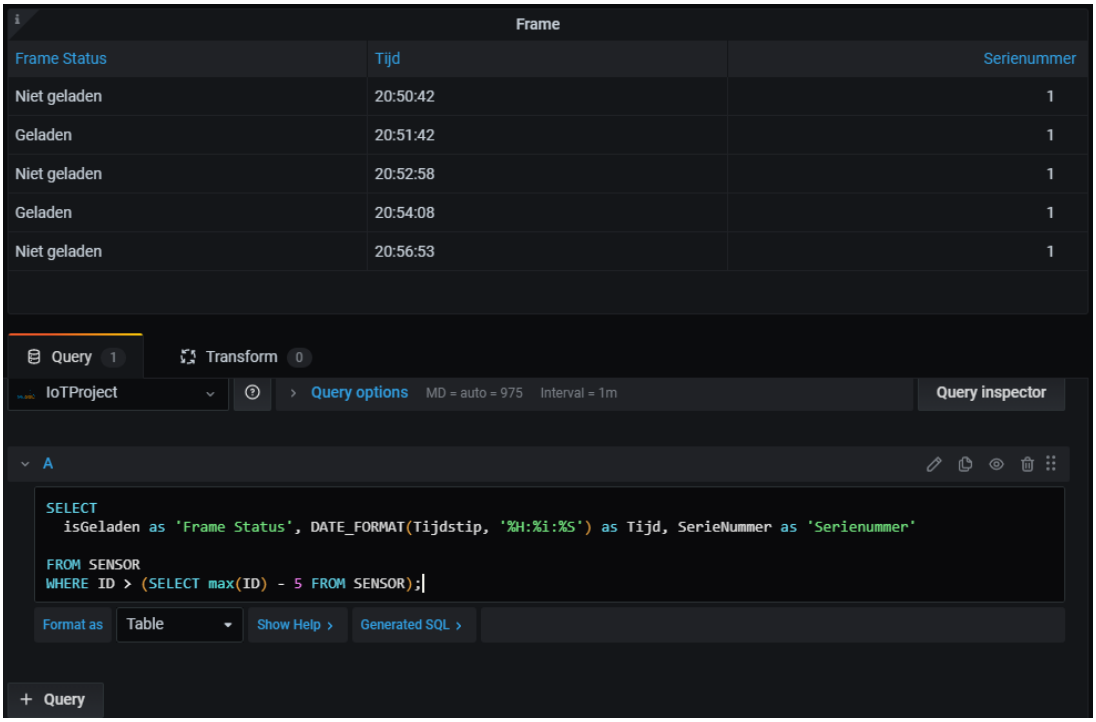
Op de manage dashboard pagina moet men dan op ‘new dashboard klikken’.
het nieuwe dashboard is natuurlijk leeg, dus men zal een panel moeten toevoegen.
Grafana komt standaard met enkele zeer nuttige panels maar er is ook een heel groot aanbod aan plugin panels, worldmap panel is zo een plugin.

Als men het panel heeft gekozen moet men dit panel gaan configureren. Voor onze toepassing gebruiken we worldmap panel. Hier moeten we de data locatie op ‘table’ zetten aangezien we met een MySQL database werken. En omdat onze gps coördinaten doorstuurd moeten we ‘table query format’ op ‘coordinates’ zetten. We kunnen ook nog andere dingen aanpassen zoals ‘initial zoom’, dit past men best aan wanneer de toepassing geen coördinaten over de hele wereld weergeeft maar enkel een kleinere regio. Wij hebben het voor ons op de coördinaten van Brussel gezet.
Vervolgens moet men zeggen welke data we willen visualiseren dit doen we in het ‘Query’ tab.



Men kan queries maken met pure SQL code of met block code zoals hierboven op de foto, als men pure SQL code wilt gebruiken klik dan op de ‘edit SQL’ knop. Het is belangrijk dat men de data source en de ‘Format as’ goed instelt anders zal de query niet werken.

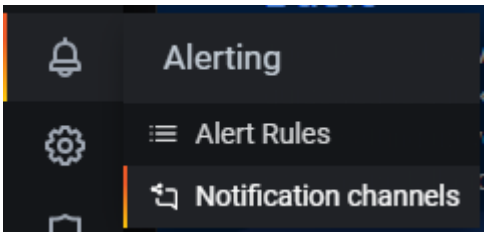
Een ander panel dat we gebruiken is de ‘table’ dit is eigenlijk een simpele tabel waar men data uit een data source kan laten zien in tabelvorm. Het enige wat men hier moet aanpassen is de Query. Men moet weeral de data source goed zetten (als de data source niet als default is ingesteld) en het ‘Format as’ goed zetten. Nu kan men gewoon de query invullen en de opgevraagde data zal in de table verschijnen.



