

Scriptie ingediend tot het behalen van de graad van
PROFESSIONELE BACHELOR IN DE ELEKTRONICA-ICT

Remote Control/Support Application using Raspberry Pi and Open Source Technologies

Michael Marivoet

Academiejaar 2015-2016

AP Hogeschool Antwerpen
Wetenschap & Techniek
Elektronica-ICT



Table of Contents

Abstract	0
Dankwoord	1
Introductie	2
Technologieën	3
Diagram systeem	3.1
Hardware	3.2
Arduino	3.2.1
Raspberry	3.2.2
Hardware	3.2.2.1
Software	3.2.2.2
Server	3.3
Node.js	3.3.1
MySQL server	3.3.2
Webapplicatie	3.4
Angular modules	3.4.1
Functionaliteit	3.4.2
Verdere uitwerking	4
Conclusie	5
Bronnen	6
Glossary	

Abstract

Open Source Remote Control & Support Application using Raspberry Pi

Deze bachelorproef bestaat uit het ontwikkelen van een systeem voor het Agentschap Wegen en Verkeer. Dit systeem moet ontwikkelt worden met behulp van open source technologieën. Het doel van dit systeem is op een kost effectieve manier kleinere installaties te voorzien van afstandsbeveiliging en het mogelijk maken deze of afstand te monitoren.

Hier wordt feitelijk verder gewerkt aan een proof of concept dat ontwikkelt is geweest tijdens het academiejaar 2014-2015.

Het systeem beschikt over de volgende functionaliteit. Het laat toe om in realtime via de webapplicatie, de status van de installatie te bekijken. Het geeft de gebruikers ook de optie om via zowel de webapplicatie als met zijn [eID](#) ter plaatse de installatie te ontgrendelen. De webapplicatie maakt het mogelijk om verschillende installaties en gebruikers aan te maken. Het laat ook toe verschillende installaties types aan te maken en deze als eigenschap mee te geven aan de installaties.

Dankwoord

Hierbij wil ik graag alle personen danken die rechtstreeks of onrechtstreeks meegewerkt of geholpen hebben aan deze bachelorproef.

Allereerst wil ik mijn externe promotor Dhr. Rik Haekens danken voor mij de kans te geven aan deze interessante stage opdracht te laten werken en voor zijn begeleiding tijdens de stage zelf. Ik wil tevens ook mijn dank uiten aan mijn interne stage promotor Dhr. Luyts Maarten voor zijn begeleiding en hulp tijdens de stage. Ook wil ik Dhr. Tim Dams en Dhr. Marc Smets danken voor de snelle en duidelijke antwoorden op mijn vragen en de informatie die ze voorzien hadden omtrent de bachelorproef.

Ik wil ook nog mijn dank uiten aan Artur Kondrashin de student die vorig academiejahr de basis heeft aangelegd voor de verdere uitwerking van deze bachelorproef en voor het snel en duidelijk beantwoorden van mijn vragen.

Tenslotte wil ook graag mijn dank uiten aan alle docenten van Artesis-Plantijn Hogeschool Antwerpen voor alle kennis en steun die ze aan ons hebben gegeven gedurende de opleiding. Dit was zeker onmisbaar tijdens het voltooien van deze bachelorproef.

Antwerpen, 30 augustus 2016

Michael Marivoet

Introductie

De opdracht van deze bachelorproef is het ontwerpen van een remote support & monitoring systeem voor technische installaties gebruik makend van open source technologieën. Deze bachelorproef situeert zich voornamelijk in twee vakgebieden, namelijk Internet of Things en webapplicatie development. Voor de webapplicatie ontwikkelen we zowel de [backend](#) als de [frontend](#).

Stagebedrijf

Het stagebedrijf waarvoor ik deze opdracht uitvoer is de EMT afdeling van het Agentschap Wegen en Verkeer, wat op zich weer een afdeling is van de Vlaamse overheid. De EMT afdeling of de afdeling voor Elektromechanica en [Telematica](#) is verantwoordelijk voor het volgende.

De projectstudie, realisatie, beheer en exploitatie van elektrische, elektromechanische en [telematica](#)-uitrustingen langs gewestwegen, langs waterwegen en waterlopen, in havens en op regionale luchthavens.
[Bron](#)

Project

Deze bachelorproef is een verdere uitwerking van een bachelorproef uit het academiejaar 2014-2015 door Artur Kondrashin. Tijdens dat academiejaar heeft men de basis gelegd van hoe dit systeem moet werken. Ook heeft men bepaalde functionaliteit ontwikkelt en uitgewerkt, die hergebruikt wordt tijdens de verdere ontwikkeling van het systeem. Er is dus voornamelijk onderzoek gedaan en een proof of concept uitgewerkt.

Het systeem moet dus over de volgende functionaliteit beschikken.

- Het monitoren en sturen van de installatie via een webapplicatie
- Toegang tot de installatie voorzien via enerzijds het uitlezen van het [nationaalnummer](#) van een elektronische identiteitskaart en anderzijds moet dit ook mogelijk zijn via de webapplicatie.
- Technici ter plaatste van ondersteuning voorzien via een video/audio chat functie.
- De webapplicatie moet de geschiedenis van de status van de installatie kunnen weergeven.

Tijdens deze bachelorproef gaan we dit proof of concept verder uitwerken tot een volwaardig systeem. Hiervoor gaan we bekijken welke technologieën en technieken er gebruikt zijn bij het proof of concept en of we deze kunnen vervangen en/of aanpassen om het systeem te verbeteren.

Technisch verslag

Technologieën

Zoals bij de introductie besproken heeft het systeem dat we gaan ontwikkelen de volgende minimum functionaliteit nodig. Dit moet geheel ontwikkelt worden door uitsluitend gebruik te maken van open source technologieën. Tevens werken we feitelijk verder aan een systeem waarvan men al een proof of concept heeft voor ontwikkelt.

Dit systeem moest beschikken over de volgende functionaliteit.

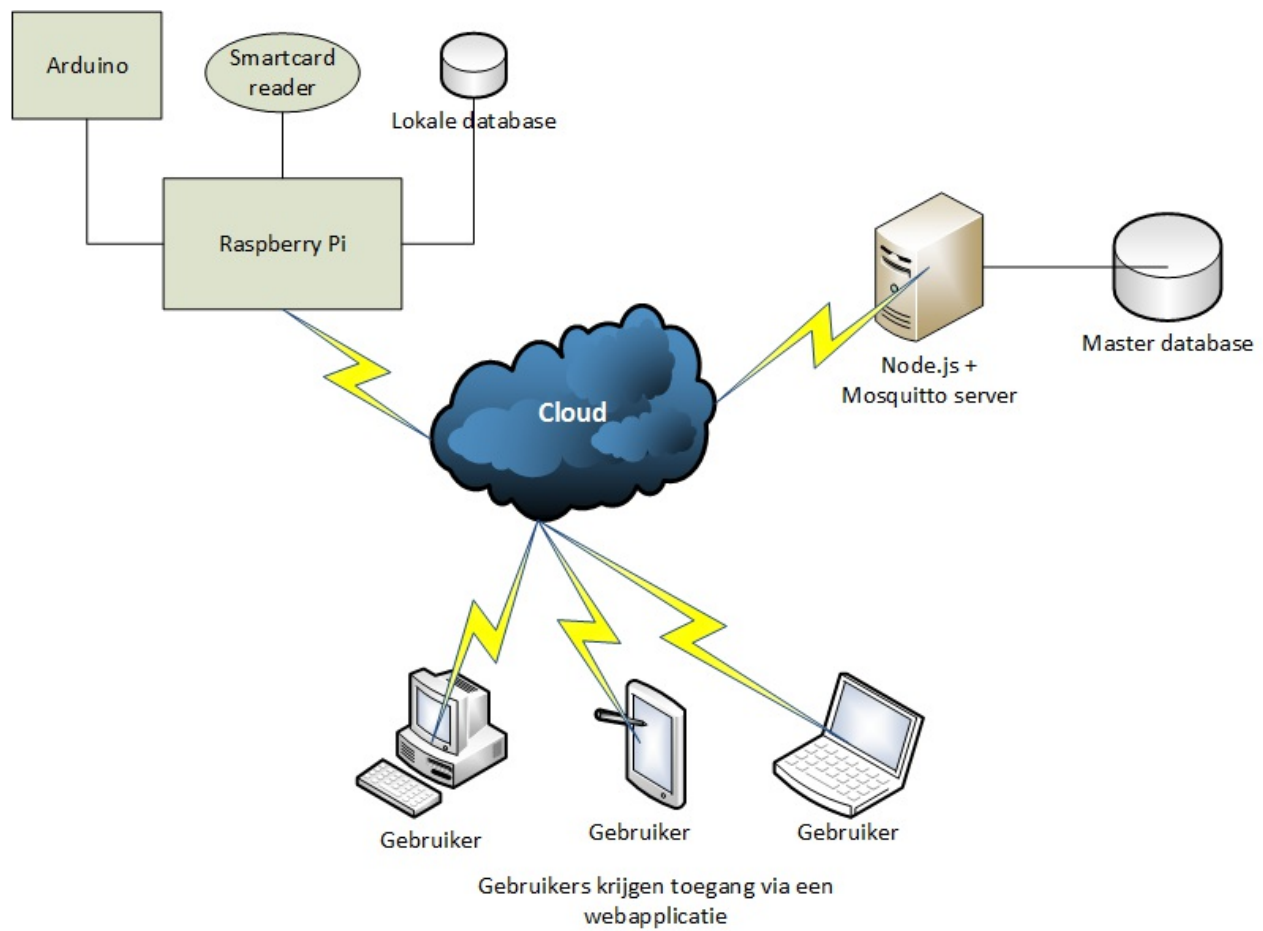
- Het monitoren en sturen van de installatie via een webapplicatie
- Toegang tot de installatie voorzien via enerzijds het uitlezen van het **nationaalnummer** van een elektronische identiteitskaart en anderzijds moet dit ook mogelijk zijn via de webapplicatie.
- Technici ter plaats te van ondersteuning voorzien via een video/audio chat functie.
- De webapplicatie moet de geschiedenis van de status van de installatie kunnen weergeven.

Als we dit geheel bekijken kunnen we afleiden dat het systeem uit 3 grote delen gaat bestaan. Deze zijn respectievelijk een hardware (**IoT**) apparaat dat we in installaties kunnen installeren, een server zijde waar we zowel data van het hardware apparaat ontvangen als ernaar versturen, dit doet de server ook voor de webapplicatie en tenslotte de webapplicatie zelf waar gebruikers toegang krijgen tot de data en functionaliteit van het systeem. Voor elk aspect van ons systeem hebben we bepaalde technologieën en hardware nodig.

Hieronder gaan we de 3 delen van deze bachelorproef individueel bespreken. We bespreken welke technologieën er zijn gebruikt en wat we juist hebben geïmplementeerd.

Diagram van het systeem

Dit diagram geeft weer hoe het systeem er ongeveer zal uitzien.



Hardware (IoT apparaat)

De hardware of IoT apparaat moet over verschillende vormen van functionaliteit beschikken. Hieronder vindt men een opsomming van de beoogde functionaliteit.

- Toegang verlenen tot de installatie door middel van eID kaart uit te lezen en het nationaalnummer na te kijken. Alsook via een webapplicatie een commando ontvangen dat ook toelaat de deur van de installatie te ontgrendelen. Deze data moet ook gelogd worden naar de server.
- Opmeten van de analoge waardes en deze versturen naar een database op een server.
- Kunnen inlezen van I/O signalen en deze naar een database op een server versturen.
- Kunnen aansturen van I/O poorten via commando's die door de server worden verstuurt.
- Video en/of audio chat voor ondersteuning tussen webapplicatie en het IoT apparaat.

Overigens moeten we ook in staat zijn tegen een bepaalde samplerate de digitale I/O en analoge waarden naar de server te sturen. Ten laatste wordt ook het externe en interne IP adres van het hardware apparaat geregeld verstuurt naar de database van de server.

Verderzetting project vorig academiejaar

Zoals al eerder vermeld is deze bachelorproef een verdere uitwerking van een proof of concept dat vorig academie jaar (2014-2015) al deels werd uitgewerkt in samenwerking met het stagebedrijf. Men heeft toen gekozen om met een Raspberry Pi (B+) te werken, we hebben hiermee verder gewerkt. Er is toen ook geopteerd om een Arduino te gebruiken voor het inlezen van de analoge / digitale waardes en het aansturen van aangesloten apparatuur via de Arduino zijn I/O poorten.

