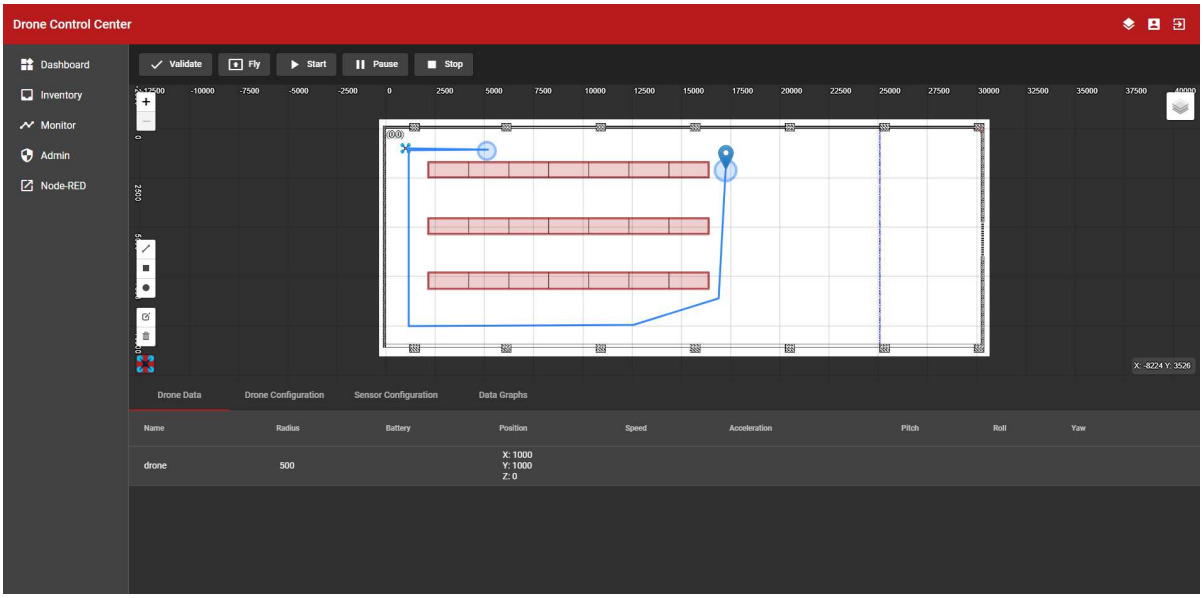
PATH PLANNER VOOR INVENTARIS MET AUTONOME DRONE



SPRINT 2

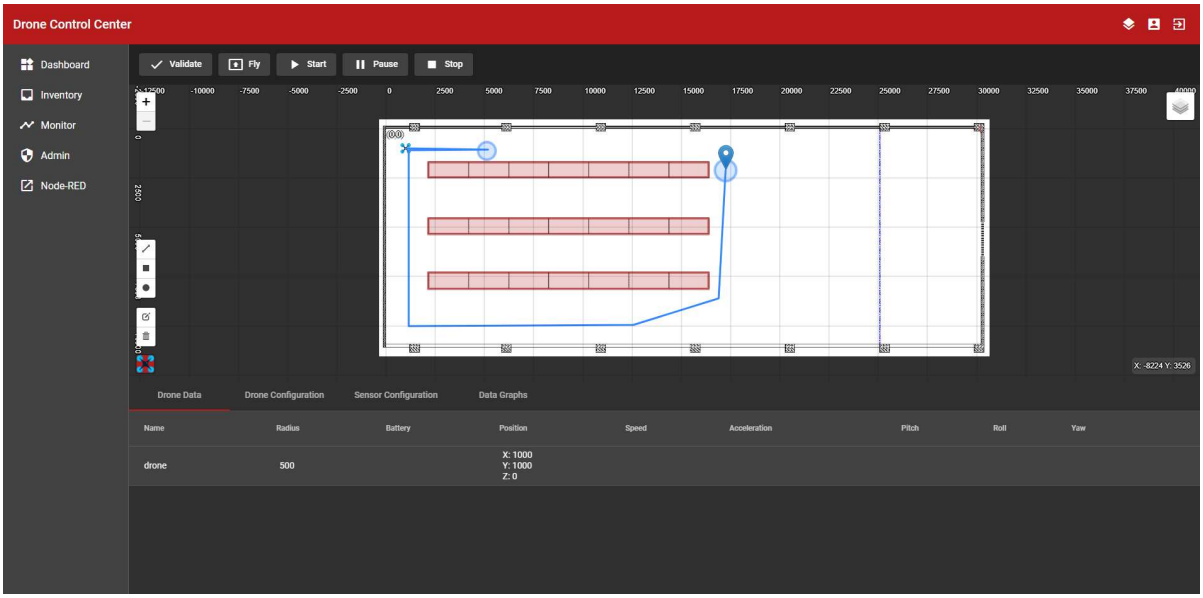
LEDEN: ROBBE DE SUTTER, KAREL EVERAERT, JOCHEN LAROY, ROEL MOEYERSOONS, WOUTER STEMGÉE

TEAM: DRONE1

Projectwebsite: http://bpvop4.ugent.be:8081/ 25/04/2019

UNIVERSITEIT GENT

PATH PLANNER VOOR INVENTARIS MET AUTONOME DRONE



SPRINT 2

LEDEN: ROBBE DE SUTTER, KAREL EVERAERT, JOCHEN LAROY, ROEL MOEYERSOONS, WOUTER STEMGÉE

TEAM: DRONE1

WOORD VOORAF

KORTE SAMENVATTING

Het IDLab is opzoek naar een manier om de inventarisering van grote warenhuizen en magazijnen te automatiseren zodat dit de hoge kostprijs, de kans op menselijke fouten en de gevaarlijke situaties voor de arbeiders kan verlagen.
Dit zou via drones verholpen kunnen worden, maar deze volgen momenteel enkel maar een manueel ingegeven pad aan de hand van coördinaten.
Het doel van deze proef is om een webgebaseerde interface en controle paneel te ontwikkelen waar de drone kan gemonitord en bestuurd worden en de volledige inventaris van het magazijn kan weergegeven en aangepast worden.
Hiervoor wordt gebruik gemaakt van de MEAN stack in combinatie met Node-RED om deze dynamische web applicatie te maken. MEAN is een acroniem dat staat voor MongoDB (een noSQL database), Express (een framework voor webapplicaties dat met Node.js werkt), Angular.js (een javascript Model-View-Controller framework) en Node.js (een server-side omgeving voor JavaScript). Door deze keuze is het mogelijk om het project modulair op te bouwen zodat er later extra functies kunnen toegevoegd worden.
Het eindresultaat is een modulaire webgebaseerde interface waarop de drone en het magazijn kan gemonitord worden.