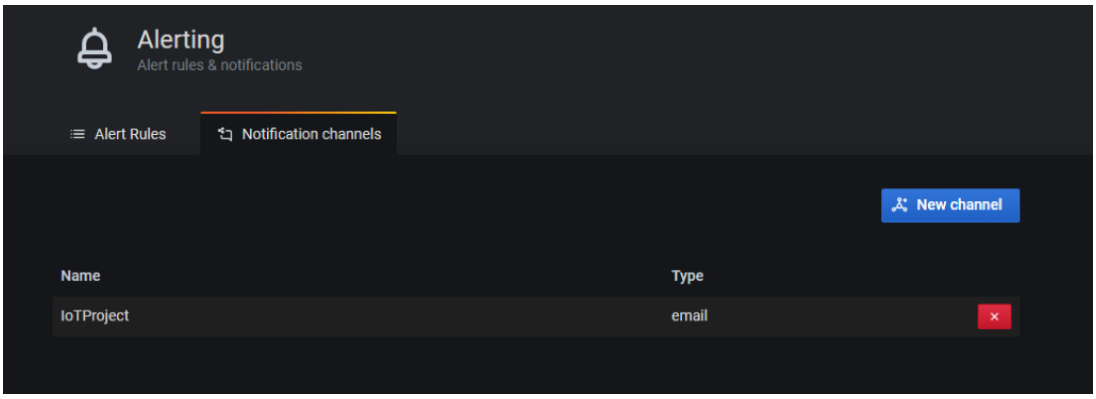
Hert is ook mogelijk om meerdere queries te schrijven voor één panel, men kan dan tussen de verschillende queries kiezen op het dashboard zelf.

3.5 Alert

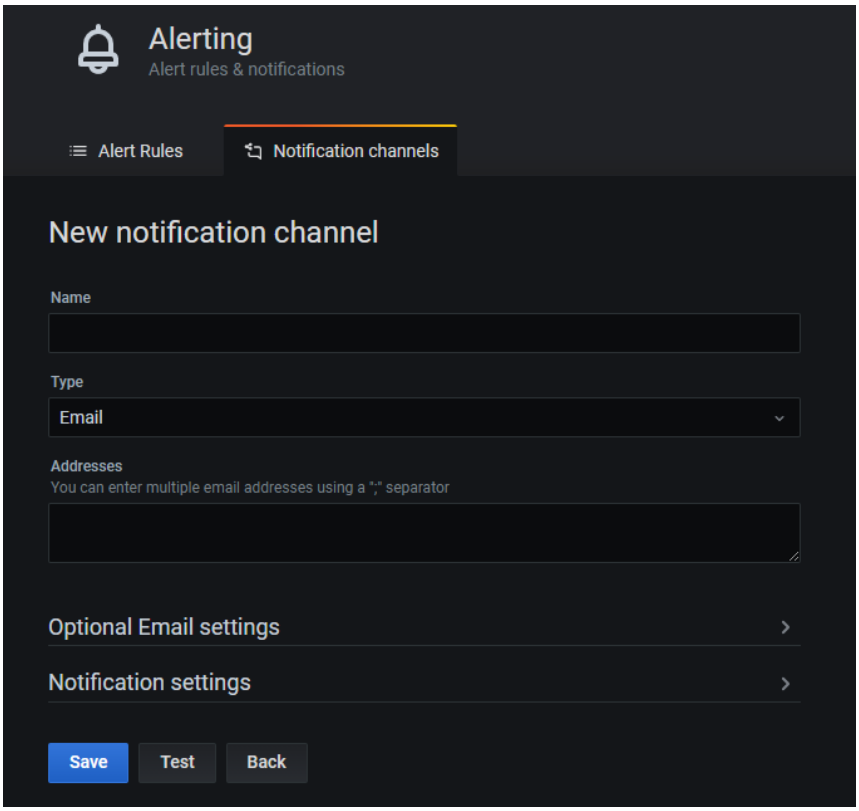
Het eerste wat men moet doen als men een alert wil instellen is een ‘notification channel’ opstellen. Dit is de manier waarop de alert naar de gebruiker word gestuurd.



