Professionele Bachelor Toegepaste Informatica

Smart campus

Cyril Knops

Promotoren:

Steven Palmaers
PXL Smart-ICT
Marijke Willems
Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2017-2018

Professionele Bachelor Toegepaste Informatica

Smart campus

Cyril Knops

Promotoren:

Steven Palmaers
PXL Smart-ICT
Marijke Willems
Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2017-2018

Dankwoord

Na een intensief traject van drie jaar is het zover het schrijven van dit dankwoord. Op het eind van elk jaar dacht ik alles geleerd te hebben, maar keer op keer bleek dat niet het geval. Met dank aan alle docenten, stagecollega's, vrienden en ouders.

Eerst en vooral wil ik mijn stagecollega's en promotors van Smart-ICT bedanken voor de fijne samenwerking. Ik wil in het bijzonder stilstaan bij mijn bedrijfspromotors, de heer S. Palmaers en K. Hermans. Ik wil beiden graag bedanken voor de fijne samenwerking en de kansen die ik heb gekregen om mijn stage en onderzoek te mogen afronden bij Smart-ICT.

Daarnaast wil ik mijn schoolpromotor, mevrouw M. Willems, bedanken voor de begeleiding bij dit eindwerk en onderzoek.

Mijn ouders zou ik willen bedanken voor de mogelijkheid om deze opleiding te volbrengen.

Natuurlijk kan ik niet mijn docenten en vrienden vergeten die ervoor gezorgd hebben dat ik sta waar ik nu sta.

Bedankt.

Abstract

In dit eindwerk komen twee problemen ter sprake.

PXL heeft een probleem met overvolle vuilnisbakken. Het campusbeheer moet elke dag elke vuilnisbak controleren om te kijken of deze vol is. Om tijd uit te sparen komt op elke vuilnisbak een knop te zitten waardoor ze in een oogopslag kunnen zien welke vuilnisbakken vol zijn.

Daarnaast zijn er ook een aantal fietsen beschikbaar die door het personeel kunnen uitgeleend worden. Het personeel kan deze fietsen gebruiken om zich te verplaatsen tussen de campussen en bedrijven. Het probleem is dat de fietsen weleens blijven staan waar ze niet thuishoren, zoals op een verkeerde campus. Om dit probleem op te lossen willen ze de fietsen kunnen traceren.

Dit traceren doen we met behulp van een sensormodule op elke fiets. Deze communiceert via Sigfox-radiocommunicatie en de locatie wordt bepaald via wifi-BSSID. Bij de vuilnisbakken wordt enkel de Sigfox radiocommunicatie gebruikt.

1. Wifi BSSID is een unieke 48-bit niet-configureerbare ID die gebruikt wordt als identificatie van een wifinetwerk. Wij gebruiken deze in combinatie met de fysieke locatie van het wifinetwerk om zo de locatie te vinden. [1]

2. Sigfox-communicatie is een Ultra Narrow Band (UNB) radiotechnologie speciaal ontworpen voor Internet of Things (IoT). Deze gebruiken we om een connectie te leggen tussen de fiets en de backend. Deze connectie kan 140 keer per dag worden gemaakt. Dit is een limitatie van het Sigfox-netwerk. [2]

Hierbij hoort ook een applicatie, deze wordt gerealiseerd met de volgende technologieën: voor de backend is dat Node.js en MySQL. De requests vanuit de Sigfox-backend, het opslaan van fietslocaties, vuilnisbak statussen en gebruiksdata wordt in deze backend gedaan. Voor de frontend wordt AngularJS gebruikt. Hier kan de beheerder van de fietsen de locatie van de fietsen zien met behulp van Google Maps, zien welke fietsen nog beschikbaar zijn en wie welke fiets gebruikt op welk moment. De gebruiker kan via dezelfde app zien waar zijn fiets op dit moment is en welke fietsen nog beschikbaar zijn. Bij de vuilnisbakken kan de campusbeheerder de status van elke bak zien en na het legen de status aanpassen.

Bij het onderzoeksaspect wordt een onderscheid gemaakt tussen een aantal grote frameworks, onder andere Node.js, ASP.net en PHP. Hier worden de grote verschillen bekeken en welke van deze platformen het best geschikt is voor Internet of Things.

Inhoudsopgave

Lijst van gebruikte figuren

Lijst van gebruikte tabellen

Lijst van gebruikte afkortingen

- UNB: Ultra Narrow Band
- IoT: Internet of Things
- API: Application Programming Interface
- HTTP: HyperText Transfer Protocol
- JSON: JavaScript Object Notation
- CEO: chief executive officer
- CSO: chief sustainability officer
- BSSID: Basic Service Set Identifier
- RSSI: Received Signal Strength Indicator
- PWO: Praktijkgericht Wetenschappelijk Onderzoek
- SPP: SpeerPunt Projecten
- OOP: Onderwijs Onderzoek Projecten
- VPS: Virtual Private Server
- NFC: Near-Field Communication

Inleiding

Tegenwoordig zie je in steeds meer contexten de term “smart” verschijnen: smartphone, smart home, smart cities, smart campus, … Afhankelijk van de context kan je aan het woord smart een eigen interpretatie geven. Maar wat is een smart city nu precies of wat maakt een campus smart? Waarom willen we dat alles “smart” wordt? Zijn we zelf wel “smart”?

Een smart city, of zoals we in het Nederlands zeggen een “slimme stad” is een stad waarbij informatietechnologie en het internet der dingen (Internet of Things of IoT) gebruikt worden om een stad te beheren en te besturen. Hierbij gaat het zowel om de administratie als om een aantal diensten zoals bibliotheken, ziekenhuizen, transport en nutsvoorzieningen. Het doel van een slimme stad is om de levenskwaliteit te verhogen door de stad efficiënter te organiseren en de afstand tussen de inwoners en het bestuur te verkleinen. Alle onderdelen van de stad zijn daarbij verbonden via een netwerk van sensoren, het internet en hoogstaande technologische apparaten, met als motor het internet der dingen. Bij een smart city gaat het ook vaak om een wijziging in de manier waarop processen geleid en opgevolgd worden, waarbij de mensen meer inspraak krijgen. Het is dus niet noodzakelijk enkel een technologisch verhaal. Dit alles maakt niet alleen een beter bestuur mogelijk, maar laat het bestuur ook toe om de inwoners in de gaten te houden, wat meteen een mogelijke keerzijde is van een smart city.

Vandaag de dag kunnen we veel steden nog niet echt “slim” noemen. Hier en daar worden verkeerslichten en -borden achter de schermen aangepast, om het verkeer beter te laten lopen of geven sensoren wat data door, maar verder dan dat gaat het vaak (nog) niet. Vaak werkt een systeem nog niet echt efficiënt wegens een gebrek aan intelligentie.

Hoe zou dat dan wel moeten werken? In eerste instantie zou het netwerk van sensoren veel groter moeten zijn. Er zouden meerdere sensoren moeten zijn die in elke publieke ruimte metingen doen zoals verkeersdrukte, waterniveau, positie, … gelinkt aan tijden van het openbaar vervoer of de beschikbaarheid van parkeerplaatsen. Een verkeersbord of -licht zou regelbaar moeten zijn en data zou beschikbaar moeten zijn voor iedereen, zodat er nieuwe producten en/of diensten kunnen gebouwd worden, die ervoor zorgen dat een stad echt “smart” wordt. [3]

In dit eindwerk gaat er vooral gekeken worden naar de smart campus. Welke aspecten maken de campus nog meer leefbaar? Er wordt concreet aan de slag gegaan met twee problemen, waarbij er ook een mogelijke oplossing wordt uitgewerkt. Enerzijds gaat het om mobiliteit, anderzijds om afval.

Wanneer meerdere bedrijven zich op een campus bevinden, wordt het al snel een logistieke nachtmerrie. Vooral afval wordt al snel een probleem voor de ondersteunende diensten. Het is niet altijd evident om elke dag een ronde te maken om na te gaan welke vuilnisbakken moeten leeggemaakt worden. Dit vergt veel kostbare tijd en zorgt soms voor frustraties. Als er bijvoorbeeld een evenement is zouden de vuilnisbakken vaker leeggemaakt moeten worden. Het doel van mijn eindwerk is om hier een oplossing voor aan te reiken, door sensoren of knoppen te plaatsen op elke vuilnisbak, zodat het personeel precies kan zien welke vuilnisbakken moeten leeggemaakt worden en het proces geoptimaliseerd kan worden. Tevens kan er op basis van de beschikbare informatie een optimale route gepland worden, om de ophalen zo efficiënt mogelijk te organiseren.

Een tweede probleem is mobiliteit. Als men een korte verplaatsing moet maken voor een meeting, lunch of korte boodschap in de buurt is het vaak interessant om een fiets te nemen, aangezien de meeste campussen zich vlakbij de stad bevinden. Een aantal campussen, waaronder die van de PXL hebben daarom campusfietsen ter beschikking, die personeelsleden ad hoc kunnen uitlenen. Momenteel is de praktische invulling hiervan echter niet optimaal. Men kan immers niet op voorhand zien of er een fiets beschikbaar is, er is geen gedetailleerd zicht op het gebruik van de fietsen, het onderhoud wordt niet uitgevoerd op basis van effectief gebruik maar op een vaste datum. Hierdoor zijn heel wat fietsen kapot of worden ze gerepareerd op een drukke dag. Het is ook niet evident om te achterhalen waar een fiets zich bevindt, na misbruik of diefstal. Om dit probleem op te lossen zal er op elke fiets een tracker geplaatst worden en daarnaast wordt ook bekeken of de sleutelkluisjes slim gemaakt kunnen worden. Hierdoor zal men op termijn inzicht krijgen op het profiel van de gebruiker van de fietsen en op het effectief gebruik van de fietsen.

1 Bedrijfsvoorstelling

Eerst wordt er in het kort de situering van Smart-ICT uitgelegd, Hierna mijn keuzen motivering en tot slot het werkproces verder wordt uitgelegd.