INHOUDSOPGAVE

overzicht figuren

1 INLEIDING

1.1 Context

Het concept van deze opdracht komt van het IDLab – Universiteit Gent. Zij zijn bezig aan een project om drones, die momenteel al op de markt zijn, van extra functionaliteiten te voorzien om deze in te zetten voor het automatiseren van het inventariseren van magazijnen of grote warenhuizen.
Het manueel inventariseren van een groot warenhuis of een magazijn vergt namelijk heel veel werk, veel tijd en veel personeel. Dit brengt dus een hoge kostprijs met zich mee en een kans op menselijke fouten of zelfs gevaarlijke situaties voor werknemers wanneer deze bijvoorbeeld een hoog gestapeld product moeten scannen.
Om dit te vermijden zou het IDLab dus gebruik kunnen maken van drones die op elk willekeurig tijdstip (bijvoorbeeld 's nachts) alle producten kunnen scannen aan de hand van een QR-code of een RFID-tag, zonder dat deze drone door een werknemer zou moeten bestuurd worden. De drone zou dus in staat moeten zijn om zelfstandig een getekend pad af te vliegen en hierbij de producten die hij onderweg tegenkomt te scannen, om zo het groot warenhuis of het magazijn te inventariseren.
Op het Technologiepark-Zwijnaarde, een technologiepark van de Universiteit Gent, gelegen in Zwijnaarde bij Gent, bezit het iGent[1] een ruimte die men omgebouwd heeft tot een klein magazijn, om hierop experimenten met zulke drones te kunnen uitvoeren.

iGent[1]: iGent is een nieuwbouw voor medewerkers van de dienst UGent TechTransfer en de vakgroepen Informatietechnologie (INTEC) en Elektronica en Informatiesystemen (ELIS) van de faculteit Ingenieurswetenschappen en Architectuur.

1.2 Probleemstelling

Op dit moment bestaat er geen geautomatiseerde manier om het pad van de drone te bepalen om een groot warenhuis of een magazijn te scannen. Elk pad moet manueel ingegeven worden aan de hand van coördinaten. Dit project bestaat dus uit de taak om op basis van scanlocaties een automatisch pad te creëren, zodat de drone dit vervolgens kan afvliegen zonder tussenkomst van een persoon. Er wordt ook gevraagd een interface en controlepaneel te voorzien waarop de drone zijn huidige locatie, de scanlocaties van de verschillende producten en het pad dat de drone zal afleggen om deze locaties te bereiken, kan op gevisualiseerd worden. Via deze webgebaseerde applicatie moet men in staat zijn om de drone te configureren, sensoren aan of uit te schakelen, producten toe te voegen of te verwijderen van een database, instructies door te geven aan de drone dat deze zijn route mag starten, pauzeren, moet stoppen, enzovoort.
Het is de bedoeling dat de webapplicatie modulair wordt opgebouwd, zodat er nadien extra functionaliteiten kunnen worden toegevoegd of onderdelen kunnen verwijderd worden indien deze niet nodig zijn of vervangen moeten worden. Nadien kan er ook nog een mobiele applicatie ontwikkeld worden, zodat het project vanop een tablet of een ander mobiel apparaat kan gebruikt worden.

1.3 Doelstelling

Het doel van deze bachelorproef is om een interface en controlepaneel te voorzien voor een drone die aan de hand van een getekend pad een groot warenhuis of een magazijn moet inventariseren. De gebruikers moeten gemakkelijk via een computer, tablet of een smartphone de drone zijn locatie kunnen monitoren, sensoren op de drone kunnen activeren of uitschakelen, de database kunnen raadplegen, om zo de beschikbare producten te bekijken en deze tevens toe te voegen, te verwijderen of aan te passen. Men moet de drone kunnen starten of stoppen met het afvliegen van zijn 'door de gebruiker getekende' vluchtroute.
De website zal natuurlijk ook over een vorm van authenticatie moeten beschikken zodat enkel gebruikers met de correcte toestemming de drone kunnen besturen of producten kunnen wijzigen in de database.
Om dit project te realiseren, zal gebruik gemaakt worden van de MEAN-stack en Node-RED. MEAN is een acroniem dat staat voor MongoDB (een noSQL database), Express (een framework voor webapplicaties dat met Node.js werkt), Angular.js (een javascript Model-View-Controller framework) en Node.js (een server-side omgeving voor JavaScript).
Angular zal samen met nginx (uitgesproken als "engine-x") de dynamische front-end van de web applicatie verzorgen. Hierop kan de gebruiker de drone live monitoren, vluchtroutes tekenen, de productinventaris raadplegen, de verschillende mappen aanpassen, de drone configureren, enzovoort.
Deze data kan dan via Express.js naar de back-end verstuurd worden. Deze back-end, dat gebruikt maakt van Node.JS, zal samen met Node-RED instaan voor het verwerken van de data afkomstig van de front-end en de drone, om bijvoorbeeld de optimale vluchtroute te bepalen en deze route terug te sturen naar de drone.
De productinventaris en de map en drone configuratie bestanden worden opgeslagen in een MongoDB database.
Door middel van Node-RED kan men makkelijk nieuwe functies toevoegen of wijzigen om zo een modulair ontwerp te bekomen. Zo zal Node-RED ook instaan voor de communicatie tussen de MQTT Broker en de andere onderdelen van dit project.

1.4 Structuur van het verslag

Het verslag bestaat naast de inleiding (=hoofdstuk 1) nog uit 6 grote onderdelen: de gebruikersaspecten, de systeemarchitectuur, het testplan, de evaluatie en discussies, de handleidingen en het besluit.

In hoofdstuk 2 kan men de gebruikersaspecten vinden. Hier kunnen de gebruikers naast de High-level requirements ook een aantal 'use case diagrammen' terugvinden, samen met een volledige lijst met alle features die de webapplicatie voorziet en in welke sprint deze gerealiseerd zijn.
In hoofdstuk 3 staat beschreven uit welke onderdelen de webapplicatie bestaat en hoe dit project gerealiseerd is.
Vervolgens in hoofdstuk 4 staat het testplan. Dit is een overzicht van alle voorziene testplannen. Deze bevatten zowel unit testen, invoercontrole, code reviews, een strategie voor het automatisch compileren, user acceptance, enzovoort.
In hoofdstuk 5 is de evaluatie en discussie terug te vinden. Hierin wordt de performantie van de webapplicatie, de beveiliging, de schaalbaarheid, de eventuele problemen en geleerde lessen in vermeld.
Hoofdstuk 6 betreft de handleidingen waaruit de gebruiker kan leren hoe hij de webapplicatie lokaal kan installeren en hoe hij ze kan gebruiken.
Tot slot volgt in hoofdstuk 7 het besluit van het project.