Arduino

Het Arduino model dat we hier gebruiken is de Arduino Uno. De Arduino is vanuit een technisch perspectief een onderdeel van het [IoT](#) apparaat, mits dat het buiten de functie van uitlezen analoge/digitale waarden en het aansturen van aangesloten apparatuur via zijn [GPIO](#) poorten geen enkele andere functie heeft. De Arduino beschikt wel over zijn eigen programma dat reageert aan de hand van vooraf ingestelde commando's. Deze commando's worden verstuurd door de Raspberry Pi over een [UART](#) verbinding.

De Arduino code is volledig opnieuw geschreven te opzichte van vorig academiajaar. De code is zo geschreven dat men enkel de volgende constants moet aanpassen om het te kunnen laten werken op bijvoorbeeld een Arduino Mega.

Constants

```
const int digitalInputs[] = {2,4,7,8,12}; //digitale input die gebruikt kunnen worden!
const int analogInputs[] = {A0,A1,A2,A3,A4,A5}; //adc poorten
const int digitalOutputs[] = {3,5,6,9,10,11,13}; //digitale output (PWM output) die gebruikt kunnen worden!
const int maxCommandsArgs = 3; //maximum argumenten per commando
```

Commando layout

Hieronder ziet u hoe de commando's eruit moeten komen te zien. De Arduino blijft karakters inlezen tot het karakter ';' voorkomt. Door het ';' karakter weet de Arduino dat het huidige commando ten einde is en zal hierna aan de hand van het commando de correct code uitvoeren.

```
Commando layout => [commando 1],[commando 2],...,[commando n];
Maximum mogelijkheden omtrent commando argumenten is afhankelijk van de variabele maxCommandsArgs
Dit commando zal de LED op pin 13 doen branden, wanneer we de 255 op het einde vervangen met nul dan zal de LED uitgaan.
Voorbeeld: 'write,13,255;'

Wanneer men een fout commando verstuurt, zal men de response "invalid" terugkrijgen!
```

Beschikbare commando's

Hieronder ziet u de commando's die momenteel beschikbaar zijn indien het Arduino programma geladen is.

```
write,[pin nummer],[0 tot 255]; => digitale output te veranderen (PWM) [opmerking! HIGH of LOW wordt niet geaccepteerd!]
test; ==> geeft test response terug
read,[digital of analog]; ==> geeft de waarden van de digitale of analoge pinnen terug in csv formaat over UART
```

Alhoewel het gebruik van een Arduino perfect mogelijk is, zeker bij het oplossen van het probleem dat een Raspberry Pi wel over [GPIO](#) pinnen beschikt maar niet over [ADC](#) poorten. Toch zou ik voor verdere kosten te besparen en het niet verkwesten van de beschikbare [GPIO](#) pinnen op de Raspberry Pi een [ADC SPI](#) module kiezen in plaats van een Arduino. Een goed voorbeeld hier hiervan is de [MCP3008](#). Deze kost een fractie van de prijs van Arduino en beschikt over 8 maal 10-bit kanalen, het heeft tevens een kleinere vormfactor en zal ook minder energie consumeren.

Raspberry

Tijdens de eerste 8 weken van deze bachelorproef is er gebruik gemaakt van een Raspberry Pi B+ als model, later zijn we overgeschakeld naar het model Raspberry Pi B2. Dit kwam omdat het eerste model geen ondersteuning heeft voor WebRTC via de [UV4L](#) library, meer informatie hierover verder in het paragraaf. We maken gebruik van [Raspbian](#) als [OS](#) voor de Raspberry Pi, dit is de standaard [OS](#).

Om [Raspbian](#) te kunnen installeren moeten we een [disk image](#) van de Raspberry Pi website downloaden en deze installeren met Win32 Disk Imager. Deze zal de image kopiëren naar het microSD kaartje van de Raspberry Pi. Win32 Disk Imager laat ons ook toe om een back-up te nemen van het gehele microSD kaartje.

Hieronder kan u delen code terugvinden die de werking van de belangrijkste functionaliteit van het Python script doorlichten.

Uitlezen eID

Hieronder bevindt zich de code voor het uitlezen van het [nationaalnummer](#) van de [eID](#) kaart die zich in de smartcard reader bevindt.

We maken hier eerst een cardrequest aan. De cardrequest heeft de eigenschappen timeout en cardType. We geven geen timeout meer, dit wil zeggen dat de cardservice zal blijven wachten tot er een [eID](#) in de smartcard reader wordt geplaatst. De cardType krijgt de variabele cardtype mee, dit is de ATR code. De ATR (Answer To Reset) code is uniek aan elk type smartcard en dus ook voor de elektronische identiteitskaarten. De cardservice zal eerst de ATR uitlezen van de [eID](#) en indien deze dan gelijk is aan dat van de cardtype zal de cardservice stoppen met de thread te blokkeren. Waardoor de overige code kan uitgevoerd worden.

Hierna gaan we het uitgelezen [nationaalnummer](#) hashen. Na de hash wordt er gekeken in de [database](#) of de hash voorkomt en indien zo zal de overige code worden uitgevoerd die de het slot aanstuurt.

```
print("Ready for card!")

cardrequest = CardRequest(timeout=None, cardType=cardtype)
cardservice = cardrequest.waitforcard()

lcd.clear()
lcd.write_string("Reading card...")

# ==[ Code vorig academiejjaar ]==
cardservice.connection.connect()

data, sw1, sw2 = cardservice.connection.transmit(CONST_SELECT)
data, sw1, sw2 = cardservice.connection.transmit(CONST_COMMAND)

nn = "" #container voor nationaal nummer

for i in enumerate(data):
    if i[1] < 128:
        if i[0] > 60 and i[0] < 90:
            if i[1] > 47 and i[1] < 58:
                nn+=str(chr(i[1]))
# ==[ Code vorig academiejjaar ]==

nn = hashlib.sha512(nn.encode('utf-8')).hexdigest()
```

Hashen van nationale nummers

Alle nationale nummers worden gehasht (SHA512) bijgehouden dit zowel in de lokale als master [database](#). Dit wordt gedaan wegens reglementering omtrent het opslaan van nationale nummers/rijksregister nummers het is bij wet verboden deze in standaard tekst bij te houden zonder toestemming van de federale overheid.

Threading

Hier maken we voor onze smartcard reader functie een aparte thread aan, hierdoor kan deze draaien op de achtergrond zonder dat het de main thread zou blokkeren. Dezelfde code wordt opnieuw gebruikt voor onder andere een thread aan te maken die de sensor data verstuurt.

```
card_thread = Thread(target=smartCardAccess, args=(dbPath, lcd, client, arduino, config,))

print("Starting card reader thread!");
card_thread.start()
```

Publisen data

Indien het python script succesvol een [eID](#) kaart heeft uitgelezen zal deze een bericht publiceren naar de MQTT broker. De payload van dit bericht is in de vorm van een json object. Het bericht wordt naar een algemeen topic gestuurd. Op de broker server draait een Node.js MQTT client die een subscriptie heeft op dit topic en deze data in de [database](#) plaatst.

```
data = {
    'installationId': config.get("InstallatieData", "InstallatieNummer"),
    'nationalNumber': nn,
    'date': str(datetime.now())
}
client.publish("hardware/door/log", json.dumps(data), qos=2)
```

Publisen sensor data

Dit is de code die gebruikt wordt om via de seriële verbinding met de Arduino de status van de ingangen op te vragen en deze te verzenden via MQTT.

```
arduino.write("read,digital;".encode())
res_digital = arduino.readline().strip().decode('utf-8')

arduino.write("read,analog;".encode())
res_analog = arduino.readline().strip().decode('utf-8')
```

```

digital = res_digital.split(';')
analog = res_analog.split(';')
data = {
    'installationId': config.get("InstallatieData", "InstallatieNummer"),
    'D2': int(digital[0]),
    'D4': int(digital[1]),
    'D7': int(digital[2]),
    'D8': int(digital[3]),
    'D12': int(digital[4]),
    'A0': int(analog[0]),
    'A1': int(analog[1]),
    'A2': int(analog[2]),
    'A3': int(analog[3]),
    'A4': int(analog[4]),
    'A5': int(analog[5])
}

# Voor testing werd er een aparte topic gebruikt, normaal '_topic_live'
client.publish("hardware/sensors", json.dumps(data), qos=0)
print("Live publish")
if sample == _sample_rate:
    client.publish("hardware/sensors", json.dumps(data), qos=1)
    print("Sample publish")
    sample = 0

```

Python Eclipse Paho Project Setup

Hier maken we een MQTT client aan. We hebben de Mosquitto server ingesteld om gebruik te maken van authenticatie. Dit stellen we in met de "username_pw_set()" functie.

De "on_connect" en "on_message" variabelen krijgen een functie toegewezen die uitgevoerd worden indien de client connectie maakt met de broker of wanneer deze een bericht ontvangt. De "on_message" variabele is zeer belangrijk mits deze wordt gebruikt om te reageren indien we een bericht binnen krijgen op één van de topics waarop we geabonneerd zijn.

Hierna connecteren we naar de broker, we doen dit aan de hand van het host adres dat we opvragen aan het configuratie bestand. We gaan ook abonneren op het topic die uniek is voor de installatie, in deze topic gaat de webapplicatie het commando publishen.

Als laatste starten we de asynchrone loop die zijn eigen thread heeft. Deze gaat pollen naar inkomende berichten, opnieuw connecteren indien de connectie naar de broker is weggefallen, etc..

```

client = paho.Client()
client.username_pw_set("username", "paswoord")

client.on_connect = on_connect_func
client.on_message = on_message_func

client.connect(config.get("ServerAdressen", "WebServices"), int(config.get("ServerAdressen", "MqttPoort")))
client.subscribe("hardware/commands/" + config.get("InstallatieData", "InstallatieNummer"), qos=1);
client.loop_start()

```

Config file

Dit is de config file die we in de "/boot" folder van de Raspberry Pi plaatsen deze. Bij elke nieuwe installatie moet het unieke installatie nummer worden meegegeven. De server adressen moeten alleen worden aangepast indien deze ook werkelijk zijn veranderd.

```
#####  
# Config file voor Agentschap Wegen & Verkeer Remote control/support applicatie  
#  
# Laatste revisie: 04/06/2016  
#  
#####  
  
## Invullen bij elke nieuwe installatie  
[InstallatieData]  
### Installatienummer is verplicht!  
InstallatieNummer: A001  
  
## Invullen bij elke nieuwe installatie indien gewijzigd  
[ServerAdressen]  
WebServices: website.com  
MqttPoort: 8883
```

Hardware

- **Smartcard reader:**

Dit gebruiken we om de data ([nationaalnummer](#)) van een [eID](#) uit te lezen en te gebruiken voor het valideren van personen. Indien de persoon de bevoegdheid heeft om aan de installatie te werken zal er toegang verleent worden. De voorwaarden waaraan smartcard readers moeten voldoen is dat ze volledig voldoen aan de [ISO/IEC 7816](#) standaard en [CCID](#) protocol

Voor dit project wordt er gebruikt gemaakt van de [h'mc \(HMC-CRID01B\)](#) smart card reader.

- **Webcam (met microfoon):**

De [UV4L](#) beschikt over een driver om met UVC (USB Video Class) webcams te werken. UVC is een USB device class voor het apparaten die in staat zijn video te streamen, zoals webcams, digitale camcorders, etc.. Het zorgt ervoor dat apparaten die bij deze klasse horen aan bepaalde standaarden voldoen. [UV4L](#) kan enkel werken met apparaten die bij deze klasse horen.

Voor dit project wordt er gebruikt gemaakt van de [eSecure USB webcam](#). Deze webcam is UVC compatibel.

- **LCD scherm:**

Dit is een simpel LCD scherm om feedback te geven aan personen die zich aan de installatie bevinden. Het wordt gebruikt om onder andere weer te geven of het systeem aan het booten is, dat het klaar is om een [eID](#) uit te lezen, slot ontgrendeld is, etc..

Software

Installatie script

Voor het installeren van alle gebruikte libraries en (Python) packages kan men gebruik maken van het *install_resources.sh* script. Het voegt ook de [UV4L](#) libraries aan de sources list toe.

```
# Voorbeeld voor gewone packages
# =====
# U4VL toevoegen aan resources!
curl http://www.linux-projects.org/listing/uv4l_repo/lrkey.asc | sudo apt-key add -
mkdir /etc/apt/sources.list.d
echo "deb http://www.linux-projects.org/listing/uv4l_repo/raspbian/ wheezy main" >> /etc/apt/sources.list.d/UV4L.list

echo "Updaten resources!"
apt-get update -y
apt-get upgrade -y

# Packages array
packs=(
    # Alle packages hieronder
    libccid
    pcscd
    ...
    libpcsc-lite1
    libpcsc-lite-dev
    # Einde
)

for pack in "${packs[@]}"
do
    sudo apt-get install $pack -y
done
```

Gebruikte libraries

- [UV4L](#)

User Video 4 Linux. Dit is een collectie van V4L (Video 4 Linux) compliant drivers die cross-platform zijn. De belangrijkste drivers zijn de UVC-driver voor UVC (USB video class) apparaten. UVC-apparaten zijn apparaten die video kunnen streamen over een USB verbinding en die voldoen aan bepaalde standaarden. De webcam die we gebruiken is UVC compliant, zodat we de bovengenoemde driver kunnen gebruiken.