1.1 Situering [4]

Het Expertisecentrum PXL Smart ICT onderzoekt en ontwikkelt allerlei slimme toepassingen op basis van emerging technologies en bundelt praktijkgericht onderzoek en dienstverlening op het snijvlak van IT en elektronica.

PXL Smart ICT spitst zich toe op mobiele technologie in de breedste zin van het woord en zoekt naar slimme oplossingen voor objecten en gemeenschappen: van slimme sensoren, uurwerken, telefoons en tablets, tot slimme gebouwen en smart cities. Daarbij staat de toepasbaarheid en inzetbaarheid van ontwikkelde toepassingen in de praktijk steeds voorop.

Als interdisciplinair expertisecentrum, ondersteunt en adviseert PXL Smart ICT ook organisaties uit de logistieke sector, de zorg en het onderwijs in hun zoektocht naar slimme applicaties.

1.2 Motivering

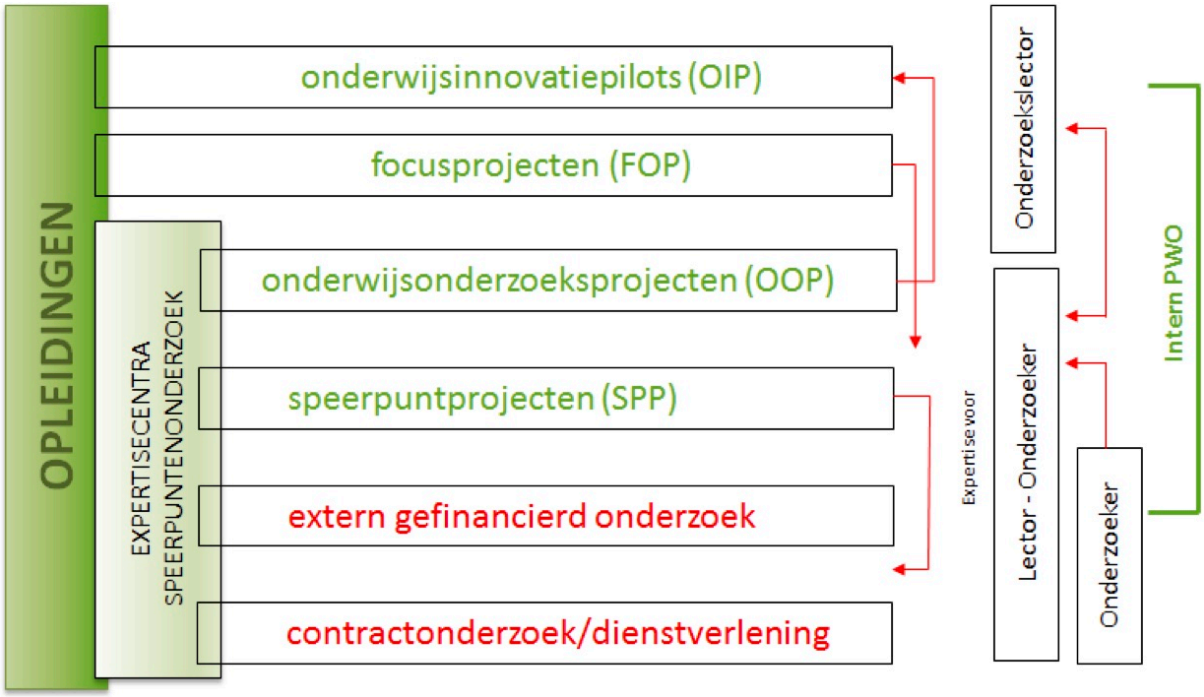
In het algemeen ben ik wel een ‘techgeek’, altijd aan het werk met de nieuwste technologieën en gadgets. Het leek mij daarom een unieke kans om mijn stage te lopen bij Smart ICT, mede doordat zij dagelijks in contact komen met nieuwe technologieën, gadgets, ideeën en werk methodes. Naast mijn studie ben ik ook zelfstandige, Hier krijg ik vaak de vraag om iets nieuws te bedenken, dit kan gaan van een programma tot een hele Escaperoom installatie. Het development aspect van Smart ICT zit hier dus zeker dik in.

1.3 Werkproces

1.3.1 Opdrachten

Binnen Smart ICT zijn er 3 soorten Functies: Praktijkgericht, dienstverlenende contracten en algemene vragen beantwoorden.

Binnen praktijkgericht zijn er ook weer 2 soorten projecten interne en externe projecten. In het algemeen zijn deze bedoeld voor onderwijs en onderzoek. Zie Figuur 1.



Figuur 1: Schema functies Smart ICT.

Intern gaat dit via PWO-middelen. Dit zijn budgetten die de hogeschool krijgt van de Vlaamse overheid. Deze projecten duren gemiddeld 2-3 jaar en wordt er vaak samengewerkt met verschillende departementen binnen research. Zoals bijvoorbeeld: Smart-ICT met Logic-ic of Bouw en energie met zorginnovatie en Smart-ICT. Een aantal project voorbeelden zijn:

-ICT innovatie op maat van de kmo
http://www.pxl.be/Pub/onderzoek/Projecten/Projecten-Smart-ICT/ICT-innovatie-op-maat-van-de-kmo.html

- Exploratie van het gebruik van “Virtual-Reality” in cliëntgericht advies voor mensen met nood aan woningaanpassingen
http://www.pxl.be/Pub/onderzoek/Projecten/Projecten-Zorginnovatie/Exploratie-van-het-gebruik-van-Virtual-Reality-in-clientgericht-advies-voor-mensen-met-nood-aan-woningaanpassingen.html

Extern wordt vaak gefinancierd door Vlajo Vlaamse Jonge Ondernemingen of Interreg Europe European Regional Development Fund. Dit is dan vaker in samenwerking met ander Hogescholen, bedrijven of Onderzoeksinstellingen in landen zoals Nederland, Duitsland, enzovoort. De groep krijg dan op het einde van het project toegang tot de resultaten. PXL-research speelt hier dan als Research and development.

Nu hebben we het gehad over de Praktijkgerichte opdrachten. Er zijn ook nog de zogenaamde dienstverlenende contract opdrachten. Hier is de opdrachtgever vaar een non- of social profit.
Je kan hier denken aan steden of ziekenhuizen. Zij hebben vaak een vraag waar PXL-research de expertise in heeft. De planning van zoon project gaat als volgt. Eerst zijn er intakegesprekken om duidelijkheid te creëren wat de klant precies wilt. Want vaak is het zo dat A zegt maar B wilt. Vervolgens wordt een offerte en een plan van aanpak gemaakt. Hier staat in wat er gedaan wordt, tegen wanneer en waar de klant voor moet zorgen. De offerte wordt bepaald aan de hand van de plan van aanpak en wordt berekend op dagtarief. Dit dagtarief mag ook niet de markt verstoren. Wat ik hier mee wil zeggen is dat De prijs niet te laag onder de concurrentie mag liggen.

Een 3de soort opdracht is algemene vragen beantwoorden. Dit is bedoeld voor bedrijven die zelf aan innovatie willen doen maar hierbij wat vragen hebben of een advies nodig hebben.

1.3.2 Development

Development hangt natuurlijk sterk af van het project. Bij projecten waar meerdere partijen betrokken worden er concrete afspraken gemaakt tussen de betrokkenen zodat alles vlot verloopt. Bij de meeste projecten zijn er 2-3 collega’s aan het werk. Deze hebben dan hun eigen Slack kanaal, dit is een programma waarin ze makkelijk met elkaar kunnen communiceren en een eigen Bitbucket repository om het project in op te slaan zodat iedereen toegang heeft tot de bestanden en verschillende takken kunnen maken voor bijvoorbeeld productie, testen en functies. Zo blijft alles gescheiden van elkaar en kunnen ze teruggaan naar een vorige versie als er iets niet meer functioneert.

Deel 1: Stageopdracht

1 Voorstelling stageopdracht

Deze stageopdracht omvangt twee aparte opdrachten, beiden opdrachten zijn probleemstellingen die bij smart cities en campussen voorkomen. Opdracht een gaat over het traceren van fietsen. In opdracht twee worden knoppen op vuilnisbakken geplaatst om te zien welke vol zijn. Eerst en vooral een kleine uitleg wat een smart city en campus nu is.

1.1 Smart city en campus

Het woord zegt het zelf al “Smart” oftewel slim. Maar wat betekend het nu om een slimme stad te zijn. Wat is er voor nodig en wat is de reden om een stad slim te maken.

Een “smart city” is een collectie van sensoren die data creëert, deze data kan hierna gebruikt worden om verschillende probleemstellingen op te lossen. Hiervoor is een groot aantal sensoren, data en een sterk netwerk nodig. Maar dit wilt niet zeggen dat je met al deze info meteen het probleem opgelost hebt, Je moet er ook iets mee doen. Een aantal gebruikssituaties zijn dan ook: Het verkeer herleiden naar beschikbare parkeerplaatsen, betere fietspaden aanleggen waar veel fietsers langs komen of een efficiëntere route uitstippelen om post te leveren. Natuurlijk zijn dit niet de enige oplossingen, het is ook nog nog maar de start van “smart cities”. Er zijn nog zoveel probleemstellingen die via deze denkwijze kan worde opgelost.

In mijn opdrachten wordt er gekeken naar de “smart campus” dit kan worden vergeleken met een “smart city” maar dan op kleine schaal. Dit wil niet zeggen dat dit niet in een stad gebruikt kan worden, het is zelfs een heel goede manier om oplossingen te testen voor dat deze geïmplementeerd worden in een stad. Bij een campus zijn al vaak systemen beschikbaar zijn zoals toegangspoorten, temperatuur sensoren en een sterk netwerk. Dit zorgt ervoor dat probleem vergemakkelijkt uit te werken zijn.

