|  |
| --- |
| path planner voor inventaris met autonome drone |
| sprint 1 |

Leden: Robbe De Sutter, Karel Everaert, Jochen Laroy, Roel Moeyersoons, Wouter Stemgée

|  |
| --- |
|  |

Inhoudsopgave

[Inhoudsopgave 2](#_Toc3451892)

[Korte Samenvatting 3](#_Toc3451893)

[1 Inleiding 4](#_Toc3451894)

[1.1 Context 4](#_Toc3451895)

[1.2 Probleemstelling 4](#_Toc3451896)

[1.3 Doelstelling 4](#_Toc3451897)

[1.4 Structuur van het verslag 4](#_Toc3451898)

[2 Gebruikersaspecten 5](#_Toc3451899)

[2.1 High-level requirements 5](#_Toc3451900)

[2.2 Use case diagrammen 5](#_Toc3451901)

[2.3 Product backlog 6](#_Toc3451902)

[3 Systeemarchitectuur 7](#_Toc3451903)

[3.1 High-level systeem model 7](#_Toc3451904)

[3.2 Klassendiagrammen 9](#_Toc3451905)

[3.3 Sequentiediagrammen 10](#_Toc3451906)

[3.4 Databank 11](#_Toc3451907)

[3.5 Object diagram 12](#_Toc3451908)

[Bijlagen 13](#_Toc3451909)

[Referenties 14](#_Toc3451910)