De andere driver die zeer belangrijk is de WebRTC driver deze laat toe video/audio chat te streamen zonder gebruik te maken van een webbrowser. Het is mij niet gelukt WebRTC aspect aan het werken te krijgen.

We hebben in totaal de volgende [UV4L](#) drivers nodig, om WebRTC te kunnen gebruiken.

- [uv4l](#)
- [uv4l-raspicam](#)
- [uv4l-raspicam-extras](#)
- [uv4l-server](#)
- [uv4l-uvic](#)
- [uv4l-server](#)
- [uv4l-webrtc](#)
- [uv4l-xmpp-bridge](#)

- **Python**

We gebruiken Python als programmeertaal voor de Raspberry Pi, we gebruikte Python 3.4 om het script te ontwikkelen. Vorig academiejaar was er gebruik gemaakt van Python 2.7 en deze heeft enkele syntax verschillen met versie 3.4.

Python is de standaard programmeertaal voor Raspberry Pi en wordt standaard met [Raspbian](#) meegeleverd.

- **SQLite**

Voor de lokale [database](#) gebruiken we een SQLite [database](#). Dit is een zeer lichte [database](#) die enkel resources gebruikt indien de [database](#) effectief gebruikt wordt. Dit komt omdat een SQLite databases niets meer zijn dan een bestand en hierdoor geen server nodig hebben zoals de meeste andere databases. Dit is zeer voordelig mits we met een Raspberry Pi werken, die over beperkte hardware resources beschikt. Vorig academiejaar werd er gebruikt gemaakt van een MySQL [database](#) op de Raspberry, dit had het nadeel dat er effectief een MySQL server op de achtergrond moest draaien indien we deze wouden gebruiken. Dit verbruikte een onnodige hoeveelheid resources van het systeem, mits er gebruik word gemaakt van een Raspberry Pi model B+ was dit een verspilling van resources.

De [database](#) wordt zogoed als enkel gebruikt voor het opslaan van en querying naar de nationale nummers.

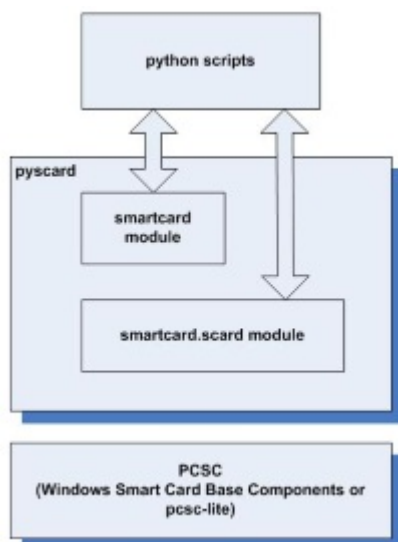
Python packages

- **pyscard**

Python package dat het mogelijk maakt om een smartcard reader te communiceren en aan te sturen. Het voegde dus smart card support toe aan python.

Pyscard bestaat uit 2 modules:

- Smartcard scard een extensie module dat als wrapper functioneert voor WinSCard API ook wel PC/SC genoemd (smartcard componenten)
- Smartcard een framework in python gebouwd boven op de PC/SC API



We hebben dus de PC/SC drivers nodig om gebruik te kunnen maken van pyscard. Deze zijn standaard niet geïnstalleerd op [Raspbian OS](#) en moeten dus geïnstalleerd worden, in totaal zijn er meerdere libraries/drivers nodig.

- Libusb library installeren:

```
$: sudo apt-get install libusb-dev
```


- Libccid package:

```
$: sudo apt-get install libccid
```

- Pcsclite package installeren

```
$: sudo apt-get install pcsclite
```

- Libpcsclite1 installeren:

```
$: sudo apt-get install libpcsclite1
```

- Libpcsclite-dev package installeren:

```
$: sudo apt-get install libpcsclite-dev
```

- Men kan naziën of er een smartcard reader is aangesloten met het volgende commando:

```
$: lsusb
```

- **paho-mqtt**

Dit package laat toe met een MQTT client in python te creëren.

MQTT is een ISO standaard publish/subscribe messaging protocol. Het is zeer licht in gebruik. Het wordt vooral gebruikt op plekken waar een kleine code voetafdruk en netwerk bandbreedte nodig is.

Ik heb gekozen om dit protocol te gebruiken om data te versturen naar de server. Het MQTT protocol is makkelijk te implementeren met een Mosquitto server en de libraries van het Eclipse Paho project.

Het Eclipse Paho project heeft MQTT libraries voor de volgende programmeertalen: C/C++, Java, JavaScript, Python, Go en C#.

- **pif**

Dit package laat toe om het externe IP adres van de host te bekomen. Dit gebeurt door externe webservices aan te spreken.

- **pyserial**

Pyserial laat ons toe om met seriële poorten te communiceren. Dit gebruiken we om met de Raspberry Pi te kunnen communiceren

- **rplcd**

Dit package laat ons toe het LCD scherm dat we gebruiken aan te sturen.

- **RPi.GPIO**

Dit package wordt gebruikt door rplcd om de [GPIO](#) pinnen te kunnen aanspreken. Het LCD scherm wordt via deze pinnen aangesloten op de Raspberry Pi

- **Requests**

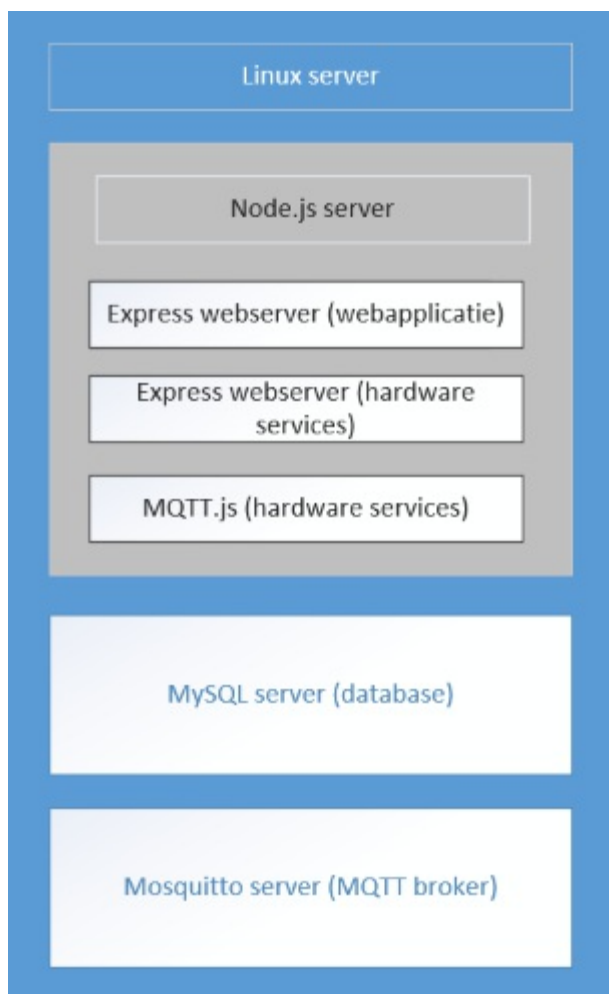
Requests is een python library die het makkelijker maakt om HTTP requests te versturen. Het wordt onder andere ook gebruikt door de Python Package Manager.

Server

Voor deze bachelorproef is er gebruik gemaakt van een virtuele linux pc om de server op te draaien. De server draait op een VMware Workstation. VMware is een Amerikaans bedrijf dat virtualisatie technologie software en services aanbied.

De server zelf bestaat uit 3 verschillende services, namelijk een Node.js service, MySQL service en Mosquitto service. Al deze services/servers zijn volledig open-source en kunnen in productie gebruikt worden.

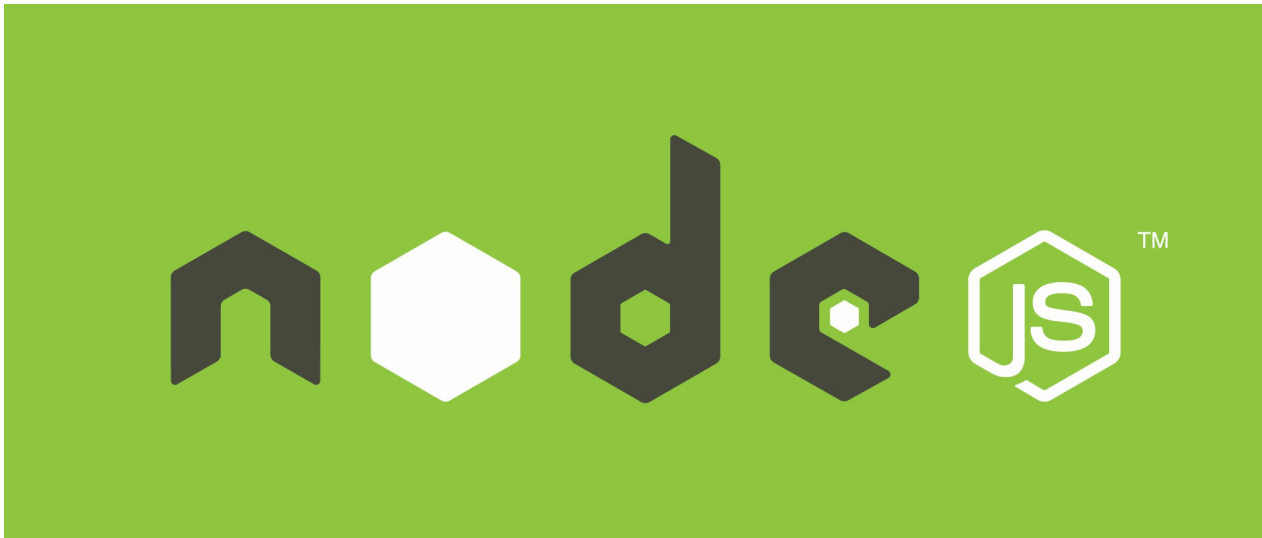
Server schema



Mosquitto

Mosquitto is een open-source MQTT broker die snel en makkelijk op te zetten is. De MQTT client gaan zich verbinden met deze server. We gebruiken deze dus voor de MQTT communicatie tussen de Raspberry Pi's en de server/webapplicatie.

Node.js



Node.js is een softwareplatform dat toelaat applicaties te ontwikkelen in javascript, men kan dus javascript server-side implementeren. De javascript code wordt niet uitgevoerd in de webbrowser maar in de Javascript-engine van Node.js. Node.js gebruikt de Javascript-V8 engine van Google.

Men kan de applicaties overal laten draaien zolang de Node.js runtime is geïnstalleerd. Node.js is beschikbaar voor zowel Windows, Mac OS X als Linux. Node gebruikt een event gedreven architectuur dat opzich gebruik maakt van asynchrone I/O.

Node.js wordt onder andere gebruikt door Microsoft, Netflix, PayPal, Yahoo, etc..

Node Package Manager

Dit is de standaard package manager van Node, het laat toe online packages/modules te downloaden en beheren. Deze kunnen op de [website van npm](#) opgezocht worden. De package manager wordt normaal standaard mee geïnstalleerd wanneer men Node.js installeert.

De Node package manager laat toe om een bestand aan te maken genaamd package.json wanneer men het onderstaande commando's gebruikt kan men een *package.json* voor het project aanmaken. Het *package.json* bestand houdt bij welke packages er zijn geïnstalleerd. Dit laat toe dat wanneer men de server bestanden wilt verplaatsen er enkel over de zelf ontwikkelde server files moet beschikken samen met het *package.json*.

Command line wizard om package.json in te stellen starten:

```
npm init
```

Node modules/packages aan package.json toevoegen

```
npm install [modules/packet] --save
```

Commando om alles benodigde packages voor het project terug opnieuw te downloaden

```
npm install
```

Gebruikte modules

Express

Express is een flexibel Node.js webapplicatie framework, dat een serie aan tools voorziet om makkelijk [backend services](#) te ontwikkelen. Express wordt vooral gebruikt om API's te ontwikkelen die door AngularJs kunnen worden aangesproken.