Een voorbeeld van zoon “smart campus” is hogeschool PXL hier wordt gebruik gemaakt van een centraal identificatie- en betaalsysteem door middel van identiteitskaarten waar NFC (Neer Field Communication) in is voorzien. Hierdoor is het mogelijk om te betalen voor kopieerapparaten, lunch en snacks, het openen van deuren waar beperkt toegang is en parkeren. In de toekomst zijn nog veel meer mogelijkheden. In opdracht een worden deze kaarten ook gebruikt.

1.2 Opdracht 1: traceren van fietsen

De PXL heeft problemen met fietsen die op de verkeerde plaats worden achtergelaten of dat personeel de fietsen een paar dagen zelf bijhoudt. Dit is natuurlijk niet de bedoeling.

Een mogelijke oplossing zou zijn om de fietsen te traceren en bijhouden wie welke fiets gebruikt. Vervolgens kan de campusbeheerder zien waar elke fiets zich bevindt en kan hij de betreffende gebruiker contacteren. De drukke dagen en meest gebruikte fiets is dan ook te zien zodat het onderhoud of vervanging van de fietsen beter kan worden gepland en niet gebeurt wanneer het druk is. Ook voor de gebruiker is dit een ideale oplossing. Deze kan dan zien welke fietsen beschikbaar is, waar deze zich bevindt, waar ze hun fiets hebben achtergelaten en statistieken bekijken hoe ver en veel ze gefietst hebben.

Deze opdracht bestaat dus uit een tracking sensor die communiceert over het sigfox netwerk naar een Node.js backend in JSON-formaat. Een client in AngularJS haalt hier de locaties en gebruiker gegevens uit en visualiseert deze in tabellen en kaarten.

1.3 Opdracht 2: monitoren van vuilnisbakken

Elke dag worden de vuilbakken nagekeken of deze vol zijn. Heel veel van deze vuilnisbakken zijn nauwelijks gebruikt en om deze elke dag te gaan controleren is tijdrovend. Andre zijn meerdere keren per dag vol en afval wordt dan vaak ernaast geplaatst.

Daarom wordt op elke bak een knop geplaatst zodat wanneer de gebruiker ziet dat de bak vol is deze gewoonweg op de knop kan drukken. Dit wordt ook weer gedaan via het Sigfox-netwerk. Eveneens wordt Node.js en AngularJS gebruikt als back- en frontend. Ook hier kan de client de locaties en gegevens zien van elke vuilbak.

Deze oplossing zorgt voor een grote tijdsbesparing omdat de campusbeheerder onmiddellijk ziet welke bak vol is en kan hierdoor gericht zijn controle uitvoeren. Als ze merken dat een vuilbak meerdere malen per dag vol is kan er bijvoorbeeld een grotere vuilbak geplaast worden. Voor de gebruiker is dit laagdrempelig vergeleken met het melden aan de campusbeheerder. Dit gaat ervoor zorgen dat alles sneller en effectiever kan worden afgehandeld.

2 Uitwerking stageopdracht 1

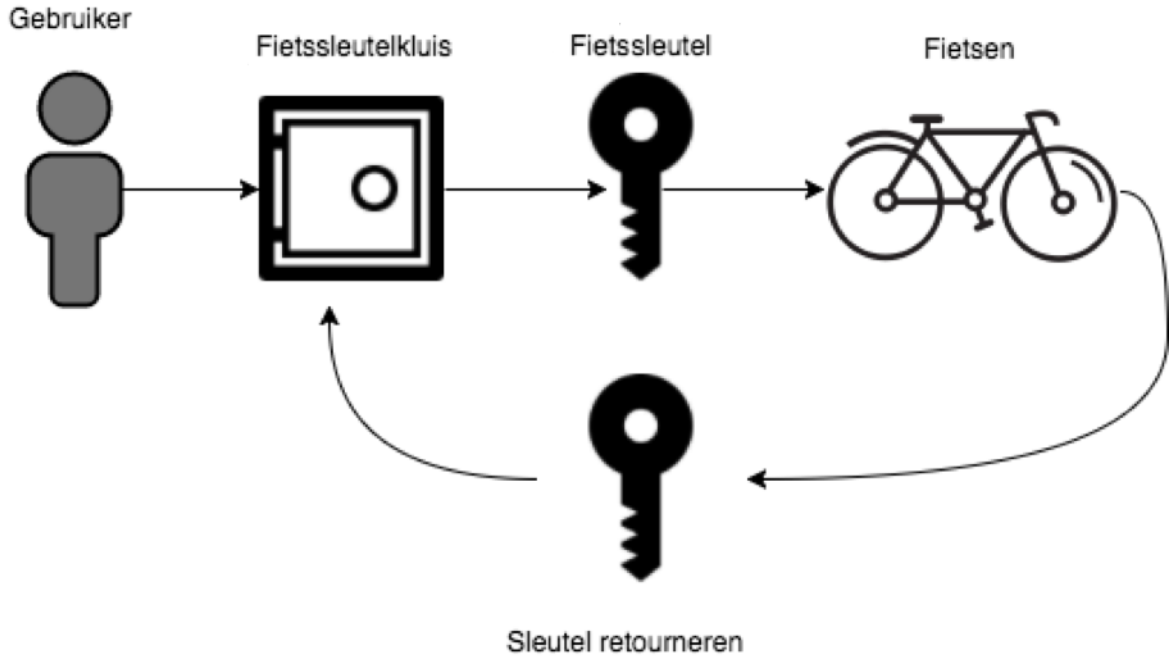
Stageopdracht 1 bestaat uit een trackingsensor die de locatiedata verstuurd via het Sigfox-netwerk waar deze data wordt opgeslagen in een Node.js-backend. Een website in AngularJS haalt deze locaties en gebruikersdata op. Hier worden de gegevens gevisualiseerd op een kaart via Google Maps en via lijsten van gegevens. Hierna wordt eerst worden de procedure uitgelegd, Hoe het nu geregeld wordt en wat verandert er in de toekomst. Vervolgens wordt de Sigfox-communicatie besproken. Ten slotte wordt uitgelegd hoe dit gecombineerd wordt.

2.1 Procedure

Hier wordt de bestaande procedure en de toekomstige procedure beschreven van het lenen van de fietsen.

2.1.1 Bestaande procedure

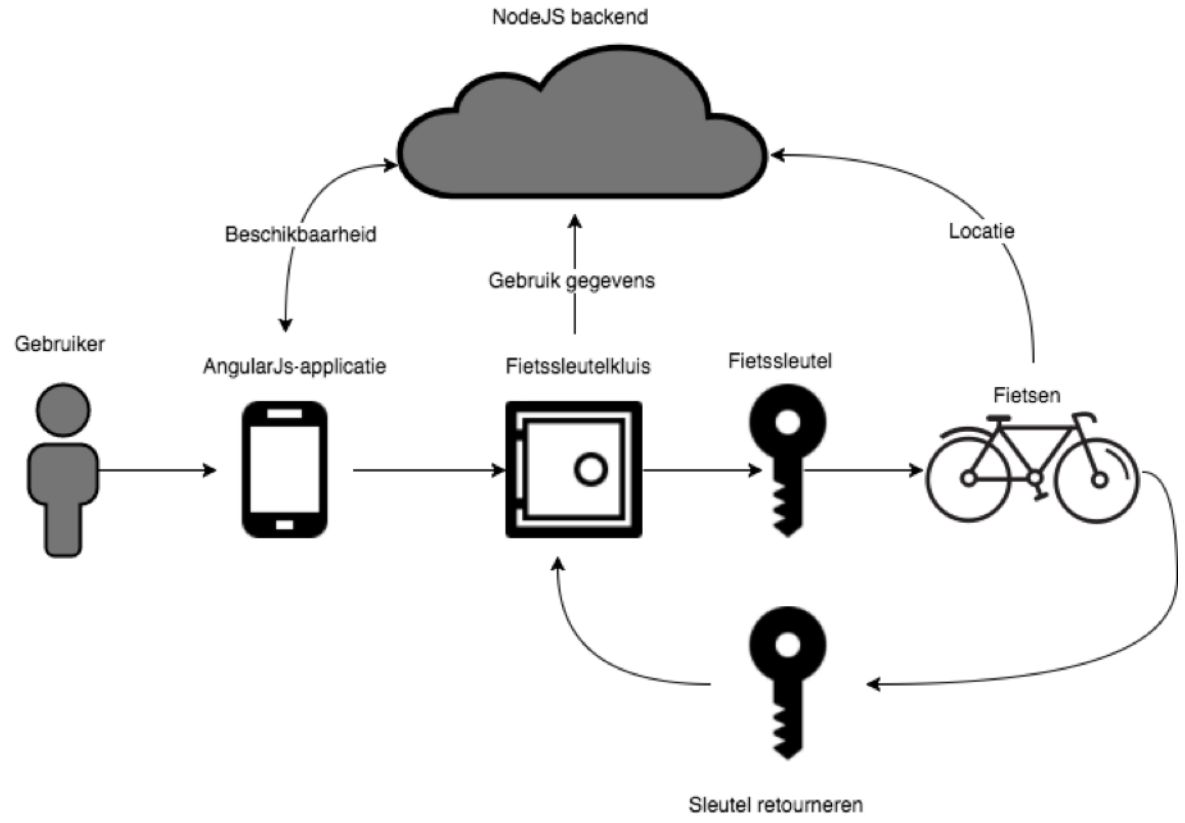
Via de bestaande procedure (zie figuur 2), zijn er kluisjes waar de sleutel van de fiets inzit. Deze kunnen geopend worden met de personeelskaart. De gebruiker kan dan de fiets openen en gebruiken. Met deze methode is er niet te zien of de fiets al dan niet genomen is of door wie, omdat de sloten van de kluisjes nog geen terugmelding hebben. Ook is de locatie waar de fiets zich bevindt onbekend.



Figuur 2: Bestaande procedure fiets lenen.