2 GEBRUIKERSASPECTEN

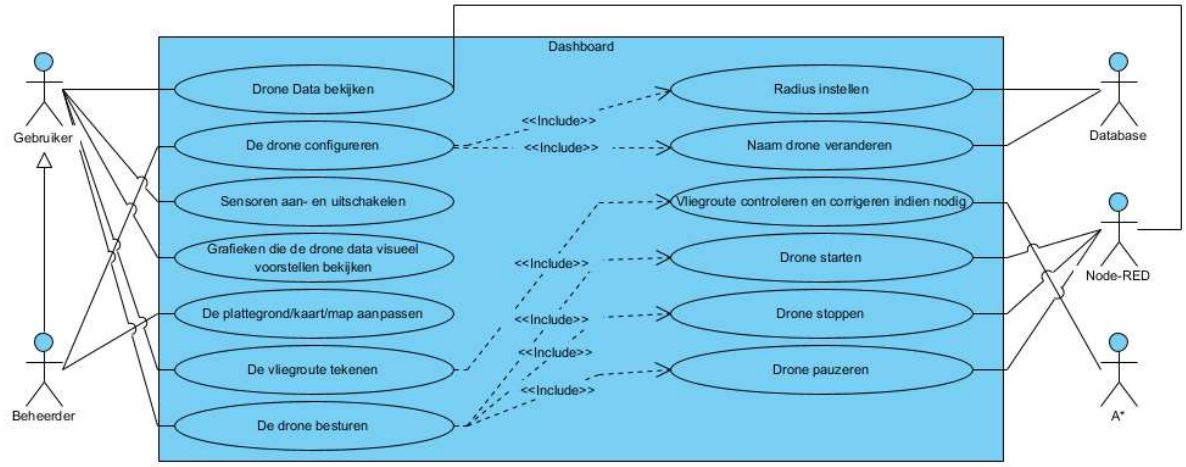
2.1 High-level requirements

Er dient een webgebaseerde interface en controlepaneel gebouwd te worden om een autonome drone, die een groot warenhuis of een magazijn moet inventariseren, moet monitoren en eventueel moet bijsturen. Deze webapplicatie moet ook connectie maken met een database waarin de geïnventariseerde producten kunnen opgeslagen worden samen met de drone en map configuraties. Enkel ingelogde gebruikers met de correcte toestemmingen mogen producten toevoegen, verwijderen of aanpassen in deze database. Hiervoor zal de webapplicatie over een authenticatie systeem moeten beschikken. De webapplicatie moet modulair worden opgemaakt zodat er makkelijk functies kunnen toegevoegd, verwijderd of vervangen kunnen worden. Eventueel kan er later een mobiele webapplicatie voorzien worden zodat de interface en het controle paneel via een tablet of een ander mobiel apparaat kan bekeken worden.

2.2 Use case diagrammen

2.2.1 Dashboard

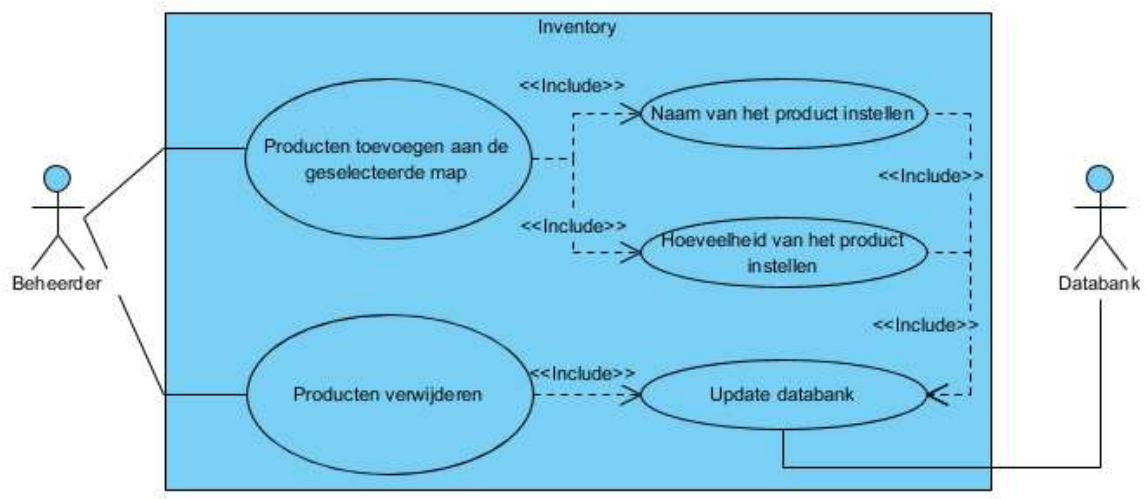
Wanneer een gebruiker zich aanmeldt, komt deze terecht op het dashboard. Hier kan de gebruiker de drone data bekijken zoals de naam, de radius, het batterij niveau, de positie, de snelheid, de acceleratie en de pitch, roll en yaw van de drone. Enkel een beheerder, een gebruiker met admin rechten, kan de naam en radius van de drone aanpassen. Deze configuratie wordt nadien opgeslagen in de database.
Een gebruiker beschikt ook over de mogelijkheid om verschillende sensoren op de drone te activeren of te deactiveren. De waardes die deze sensoren meten kunnen op het dashboard aan de hand van grafiek gemonitord worden.
Een gebruiker kan via de kaart een vliegroute tekenen en nadat deze gecontroleerd is en eventueel gecorrigeerd via het A* algoritme kan de gebruiker de drone de gekozen vliegroute laten afvliegen. Deze commando's worden via Node-RED naar de drone verstoord. Enkel een beheerder beschikt over de mogelijkheid om een kaart aan te passen door scanlocaties of een obstakel toe te voegen.



Figuur 1: Use case diagram van het dashboard

2.2.2 Inventory

Als beheerder ben je instaat om via het inventory scherm de geïnventariseerde producten op te halen die bij de gekozen map horen. Je kan ook extra producten toevoegen door deze een naam en hoeveelheid mee te geven. Na het toevoegen van een product zal de databank ook geüpdatet worden door dit nieuwe product toe te voegen. De beheerder is ook in staat om een product uit deze lijst te verwijderen.