Hieronder bevindt zich een voorbeeld voor een Express API die een gebruiker verwijderd.

```
app.delete("/api/users/:userId", auth, function(req, res){
  dbPool.getConnection(function(err, conn){
    if(err){
      conn.release();
      res.status(503).send("ERROR");
      return;
    }
    conn.query("DELETE FROM users_db.users WHERE Id = ?", req.params.userId, function(err, rows){
      conn.release();
      if(!err){
        res.status(200).send("OK");
        return;
      }
      res.status(500).send("ERROR");
    });
  });
});
```

Express-session

Express session is een extensie die toelaat sessies te genereren als men gebruik maakt van het express framework. We gebruiken dit voor bij de webapplicatie een sessie aan te maken, indien een gebruikers zich aanmeld met een geldige gebruikersnaam en paswoord.

Hieronder vinden we de instellingen voor de onze sessies terug. De belangrijkste instellingen zijn:

- **Secret:** dit is de string die gehashed wordt een meegegeven wordt met onze cookie die we aanmaken bij de client. Het sessie geheim moet uniek zijn voor onze sessie.
- **Resave:** dit forceert het opslaan van de sessie data naar de sessie store. Kan best enkel gebruikt worden indien we in het programma handmatig de sessie variabelen gaan opslaan in plaats van dit automatisch te laten gebeuren.
- **SaveUninitialized:** zorgt ervoor dat de sessie niet wordt opgeslagen vooraleer deze gebruikt wordt en er effectief variabelen worden toegewezen.
- **Voor de cookies:**
 - **Secure:** Dit zorgt ervoor dat cookies alleen worden geïnitieerd wanneer er een HTTPS verbinding aanwezig is.
 - **maxAge:** Dit is de maximum leeftijd van een cookie. Dit geeft ook de maximale levensduur van de sessie weer, dat maxAge geen waarde mee te geven zal de cookie geen vervaldatum hebben. Het instellen van een cookie zijn maximale levensduur bepaald ook de levensduur van een sessie, indien de cookie vervalt dan vervalt de sessie ook.

De sessies zelf hebben een levensduur van 60 minuten, dit zorgt ervoor dat een sessie na 60 minuten vervalt indien er geen communicatie is met de server. Wanneer er wel communicatie is met de server zal de levensduur van de sessie verlengt worden.

```

``` javascript
app.use(session({
 name: 'awv_webapp',
 secret: process.env.SESSION_SECRET,
 store: mySqlSessionStore,
 resave: true,
 saveUninitialized: false,
 proxy: false,
 cookie: {
 secure: true,
 maxAge: process.env.SESS_COOKIE_MAXAGE
 }
}));
```

```

We maken dus een sessie aan bij het aanmelden, een sessie is pas geldig indien deze een variabele meekrijgt voor in de [database](#) store te plaatsen. Door "req.session.variabele" te doen kunnen we data toevoegen aan onze sessie, deze data wordt als een JSON string opgeslagen in de [database](#).

```

``` javascript
// Logica voor het aanmelden
if(result == true){
 req.session.loggedIn = true;
 req.session.userId = rows[0].Id
 req.session.username = rows[0].Voornaam + " " + rows[0].Achternaam;
 req.session.securityLevel = rows[0].SecurityLevel;
 data.answer = "Login geslaagd, u wordt doorverwezen!";
}
if(result == false){
 data.answer = "E-mail adres en/of paswoord zijn incorrect!";
}
// Logica voor het aanmelden af te ronden
```

```

express-mysql-session

Dit laat toe een [session store](#) te maken in een MySQL [database](#), standaard slaat express-session de sessions op in het RAM-geheugen. De sessions opslaan in een [database](#) is vele malen efficiënter.

mysql

Dit is [middleware](#) dat gebruikt wordt om Node.js te laten communiceren met een MySQL [database](#). We maken in onze applicatie gebruik van de connectie pooling om zo efficiënt mogelijk te werken met onze [database](#) en goede prestaties te verkrijgen.

dotenv

Dit package laat ons toe een environment file (.env) te parsen. Deze file wordt gebruikt zodat we niet hardcoded gegevens moeten plaatsen in onze server files, het zorgt er ook voor dat we maar één maal gegevens moeten aanpassen indien deze veranderd zijn.

```
```` ini
Env variabelen voor gebruik in server applicatie
#
Auteur: Michael Marivoet
Laatste revisie: 02/06/2016
#
#####

DB_HOST= 127.0.0.1
DB_USER= [username]
DB_PASS= [database]

SESSION_SECRET= [secret]

APP_PORT = 443

EMAIL_CONN_STRING = [connection string]

BCRYPT_SALTROUNDS = 14

De tijden hieronder zijn uitgedrukt in milliseconden
SESS_MAXAGE = 3600000
SESS_CLEAN_DB = 300000
SESS_COOKIE_MAXAGE = 14400000

Voor mqtt service
MQTT_USERNAME = [username]
MQTT_PASSWORD = [paswoord]
````
```

nodemailer

Nodemailer laat ons toe e-mails te versturen met Node en wordt gebruikt in onze applicatie om een recovery e-mail te versturen. Deze e-mail bevat een link naar de form om het paswoord opnieuw in te stellen.

Nodemailer laat het toe om met Node een verbinding te maken naar een SMTP e-mail server, voor het testen van de applicatie is er gebruik gemaakt van een gmail account.

MQTT

MQTT package voor Node, het laat toe een MQTT client aan te maken in node. Het wordt gebruikt om een service te creëren die naar algemene topics abboneert. Deze service gaat hierna de binnenkomende berichten verwerken. Dit wordt gebruikt voor sensor data op te slaan, deurlog aan te vullen en om het Ip adres up te daten.

pm2

Dit is een proces manager die gebruikt kan worden om Node te draaien, het zorgt er onder andere voor dat Node server onmiddellijk opnieuw wordt opgestart, houdt een log bij van wat de server doet. Het geeft ook de mogelijkheid om gebruikte resources, etc. te bekijken van de verschillende Node instances die aan het draaien zijn op de achtergrond.

Men kan ook meerdere type processen/programmeertalen regelen met pm2, zoals Python, Ruby, bash, etc..

Enkele belangrijke pm2 commando's zijn als volgtend:

```
``` bash
Start een Node.js applicatie en geeft deze de naam webserver en creeërt een watchdog die
de applicatie opnieuw start indien er applicatie bestanden worden aangepast.
$ pm2 start webapp.js --name="webserver" --watch

Stopt/herstart/verwijdert de webserver proces
$ pm2 stop webserver
$ pm2 restart webserver
$ pm2 delete webserver

Geeft een lijst weer van alle actieve processen.
$ pm2 list

Laat toe om in real-time de processen te monitoren.
$ pm2 monit
```
```

bcrypt

Bcrypt package wordt gebruikt om paswoorden veilig in de [database](#) op te slaan en te beschermen tegen [brute force](#) aanvallen op gebruikersaccounts.

Bcrypt beschermt tegen [brute force](#) aanvallen door een kost toe te voegen, deze kost is het aantal keer dat bcrypt de interne hash functie gaat uitvoeren. Deze kost wordt ook wel 'log rounds' of 'salt rounds' genoemd. Hoe hoger deze kost is hoe langer het duurt om de bcrypt string te genereren. De salt/log rounds waarde kan in de environment file worden aangepast.

Bcrypt genereert één enkele string die de volgende eigenschappen bevat: de versie van bcrypt die wordt gebruikt, het aantal salt/log rounds, salt en het paswoord.

Voorbeeld van een bcrypt string:

```
$2a$12$8vxYfAWCXe0Hm4gNX8nzwuqWNukOkcMJ1a9G2tD71ipotEZ9f80Vu

$bcrypt_id$log_rounds$128-bit-salt184-bit-hash
```

helmet

Helmet is [middleware](#) dat helpt bij het beter beveiligen van een webapplicatie dat gebruik maakt van Express. Helmet doet dit door automatisch verschillende HTTP headers in te stellen voor ons.

Helmet helpt te beveiligen tegen de volgende kwetsbaarheden:

```
```\n  - contentSecurityPolicy for setting Content Security Policy\n  - dnsPrefetchControl controls browser DNS prefetching\n  - frameguard to prevent clickjacking\n  - hidePoweredBy to remove the X-Powered-By header\n  - hpkp for HTTP Public Key Pinning\n  - hsts for HTTP Strict Transport Security\n  - ieNoOpen sets X-Download-Options for IE8+\n  - noCache to disable client-side caching\n  - noSniff to keep clients from sniffing the MIME type\n  - xssFilter adds some small XSS protections\n```\n
```

### bron

Onderstaande code geeft aan hoe we de helmet [middleware](#) toevoegen aan onze Node applicatie. Standaard zal 7 van de 10 middlewares geactiveert zijn. De volgende middlewares zijn inactief contentSecurityPolicy, hpkp en noCache.

```
```\n  javascript\n  var express = require("express");\n  var app = express();\n  app.use(helmet());\n```\n
```

body-parser

Door gebruik te maken van body-parser kunnen we de gegevens in de body van een request rechtstreeks omzetten naar onder andere een object, zoals dit het geval indien we body-parser instellen voor JSON request te verwerken. De JSON gegevens worden in het "body" object geparsed van de express request object, de request data kan dan via "req.body" worden aangesproken.

Voorbeeld:

```
```\n  javascript\n  app.use(bodyParser.json({type: '*/json'}));\n\n  app.get("/check-number", function(req, res){\n    res.send(req.body.number);\n  })\n```\n
```



## morgan

Morgan is een HTTP request logger. Het gaat informatie omtrent HTTP request die de server afhandelt uitprinten in console, dit is zeer handig tijdens het ontwikkelen van de applicatie en kan ook gebruikt worden om errors in HTTP requests te vinden indien de output van de Node.js server wordt bijgehouden in een log file.