2.1.2 Toekomstige procedure

Bij de toekomstige procedure (zie figuur 3), is er wel een terugmelding op de kluisjes zodat te zien is wanneer een sleutel eruit wordt genomen en wie deze sleutel heeft. Er kan nu ook bekeken worden hoelang de gebruiker de sleutel bijhoudt vooraleer hij een fiets neemt. Elk kluis heeft een slot die opengemaakt kan worden met een personeelskaart, dit is dezelfde kaart die ook gebruikt wordt voor de andere doeleinde binnen de PXL. Deze sloten zijn verbonden met het netwerk en kunnen daarom gebruikt worden als een terugmelding naar de backend. De fietsen worden verzien van een tracker die verbonden zijn met het sigfox-netwerk. Hierdoor kunnen ze overal getraceerd worden. Het is niet nodig om als gebruiker een account te hebben. De personeelskaart ID wordt dan gewoon meegegeven. De campus beheerder kan zo nog steeds achterhalen wie de fiets heeft. Wanneer de gebruiker toch een account aanmaakt kan hij ook al zijn vorige ritten zien. Gebruikers die geen toegang hebben tot de fietsen krijgen het kluisje ook niet open.



Figuur 3: Toekomstige procedure fiets uitlenen.

2.2 Sigfox

Sigfox is radioprotocol speciaal ontworpen voor IoT. In volgende punten wordt een korte geschiedenis, een aantal use cases en tot slot welke technologie dat het Sigfox-protocol gebruikt, besproken.

2.2.1 Geschiedenis

Sigfox een Frans bedrijf is opgericht door Ludovic Le Moan (CEO) en Christophe Fourtet (CSO) in 2010. Beiden hebben een achtergrond in machine-to-machine en radiotechnologieën. In 2012 werd het eerste Sigfox netwerk gelanceerd en slechts drie jaar later was deze technologie beschikbaar in 36 landen. Vandaag zijn er meer dan 32 serviceproviders in 42 landen.

2.2.2 Use cases

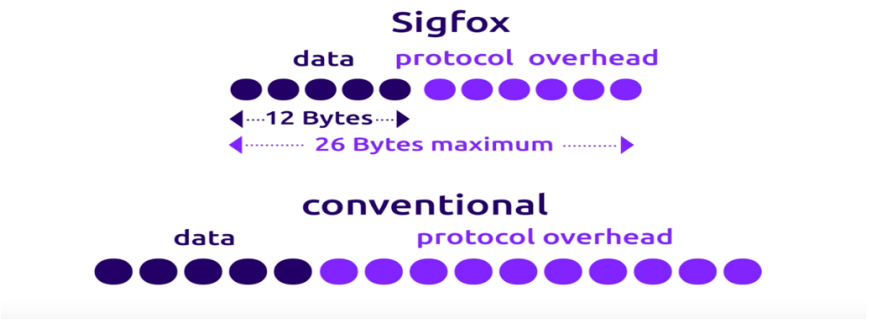
Het grootste probleem van IoT is om aan de gegevens van de sensors te komen, dit is vaak duur of ingewikkeld. Gelukkig biedt sigfox daar er een oplossing voor met een goedkoop jaarlijks abonnement en een makkelijk te gebruiken configuratie en communicatie platform.

Ook de radio technologie is ook een plus. Het is mogelijk om de sigfox zenders ondergronds te plaatsen, dit biedt wat interessante mogelijkheden zoals parkeersensoren, gasmeters en waterpijlmeters. Ook de grote afstanden en batterijduur zorgen er voor dat containers, taxi's en, in ons geval fietsen kunnen worden getraceerd zonder te veel aanpassingen.

Een werkend voorbeeld is een taxicentrale in Hasselt die in elk hotel een knop heeft staan. Op het moment dat op de knop wordt gedrukt krijgt deze centrale een melding en stuurt direct een taxi naar het hotel.Doordat deze knop op sigfox werkt hoeft het hotel geen rekening te houden met het netwerk en kan de knop ook zonder bekabeling worden geplaatst.

2.2.3 Technologie

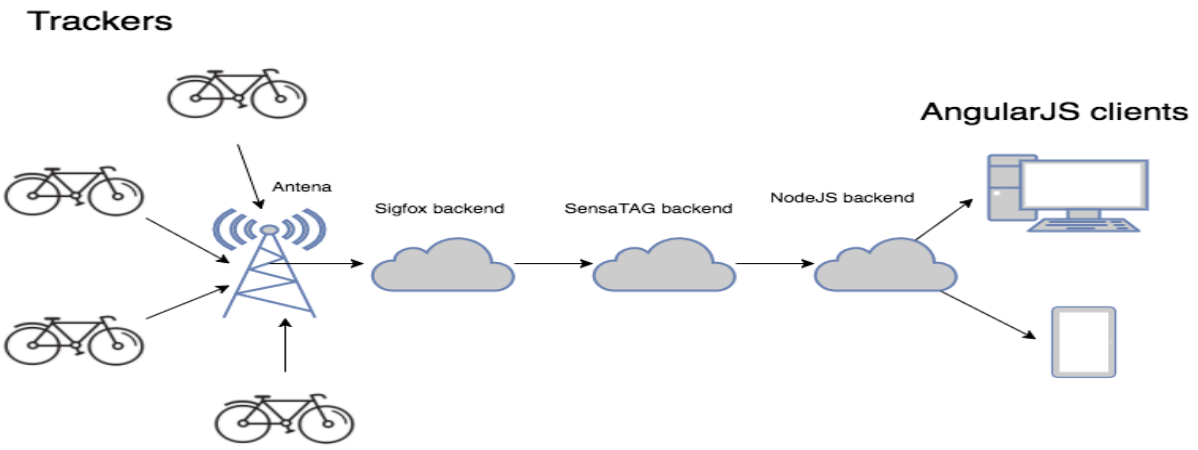
Sigfox werkt via de Ultra Narrow Band (UNB) radiotechnologie. Deze ligt tussen de 868-869Mhz en 902-928 MHz. Afhankelijk van de regio's in Europa is dit 868-869Mhz. Elk bericht kan maximum 12-bytes groot zijn. Omdat dit protocol minder data gebruikt om een verbinding te leggen is het mogelijk om over grotere afstanden data te versturen (zie figuur 4). Hierdoor is de batterijduur langer en storingsgevoeligheid lager. Dit geeft de mogelijkheid om de zenders ondergronds te plaatsen. Een nadeel van deze technologie is dat er maximaal 140 berichten per dag kunnen worden verstuurd. Ontvangen is maximaal 4. Daarom is het zeker niet te bedoeling om dit te gebruiken als een hoge en snelle datacommunicatie. Voor meeste toepassingen is meer ook niet nodig.



Figuur 4: Sigfox vergeleken met andere radio technologieën.

2.3 Communicatie

In dit onderdeel wordt uitgelegd hoe dit alles werkt en met elkaar communiceert. Er zal eerst beschreven worden hoe de configuratie van van de Sigfox-sensoren gebeurt. Vervolgens wordt de communicatie tussen de Sigfox-backend en de SensaTAG-backend beschreven. Deze zorgt ervoor dat de data van de tracker wordt omgezet in een begrijpbare lengte- en breedtegraad. Hierna volgt de communicatie tussen de SensaTAG-backend en de Node.js backend. (zie figuur 5). We sluiten af met de werking van de frontend.



Figuur 5: Communicatie tussen trackers, backends en clients.

2.3.1 Sensor naar Sigfox-backend

Eerst wordt er een categorie aangemaakt (zie figuur 6). Dit zorgt ervoor dat de sensoren opgesplitst kunnen worden in verschillende groepen waardoor voor elke groep een “callback” kan worden ingesteld voor de Node.js backend. Hier zijn de twee categorieën te zien van beide opdrachten.

Description	Display type	Group	Keep alive	Name
SensaTAG Wifi	None	PXL Hogeschool	1 day	SensaTAG Wifi
SigFox Button	None	PXL Hogeschool	1 day	SigFox Button

Figuur 6: Apparaat categorieën

Hierna kan er een apparaat worden toegevoegd en gekoppeld aan een categorie, (zie figuur 7). Zoals te zien in de afbeelding zijn enkele bollen rood. Dit wil zeggen dat dit apparaat langer dan een dag geen melding heeft gegeven. Elk apparaat heeft ook zijn eigen ID deze wordt door de fabrikant ingesteld en kan niet veranderd worden. Hierdoor weet de Sigfox backend welke sensor bij welk administratief account hoort.

Average Rssi	Average SNR	Communication status	Device type	Id	Last seen	Name	Token state
-100.46	50.43		SensaTAG Wifi	368868	2017-11-26 11:59:17	TAG3	
-122.12	28.93		SensaTAG Wifi	36BCEA	2017-11-26 12:20:00	TAG1	
-85.01	66.12		SensaTAG Wifi	36BD08	2017-11-26 10:48:56	TAG4	
-92.28	58.89		SensaTAG Wifi	36C0F3	2017-11-26 12:13:20	TAG2	
-85.90	65.22		SensaTAG Wifi	370484	2017-11-26 10:20:15	TAG5	
-91.23	59.77		SigFox Button	4D8492	2017-11-23 15:17:02	Button 5	
-94.89	56.24		SigFox Button	4D85A3	2017-11-23 08:15:23	button1	
-94.04	57.07		SigFox Button	4D86FA	2017-11-23 08:15:21	Button 4	
-89.66	61.57		SigFox Button	4D8E70	2017-11-23 08:15:22	Button 3	
-91.35	59.75		SigFox Button	4D91BC	2017-11-24 13:44:19	button 2	

Figuur 7: Lijst van apparaten.