Figuur 2: Use case diagram van het inventory scherm

2.3 Product backlog

2.3.1 Volledige feature list

- Responsieve webapplicatie als front-end gebruikersinterface en controle paneel (eventueel later mobiele app ontwikkelen).
- Architectuur moet zeer modulair zijn zodat men gemakkelijk modules kan toevoegen of aanpassen (weinig afhankelijkheden gebruiken).
- Databank die de producten in het magazijn of groot warenhuis bijhoudt, wanneer de drone items scant, wordt de database bijgewerkt met de recentste informatie.
- Mogelijkheid aanvinken welke specifieke data men wil weergeven of verbergen op de UI.
- Camera feed streamen naar canvas van webapplicatie.
- Dummy drone ontwikkelen die de sensor data stream van een vlucht simuleert.
- IP van drones moet dynamisch configureerbaar zijn.
- MQTT protocol gebruiken om data uit te wisselen tussen server en drone.
- Applicatie moet alerts geven indien drone afwijkt van pad.
- Dynamisch wisselen van datastream tussen verschillende sensoren indien bepaalde sensoren inaccuraat blijken te zijn.
- De drone besturen door een vliegroute mee te geven, deze vliegroute moet eerst gecontroleerd worden of de route veilig vliegen is voor de drone.
- De webapplicatie moet over een authenticatie en beveiligingssysteem beschikken.

2.3.2 Geselecteerde features per sprint

2.3.2.1 Sprint 1

- Communiceert via MQTT met de mockup voor het ontvangen van drone data en versturen van commando's. **15 story points.**

- Een API ter beschikking stellen met express voor de front-end UI zodat deze:
  - kaarten kan inladen, wijzigen en verwijderen
  - vliegroutes kan berekenen en uitwisselen met de drone
  - commando's vanaf de front-end kan doorsturen naar de mockup drone of naar een echte drone
  - de productinventaris kan opvragen/aanpassen

  20 story points.

- zorgen voor persistentie met MongoDB voor het bijhouden van kaarten, droneconfiguraties en de productinventaris. **10 story points.**

- Berekenen van de optimale vliegroute tussen de gewenste waypoints of het controleren en corrigeren van een zelfgetekende vliegroute rekening houdend met obstakels. **15 story points.**

- Een simulator voorzien die de functionaliteit en alle dataflow kan testen, deze dient echter niet als eindproduct.
  - dynamisch inladen/aanpassen/opslaan van kaarten die de obstakels, waypoints en producten visualiseert
  - selecteren van waypoints en visualiseren van het optimale pad
  - visualiseert een vlucht van de mockup drone op basis van gesimuleerde data, deze data wordt in aparte componenten weergegeven

  20 story points.

- Weergeven van de productinventaris **5 story points**.

- Besturingselementen voorzien om de drone aan te sturen. **10 story points.**

2.3.2.2 Sprint 2

- De UI converteren naar een realistische 2D visualisatie door middel van leaflet.
    - Het voorzien van een continu coördinatenstelsel met de plattegrond van het gekozen magazijn of grootwarenhuis.
    - Voorzien van besturingselementen om de drone te besturen.
    - Toevoegen van scanlocaties waar de drone op basis van een oriëntatie de geselecteerde producten scant.
    - Voorzien van meldingen in verband met foutieve vliegroutes.

  15 story points.

- Node-RED volledig implementeren en voor alle datastreams flows genereren. **15 story points.**

- Ondersteuning bieden voor mobiele apparaten door touch, scaling en gestures te voorzien. **15 story points.**

- De vliegroutes valideren met obstakels, met behulp van het snellere A* algoritme en collision detectie zodat niet correcte paden kunnen gecorrigeerd worden. **15 story points.**

- Het maken van een mockup drone voor het simuleren van vlucht data. **15 story points.**

- De ontvangen drone data van een echte drone of van de mockup drone verwerken en visueel weergeven op de webapplicatie door middel van grafieken, etc. **15 story points.**

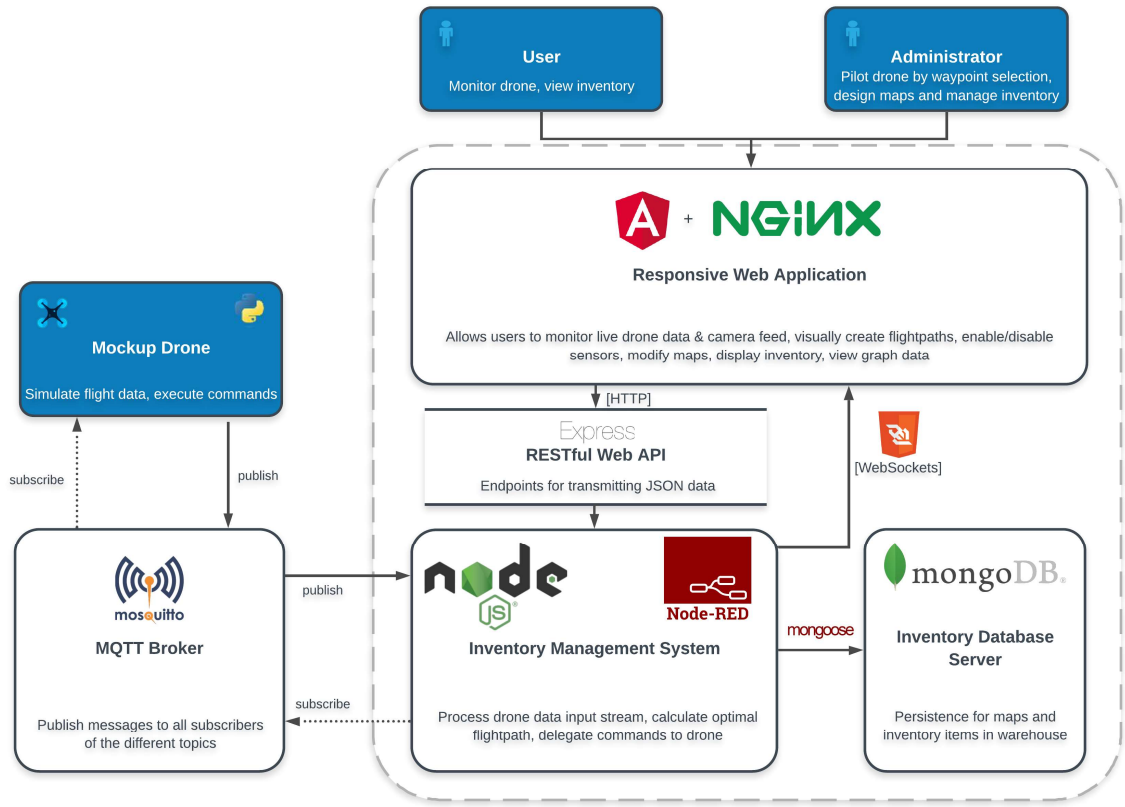
- Toevoegen van authenticatie en beveiliging zodat niet geautoriseerde gebruikers geen toegang krijgen tot de webapplicatie. **5 story points.**