```
webserver-0 POST /login 250 1144.135 ms - 2
webserver-0 GET /stage.michaelmarivoet.be 302 5.060 ms - 46
webserver-0 GET / 200 1.987 ms - 7680
webserver-0 GET /bower_components/bootstrap/dist/css/bootstrap.min.css 200 2.310 ms - 121260
webserver-0 GET /bower_components/angular-chart.js/dist/angular-chart.min.css 302 3.390 ms - 23
webserver-0 GET /css/sb-admin.css 200 1.950 ms - 3592
webserver-0 GET /font-awesome/css/font-awesome.min.css 200 3.023 ms - 21984
webserver-0 GET /javascript-plugins/browserMqtt.js 200 2.709 ms - 482851
webserver-0 GET /bower_components/jquery/dist/jquery.min.js 200 2.780 ms - 85578
webserver-0 GET / 200 1.614 ms - 7680
webserver-0 GET /bower_components/bootstrap/dist/js/bootstrap.min.js 200 2.531 ms - 36868
webserver-0 GET /bower_components/angular/angular.min.js 200 1.387 ms - 157919
webserver-0 GET /bower_components/angular-ui-router/release/angular-ui-router.min.js 200 1.317 ms - 32913
webserver-0 GET /bower_components/angular-smart-table/dist/smart-table.min.js 200 1.391 ms - 6704
webserver-0 GET /bower_components/ng-confirm/dist/ng-confirm.ignore.min.js 200 2.218 ms - 795
webserver-0 GET /bower_components/dygraphs/dygraph-combined.js 200 1.558 ms - 125468
webserver-0 GET /scripts/myApp.js 200 4.529 ms - 4333
webserver-0 GET /scripts/controllers/userCtrl.js 200 2.742 ms - 788
webserver-0 GET /scripts/controllers/dashboardCtrl.js 200 2.716 ms - 2128
webserver-0 GET /scripts/controllers/userAccountCtrl.js 200 3.456 ms - 4519
webserver-0 GET /scripts/controllers/usersOverviewCtrl.js 200 3.445 ms - 2516
webserver-0 GET /scripts/controllers/usersAddCtrl.js 200 2.429 ms - 4661
webserver-0 GET /scripts/controllers/validInstallationsOverviewCtrl.js 200 2.515 ms - 2667
webserver-0 GET /scripts/controllers/usersChangeCtrl.js 200 2.066 ms - 5231
webserver-0 GET /scripts/controllers/invalidInstallationsOverviewCtrl.js 200 2.691 ms - 3333
webserver-0 GET /scripts/controllers/installationTypesOverviewCtrl.js 200 8.379 ms - 3677
webserver-0 GET /scripts/controllers/installationTypePageCtrl.js 200 3.931 ms - 2277
webserver-0 GET /scripts/controllers/installationPageCtrl.js 200 5.032 ms - 1792
webserver-0 GET /scripts/controllers/installationPageControlCtrl.js 200 3.335 ms - 2161
webserver-0 GET /scripts/controllers/installationPageFeedCtrl.js 200 3.195 ms - 2307
webserver-0 GET /scripts/controllers/installationPageInformationCtrl.js 200 5.203 ms - 1864
webserver-0 GET /scripts/controllers/installationPageVideochatCtrl.js 200 1.463 ms - 546
webserver-0 GET /scripts/controllers/installationPageSensorHistoryCtrl.js 200 1.402 ms - 2682
webserver-0 GET /scripts/controllers/installationPageDoorHistoryCtrl.js 200 1.343 ms - 1000
webserver-0 GET /currentUser 200 1.443 ms - 49
```

Bovenstaande afbeelding is een voorbeeld van de output die morgan creeërt wanneer er HTTP requests worden verwerkt door de server. De output string bevat de volgende informatie: HTTP methode (POST, GET, etc.), URL, HTTP status code, response tijd en de grootte van de response. Het is ook mogelijk de output strings aan te passen zodat deze bijkomende informatie bevat zoals onder andere de tijd en/of datum van de requests, niet weergeven van requests met bepaalde HTTP status codes, etc..

# MySQL



MySQL is een open-source relationele [database](#) management systeem. Het laat toe om relationele databases te beheren en wordt onder andere gebruikt door Facebook, Google, Twitter, etc..

We maken gebruik van een MySQL server om onze primaire databases te beheren. De server bevat twee databases deze zijn als volgend, [database](#) voor gebruikers (users-db) en installatie (installaties-db).

De gebruikers [database](#) bevat de volgende tabellen:

- Een tabel voor de sessies deze wordt gecreëerd door de express-mysql-session dit is een toolset die toelaat dat express-session de [database](#) als [session store](#). Voor verdere informatie zie paragraaf 3.3.1 Node.js.
- Een tabel waarin alle gegevens omtrent de gebruiker wordt opgeslagen zoals naam, paswoord, adres, etc..
- Een tabel voor berichten (messages) in op te slaan. Het is mogelijk om berichten op het dashboard van de webapplicatie achter te laten, voor verdere informatie omtrent zie het hoofdstuk van de webapplicatie.

De installatie [database](#) bevat de volgende tabellen:

- Een installaties tabel met alle installaties die worden gebruikt en de gegevens van die installatie zoals onder andere adres, datum van eerste aanmelding, etc..
- Een installatie types tabel. Deze tabel bevat alle verschillende installatie types en de behorende gegevens van deze installaties zoals de gegevens en toepassingen van de Arduino poorten.
- Een meetgegevens tabel. Deze tabel bevat alle meetgegevens van actieve installaties.
- Een deurlog tabel. Deze tabel houdt bij wie de deur heeft geopend.

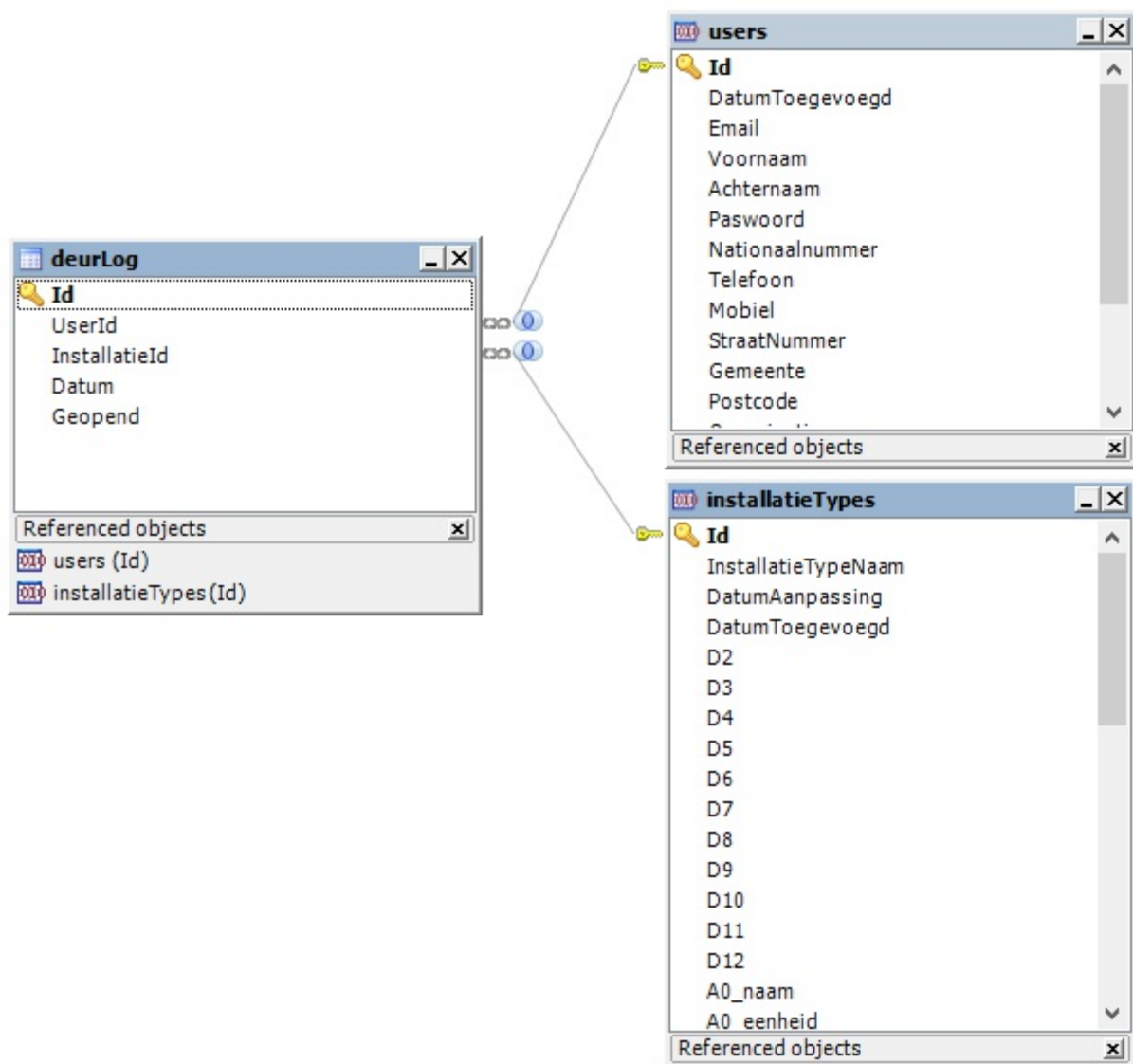
## MySQL workbench

Dit is een [database](#) ontwerp tool, deze maakt het mogelijk om via een visuele omgeving een [database](#) te managen. Het is gratis te downloaden en is ontwikkelt door [Oracle](#). [Oracle](#) is tevens de ontwikkelaar van MySQL.

## Ontwerp database

Omdat er gebruik gemaakt wordt van een relationele database gaan we relaties creëren tussen verschillende tabellen indien deze data van elkaar nodig hebben. Dit gebeurt doormiddel van foreign keys, dit zijn connecties van één tabel naar de primaire key van een andere tabel. Dit verzekert data integriteit tussen de verschillende tabellen en laat toe dat MySQL bepaalde acties onderneemt als er data wijzigt in de tabellen met de primaire key of omgekeerd.

Hieronder bevindt zich een voorbeeld met de deurlog tabel, deze maakt gebruik van data uit zowel de gebruikers als installatie database. Wanneer we bij de webapplicatie opvragen om de deurlog van een bepaalde installatie te bekijken zal deze gebruik maken van deze foreign keys om data van de gebruiker en installatie op te vragen.



# Webapplicatie

## Technologieën

### Bootstrap 3

Bootstrap is een framework met hulpmiddelen voor het maken van [frontend](#) webtoepassingen. Het is gratis, open source en maakt het ontwikkelen van frontends zeer makkelijk. Het is een verzameling van sjablonen gebaseerd op HTML en CSS. Dit project maakt gebruik van versie 3 van het framework.

Bootstrap beschikt over verschillende formulieren, knoppen, iconen en vele andere interfaceonderdelen. Dat vergemakkelijkt het ontwerpen van de [GUI](#) en zorgt voor uniformiteit tussen de verschillende webpagina's doordat alerts, knoppen, etc. telkens identiek zijn in design.

Bootstrap heeft tevens jQuery nodig om te kunnen functioneren, dus deze moet als dependency voorzien worden.

### Bootstrap sjabloon

Voor dit project wordt er gebruik gemaakt van een open source Bootstrap 3 patroon genaamd [SB Admin](#). De template is 100% compatibel met mobiele apparaten zoals tablets en smartphones.

### AngularJs

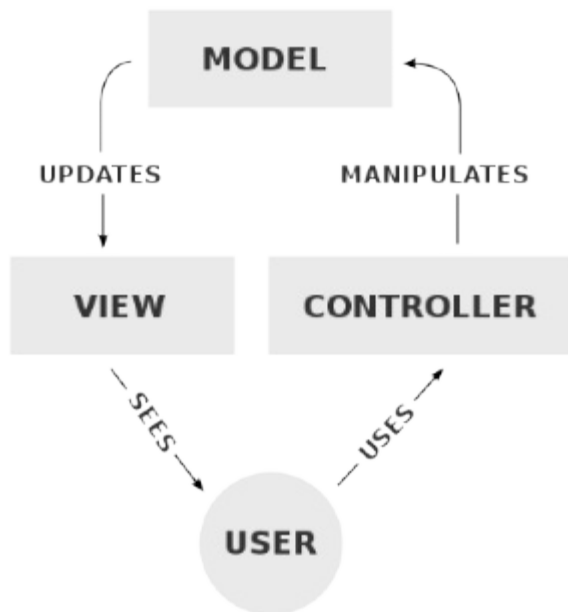
Dit is een open source JavaScript framework voor het helpen ontwikkelen van web applicaties dat mede ontwikkelt wordt door Google. Het is ontwikkelt om de problemen op te lossen die zich voor doen bij [single-page applicaties](#). Het wordt onder andere gebruikt door Intel, Sprint, NBC, etc..

AngularJS laat onder andere toe om webpagina elementen te manipuleren, het beschikt over data binding, etc.. Data binding is zeer handig mits dit toelaat variabelen rechtstreeks met HTML elementen te binden, hierdoor moet men enkel de variabele manipuleren in de controller en dit zal zich rechtstreeks in de view weergeven.

Het is gebaseerd op het MVC (Model-View-Controller) patroon. MVC patroon is een software architectuur patroon voor het implementeren van gebruikersinterfaces en is één van de populairste patronen voor het ontwerpen van web applicaties.

Het patroon wordt in drie delen opgesplitst:

- **Model:** Hier wordt de data waarmee de applicatie werkt opgeslagen.
- **View:** Hier wordt informatie weergegeven aan de gebruiker, hier worden geen gegevens verwerkt.
- **Controller:** De controller reageert op en verwerkt op events die meestal handelingen van de gebruiker zijn.



## Bower

Bower is een package manager voor [frontend](#) packages, het heeft verschillende overeenkomsten met npm en heeft Node.js + npm nodig om te kunnen functioneren (zie hoofdstuk 3.3.1 Node.js). Om bower op Windows te gebruiken moet ook [Git](#) voor Windows correct geïnstalleerd zijn.

Het laat dus toe om makkelijk alle libraries/packages/dependencies van de webapplicatie te downloaden en updaten. We gebruiken dit voor al de packages van de webapplicatie.

Net zoals bij npm maakt bower een json file aan waar alle dependencies worden opgeslagen, deze wordt 'bower.json' genoemd.

Installeert alle dependencies die in de bower.json file zijn opgeslagen.

```
$ bower install
```

Bij eerste gebruik maakt bower bij dit commando de 'bower.json' file aan en voegt de dankzij de '--save' parameter de dependency op in de file.

```
$ bower install [dependency name] --save
```

Verwijdert packages en slaagt die wijziging ook op naar de 'bower.json' file.