2.3.2 Sigfox-backend naar SensaTAG-backend

De data die de Sigfox-backend krijgt van de tracker is enkel de BSSID- en de RSSI-waarde van de sensor. BSSID is een unieke id die elke wifi-modem heeft. Zonder een database te hebben waar instaat welke BSSID bij welke locatie hoort kan er niet veel mee gedaan worden. Gelukkig heeft de fabrikant van deze trackers hiervoor een backend gemaakt die de BSSID omzet naar een lengte- en breedtegraad. Mocht er nu geen wifi-netwerk beschikbaar zijn, wordt er een triangulatie gedaan met behulp van de RSSI-waarde (zie figuur 8). RSSI staat voor de signaalsterkte tussen de zender (sensor) en ontvanger (antenne). Door de gekende locatie van de antennen en de signaalsterkte tussen de antennen en de sensor kan de locatie bepaald worden van de sensor. Dit is lang niet zo accuraat als via wifi. Het is dus geen optie om enkel op deze manier te werken.

Figuur 8: Triangulatie tussen drie antennes.

2.3.3 SensaTAG-backend naar Node.js-backend

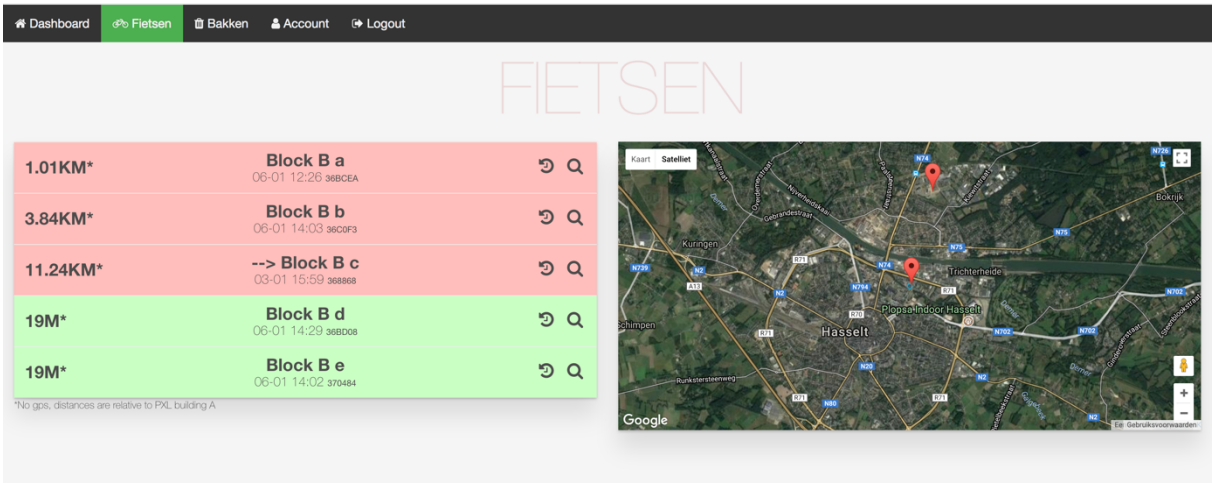
Wanneer de SensaTAG-backend de data heeft vertaald, wordt deze door middel van een “callback” doorgestuurd naar de Node.js-backend. Deze “callback” bevat de volgende informatie (zie Tabel 1). Deze gegevens verzamelen wij in een MySQL database om later te kunnen gebruiken door de client. ‘Latitude’ en ‘longitude’ zijn de lengte- en breedtegraad coördinaten. Radius is de nauwkeurigheid. Hoe kleiner dit getal is, hoe dichter de sensor bij de coördinaten ligt. ‘Time’ is het tijdstip dat de sensor zijn locatie heeft doorgestuurd aan de backend. De deviceID is de unieke id die elke sensor heeft.

Key	Value	Example
lat	latitude	55.123456
lng	longitude	4.123456
radius	radius	24
time	YYYY-MM-DD hh:mm:ss	2017-07-28 13:15:21
deviceID	Device ID (modem code)	A1B2C3

Tabel 1: Data van “callback”.

2.3.4 Node.js-backend naar AngularJS-frontend

Dit is het gedeelte waar de backend communiceert met de frontend. De frontend is waar de gebruiker alle informatie te zien krijgt, dan hebben we het over de locaties van de fietsen, gebruiker gegevens, statistieken en zo voort. Hier worden geen “callbacks” gebruikt omdat er enkel communicatie nodig is als de gebruiker op de webpagina is (zie figuur 9). Deze communicatie is beveiligd met een token, deze token wordt gegenereerd bij het aanmelden en geldt voor 24 uur. Dit zorgt ervoor dat gebruikers die niet ingelogd zijn ook geen data kunnen krijgen of aanpassen, omdat deze token bij elke aanvraag moet worden meegestuurd.



Figuur 9: Webpagina fietsen.

3 Uitwerking stageopdracht 2

Stageopdracht 2 bestaat uit knoppen die net als de voorgaande opdracht ook wordt verstuurd via het Sigfox-netwerk naar een Node.js-backend, waar een clientwebsite in AngularJS deze visualiseert in een lijst en op een Google-Maps kaart. De procedure wordt uitgelegd, hierbij zijn enkele verschillen met de stageopdracht 1. Tenslotte wordt de communicatie uitgelegd. Deze heeft een aantal verschillen met de vorige opdracht.

3.1 Procedure

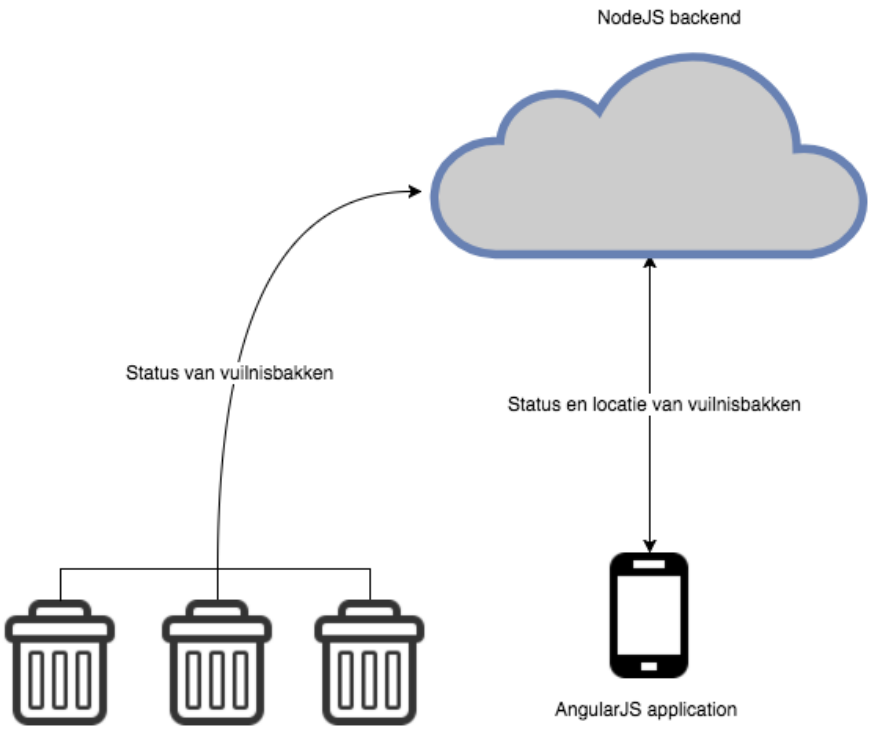
Hier wordt de bestaande procedure en de toekomstige procedure beschreven van het onderhouden van de vuilnisbakken.

4.1.1 Bestaande procedure

Vandaag de dag worden alle vuilnisbakken dagelijks manueel nagekeken en leeggemaakt. Dit zorgt vaak voor onnodige afstanden en frustratie. Vaak zijn de bakken niet vol of zelfs overvol. In sommige gevallen is het ook storend als er bijvoorbeeld een meeting of evenement is. Het is ook niet gemakkelijk te overzien welke vuilnisbakken veel gebruikt worden en waar het best nog een extra vuilnisbak kan komen te staan.

4.1.2 Toekomstige procedure

Dit probleem willen we oplossen met behulp van sensoren of knoppen op elke vuilnisbak, zodat de gebruiker op de knop kan drukken wanneer de bak vol is. Zo krijgen de campusmedewerkers een melding en hoeven ze niet meer onnodig alle bakken te controleren. Ook is het mogelijk om zo te zien welke vuilnisbakken het meest gebruikt worden en op weke plaats er een extra bak moet komen te staan (zie figuur 10).



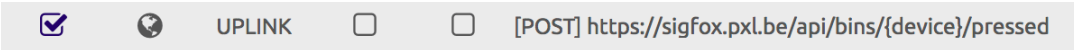
Figuur 10: Toekomstige procedure vuilnisbakken.

3.2 Communicatie

Bij deze opdracht loopt de communicatie tussen de knoppen en webpagina net wat anders dan bij de fietsen. Omdat er geen vertaling tussen de BSSID en de coördinaten moet gebeuren, communiceert deze niet via de SensTAG-backend, maar communiceert de Sigfox backend rechtstreeks met de Node.js backend. De sensor communiceert op dezelfde manier als bij de stageopdracht een met de Sigfox backend.

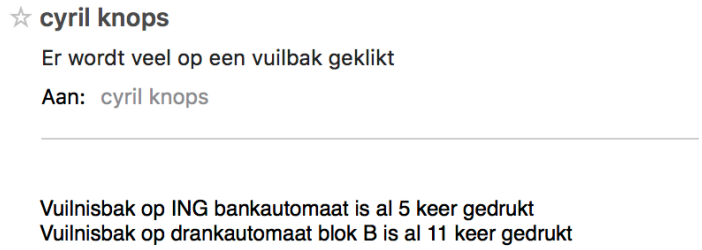
3.2.1 Sigfox-backend naar Node.js-backend

Ook bij de Sigfox backend kunnen “callbacks” worden ingesteld (zie figuur 11). Omdat hier enkel de deviceID van toepassing is, wordt deze enkel meegestuurd in de “Callback”. Deze “Callbacks” zijn ook beveiligd met een token, deze token vervalt niet omdat anders er steeds een nieuwe token moet worden ingevoerd in de Sigfox-backend.