# Korte Samenvatting

# Inleiding

## Context

## Probleemstelling

## Doelstelling

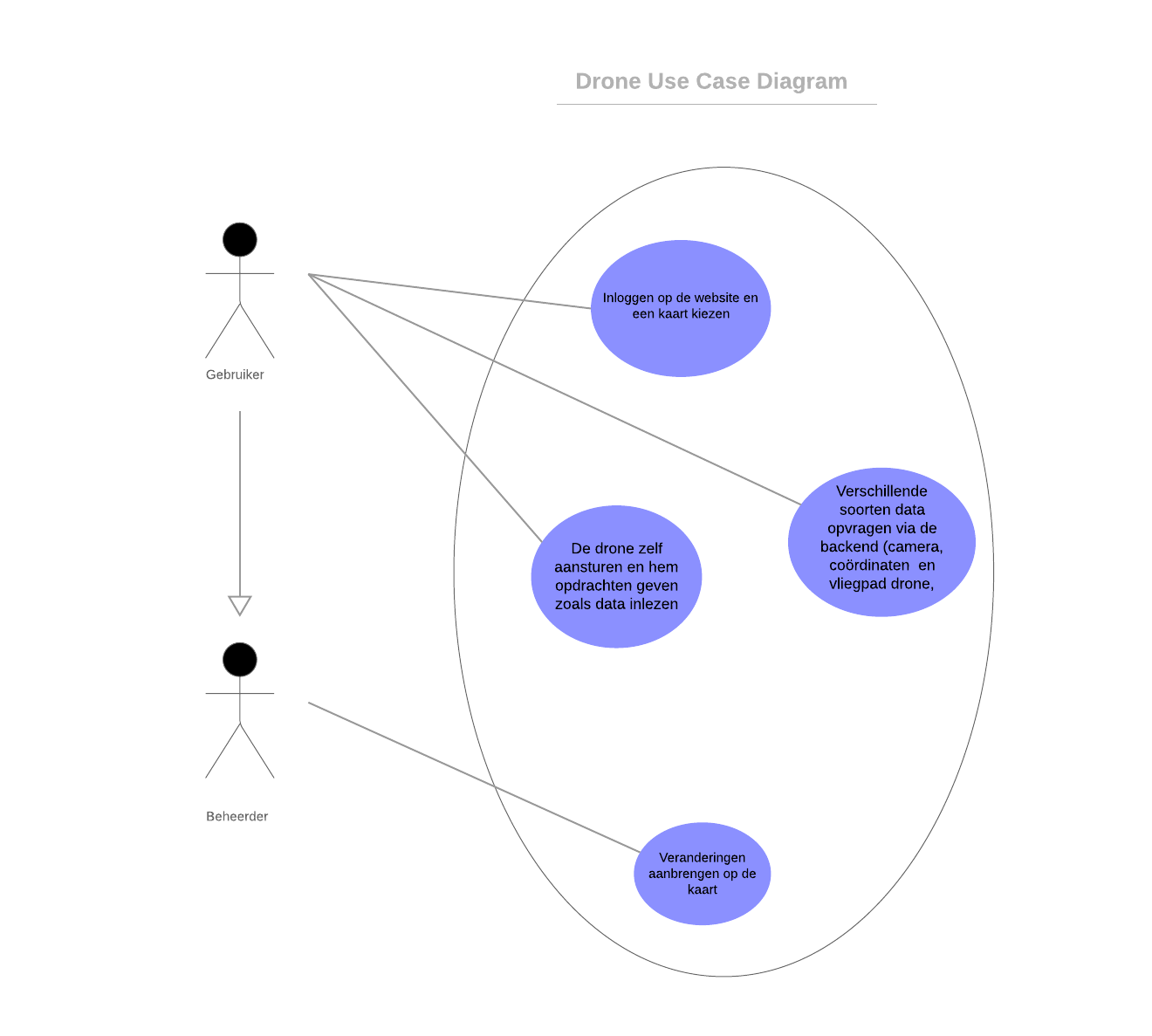
## Structuur van het verslag

# Gebruikersaspecten

## High-level requirements

Om van de applicatie gebruik te kunnen maken heeft de client enkel een browser nodig. De applicatie kan zowel op een computer als op een mobiel apparaat gebruikt worden.

## Use case diagrammen



Figuur 1: Use case diagram

## Product backlog

### Volledige feature list

* Responsive webapplicatie als front-end gebruikersinterface (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 bijhoudt, wanneer 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

### Geselecteerde features per sprint

#### Sprint 1

* communiceert via MQTT met de mockup voor het ontvangen van drone data en versturen van commando's **15 story points.**