```
$ bower uninstall [dependency name] --save
```

## Overige libraries

### browserMqtt

Dit package is de browser versie van de MQTT client die we voor de Node.js server gebruiken, het beschikt over dezelfde functionaliteit. Voor verdere informatie zie hoofdstuk 3.3.1 Node.js.

# AngularJS modules

## angular-ui-router

Angular-ui-router is een router framework voor AngularJS [single-page applicaties](#), het laat toe om de browser URL te manipuleren wanneer de gebruiker door de webapplicatie navigeert. De gebruiker kan dan een bookmark/bladwijzer/favoriet maken van een locatie binnen de webapplicatie.

Een ui-router webapplicatie heeft een hiërarchische boom structuur van states, elke state kan over child states beschikken. Deze child states kunnen dan opzich ook weder over child states beschikken, etc..

Een state heeft minimum twee eigenschappen, namelijk een url en een template. Het is wel mogelijk om een naamloze rootstate (zonder url) te maken, maar dit kan maar éénmaal. Het toevoegen van een controller is optioneel.

Er zijn nog veel meer opties/eigenschappen die men aan een state kan meegeven zoals,

- Een state kan over meerdere templates en controllers beschikken.
- Callbacks voor als men navigeert naar de state, als wanneer met deze terug verlaat.
- ...

Voorbeeld van hiërarchische boom:

- De `$stateProvider` heeft opzich een state object waarin alle mogelijke states zich bevinden.
- De installatie pagina (`installationPage`) heeft één child state genaamd (`installationPage`).`control`.
- De installatie pagina heeft ook een variabele in zijn URL, namelijk `{id}`. Men kan dus variabelen in de URL plaatsen, door de `'$stateParams'` dependency toe te voegen aan de installatie pagina controller kan men dan deze variabele aanspreken.
- De `otherwise` eigenschap van de `$urlRouterProvider` object zal de webbrowser URL laten rerouten naar het dashboard indien de gebruiker een niet geldige state/URL gebruikt.

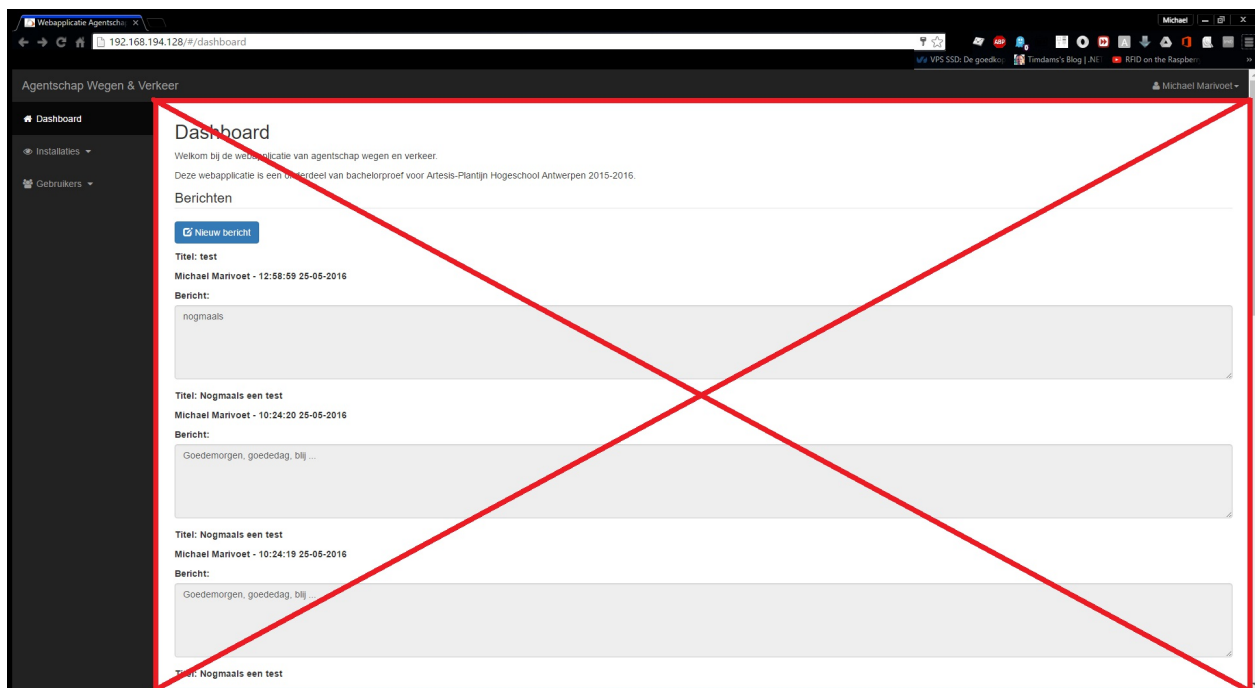
```
(function(angular){
angular
 .module("myApp", [
 "ui.router",
 "smart-table",
 "ngConfirm"
])
 .config(["$urlRouterProvider", "$stateProvider", function($urlRouterProvider, $stateProvider){
 $urlRouterProvider
 .otherwise("/dashboard")

 $stateProvider
 // Installation page
 .state("installationPage", {
 url: "/installaties/installatie/{id}",
 templateUrl: "templates/installationPage.html",
 controller: "installationPageCtrl"
 })
 // Installation page children
 .state("installationPage.control", {
 url: "/aansturen",
 templateUrl: "templates/installationPageControl.html",
 controller: "installationPageControlCtrl"
 })
 // Meerdere child states
 // End installation page children