Figuur 11: “Callback” naar Node.js backend.

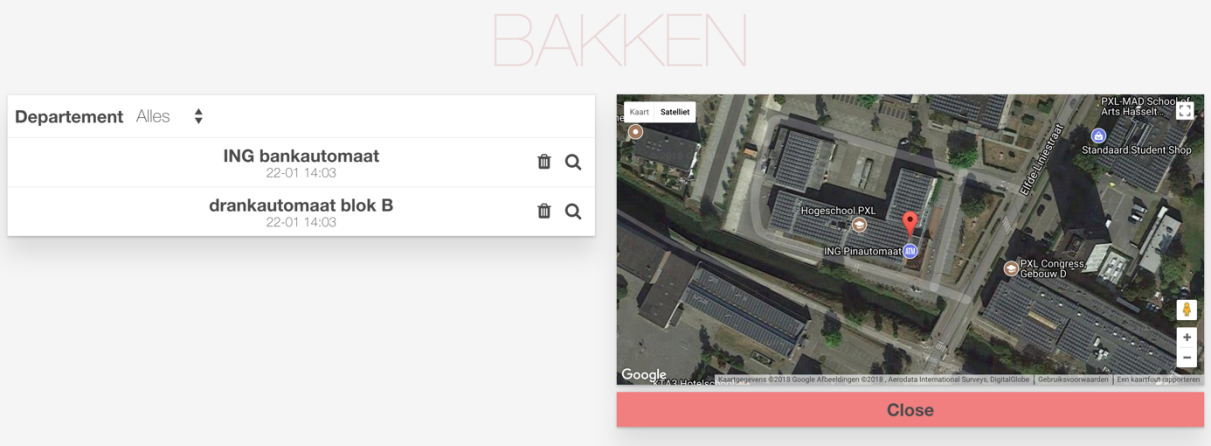
De Node.js backend gaat heir nu kijken of de knop de afgelopen 5 minuten in is gedrukt. Zo niet wordt geteld hoevaak is gedrukt sinds de laaste leging. Is er meer als 5 keer gedrukt dan wordt er een email gestuurd naar de campus beheerder (zie figuur 12).



Figuur 12: E-mail campus beheerder.

3.2.2 Node.js-backend naar AngularJS-frontend

Tot slot de communicatie van de backend naar de frontend. Hier krijgt de campusbeheerder te zien welke vuilnisbakken vol zijn, waar deze zich bevindt en hoe laat deze is gedrukt. Ook kan er een filter worden ingesteld om enkel de resultaten van een bepaald gebouw te zien (zie figuur 13). Ook hier is de communicatie tussen Node.js-backend en AngularJS-frontend beveiligd met een token. Dit is de zelfde token als bij opdracht een, dit komt omdat de twee opdrachten op de login pagina bereikt kunnen worden.



Figuur 13: Webpagina vuilnisbakken.

Deel 2: Vraagstelling onderzoek

1 Onderzoeksvraag

Campussen worden alsmaar interessanter voor bedrijven en scholen. De voordelen die instanties hebben om bij elkaar te zitten zijn oneindig. Helaas is het moeilijk voor de campusbeheerders om al het personeel, vragen en problemen gestructureerd te houden. Gelukkig biedt “Internet of Things” hier een oplossing voor. Ook meer en meer bedrijven bieden nu frameworks aan om deze IoT-apparaten aan te spreken. Ook worden de verschillende hosting mogelijkheden beken waar deze frameworks op werken. Maar wat zijn hier nu de verschillen?

- Welk framework en hosting is het goedkoopst?
- Welk is het gemakkelijkste op te zetten?
- Welk is makkelijker uit te bereiden?
- Welke oplossing is het snelst?

In dit onderzoek gaan we een vergelijking maken tussen drie frameworks namelijk “Node JS”, “ASP .NET” en PHP. Het antwoord op de voorgaande vragen is sterk afhankelijk van de hosting, daarom worden deze ook bekeken. Het onderzoek is geslaagd als er een duidelijk verschil/antwoord is op de vier voorgaande vragen.

2 Onderzoeksmethode

De bedoeling van dit onderzoek is hoe de Sigfox-sensor gaat communiceren (in JSON-formaat) via een van deze frameworks. Hier wordt met verschillende tools gekeken naar de performantie van elk framework. Ook wordt bekeken wat de kosten gaan zijn en hoe gemakkelijk het is om dit op te zetten. Moet dit op een ‘dedicated server’ of kan dit via een webserver. Is het scalable of moet er te veel worden aangepast?

2.1 Prijs

Node.js en ASP.NET zijn niet gemaakt om op een webhosting te draaien. Hiervoor is een krachtigere server nodig waar meer toegang op verleend kan worden. Dit kan op verschillende platformen, de meest bekende zijn Amazon AWS, Microsoft Azure en Google Cloud. Dit zijn vooral de zogenaamde Cloud hostings. De prijs is tussen de 1.60-170 dollar per maand gemiddeld maar kan oplopen tot miljoenen per maand. Dit komt omdat je enkel betaald voor wat je nodig hebt.

Het is ook mogelijk om een VPS of dedicated server te gebruiken. Hier wordt wel een vast bedrag aangerekend en deze liggen tussen de 10-800 dollar per maand.

Een PHP framework kan perfect gehost worden op een shared webhosting. Een shared webhosting is eveneens als de VPS en dedicated server een hosting met een vaste prijs en kost 2 tot 25 dollar per maand.

Er is geen bijkomende prijs voor de frameworks deze zijn allemaal “free to use”. Dit wil niet zeggen dat de bewerking programma’s niet betaald zijn, zeker als deze gebruikt worden in een bedrijfsomgeving. Is de prijs voor Visualstudio een tekst bewerker voor ASP.NET 499 dollar per jaar per gebruiker gevraagd. De prijs voor Webstorm een tekst bewerker voor Node.js is dit 160 dollar per jaar per gebruiker. Tot slot de prijs voor Phpstorm een php tekst bewerker 250 dollar per jaar per gebruiker gevraagd.

2.2 Opzetten

2.2.1 Hostings

Shared webhosting is het makkelijkste op te zetten op te zetten omdat alles wat je nodig hebt om te beginnen al geïnstalleerd is. Vaak is hier ook goede klantenservice bij omdat dit vaak de keuzen is voor iemand die niet veel van servers kent.

Cloud hosting is moeilijker omdat je vaak al veel moet configureren om te krijgen wat je nodig hebt en vaak zijn de hosting sites onoverzichtelijk.

VPS of dedicated hosting is het moeilijkste, omdat je een hele server ter beschikking hebt en ook alle beveiliging voor jou rekening moet nemen. Dit is ook meteen een voordeel omdat je ook geen software limitaties hebt.

2.2.2 Frameworks

PHP is het gemakkelijkste om op te zetten doordat deze vaak standaard staat geïnstalleerd op de server, zeker bij een Shared webhosting. Mocht je deze nu toch meer gaan gebruiken dan dat Shared webhosting kan bieden kan je nog altijd met PHP kiezen om voor een Cloud, VPS of dedicated hosting te gaan.

Met frameworks zoals Node.js en ASP.NET heb je enkel de keuze om te werken op een cloud, VPS of dedicated hosting. Dit is ook vaak moeilijker omdat deze frameworks niet standaard op de server staan geïnstalleerd.

2.3 Uitbreiden

2.3.1 Hosting

Bij shared webhosting is het redelijk makkelijk om over te stappen naar een duurder tarief met meer mogelijkheden. Om terug te gaan naar een goedkoper tarief is vaker minder evident. Deze overstap gaat ook niet automatisch je moet dus op het moment van een tekort aan rekenkracht of geheugen zelf de overstap doen.

Cloud hosting is het makkelijkste om uit te breiden. Dit komt doordat het schalen automatisch kan en er enkel wordt aangerekend voor wat je gebruikt. Het nadeel hiervan is dat je vaak voor onverwachte kosten komt te staan.

VPS is makkelijker uit te bereiden dan een dedicated server. Omdat het een Virtuele server is, kan de CPU, het geheugen en de opslag aangepast worden. Dit is bij een dedicated server niet het geval. Hier krijg je een complete server en moet je het doen met wat je oorspronkelijk aangekocht hebt.

2.3.2 Frameworks

Voor PHP zit je al snel vast, eenmaal je aan het Shared webhosting premium limiet zit. Als je nu nog meer rekenkracht of geheugen nodig zou hebben moet je alles overzetten naar een van de andere opties.

Voor Node.js en ASP.NET is het uitbreiden al een minder groot probleem omdat je niet op een shared webhosting zit. Bij cloud hosting of VPS-hosting is het al een stuk makkijker omdat hier vaak meer opties beschikbaar zijn. Enkel bij de dedicated hosting zou je moeten overschakelen naar een van de andere opties als je het limiet hebt bereikt.

2.4 Snelheid

Snelheid van het framework hangt af van de server snelheid. Daarom ga ik deze eerst vergelijken.

2.4.1 Hostings

Shared webhosting is vaak het traagst omdat je met meerdere mensen de CPU en Geheugen deelt.

VPS en Cloud hosting is een stap sneller. Hier deel je nog steeds de server maar dit zijn vaker snellere servers en kunnen makkelijker aangepast worden.

Dedicated hosting is het snelste mits dat je een krachtige server hebt. Dit komt omdat je niks deelt, de hele server is van jou.

2.4.2 Frameworks

Voor elk framework drie testen geschreven (zie tabel 2,3,4). In deze test wordt een lijst van nummers gestuurd naar het framework. Deze telt de nummers op en stuurt het resultaat terug. De tests lopen gedurende 60 seconden. Deze testen zijn uitgevoerd op de volgende systemen.