* stelt een API ter beschikking met express voor de frontend UI zodat deze:
  + kaarten kan laden en opslaan
  + flightpath's kan uitwisselen
  + commando's vanaf de frontend kan doorsturen naar de mockup drone
  + de inventaris kan opvragen/aanpassen

**20 story points.**

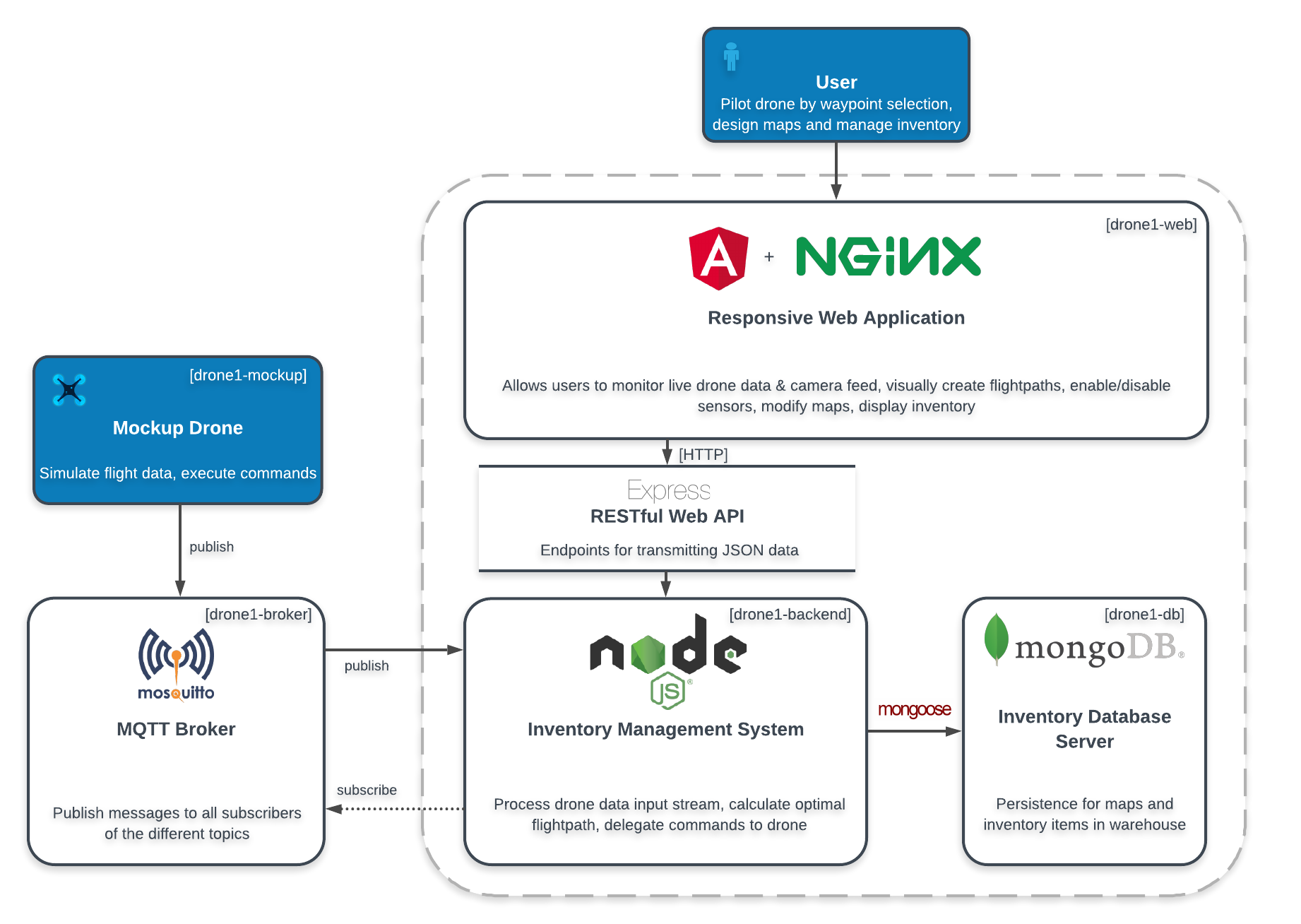
* zorgt voor persistentie met MongoDB voor het bijhouden van kaarten en de inventaris **10 story points.**
* berekent aan de hand van het algoritme van Dijkstra het optimale pad tussen de gewenste waypoints, rekening houdend met obstakels **15 story points.**
* prototype simulator die de functionaliteit 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.**