 })
})(angular)
```

Het maakt de webapplicatie ook meer modulaair tijdens het ontwikkelen doordat elke navigatie link zijn eigen controller en view heeft die dan binnen een ui-view tag kan worden geladen. Dit maakt het zeer makkelijk om functionaliteit modulaair toe te voegen en onderhouden.

Onderstaande afbeelding toont waar de verschillende states worden ingeladen.



## smart-table

Smart table is een module om makkelijk data in tabellen te tonen aan de gebruiker. Het laat toe om items in de tabel te rangschikken, filteren, etc.. Een voordeel aan deze module is dat het geen andere dependencies nodig heeft dan Angular zelf.

Voorbeeld smart-table:

### Installaties

| Id                                  | Installatie nummer                  | Installatie type                    | Naam                                | Straat en nummer                    | Gemeente                            | Postcode                            | Bekijk               | Verwijder                 |
|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|----------------------|---------------------------|
| <input type="text" value="Zoeken"/> | <input type="text" value="Zoeken"/> | <input type="text" value="Zoeken"/> | <input type="text" value="Zoeken"/> | <input type="text" value="Zoeken"/> | <input type="text" value="Zoeken"/> | <input type="text" value="Zoeken"/> |                      |                           |
| 1                                   | A001                                | Generic                             | Test installatie 2                  | Kievietplein 112/113                | Antwerpen                           | 2000                                | <a href="#">Info</a> | <a href="#">Verwijder</a> |

## angular-cookie-law

AngularJS module die een popup banner genereert om aan te tonen dat de website gebruik maakt van cookies. Wordt gebruikt om conform te zijn met de reglementering van de Europese Unie.

## ngConfirm

Dit is een simpele module die aan Angular kan worden toegevoegd om aan de gebruiker te vragen of hij ervan zeker is om een bepaalde actie uit te voeren, zoals het verwijderen van een installatie.



# Functionaliteit webapplicatie

Hieronder zullen enkele aspecten van de functionaliteit van de webapplicatie worden toegelicht.

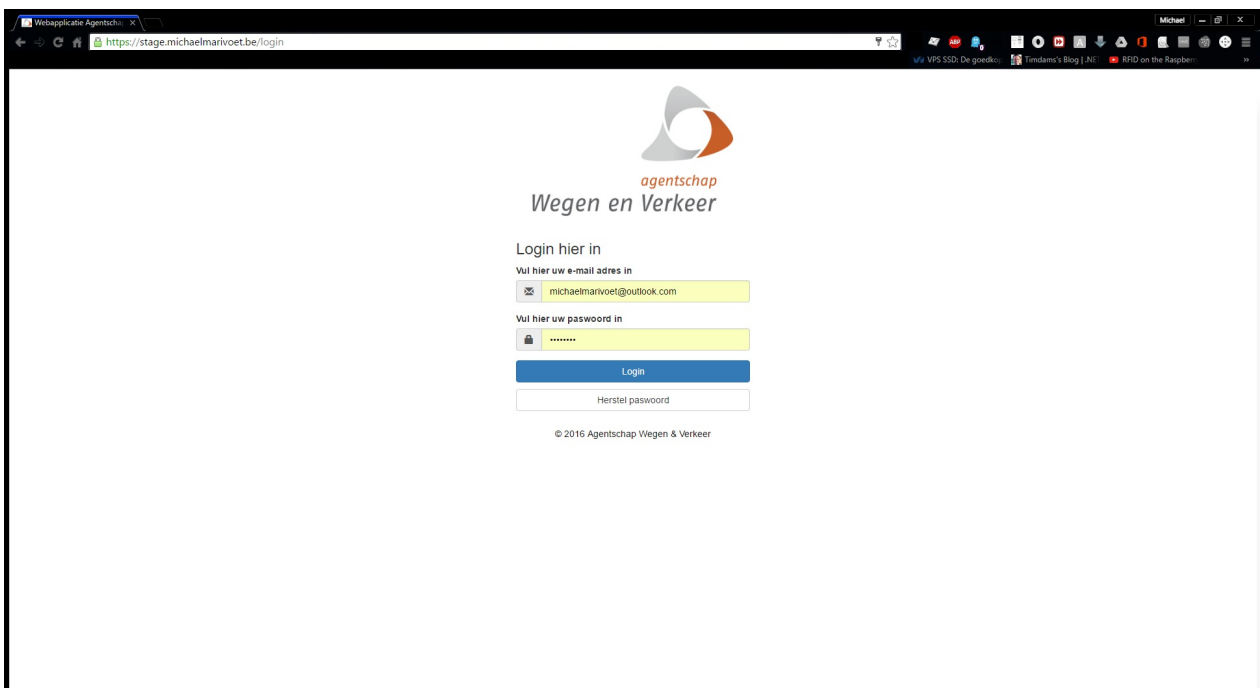
De webapplicatie beschikt over accounts met verschillende vormen van privileges/securitylevels. Momenteel is dit beperkt tot vier types.

1. Administrator (alle privileges)
2. Geen gebruikers manipuleren en/of toevoegen
3. Geen gebruikers en installaties manipuleren, men kan nog wel de installatie aansturen.
4. Enkel installaties en gebruikers uitlezen.
5. Gebruiker heeft geen rechten.

## Login en paswoordherstel

Hier krijgt de gebruiker de mogelijkheid om zich in te loggen en/of zijn paswoord te herstellen. Wanneer de gebruiker zijn e-mail opgeeft voor paswoordherstel dan zal deze een e-mail ontvangen met een link naar een webpagina waar men een nieuw paswoord kan aanvragen.

Wanneer de gebruiker succesvol is ingelogd zal hij op het dashboard terechtkomen.



The screenshot shows a web browser window with the URL <https://stage.michaelmarvoet.be/login>. The page features the logo of 'agentschap Wegen en Verkeer' at the top center. Below the logo, the text 'Login hier in' is displayed. There are two input fields: 'Vul hier uw e-mail adres in' with the email 'michaelmarvoet@outlook.com' entered, and 'Vul hier uw paswoord in' with a masked password '\*\*\*\*\*'. Below these fields are two buttons: a blue 'Login' button and a white 'Herstel paswoord' button. At the bottom of the page, the copyright notice '© 2016 Agentschap Wegen & Verkeer' is visible. The browser's address bar and various extension icons are also visible at the top of the window.

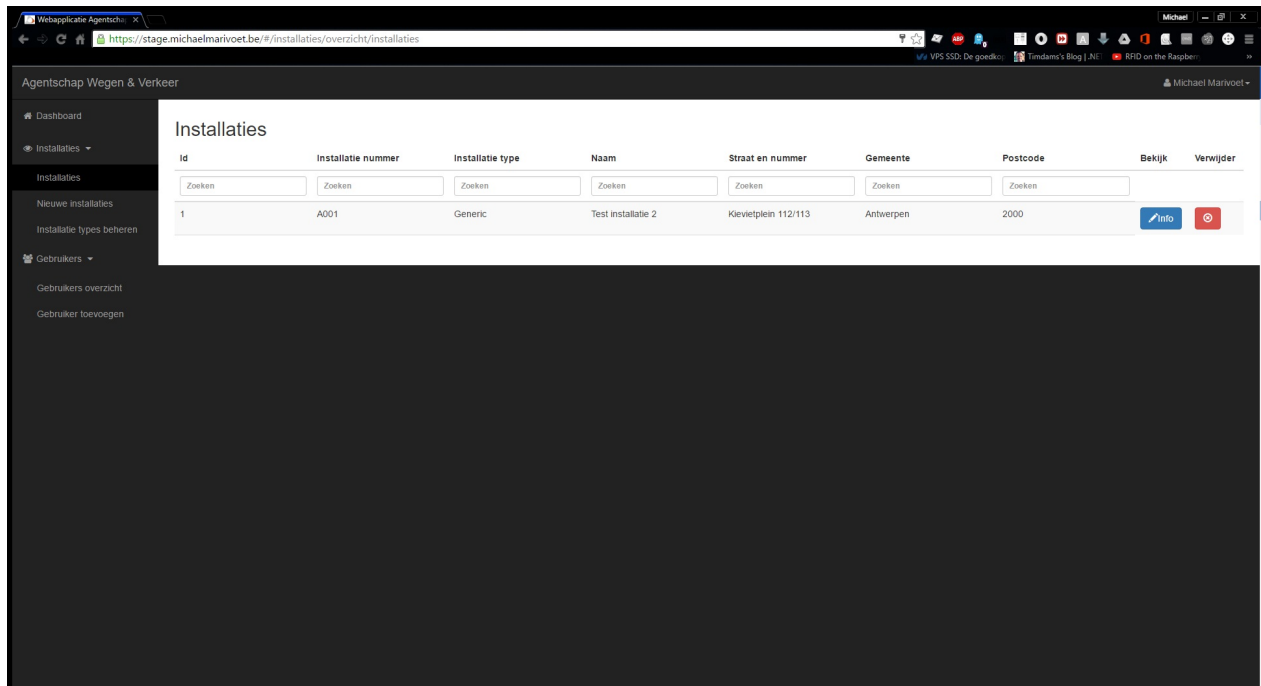
## Persoonlijke instellingen en uitloggen

De gebruiker kan rechtsbovenaan op zijn naam klikken, wanneer de gebruiker dit doet krijgt hij/zij twee opties. De optie om zijn instellingen bij te werken en/of na te kijken. De pagina met persoonlijke instellingen laat de gebruiker dan toe om zijn gegevens aan te passen en/of zijn paswoord te wijzigen. De tweede optie is om de uit te loggen en dus de huidige sessie te beëindigen.

## Installaties

### (Overzicht) Installaties

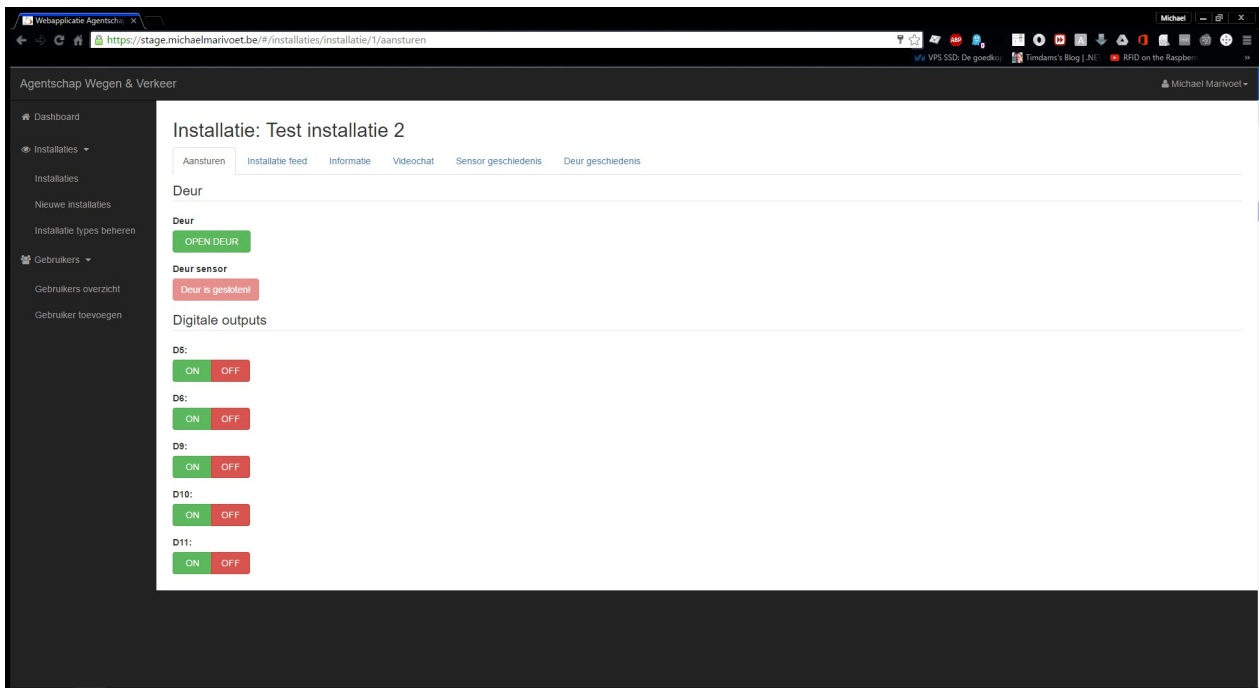
De gebruiker krijgt hier in tabelvorm alle actieve en goedgekeurde installaties te zien, door op de 'info' knop te klikken wordt er een nieuwe pagina geopend die verdere informatie en mogelijkheden omtrent de gekozen installatie weergeeft. Door op de rode knop met een kruis te drukken kan men een installatie verwijderen.



### Enkele (gekozen) installatie

Hieronder bevindt zich een voorbeeld van de pagina die wordt getoond voor één enkele installatie. De gebruiker krijgt enkele opties/tabs te zien die ieders een bepaalde functionaliteit hebben.

- **Aansturen:** Hier kan de gebruiker zien of de deur open/gesloten is en deze openen. De gebruiker kan hier ook de overige uitgangen van de Arduino aansturen.
- **Installatie feed:** Hier kan de gebruiker een aanvraag doen om een live feed te krijgen van de status van de installatie. De analoge poorten geven de eenheid weer van de installatietype waarmee de gekozen installatie wordt geassocieerd.
- **Informatie:** Hier kan de gebruiker algemene informatie over de gekozen installatie verkrijgen. Men kan onder andere de type Raspberry Pi, adres, naam, ip adres, etc. terugvinden en bepaalde gegevens aanpassen.
- **Videochat:** Er is momenteel geen videochat implementatie, mits dit deel van het stageproject niet geslaagd is.
- **Sensor geschiedenis:** Hier kan men een aanvraag doen om de status geschiedenis van de verschillende ingangen op te vragen en deze laten weergeven in een lijndiagram.
- **Deur geschiedenis:** Hier wordt een tabel gegenereerd met de gegevens van alle personen die succesvol de deur hebben laten openen.

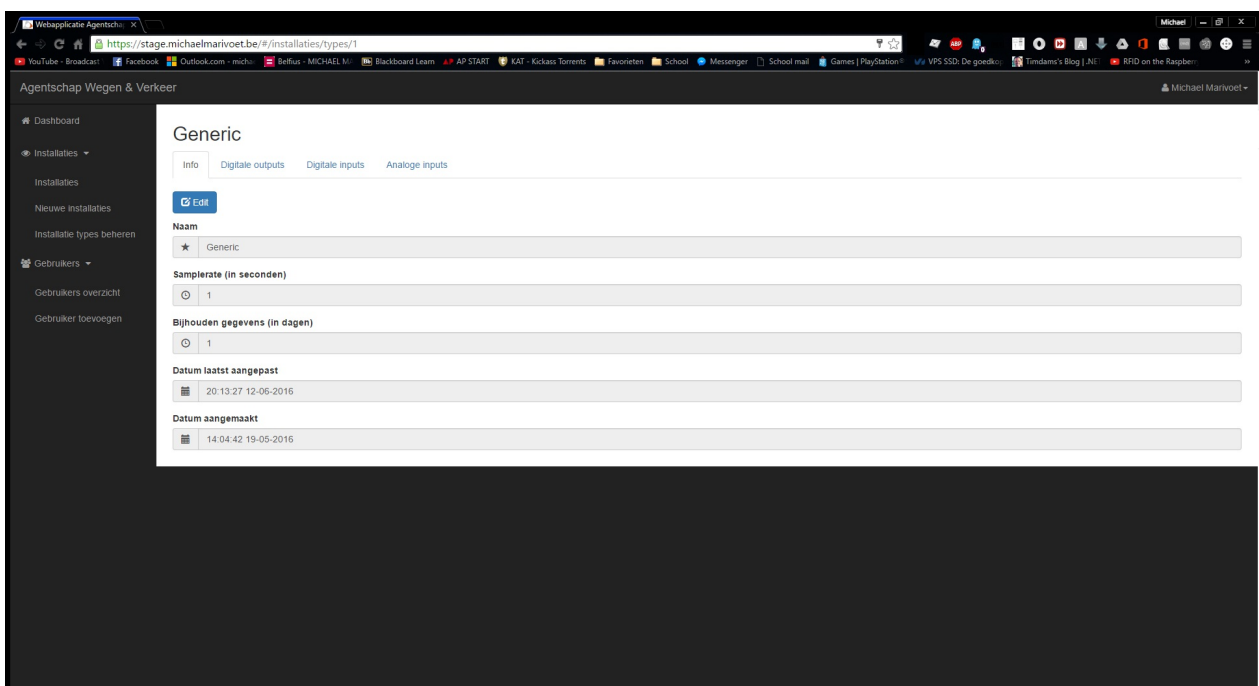


## Nieuwe Installaties

Dit overzicht is ongeveer gelijk aan dit van installaties, enkel hier is de extra optie om een installatie na te kijken en deze te valideren. Wanneer een installatie zich aanmeldt aan de server zal deze hier terechtkomen, voor veiligheidsredenen moeten alle nieuwe installaties eerste gevalideerd worden door een gebruiker. Dit komt omdat het aanmeldingssysteem geautomatiseerd is en dit extra controle biedt zodat men zeker is dat de installatie is toegevoegd en goedgekeurd voor gebruik.

## Installatietypes beheren

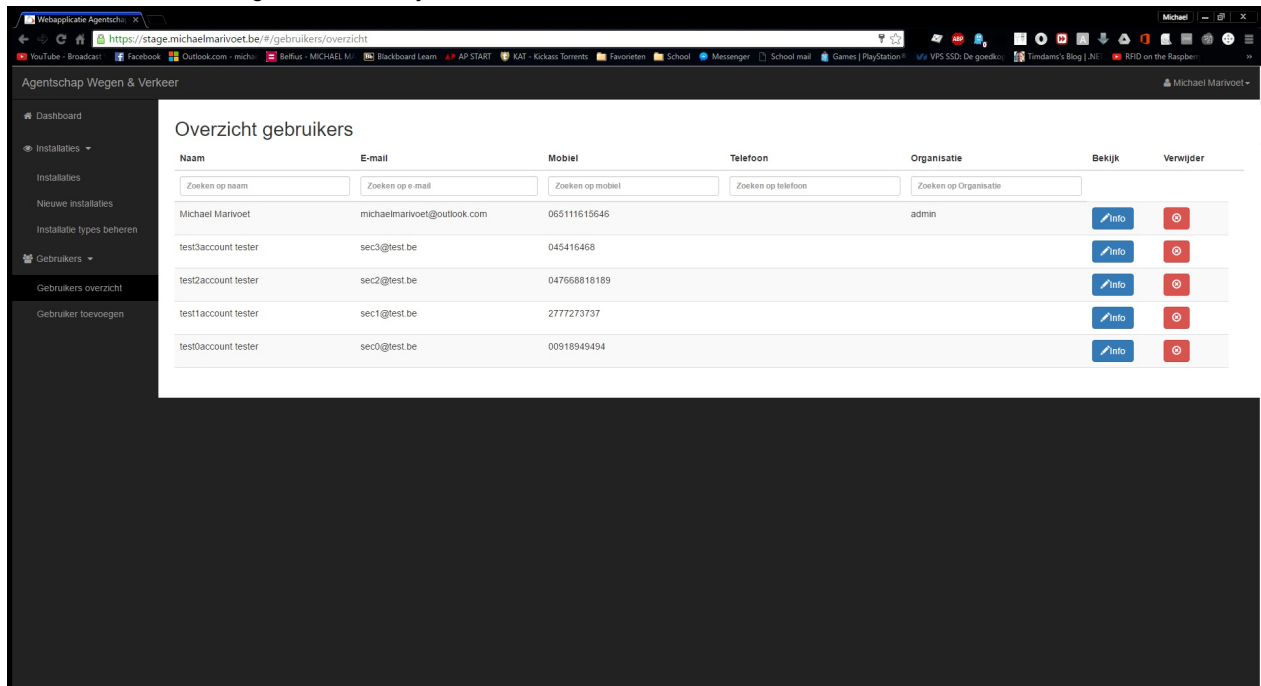
Men krijgt eerst een overzicht van alle installatietypes met de extra optie om een nieuw installatietype aan te maken. Dit overzicht is een tabel die te vergelijken valt met de installaties overzicht tabel. Wanneer men op de info knop klikt bij een installatietype dan navigeert de gebruiker naar een pagina waar men in staat is de installatietype zijn eigenschappen aan te passen.



## Gebruikers

### Gebruikers overzicht

Hier wordt in tabelvorm alle gebruikers weergegeven die zich in de [database](#) bevinden. Wanneer men op de info knop klikt krijgt men een nieuwe pagina voorgeschoteld waar aan de hand van de privileges waarover de gebruiker beschikt enkel de informatie van de gebruiker kan lezen of deze ook aanpassen. Het is ook mogelijk om het paswoord aan te passen indien men over de juiste privileges beschikt. Door op de rode knop met een kruis te drukken kan men een gebruiker verwijderen.



| Naam                | E-mail                     | Mobiel       | Telefoon | Organisatie | Bekijk               | Verwijder                 |
|---------------------|----------------------------|--------------|----------|-------------|----------------------|---------------------------|
| Michael Marvoet     | michaelmarvoet@outlook.com | 065111615646 |          | admin       | <a href="#">Info</a> | <a href="#">Verwijder</a> |
| test3account tester | sec3@test.be               | 045416468    |          |             | <a href="#">Info</a> | <a href="#">Verwijder</a> |
| test2account tester | sec2@test.be               | 047668818189 |          |             | <a href="#">Info</a> | <a href="#">Verwijder</a> |
| test1account tester | sec1@test.be               | 2777273737   |          |             | <a href="#">Info</a> | <a href="#">Verwijder</a> |
| test0account tester | sec0@test.be               | 00918949494  |          |             | <a href="#">Info</a> | <a href="#">Verwijder</a> |

### Gebruiker toevoegen

Hier kan een gebruiker met de juiste bevoegdheid gebruikers aanmaken. Een nieuwe gebruiker moet wel altijd over de volgende eigenschappen beschikken: e-mail adres, [nationaalnummer](#), voornaam/achternaam, mobiel telefoonnummer, securitylevel (deze zal gelijk worden aan geen rechten indien met dit blanco laat) en een paswoord.

# Verdere uitwerking/uitbreiding

## Videochat

De videochat functionaliteit is momenteel niet werkende, er zijn wel meerdere pogingen gedaan om zo'n functionaliteit te voorzien maar zonder resultaat. Uit onderzoek is gebleken dat een webRTC implementatie heeft effectiefste zou zijn omtrent comptabiliteit met de webapplicatie. De pogingen om webRTC te voorzien door middel van [UV4L](#) is niet gelukt en zou uiteindelijk moeilijk te implementeren zijn mits er telkens een reeks manuele stappen moet worden ondernomen om de drivers 100% te laten functioneren.

Tenslotte is er wel onderzoek gedaan richting het vinden van een oplossing voor het videochat probleem. Een mogelijke oplossing is om Node.js als cliënt te laten functioneren op de Raspberry Pi. Op het moment van schrijven zijn er verschillende Node packages die het toelaten een webRTC client aan te maken. Het zou dan verder ook mogelijk zijn de volledige Python applicatie om te zetten naar Node. Dit laatste is perfect mogelijk mits er packages zijn die ondersteuning aan Node toevoegen voor onder andere het communiceren met SQLite, seriële apparaten, etc.. Wegens tijdsgebrek was het niet mogelijk om dit nog te implementeren.