Cliënt systeem:	Node.js & PHP Server:	ASP.NET Server:
Bandbreedte: 512 kbps,	Bandbreedte: 2 Mbps,	Bandbreedte: 2 Mbps,
2.8Ghz Dual Core Processor,	2.0Ghz Single Core Processor,	2.8Ghz Quad Core Processor,
4 GB RAM	512 MB Ram	4 GB Ram

Case 1: 10 agents
Case 2: 30 agents
Case 3: 50 agents

ASP.NET

Name	Case 1	Case 2	Case 3
Total Requests	134	129	143
Errors	0	0	0
Avg Response Time	4.675	15	25.2
Avg Throughput	2.161	1.9	2.3
Max Throughput	5	7	9

Tabel 2: Resultaten ASP.NET.

PHP

Name	Case 1	Case 2	Case 3
Total Requests	156	162	165
Errors	1	0	4
Avg Response Time	3.9	12.3	19.7
Avg Throughput	2.5	2.3	2.17
Max Throughput	5	5	8

Tabel 3: Resultaten PHP.

Node.js

Name	Case 1	Case 2	Case 3
Total Requests	231	256	260
Errors	0	0	0
Avg Response Time	2.6	7.6	12.6
Avg Throughput	3.6	3.6	3.56
Max Throughput	6	5	6

Tabel 4: Resultaten Node.js.

Zoals te zien in de resultaten komt Node.js uit als de winnaar. Dit komt omdat Node.js “non-blocking” en “event-driven” is. Dit zorgt ervoor dat het famework licht en efficiënt is om meerdere cliënts tegelijk te dienen. Zelfs met minder reken- en geheugenkracht.

2.5 Bronnen

Website setup. What is web hosting [Online] [11 12 2014]
https://websitesetup.org/what-is-web-hosting/

Hack witch mak. ASP.NET vs PHP vs Node.js benchmark [Online][2 4 2014]
http://www.hackwithmak.com/2014/04/aspnet-vs-php-vs-nodejs-benchmark-test.html#

3 Proof of Concept

3.1 Prijs

Voor een standaard AWS, Azure of Google Cloud server ligt de prijs gemiddeld tussen de 0,1 en 0,21 Dollar per uur (Zie figuur 14). Bij een dedicated server wordt vaak een prijs per uur gerekend, dit maakt het schalen van een server veel gemakkelijker omdat je enkel betaalt voor hetgeen dat je gebruikt.

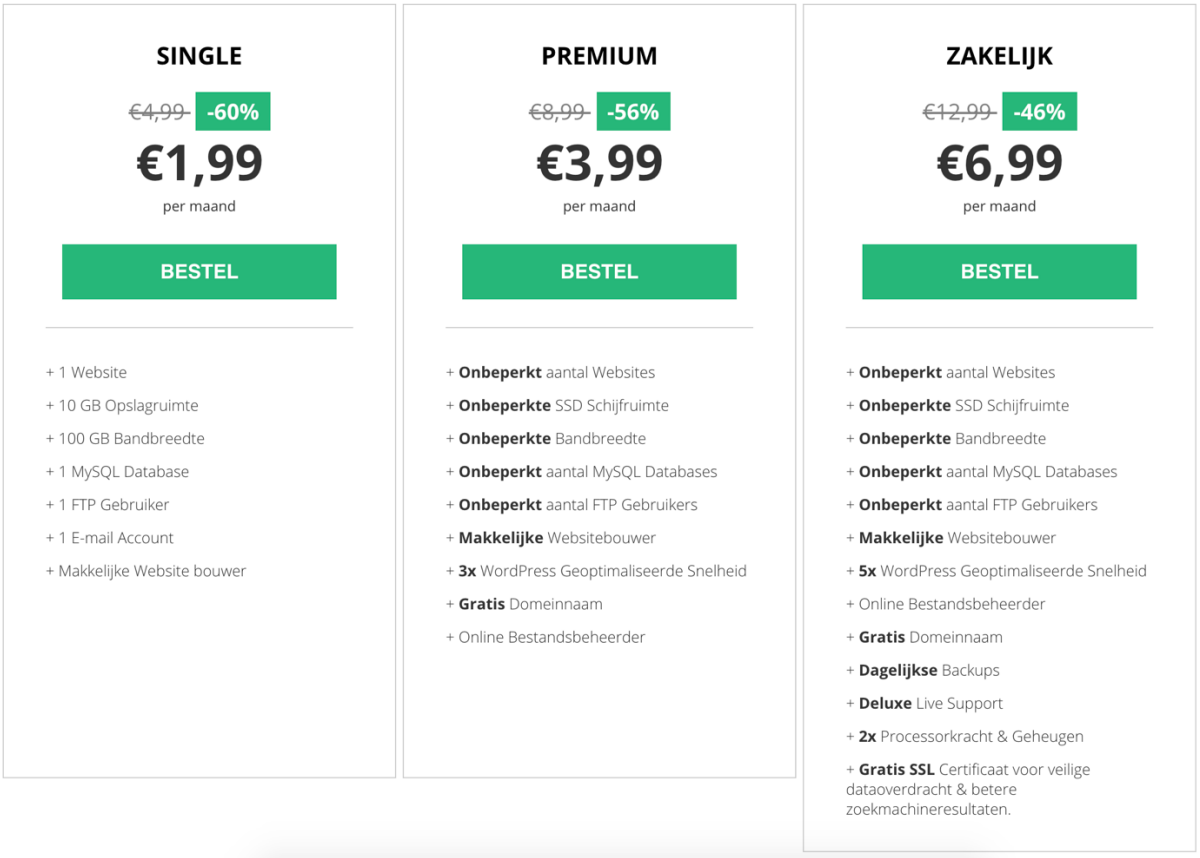
AWS vs. Azure vs. Google On-Demand Prices

Resource Type (us-east, Linux)	AWS Instance	Azure Instance	Google Instance	AWS OD Hourly	Azure OD Hourly	Google OD Hourly	AWS /GB RAM	Azure /GB RAM	Google /GB RAM
Standard 2 vCPU w SSD	m3.large	D2 v2	n1-standard-2	$0.133	$0.114	$0.212	$0.017	$0.016	$0.028
Highmem 2 vCPU w SSD	r3.large	D11 v2	n1-highmem-2	$0.166	$0.149	$0.238	$0.011	$0.011	$0.018
Highcpu 2 vCPU w SSD	c3.large	F2	n1-highcpu-2	$0.105	$0.099	$0.188	$0.028	$0.025	$0.104
Standard 2 vCPU no SSD	m4.large	D2 v2	n1-standard-2	$0.108	$0.114	$0.100	$0.014	$0.016	$0.013
Highmem 2 vCPU no SSD	r4.large	D11 v2	n1-highmem-2	$0.133	$0.149	$0.126	$0.009	$0.011	$0.010
Highcpu 2 vCPU no SSD	c4.large	F2	n1-highcpu-2	$0.105	$0.099	$0.076	$0.027	$0.025	$0.042

As of Dec 2, 2016 *Source: RightScale*

Figuur 14: Prijzen cloud servers.

Bij shared webhosting is dit meestal een maandelijks of zelfs jaarlijks bedrag. Hierdoor weet je precies wat je gaat moeten betalen maar dit is in vergelijking met dedicated servers minder handig om uit te breiden. Hostinger.nl is een bekende hosting site en zij hanteren 3 verschillende formules (Zie figuur 15).



Figuur 15: Prijzen webservers.

Laten we veronderstellen dat de server 24 uur per dag 7 dagen per week zou opstaan, dan ben je voor een Cloud service tussen 74,4 en 156,24 euro kwijt. Voor de shared webhosting kom je tussen de 4,99 en 12,99 te zitten, afhankelijk van de processorkracht die je nodig hebt. Het voordeel van de Cloud service is dat wanneer je minder gebruikt, je automatisch minder gaat moeten betalen.

3.2 Opzetten

Voor iedereen is dit verschillend. Iemand die liever programmeert dan een server op te zetten gaat zich eerder richten op een shared webhosting of cloud hosting. Persoonlijk vind ik de voordelen van een dedicated of VPS belangrijker. Het idee dat de beveiliging van je website bij een onbekende ligt is toch vreemd. Daarom kies ik liever voor een dedicated of VPS voor kleinere applicaties. Mocht ik nu toch voor een groot bedrijf applicaties maken dan zou ik voor cloud hosting omdat dit simplere is om voor meerdere programmeurs op te zetten. Als framework zou ik daarom kiezen voor Node.js. In het verleden heb ik meerdere malen Node.js over moeten zetten van een hosting naar de andere en heb hier nooit problemen gehad. Bij het toevoegen van packeten worden deze toegevoegd aan een configuratie bestand waardoor het opzetten gemakkelijk kan gebeuren.