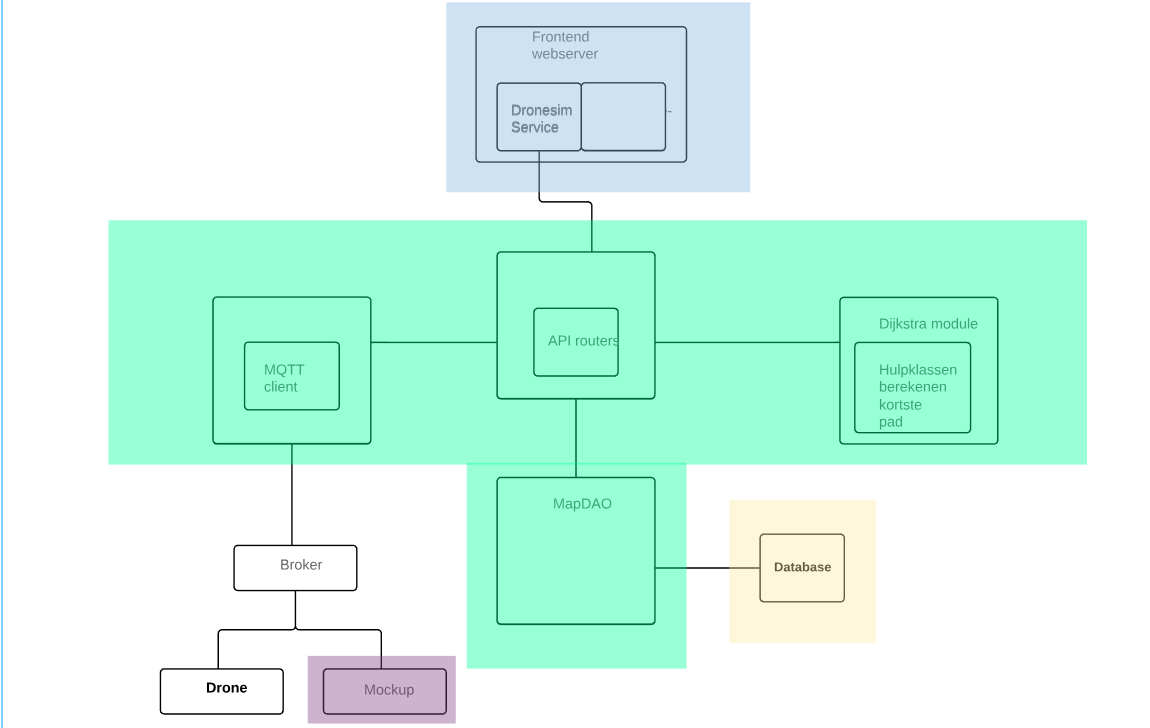
* weergeven van de inventaris **5 story points**.
* voorziet besturingselementen om de drone aan te sturen met commando's **10 story points.**