- Verschillende droneconfiguraties ondersteunen zodat er op een makkelijke manier tussen verschillende drones kan gewisseld worden. **5 story points.**

3 SYSTEEMARCHITECTUUR

3.1 High-level systeem model

Om dit project te realiseren zal gebruik gemaakt worden van de MEAN stack en Node-RED. MEAN is een acroniem dat staat voor MongoDB (een noSQL database), Express (een framework voor webapplicaties dat met Node.js werkt), Angular.js (een javascript Model-View-Controller framework) en Node.js (een server-side omgeving voor JavaScript).
Angular zal samen met nginx de dynamische front end van de web applicatie verzorgen.
De data afkomstig van de front end kan via Express.js naar de back end verstuurd worden. Deze back end, dat gebruikt maakt van Node.JS, zal samen met Node-RED instaan voor het verwerken van de data afkomstig van de front-end en de drone om bijvoorbeeld de optimale vluchtroute te bepalen en deze route terug te sturen naar de drone.
De product inventaris en de map en drone configuratie bestanden worden opgeslagen in een MongoDB database.
Doormiddel van Node-RED kan men makkelijk nieuwe functies toevoegen of wijzigen om zo een modulair ontwerp te bekomen. Zo zal Node-RED ook instaan voor de communicatie tussen de MQTT Broker en de andere onderdelen van dit project.



Figuur 3: Beschrijving van de containers met de gebruikte framework en protocollen.

3.1.1 Front-end UI: Angular 7 op een nginx webserver

De front-end voorziet een continu coördinatenstelsel gebaseerd op een plattegrond van een groot warenhuis of magazijn. Een beheerder kan op deze plattegrond verscheidende obstakels en scanlocaties aanduiden, deze aanpassingen worden doorgestuurd naar de MongoDB databank waar de map wordt opgeslagen.
Deze map wordt gecreëerd door middel van leaflet, dit is een open-source JavaScript library voor het creeeren van interactieve kaarten. Leaflet is volledig geïntegreerd in de front-end als Angular Directive en bied ondersteuning aan voor verschillende bestaande plugins zoals: live data, drawing, heatmaps, etc. Hierdoor is het mogelijk om zelfs op een mobiel apparaat met behulp van touch en gestures de map te vergroten, te verkleinen of aan te passen. De beheerder kan deze kaarten ook dynamisch aanpassen door scanzones en obstakels toe te voegen en de vliegroute voor de autonome drone te tekenen. Deze vliegroute wordt via Express naar de back-end gestuurd waar deze gecontroleerd en eventueel gecorrigeerd wordt.

3.1.2 Back-end: Node.js en Node-RED

De back-end bestaat uit twee delen: Node.js en Node-RED.

De Node.js server stelt een API ter beschikking via Express waarmee data van de front-end naar de back-end kan gestuurd worden en deze doormiddel van mongoose op de MongoDB databank kan worden opgeslagen.
Deze API kan getest worden door middel van de meegeleverde unit testen, zoals beschreven staat in hoofdstuk 4: Testplan.

Naast de API en de unit testen staat Node.js ok in voor het valideren van de vliegroutes. In een eerste versie werd dit gedaan door middel van Dijkstra.
Het doel van Dijkstra is om voor de routeplanner het optimale pad van de drone te berekenen doorheen alle waypoints. Het algoritme maakt gebruik van een graaf om het kortste pad te vinden. Een graaf bestaat uit verschillende knopen die met elkaar verbonden kunnen worden. Deze verbindingen bevatten op zijn beurt een waarde/gewicht die bepaalt hoe lastig de verbinding is om af te leggen.
Het algoritme van Dijkstra werkt als volgt:

- Eerst wordt de afstand tot elke knoop op oneindig gezet behalve deze van de startknoop, deze wordt op 0 geinitialiseerd.
- Deze startknoop wordt nadien toegevoegd aan een prioritaire wachtrij, de positie van een knoop binnen deze wachtrij wordt bepaald aan de hand van de gekregen afstandswaarde. Hoe lager de afstandswaarde, hoe hoger de positie in de wachtrij.
- De startknoop wordt als bezocht gemarkeerd en als huidige knoop ingesteld.
- Vervolgens wordt er in de lijst met verbindingen gezocht naar elke verbinding met de huidige knoop, indien het de eerste keer is dat het algoritme deze knoop bezoekt wordt deze nieuwe knoop toegevoegd aan de prioritaire wachtrij en als bezocht gemarkeerd.
- De totale afstand/gewicht tot deze knoop vanaf de startknoop wordt berekend en indien deze lager is dan de vorige afstand/gewicht tot deze knoop wordt deze geüpdatet en verplaatst binnen de prioritaire wachtrij.
- Nadien wordt de knoop verwijderd en indien men de eindknoop bereikt heeft wordt het gevonden pad terug gegeven, indien dit nog niet het geval is wordt de meest prioritaire knoop opgehaald en als bezocht gemarkeerd en herhalen de laatste 2 stappen zich.

Dit werd verder geoptimaliseerd door gebruik te maken van de hierboven vermelde prioritaire wachtrij zodat de nabijgelegen knopen eerst worden onderzocht alvorens de verder gelegen knopen worden bekeken om zo sneller het gewenste pad te vinden. In dit project moet het kortste pad tussen de verschillende waypoints berekend worden. Hiervoor wordt het pad vanaf de begin positie tot elke waypoint berekend en vervolgens het dichtstbijzijnde waypoint gekozen als eerste tussenstop, nadien wordt dit herhaald met de tussenstop als startknoop totdat alle waypoints zijn opgenomen.
Dit was echter niet performant genoeg en bevatte nog geen manier om een getekend pad te valideren. Hierdoor werd besloten om over te stappen op het A* algoritme. A* verkiest namelijk om knopen te bekijken waarvan hij denkt dat deze tot een lagere globale kost zal eindigen, terwijl

het algoritme van Dijkstra rekening blijft houden met alle omliggende knopen. Dit halveerde de zoektijd naar de kortste vluchtroute. Om de tijd om een vluchtroute te genereren nog meer te verlagen werd ervoor gekozen om de graaf, dat tot nu toe uit een volledige grid bestond bovenop de map, te vereenvoudigen. De autonome drone hoefde namelijk enkel van richting te veranderen indien er een obstakel in de weg stond. Zo werd er voor gekozen om enkel in het midden van een scanzone een knoop te plaatsen en 4 knopen rondom een obstakel. Vanuit deze knopen wordt nadien gekeken welke met elkaar te verbinden zijn zonder te dicht bij een obstakel te komen. Zo wordt het aantal knopen in de graaf sterk omlaag gebracht wat de zoektijd minimaliseert. Hierdoor wordt het ook mogelijk om te controleren of een zelf getekend pad niet te dicht bij een obstakel kwam en indien dit wel het geval is via het A* algoritme dit pad te corrigeren.