Node.js packages voor webRTC client:

- [rtc-everywhere](#)
- [webrtc-native](#)
- [node-webrtc](#)

[Stackoverflow discussie omtrent Node webrtc client](#)

## MQTT

Momenteel beschikt de MQTT broker over geen encryptie, dit is onveilig en zou aangepast moeten worden. Er deden zich problemen voor met het implementeren van encryptie bij de Mosquitto broker zowel bij gewone [websocket](#) als gewone connecties. De [websocket](#) connectie wordt gebruikt voor communicatie tussen de webbrowser en de MQTT broker. Er is gebruik gemaakt geweest van legale certificaten die door '[Let's Encrypt](#)' zijn gegenereert, dit was nodig voor de [websocket](#) connectie. Om toch encryptie te implementeren zou men eventueel moeten opteren om over te schakelen naar een andere broker.

Verder moet de manier waarop de Raspberry Pi verbinding maakt met de MQTT broker verbeterd worden, momenteel wordt voor elke Raspberry Pi hetzelfde account gebruikt om zich aan te melden. Dit zou zo moeten worden uitgewerkt dat de broker gegevens van de installatie uit de master [database](#) haalt, zodat die gegevens gebruikt kunnen worden door de Raspberry Pi om zich aan te melden. Hierdoor kunnen de aanmeldings gegevens uniek zijn voor elke installatie. Het implementeren van zo een systeem is mogelijk met de Mosca broker, het is bij de broker mogelijk om het authenticatie systeem zelf te ontwikkelen.

Alternatieve brokers:

- [emqtt \(erlang mqtt\)](#)
- [mosca \(Node.js broker\)](#)

# Conclusie

## Project

Dankzij de student die vorig academiejaar al de basis heeft ontwikkeld van de structuur van het project, kon hier makkelijk op verder gewerkt worden. Uiteindelijk is er wel gekozen om enkel fundamenteën omtrent het project van vorig academiejaar te wijzigen, er is onder andere gekozen om de PHP webapplicatie om te vormen in een moderner systeem in de vorm van een single-page webapplicatie die dan met REST API's communiceert. Ook is er gekozen om de Python code en Arduino code grotendeels opnieuw op te bouwen.

Het grootste deel van de door het stagebedrijf gevraagde functionaliteit is uiteindelijk geïmplementeerd. De grootste struikelblok is de videochat functionaliteit, hiervoor is er geen oplossing beschikbaar op dit moment.

## Stage

De stageperiode bij Agentschap Wegen & Verkeer was een leerrijke ervaring, het was zeer leuk om een langere tijd te kunnen werken aan één enkel project. Het was ook zeer interessant om een grotere applicatie te ontwikkelen met Node.js en AngularJS. Er waren wel wat problemen in het begin omtrent het begrijpen van de structuur en code van het project vorig academiejaar. Ik heb hiervoor enkele keer contact moeten opnemen met de student die vorig academiejaar aan het project had gewerkt.

Tenslotte ben ik tevreden met de behaalde resultaten, toch moet ik toegeven dat ik achteraf bepaalde aspecten van het ontwikkelingsproces anders zou hebben aangepakt. Ook zou ik nog opteren om van bepaalde technologieën te veranderen.

# Bronnen

1. [MQTT starters guide](#)
2. [Tutorial voor HTTPS met Node.js](#)
3. [MQTT best practices](#)
4. [UV4L](#)
5. [Forum Raspberry Pi - PS3 streaming](#)
6. [Mosquitto setup tutorial](#)
7. [Mosquitto website](#)
8. [Nodemailer](#)
9. [Express.js](#)
10. [Bcrypt node](#)
11. [Python wikia - instructies packages & beginners guide](#)
12. [Node.js](#)
13. [Eclipse Paho project \(MQTT\)](#)
14. [SB admin template](#)
15. [Pyscard](#)
16. [Helmet](#)
17. [Angula-ui-router](#)
18. [angular-cookie-law](#)
19. [MySQL workbench](#)

# Glossary

## ADC

Analog to Digital Converter, dit laat toe om analoge signalen in te lezen en om te zetten naar een digitale waarde. De digitale waarde is een getal, waarvan de grootte afhankelijk is van de range van de ADC.

[3.2.1. Arduino](#)

## backend

Het gedeelte van een webapplicatie waar aanvragen van de frontend worden verwerkt. Zoals het verkrijgen van gegevens uit een database. De backend is dus de feitelijke server van ons project.

[2. Introductie](#)   [3.3.1. Node.js](#)

## brute force

Brute force is een tactiek gebruikt door hackers om paswoorden van gebruikersaccounts te achterhalen doormiddel van het uitproberen van alle mogelijke combinaties tot men het juiste paswoord kan raden en hiermee in te loggen.

[3.3.1. Node.js](#)

## CCID

Chip Card Interface Device is een protocol dat toelaat smartcards te verbinden met een computer via een reader met een standaard USB interface.

## database

Digitaal archief waar data gestructureerd kan in opgeslagen worden, met een oog op het makkelijk raadpleging en aanpassing van deze data.

[3.2. Hardware](#)   [3.2.2. Raspberry](#)   [3.2.2.2. Software](#)   [3.3.2. MySQL server](#)   [3.3.1. Node.js](#)  
[3.4.2. Functionaliteit](#)   [4. Verdere uitwerking](#)

## disk image

Dit is een computerbestand dat een exact kopie van een opslagmedium zoals een hardeschijf, cd/dvd, etc. bevat. In dit project gebruiken we het voor onder andere Raspbian op de Raspberry Pi te installeren.

[3.2.2. Raspberry](#)



## eID

Elektronische identiteitskaart, dit is de nieuwe identiteitskaart waarover elke Belg sinds 2009 moet over beschikken. De kaart bevat een chip waarop er extra digitale informatie staat over de persoon in kwestie. Deze kaart gebruiken we voor toegang te verlenen aan bevoegde personen door het nationaalnummer uit te lezen.

[0. Abstract](#)   [3.2. Hardware](#)   [3.2.2.1. Hardware](#)   [3.2.2. Raspberry](#)

## frontend

Het gedeelte van een webapplicatie dat de gebruiker gebruikt om toegang te krijgen tot de services van de backend.

[2. Introductie](#)   [3.4. Webapplicatie](#)

## Git

Git is een open source versiebeheersysteem voor het managen van software projecten. Het wordt gebruikt tijdens de ontwikkeling van het stageproject.

[3.4. Webapplicatie](#)

## GPIO

General-purpose Input/Output. Poorten die zowel digitale waardes kunnen inlezen als uitsturen aan de hand van het gebruik.

[3.2.1. Arduino](#)   [3.2.2.2. Software](#)

## GUI

GUI oftewel Graphic User Interface is het visuele gedeelte van een applicatie waarmee de gebruiker in aanraking komt.

[3.4. Webapplicatie](#)

## IoT

Internet of Things oftewel het internet der dingen. Dit is het principe dat de meerderheid van apparaten op het internet bestaat uit intelligente apparaten. Deze apparaten gebruiken het internet om gegevens tussen elkaar en gebruikers uit te wisselen.

[3.2.1. Arduino](#)   [3.2. Hardware](#)   [3. Technologieën](#)

## Let's Encrypt

Service die toelaat om automatisch officiële certificaten te laten genereren voor TLS-encryptie.

#### [4. Verdere uitwerking](#)

## middleware

Middleware is een functie (code blok) die toegang heeft tot de request (req), response (res) en next objecten van een Express.js request/response cyclus. Het kan dus toegang krijgen tot de request eigenschappen en de response aanpassen. Het kan ook met behulp van de next functie doorverwijzen naar een volgende middleware functie die dan de request kan afhandelen.

#### [3.3.1. Node.js](#)

## nationaalnummer

Het nationalenummer of het rijksregisternummer is een uniek identificatienummer dat de Belgische overheid aan alle natuurlijke personen in België geeft. Het staat vermeld elk persoon zijn eID (elektronische identiteitskaart) dit zowel fysiek gedrukt op de eID als op de datachip van de kaart.

[2. Introductie](#)   [3.2. Hardware](#)   [3.2.2.1. Hardware](#)   [3.2.2. Raspberry](#)   [3. Technologieën](#)  
[3.4.2. Functionaliteit](#)

## Oracle

Oracle Corporation is een Amerikaans softwarebedrijf dat onder andere de ontwikkelaar is van MySQL en MySQL Workbench.

#### [3.3.2. MySQL server](#)

## OS

Operating system oftewel besturingssysteem.

[3.2.2. Raspberry](#)   [3.2.2.2. Software](#)   [3.3.1. Node.js](#)

## Raspbian

Raspbian is het standaard OS van de Raspberry Pi en is een op Debian gebaseerd besturingssysteem. Het grootste verschil met Debian is dat de kernel van Raspbian gecompileerd is voor gebruik met de ARM processoren van de Raspberry Pi.

[3.2.2. Raspberry](#)   [3.2.2.2. Software](#)

## samplerate

Dit is de snelheid (rate) waarmee metingen (samples) genomen worden.

[3.2. Hardware](#)

## session store

Wordt gebruikt om gegevens bij te houden van actieve sessies.

[3.3.2. MySQL server](#)   [3.3.1. Node.js](#)

## single-page applicaties

Dit zijn applicaties die uit één enkele pagina bestaan.

[3.4.1. Angular modules](#)   [3.4. Webapplicatie](#)

## SPI

Serial Peripheral Interface, dit is een synchrone seriële data link tussen tenminste 2 apparaten. Het maakt gebruik van een master/slave verhouding, er is dus altijd tenminste één slave en één master. De master kan kiezen met welke slave het wilt communiceren.

[3.2.1. Arduino](#)

## Telematica

Discipline die zich bezighoudt met de telecommunicatie en informatie, maar vooral omtrent de combinatie van ervan.

[2. Introductie](#)

## UART

Universal Asynchronous Receiver/Transmitter. Dit is een standaard dat toelaat data te verzenden tussen 2 apparaten over een seriële verbinding.

[3.2.1. Arduino](#)

## UV4L

User Video for Linux is een collectie van tools en drivers om (media) streaming toe te laten. Het heeft een grote compatibiliteit graad met de Raspberry Pi.

[3.2.2.1. Hardware](#)   [3.2.2. Raspberry](#)   [3.2.2.2. Software](#)   [4. Verdere uitwerking](#)

# websocket

Dit is een protocol dat full-duplex communicatie (communicatie in twee richtingen) toelaat over één enkele connectie.

## [4. Verdere uitwerking](#)