# Systeemarchitectuur

## High-level systeem model



Figuur 2: Beschrijving van de containers met de gebruikte frameworks en protocollen.

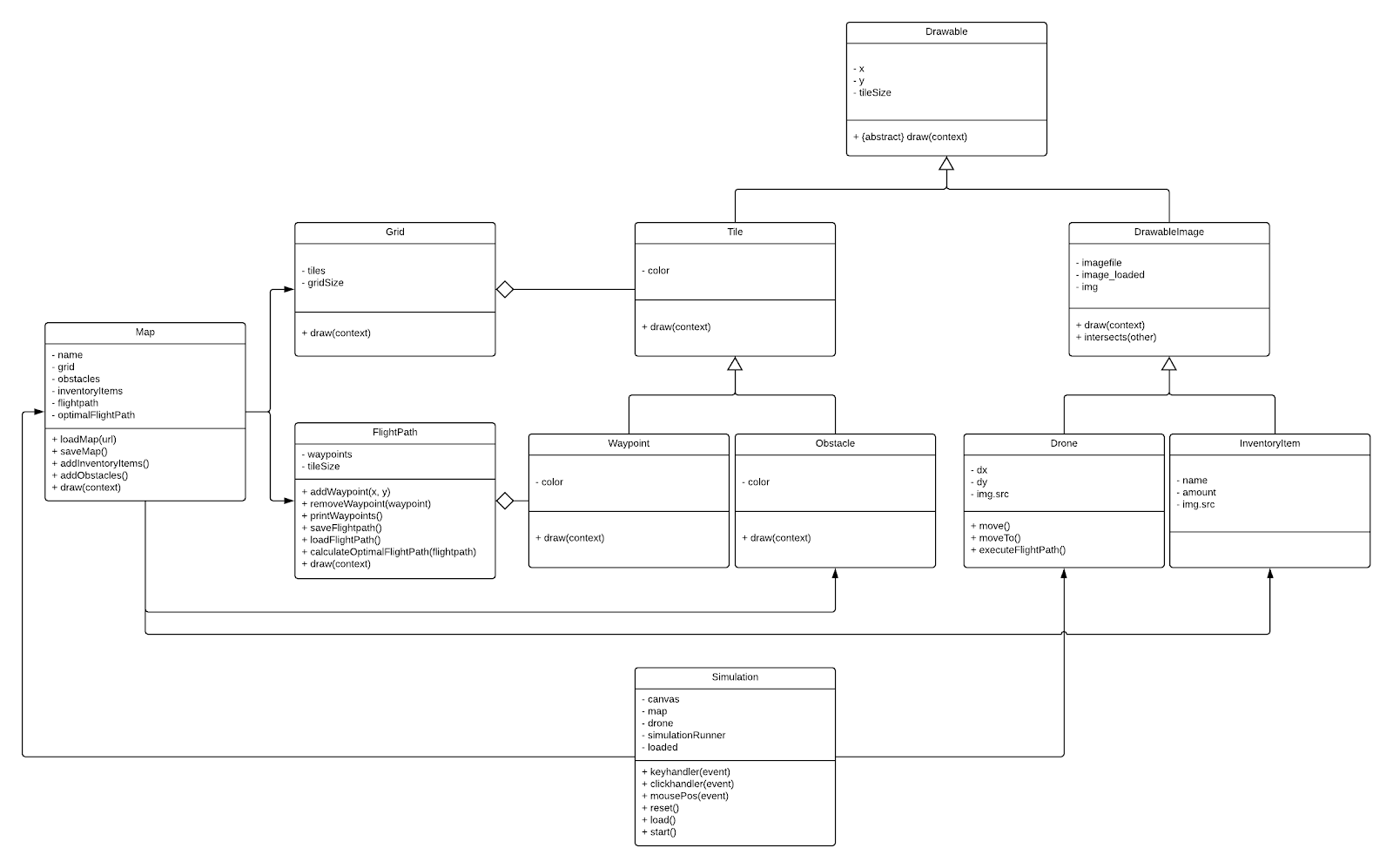


Figuur 3: Algemenere structuur van de applicatie

### Deployment diagram

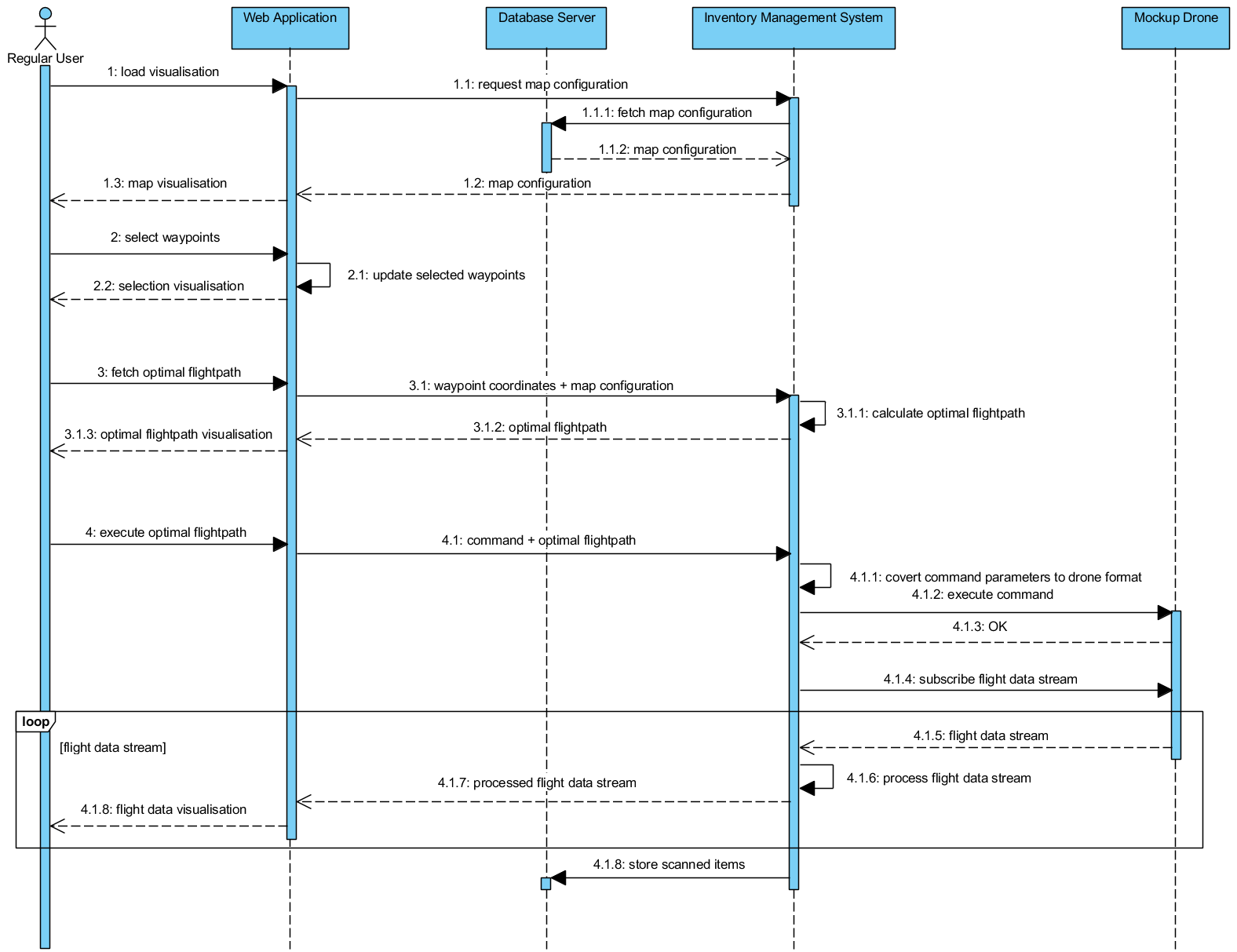