Node-RED verzorgt vervolgens nog de communicatie tussen de front-end en de MQTT broker. Zo zal de data ontvangen van de drone of mockup drone in real time via Websockets verstuurd worden naar de front-end waar deze gevisualiseerd kan worden. De Node-RED flow bevat ook nog verschillende functies om bijvoorbeeld na te gaan of de drone zich op een scanlocatie bevindt en hier producten moet scannen, om na het toekomen op een locatie de nieuwe coördinaten door te sturen naar de drone zodat deze verder kan vliegen, etc.

Node-RED wordt gebruikt om flow-gebaseerd visueel te programmeren voor IoT of Internet of Things. Dit werkt volledig in de browser en is gemaakt op Node.js. Dit biedt de gebruiker de mogelijkheid om zelf delen van het programma te vervangen door een eigen alternatief. Er werd voor Node-RED gekozen aangezien deze MQTT, HTTP en WebSocket eindpunten bevat, zodat er makkelijk connectie kan gemaakt worden tussen de verschillende componenten van deze webgebaseerde applicatie.

3.1.3 MQTT en de Mockup drone

MQTT of message Queuing Telemetry transport is een publish/subscriber messaging protocol. Via dit protocol wordt er data vanaf de drone of mockup drone verstuurd naar de Node-RED back-end en omgekeerd.
Bij MQTT wordt er vanuit een client een bericht uitgestuurd naar de broker op een bepaalde topic. Andere apparaten kunnen subscriben op deze topic, de broker zal vervolgens telkens het data ontvangt deze doorsturen naar alle gesubscribde apparaten.

Voor een drone te simuleren wordt er gebruik gemaakt van een mockup drone geschreven in python. Deze mockup is op dit moment slechts een testversie. De python mockup drone wordt tijdelijk vervangen door een simpelere mockup in Node-RED omdat de python mockup drone door zijn complexiteit en slechtere performantie niet klaar is na sprint 2. De mockup in Node-RED is minder complex aangezien deze minder features bevat en ondervindt geen performantie problemen.

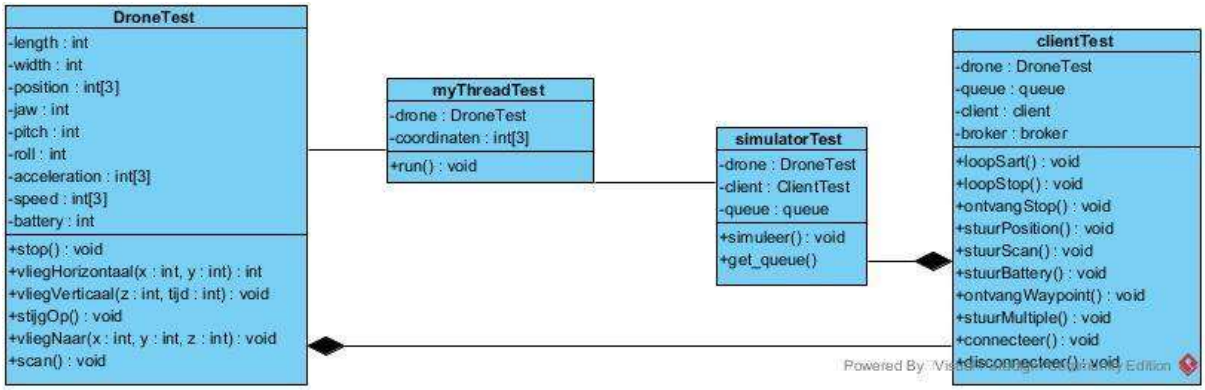
3.1.4 Android application

Naast het feit dat de volledige applicatie zodanig is gebouwd dat deze ook op een mobiel apparaat in de browser kan getoond worden, wordt er nog gewerkt aan een native Android applicatie. Hiermee kan de gebruiker de applicatie ook als app instaleren op een Android toestel.

3.2 Klassendiagrammen

3.2.1 Klassendiagram simulator

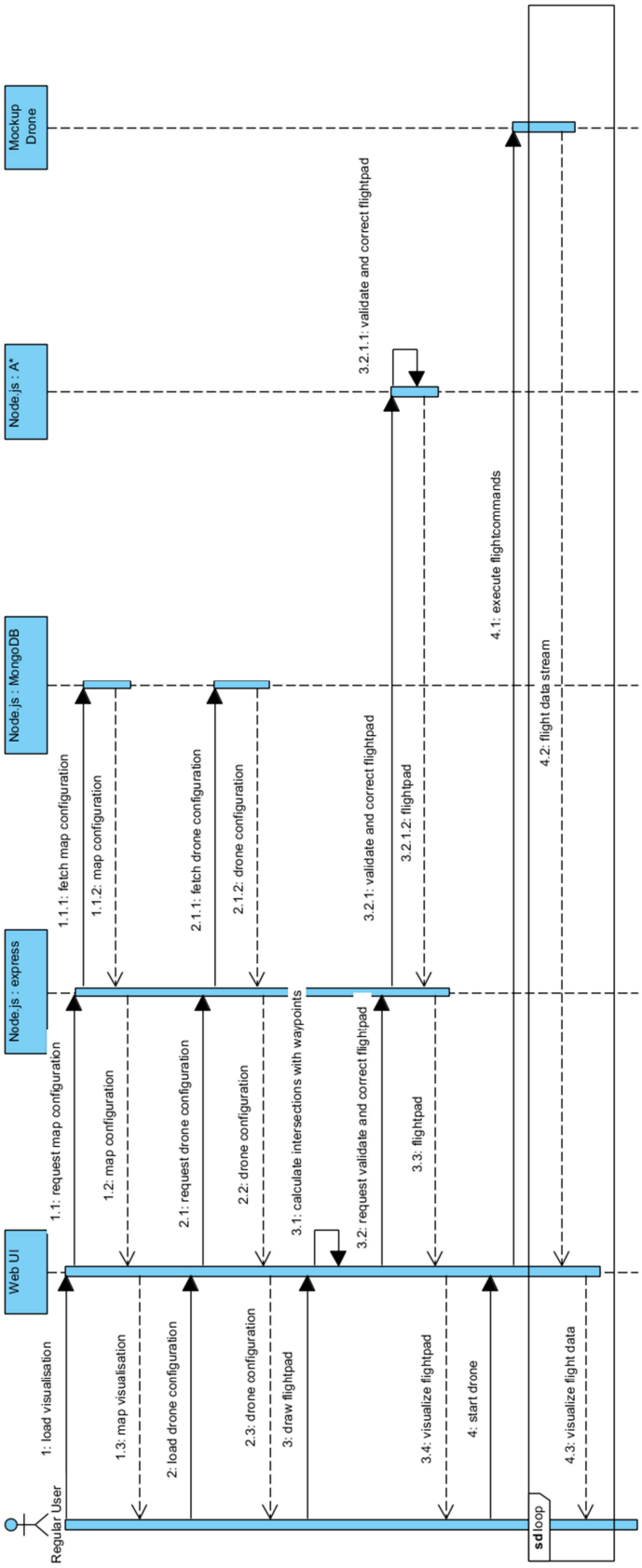
DE volledige python mockup drone bestaat uit verschillende klassen.
De klasse drone bevat alle data van de drone, zoals de positie, breedte, oriëntatie, snelheid, etc., en methodes om het vliegen van de drone te kunnen simuleren. Voor alle data die de drone bevat worden getters en setters voorzien, deze worden echter niet op het klassendiagram weergegeven zodat het klassendiagram overzichtelijk blijft.
De klasse client zou het mogelijk maken om data te kunnen sturen en ontvangen met MQTT door te pushen en subscriben op een broker.
Tenslotte wordt er in de klasse simulator een eindeloze while-lus gestart waarin het gedrag van de drone gesimuleerd ging worden. Deze lus maakt gebruik van de klasse myThread, zodat de client de data van de drone kan blijven verzenden tijdens het vliegen van de drone.