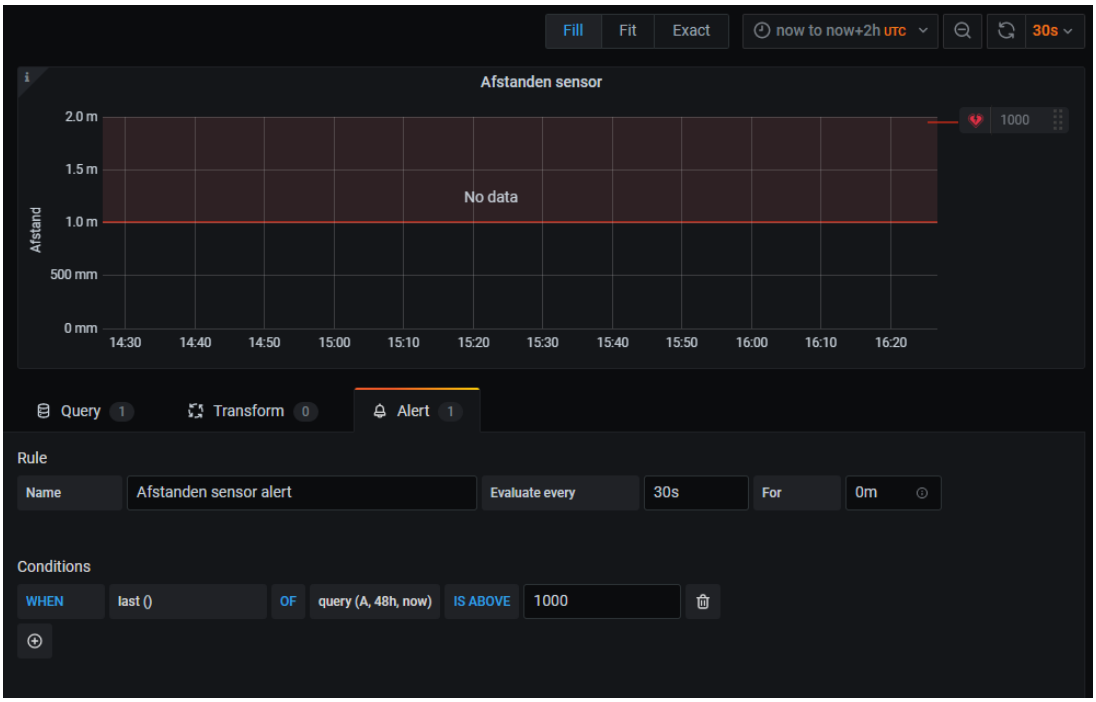
Op deze pagina kan men, net zoals bij de data sources, een nieuw channel aanmaken en bestaande channels beheren.



Een nieuw notification channel heeft een naam nodig en een type van notification, dit kan gaan van webhooks tot e-mails tot Discord notifications.

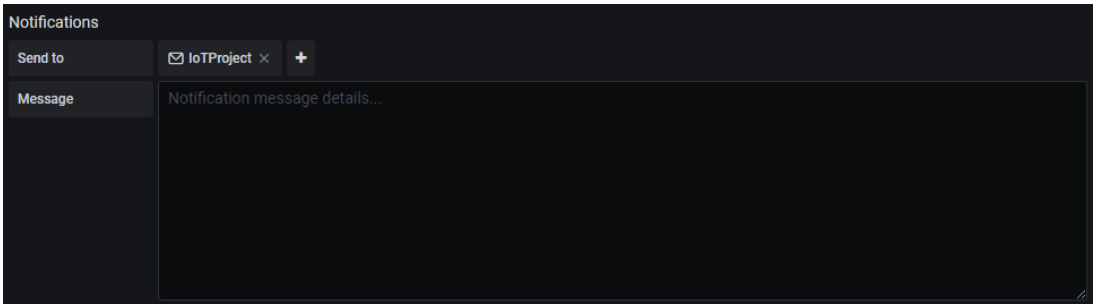


Om email te gebruiken moet je ook wel de SMTP settings van Grafana instellen. Het heeft onder andere een email nodig van waaruit je de mails gaat versturen.
Als je dit gedaan hebt kan je in je graph panel een alert rule en condition opstellen.