## Klassendiagrammen

### Klassendiagram simulator



Figuur 4: Klassendiagram van de simulator/front-end

## Sequentiediagrammen

**

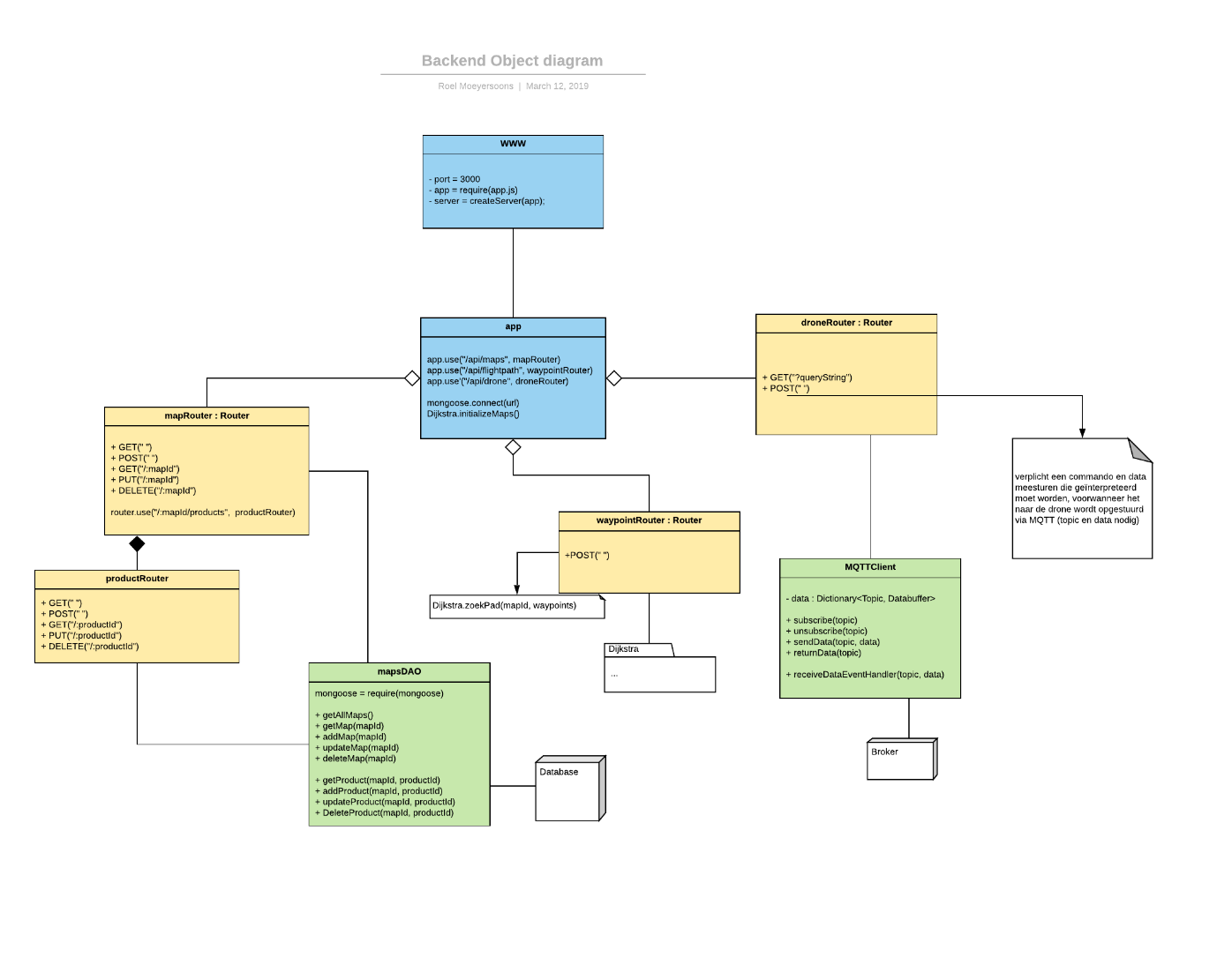
Figuur 5: Use case diagram

## 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.

## Object diagram



Figuur 6: Object diagram van de backend

Bijlagen

1. :

Referenties

*.*