Figuur 4: Klassendiagram van de simulator/mockup drone

3.3 Sequentiediagrammen

Bij het opstarten van de webapplicatie wordt zowel de map, de scanlocaties, de producten en de drone configuratie opgehaald uit de database. Indien de gebruiker van map veranderd zal de map en de bijhorende producten van de nieuwe map opgehaald worden.
Vervolgens kan de gebruiker een vliegroute tekenen op de kaart, hierbij zal dan berekend worden welke scanlocaties op dit getekend pad liggen en wordt het pad gevalideerd en eventueel gecorrigeerd via het A* algoritme indien het pad door of te dicht bij een obstakel liep.
Hierna kan de gebruiker de drone starten en zal deze beginnen met het aflopen van het pad richting de scanlocaties. Hierbij zal de drone periodiek zijn nieuwe positie, snelheid, etc. doorsturen en zal dit visueel weergegeven worden op de kaart.
Wanneer de drone klaar is met zijn vliegroute zal deze terugkeren naar zijn oorspronkelijke positie en landen.
De gebruiker zal ook instaat zijn om de drone te pauzeren en eventueel te stoppen tijdens zijn vluchtroute.



Figuur 5: Sequentiediagram

3.4 Databank

Om de producten in de database te bewaren moet er eerst en vooral een productschema opgesteld worden. In dit schema wordt de structuur vastgelegd van hoe de verschillende producten in de database zullen opgeslagen worden. In dit schema wordt bepaald dat een product een id, een naam, een hoeveelheid en een positie kan bezitten. Het id van het product zal automatisch door mongoose gegenereerd worden. Verder zijn de naam, de hoeveelheid en de coördinaten van het product verplicht op te geven bij het aanmaken. De producten kunnen vervolgens opgeslagen worden in de database. De toegang tot de database wordt geregeld via een REST API.

Er kunnen specifieke producten worden opgevraagd uit de database, dit gebeurt met behulp van een uniek id. Deze producten kunnen met behulp van hetzelfde id vervolgend gemakkelijk aangepast of verwijderd worden. Tot slot bestaat er ook de mogelijkheid om de volledige inhoud van de databank op te vragen.

4 TESTPLAN

Alvorens een gebruiker de webapplicatie kan uittesten moet men deze eerst correct lokaal instaleren. Een uitgebreide handleiding over hoe dit gedaan wordt staat in het hoofdstuk Handleidingen.

4.1 Unit testen

Om de verschillende API calls voor het toevoegen, aanpassen en verwijderen van de verschillende mappen, producten en droneconfiguraties en het valideren en corrigeren van een vluchtroute is een unit test meegegeven. Met behulp van macha en chai kan iemand gemakkelijk de back-end te testen. Deze test kan uitgevoerd worden door via een CMD-venster naar *~/project/backend/management-server* te navigeren en het commando *npm test* uit te voeren. Hiermee wordt het testbestand *~/project/backend/management-server/test/testAPI.js* uitgevoerd.

4.2 Invoercontrole

Elk invoerveld is voorzien van een controle zodat er in velden waar enkel numerieke waardes mogen ingevoerd worden enkel nummers kunnen ingevoerd worden. Ook wanneer er geen waarde ingevoerd wordt, wordt er een correcte foutboodschap op het scherm getoond.

5 EVALUATIE EN DISCUSSIES (SPRINT 3)

6 HANDLEIDINGEN

6.1 Installatiehandleiding

De gebruiker kan de webapplicatie lokaal testen of deze via de projectwebsite bereiken. De projectwebsite is in de voetregel of op het voorblad te vinden.

Om de webapplicatie lokaal te testen zijn er een aantal vereisten.
Namelijk:

- De broncode moet gedownload worden zodat deze lokaal beschikbaar is.
- MongoDB moet geïnstalleerd worden en de database locatie moet worden aangemaakt.
- Node.js en een Node Package Manager die automatisch met Node.js geïnstalleerd wordt.
- Angular, Express en andere modules moeten correct geïnstalleerd worden.
- Node-RED.
- De Mosquitto broker.

- Om de broncode lokaal beschikbaar te maken, moet de GitHub repository gecloned worden. Hiervoor navigeert de gebruiker via een browser naar de hoofdpagina van de repository. In dit geval is dat dus https://github.ugent.be/bp-vop-2019/drone1. Op deze pagina klikt men op de 'Clone or download' knop waarna men 2 opties heeft. Men kan op download ZIP klikken om vervolgens het ZIP-bestand uit te pakken in een locatie naar keuze of men maakt gebruik van de clone URL en Git Bash. Bij Git Bash verandert men de huidige working directory naar een locatie naar keuze en kan men de GitHub repository downloaden via het git clone commando. Hier is dit dus git clone https://github.ugent.be/bp-vop-2019/drone1.git. Nu is de broncode lokaal beschikbaar.