In dit voorbeeld gaat de condition kijken naar de uitput van ‘query A’, wanneer de laatste waarde van 48u geleden tot nu hoger is dan 1000 gaat het alert actief. Het is mogelijk om met een ‘AND’ en een ‘OF’ condition meerder conditions aan elkaar te hangen.

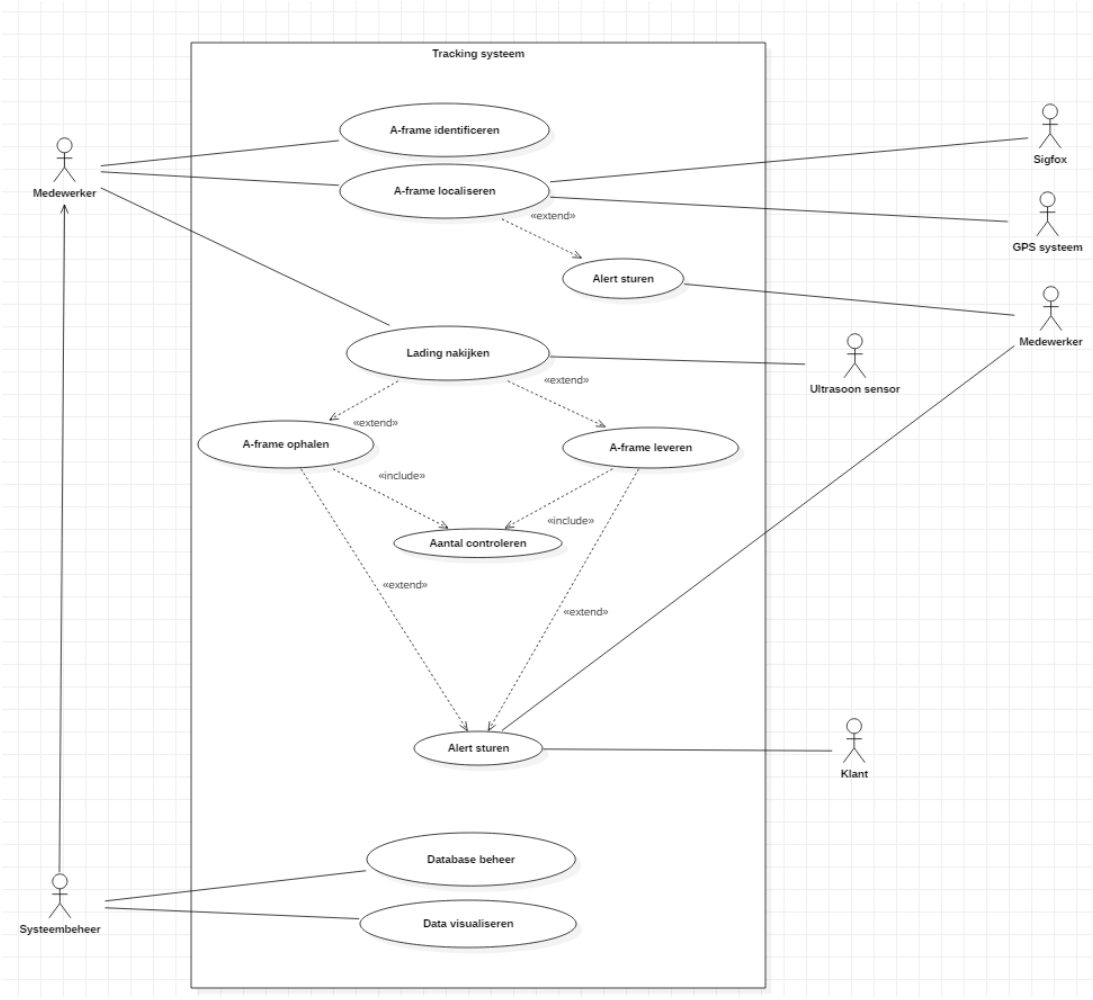
In de rule kan je de naam van het alert ingeven alsook wanneer het hierop moet checken (elke 30s) alsook voor hoelang het alert actief moet zijn eer er effectief een bericht moet gestuurd worden (‘For’) wat in dit geval leeg gelaten is.
Vervolgens kan men de inhoud van het alert bericht ingeven en over welk ‘notification channel’ het verstuurd moet worden.



Nu zal Grafana via het ingestelde email een mail sturen telkens wanneer de laatste waarde uit ‘Query A’ boven 1000 is.