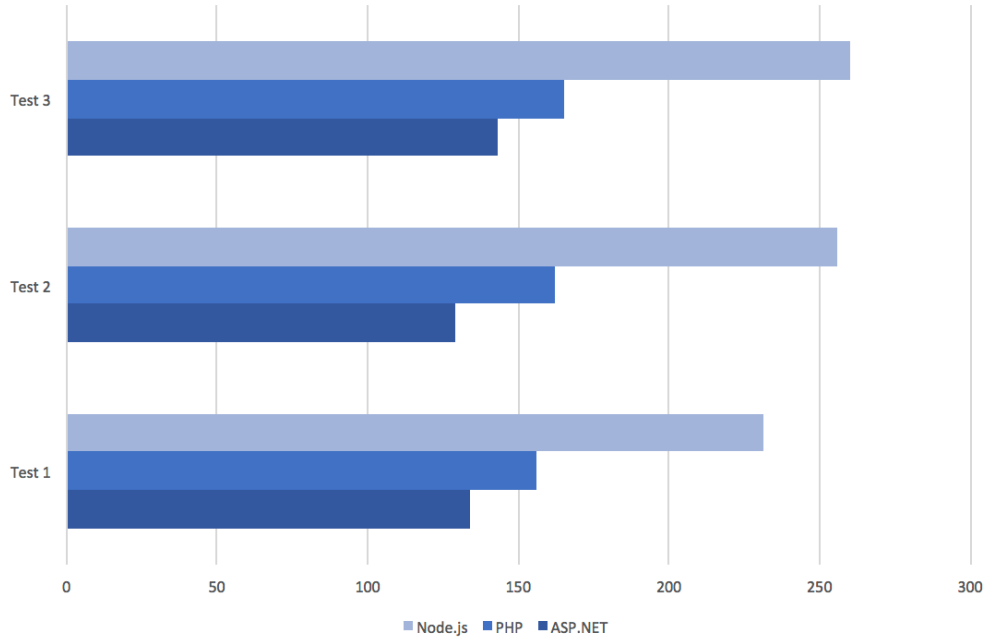
3.3 Uitbereiden

Voor uitbereiding zie ik maar een duidelijke winnaar, cloud hosting is hier de beste oplossing. Als nieuw bedrijf zou ik hier mijn energie en tijd insteken omdat deze optie met het bedrijf mee groeit. Zeker als een applicatie of website online moet blijven. Bij een dedicated of VPS is dit vaak eigenverantwoordelijkheid, zeker als het een van de systemen is die jezelf toegevoegd hebt. Bij cloud hosting is de hosting altijd verantwoordelijk en zij doen hier dus ook alles aan om geen uitval te krijgen.

3.4 Snelheid

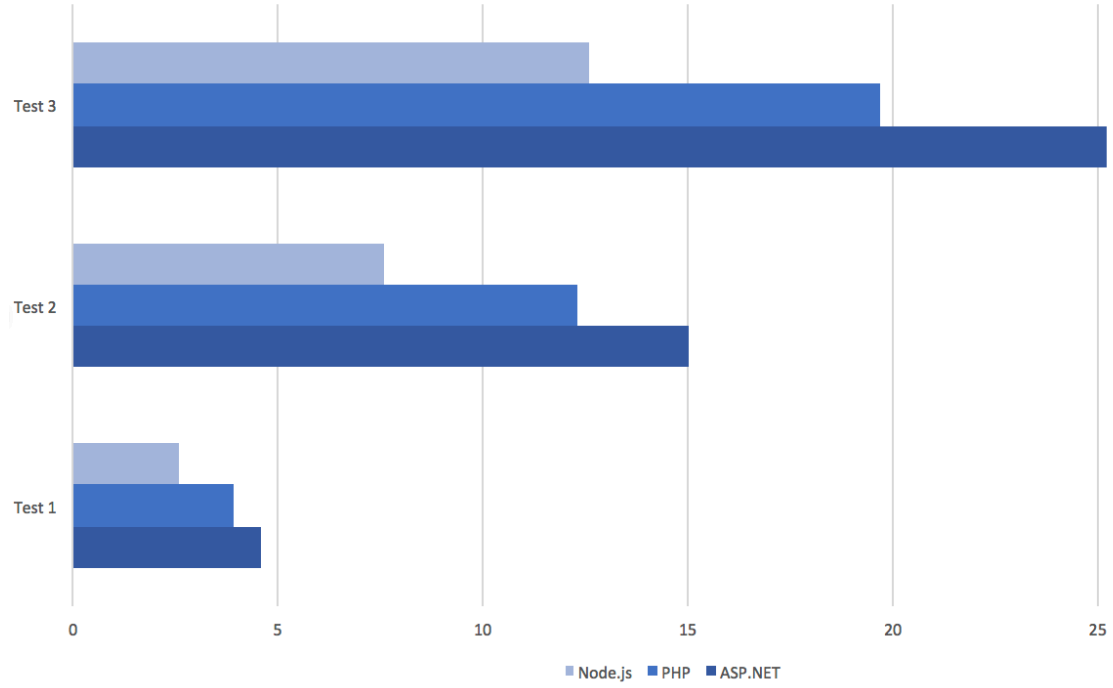
De snelheid van een framework hangt af van de server of hosting waar deze op draait. In deze testen heb ik een shared webhosting van Hostinger gebruikt en een Blade server van Hogeschool PXL. De specificaties wordt er bij shared webhosting dit vaak niet vermeld. De blade server is een 4 core 2.6 GHz met 4 GB ram. De testen van ASP.NET en Node.js zijn via de blade en PHP draait op een shared webhosting. De tool die hier gebruikt wordt is Pylot dit is een Python gebaseerde speed testing tool. Hier kan een configuratie file gemaakt worden waar in staat hoeveel clients er gesimuleerd worden, hoelang de test duurt en hoe de aanvragen eruitzien. In dit voorbeeld worden willekeurig getallen gestuurd, de server moet deze getallen optellen en vermenigvuldigen en de oplossing hiervan wordt teruggestuurd. Er wordt gekeken hoeveel keer dit kan gebeuren per minuut en hoelang elke aanvraag duurt.

In de volgende figuur (figuur 16) is getest hoeveel aanvragen per minuut er kunnen worden gedaan voor elk framework. Bij test een zijn er 10 clients, test twee 25 clients en bij test drie 50 clients. Hoe meer aanvragen hoe beter. En zoals te zien is Node.js de koploper in elke test.



Figuur 16: Aanvragen per minuut.

Hier (figuur 17) wordt gekeken hoe lang elke aanvraag duurt dit is uitgedrukt in ms. Hoe korter de tijd hoe beter. Zoals vorige is hier weer te zien dat Node.js de snelste is. Natuurlijk zijn deze twee grafieken sterk met elkaar verbonden. Hoe sneller de aanvragen hoe meer aanvragen er passen in een minuut.



Figuur 17: Duur aanvraag in milliseconde.

Een terugblik naar de andere ondervindingen van verschillende bronnen is te zien dat de gegevens met elkaar overeenkomen. Hiermee kunnen we ervan uitgaan dat in de tijd tussen de twee testen er nog geen verandering is gekomen in welke framework de beste is.

Conclusie

1.1 Deel 1

Om efficiëntie te verhogen en ergernis te voorkomen binnen de PXL, is er de vraag om verschillende processen te verbeteren. Het bijhouden van de fietsen en het controleren van de vuilnisbakken is een arbeidsintensief proces. Deze beide probleemstellingen kunnen worden verholpen met behulp van IoT.

Het bijhouden van de fietsen en het controleren van de vuilnisbakken is een dagelijks proces die tot heden manueel uitgevoerd moest worden. Elke dag wordt er gekeken of alle fietsen terug staan op hun plaats en of er geen vuilnisbakken vol zijn. Dit wil zeggen dat de campus beheerder deze dagelijks moet gaan controleren en in het geval van een grote campus zoals de PXL is dit tijds intensief. Door gebruik te maken van IoT kan de status van de fietsen en vuilnisbakken doorgegeven worden zonder dat hij ervoor moet gaan kijken.

De fietsen krijgen nu elk een tracker waardoor de exacte locatie kan worden bepaald. De kluisjes waar de fietssleutels in bewaard worden krijgen nu een terugmelding, zo wordt de persoons ID meegegeven zodat er duidelijk is wie de fiets gebruikt.

Bij de vuilnisbakken komt een knop te staan zodat personeel of studenten er op kunnen drukken wanneer deze vol is. Hierdoor kan de campus beheerder een planning maken om deze het beste te legen, ook krijgt hij een zicht op het gebruik van de vuilnisbakken en kan hierdoor extra vuilnisbakken voorzien.

Doordat deze processen nu via IoT worden opgelost kan de campus beheerder zich richten op andere belangrijkere taken. Niet enkel voor de campus beheerder is dit een ideale oplossing maar de gebruiker profiteert hier ook van. Zo kan de gebruiker zien welke fietsen nog beschikbaar zijn, Waar hij zijn fiets heeft gezet en hoeveel KM hij al gefietst heeft. Bij de vuilnisbakken is dit duidelijk. Niemand zoekt graag naar een vuilnisbak die nog niet vol is, laat staan kijken naar een uitpuilende vuilnisbak of vuilnis op de grond.

Door het blijven toepassen van IoT in campussen en het blijven bouwen van zulke “smart campussen” en “smart cities” gaat de efficiëntie en levenskwaliteit omhoog en blijven kosten en frustratie laag. Daarom zie ik zeker de belangen om dit nu en in de toekomst te doen en raad ik Smart-ICT aan om hier zeker verder onderzoek in te doen zodat we uiteindelijk een compleet zorgeloos en efficiënte samenleving tegemoet kunnen komen.

1.2 Deel 2

Bij het vergelijken tussen eigen en ondervindingen van derde zien we dat Node.js nog steeds op nummer een staat als framework voor IoT dit is mede doordat Node.js geschreven is in Javascript waardoor het ontvangen, verzenden en begrijpen van JSON een stuk sneller gaat dan bij een van de andere frameworks. JSON is het meest gebruikte structuur van communicatie bij IoT waardoor dit framework zeer geschikt is. Tweede sterk punt is dat Node.js “non-blocking” is dit wil zeggen dat er geen aanvragen in de wacht worden gezet, dit helpt zeker bij de snelheid van het framework.

Voor Prijs en Uitbreidbaarheid is dit ook zeer gemakkelijk. Doordat er een configuratie bestand wordt aanpast bij toevoegingen en aanpassingen van de modules. Hierdoor is het opzetten, migreren en uitbreiden een stuk gemakkelijker.

Bibliografie

1 SourceDaddy. BSSID, SSID and ESSID [Online] [27 11 2017]
http://sourcedaddy.com/networking/bssid-ssid-and-essid.html/

2 Sigfox Radio Technology Keypoints. [Online] [30 11 2017]
https://www.sigfox.com/en/sigfox-iot-radio-technology/

3 Joni Horemans. HLN [Online] [2 /2 2017]
https://www.hln.be/geld/vacature/waarom-de-toekomst-van-onze-steden-slim-is~ad196e29/

4 PXL. PXL-research [Online] [16 12 2017]
https://www.pxl.be/onderzoek.html