Accesstoken = i8k0j9xr9uk1bmrlx7j6

# Data naar data server

De data die de arduino genereerd moet naar een database verstuurd worden. Er is gekozen voor een mySQL database die op een losse dataserver zal draaien. De makkelijkste manier om hier data naar te versturen is door het maken van een script op deze server. Deze accepteert een parameter in de URL die het nieuwe aantal bevat. Echter is dit natuurlijk niet veilig omdat zomaar iedereen op deze manier de database aan kan passen. Een andere oplossing kan zijn een extra parameter meesturen waar een key in zit. Wanneer deze correct is kan het script iets aanpassen. Echter het veiligste is om een RESTful API te maken die deze zaken afhandelen. De manier waarop de arduino data zal verzenden is door een Ethernet module. Deze ethernet module zal naar een specifiek URL een request sturen met de juiste api call. Binnen deze api call zal de juiste data staan die wordt aangepast.

Deze API zal plaatsvinden op een LINUX server besturingssysteem. Ook zal hierop de MySQL database draaien waar de api mee zal communiceren. Vanuit buitenaf zal er een request binnen komen bij de server. Deze zal luisteren naar de request en hem doorsturen naar de database. Dat api zal geschreven worden door middel van Javascript, NodeJs als server logic en Express als http routing.

# Dataserver documentatie

Om de product informatie op te slaan wordt er gebruik gemaakt van een dataserver. Deze draait op een raspberry pi die is verbonden aan het netlab. Op deze manier kan er via een vpn verbinding connectie gemaakt worden met de server. Het ip adress van deze server is 5.104.119.149:8000. Op de server draait nodeJS en een distributie van mySQL. Via een package heb ik een nodeJS server gemaakt die luistert op poort 8000. Wanneer er een request binnen komt zal de backend dit afhandelen en routen naar de juiste database query.

Om de server veiliger te maken zijn er een aantal maatregelen getroffen. De eerste maatregel om indringers buiten te houden is een Public/ Private key pair te gebruiken. Dit is een manier om ongewilde gasten buiten te houden die het wachtwoord van een gebruiker weten of het wachtwoord brute forcen. Hoe dit werkt is: op de server staat een bestand genaamd “authorized\_keys” in dit bestand staat een string waarin staat wat voor type het keypair is en een daarna een public key. Deze is gekoppeld aan één specifieke private key die goed beveiligd staat op het device van de gebruiker. Deze key combinatie is aangemaakt op het device van de client door het volgende command

*ssh-keygen -t ecdsa*

Dit command zorgt er voor dat er een public en private keypair word gemaakt. Het ecdsa gedeelte is de manier waarop de encryptie werkt. Hierna word gevraag waar de keys opgslagen moeten worden. Dit is meestal de .ssh map. Ook moet er een naam van het bestand worden opgegeven. Deze is later nog belangrijk voor het verbinden met de server. Als laatste word er gevraagd een wachtwoord op te geven voor het bestand. Deze is ook belangrijk voor de connectie met de server.

Wanner de keys zijn gegenereerd op de client device kan de **public** key worden overgezet naar de server. Dit heb ik gedaan door op de server in het bestand authorized\_keys handmatig de public key te kopieren. Dit gebeurt door het volgende command

*Sudo nano ~/.ssh/authorized\_keys*

Dit command zorgt er voor dat het bestand kan worden aangepast.

Na getest te hebben dat het werkte met keyfile kon het ssh wachtwoord worden uitgeschakeld. Op deze manier kan er enkel doormiddel van een public key worden ingelogd.

Om nu verbinding te maken met de server moet het volgende command worden gebruikt

ssh -i ~/.ssh/superroute\_laptop\_thijmen-ecdsa [pi@5.104.119.149](mailto:pi@192.168.218.103)

Naast ssh beveiliging beschikt de server ook over een firewall die enkel de poorten toe staat die er gebruikt worden. Dit zijn poort 8000 voor de API en poort 22 voor de ssh verbinding. Het toestaan van poorten worden gedaan door het volgende command

*Sudo ufw allow <poort>*

# API documentatie

De data zal vanuit de arduino doormiddel van een Ethernet module naar de dataserver verzonden worden. Deze ethernet module zal een http request sturen naar het ip adress van de dataserver, deze is 5.104.119.149:8000.

## Getting started

Om verbinding te maken met de api heeft de client een acces token nodig. Deze moet worden verwerkt in de url van de api call. Wanneer deze correct is kunnen er verschillende URL’s worden opgevraagd die verschillende data opvragen of aanpassen.

## Use cases

De API heeft een aantal verschillende doelen. Het moet data weergeven binnen de mobiele app. Hier word alle data voor opgevraagd om dit in een lijst weer te geven. Daarnaast moet er een enkel product te zien zijn, Dus de mogelijkheid bestaat doormiddel van een ID een product op te vragen via de api waarna deze data in JSON vorm zal worden weergegeven. En ten derde zal de api gebruikt worden om data aan te passen. Voorraden moeten worden aangepast van specifieke producten. Dit kan worden gedaan doormiddel van een PUT request.

## URI

Het eerste endpoint is om alle producten in een keer op te vragen. Dit zal gebeurt door een GET request te sturen naar het IP adres met het *allProducts* keyword. \*De api is Case sensitive

[*Http://5.104.119.149:8000/<AccesToken>/allProducts*](Http://185.113.172.65:8000/%3cAccesToken%3e/allProducts)

Het tweede endpoint is om een enkel product op te vragen aan de hand van een ID. Dit id zal meegegeven worden over de url als parameter.

[*Http://* *5.104.119.149:8000/<AccesToken/getById/<id>*](Http://192.168.218.103:8000/%3cAccesToken/getById/%3cid%3el)

Het derde en laatste endpoint is om een voorraad van het product aan te passen. Dit word gedaan door wederom het ID mee te sturen over de url, Echter word get dit keer ook een body mee gestuurd over het http request. In de body word in JSON formaat het aantal meegegeven.

[*Http://* *5.104.119.149:8000/<AccesToken/update/<id>*](Http://192.168.218.103:8000/%3cAccesToken/update/%3cid%3e/l)

*Body:*

*{“Aantal”: <Aantal>}*