4 USECASEDIAGRAM

Er is ook een usecasediagram gemaakt om de requirements van ons systeem in kaart te brengen.



Een **Medewerker** kan ons Tracking systeem gebruiken om een A-frame te identificeren (aan de hand van serienumber / frameID). Ook bestaat er de mogelijkheid om een A-frame te localiseren. Hierbij wordt gebruik gemaakt van een extern GPS module die in de code verwerkt is. Om data te kunnen transmitten ga je ook beroep doen op een extern netwerk, namelijk Sigfox. Dit is een Low Power, Wide Area Network. Afhankelijk van de locatie van de A-frame kan de Medewerker ook alert aankrijgen.

Een **Medewerker** kan ons systeem ook gebruiken om een lading van een A-frame na te kijken. Dit gebeurt aan de hand van een ultrasoon sensor, gemonteerd op het A-frame. Er zijn 2 mogelijke scenario's bij 'lading nakijken':

- A-frame leveren: Indien de sensor detecteert dat een A-frame geladen is moet de klant een melding kunnen ontvangen met details over het afhalen van de A-frames bij het magazijn of om de klant op de hoogste te brengen van zijn bestelling.

- A-frame ophalen: Indien de sensor detecteert dat een A-frame niet geladen is, dan moet de klant een melding kunnen ontvangen met details over wanneer de lege A-frames opgehaald kunnen worden. Het is cruciaal dat de klant hiervan op de hoogte

wordt gesteld zodat zij niet denken dat de frames gestolen zijn op de werf en dergelijke.

- In beide gevallen moeten we het principe aantal controleren toepassen. Zodat we efficiënt te werk kunnen gaan bij het ophalen en afhalen van A-frames.

Een **Systeembeheerder** kan ons systeem gebruiken met alle machtigingen die een gewone medewerker heeft, maar kan bovenop deze functionaliteiten ons systeem nog gebruiken voor extra zaken. Waaronder het beheren van de database waarin alle data van de GPS, sensor en frame. Ook is er de mogelijkheid om via ons systeem een control panel te beheren met de nodige visualisatie van de data.

5 KOSTENANALYSE

- GPS tracker – Oyster industrial: €74,42 per stuk
  - Wij hebben deze gekozen omdat deze GPS een zeer lange levensuur heeft (7jaar in low power mode) en omdat deze accuraat is.

- Sigfox met max. 100 berichten per dag: €12,00 per jaar per gps unit
  - Dit hebben we gekozen omdat we verwachten dat 100 berichten per dag meer als genoeg is, en dit is een vrij goedkope oplossing (als het nodig is kan je het abonnement nog aanpassen).

- Database hosting – AWS RDS M5 instance: €44,76 per jaar voor de hosting
  - Wij hebben voor de database gekozen AWS omdat dit de goedkoopste is waar ook makkelijk uitbreiding mogelijk is (als dit nodig zou zijn).

- Ultrasone sensor - Schneider Electric XXV18B1PAM12: €129
  - Dit is een zeer goede sensor maar naar ons mening iets te duur.

- Ultrasone sensor - Cylindrical distance sensor UFA-1500
  - Dit is ook nog een goede optie voor de sensor maar hier hebben we helaas de prijs niet van omdat deze op aanvraag is.

6 BESLUIT

In dit project hebben we gezien dat goed teamwork altijd beter is om een doel te bereiken, en niet te vergeten ook veel leuker om zo te werken. Er waren verschillende keren dat we vast zaten of niet goed wisten hoe we iets moesten doen/oplossen, maar dan vroegen we het gewoon aan de rest van het team en kregen zo ons antwoord. In de gevallen dat zij het ook niet wisten gingen we met de groep opzoek naar een oplossing. Vaak hebben we ook lange meetings gehad waar we allerlei dingen deden, afspreken hoe de taakverdeling is, elkaar helpen met problemen, oplossingen zoeken voor dingen die we niet gedaan kregen of niet werkten zoals we wouden. Bepaalde dingen waren makkelijker als anderen maar alles is volgens ons vlot verlopen van samenwerking en opbouw. We moeten ons soms wel iets beter houden aan deadlines die we zelf zetten (bv wanneer we een deel af willen hebben of wanneer we afspreken om een meeting te houden, dit gaat over het algemeen wel vrij goed maar soms lopen we wel eens uit of starten we later).