- MongoDB is beschikbaar via https://www.mongodb.com/download-center/community. Op deze pagina selecteert men de versie en het besturingssysteem naar keuze. Men kiest best voor de recentste versie en download het MongoDB Server bestand. Tijdens deze installatie wordt aangeraden om de standaard instellingen te behouden. Naast MongoDB Server zal dit ook MongoDB Compass installeren, dit is een handige tool om connectie te maken met de database en deze ook te visualiseren. Collecties kunnen met MongoDB Compass ook aangesproken worden via CRUD operaties. Vervolgens moet de map *'Directory waar MongoDB is geïnstalleerd'/data/db* (of */data/db* op linux) worden aangemaakt. Voor de meeste Windows gebruikers zal dit dus *C:/data/db* zijn. Hierin zal MongoDB zijn collecties bewaren. Indien de gebruiker een andere map locatie wenst te gebruiken, moet de gebruiker bij het runnen steeds de optie --bpath met als argument het gewenste pad mee geven.

- Node.js is een run-time environment dat alles bevat om de Javascript code van dit project uit te voeren. Node.js wordt ook gebruikt voor het installeren van Angular en Express. Node.js

kan geïnstalleerd worden door naar https://nodejs.org/en/download/ te surfen en vervolgens de correcte installer te downloaden. Voor de meeste gebruikers zal dit de Windows Installer zijn onder het LTS tabblad. Tijdens deze installatie wordt aangeraden om de standaard instellingen te behouden.

- Vervolgens kunnen alle bijhorende modules waaronder ook Angular en Express geïnstalleerd worden. Hiervoor moet de gebruiker een CMD-venster openen en naar de correcte folder locatie navigeren. Eenmaal de gebruiker naar *~/project/backend/management-server* genavigeerd is, moet hij het commando *npm install* uitvoeren. Zo worden de nodige afhankelijkheden voor de back-end automatisch geïnstalleerd. Na afloop doet de gebruiker hetzelfde maar nu navigeert hij eerst naar de folder *~/project/web-ui/drone-control-center*. Nu worden de afhankelijkheden van de front-end automatisch geïnstalleerd.

- Om de mosquitto broker, die voor de communicatie tussen de drone en de webapplicatie staat, te installeren surft de gebruiker naar https://mosquitto.org/download/ en selecteert deze de correcte installer. Voor de meeste gebruikers zal dit de windows-x64.exe zijn onder Binary Installation. Tijdens deze installatie wordt aangeraden om de standaard instellingen te behouden.

Nu alle onderdelen correct geïnstalleerd zijn, kan de gebruiker de webapplicatie opstarten door de verschillende services op te starten. Hiervoor moet de gebruiker:

- De front-end te starten door naar de folder *~/project/web-ui/drone-control-center* te navigeren in een CMD-venster. Vervolgens voert men het *ng serve* commando uit.
- De back-end te starten door naar de folder *~/project/web-ui/drone-control-center* te navigeren in een CMD-venster en vervolgens het *nodemon api/bin/www* commando uit te voeren.
- De database kan gestart worden door in een CMD-venster te navigeren naar de installatie folder van MongoDB en vervolgens naar de folder *~/Server/4.0/bin*. In deze map voert men *het mongod* commando uit.
- Als laatste start men de mosquitto broker door opnieuw in een CMD-venster te navigeren naar de mosquitto folder en hier het commando *mosquitto -v* uit te voeren.

Door via een browser naar keuze te navigeren naar http://localhost:4200/dashboard kan de gebruiker de webapplicatie uittesten. Men kan de Node-RED flows bekijken en eventueel aanpassen door naar http://localhost:3000/editor te surfen.

6.2 Gebruikershandleiding

Vooraleer een gebruiker de webapplicatie kan gebruiken, moet deze zich registreren of inloggen met een geregistreerd account. Indien de webapplicatie lokaal wordt gebruikt, is men vrij om zelf een account aan te maken. Indien men de webapplicatie via de projectwebsite raadpleegt, dient men in te loggen via het admin account om alle functies te gebruiken. Het admin account gebruikt als email: 'admin@ugent.be' en het wachtwoord is 'Roeliewoelie'.

Vervolgens komt men op het dashboard scherm. Hierbij zal er een map ingeladen worden van de database of, indien deze leeg is, een nieuwe map worden aangemaakt. Via het dashboardscherm ziet men een plattegrond van het magazijn of groot warenhuis met hierop de scanlocaties aangeduid via een blauwe cirkel en de obstakels via een rode rechthoek. Naast de scanlocaties en de obstakels wordt ook de huidige locatie van de drone aangeduid samen met zijn rotatie. Via de knoppen op de plattegrond kan men scanlocaties en obstakels toevoegen, scanlocaties en obstakels aanpassen of verwijderen, kan men een vliegroute tekenen en kan men het centrum van de map zich laten focussen op de locatie van de drone.
Onder de map zijn er 4 tabbladen te vinden. Eén voor de drone data: hier kan men de naam, de radius, het batterij niveau, de positie, de snelheid, de acceleratie en de pitch, roll en yaw van de drone bekijken.
In het tweede tabblad vindt de gebruiker de drone configuratie waar hij de naam en de radius van de drone kan configureren. Deze configuratie wordt nadien opgeslagen in de database.
Op het derde tabblad kan men de sensoren van de drone aanzetten of uitschakelen.
Het laatste tabblad bevat vervolgens nog enkele visuele grafieken waar de data van de drone visueel wordt voorgesteld.
Via de knoppen boven de plattegrond kan men de getekende vluchtroute valideren, de drone het pad laten afvliegen en de drone tijdens zijn vluchtroute laten pauzeren, terug verder laten vliegen of de drone volledig laten stoppen met het vliegen van zijn vluchtroute.

Links op het scherm kan men naast het dashboard de andere functionaliteiten van de webapplicatie bekijken. Het inventory scherm bevat een lijst met alle producten die gekoppeld zijn aan de huidige geselecteerde map. Hier kan een gebruiker met admin rechten ook producten aan toevoegen of deze weer verwijderen.
Op het monitor scherm kan de drone data die op het dashboardscherm aanwezig was in meer details bekijken.
Via het Admin scherm kan een gebruiker met admin rechten kiezen om alle mappen te verwijderen uit de database.
En als laatste is er ook een Node-RED tabblad voorzien zodat een gebruiker de flows kan bekijken en aanpassen.

In de rechterbovenhoek vindt de gebruiker 3 knoppen waarmee hij een plattegrond kan selecteren, zijn profiel kan bekijken en waarmee hij kan uitloggen uit de webapplicatie.

7 BESLUIT (SPRINT 3)

BIJLAGEN

1. < datum - {dd-mm-jj} > : < titel >

REFERENTIES

< *referentienaam - cursief* > . < referentiebeschrijving >