

# Bachelorproef: eindverslag

## Droneplanning-tool





[Drone Planner](#) [Map](#) [Projects](#) [Drone Flights](#) [Drones](#) [Pilots](#) [Contact](#) [About](#) Hello admin! [Log off](#) [Register](#)

### Drone Flights

Drone Flights /

Create new Flight

Show 10 entries Search:

Project	Location	Date	Drone	Pilot	Map	Files	Upload	Actions
PRJ-001	Avelgem	06/02/2020	Happy	Bryan Van Huyneghem		<div>QR GCP CTRL TFW</div> <div>Images XYZ Drone Log</div>		<a href="#">Details</a> <a href="#">Edit</a> <a href="#">Delete</a>
PRJ-001	TBD	NA.	Happy	Bryan Van Huyneghem		<div>QR GCP CTRL TFW</div> <div>Images XYZ Drone Log</div>		<a href="#">Details</a> <a href="#">Edit</a> <a href="#">Delete</a>

Showing 1 to 2 of 2 entries Previous 1 Next

© 2020 - Jan De Nul, Drone Flights Application

<https://github.ugent.be/bp-2020/drone1>

Bryan Van Huyneghem  
Philip Kukoba  
Nathan Beyne  
Niels Hauttekeete

Promotoren: Prof. Helga Naessens, Prof. dr. Veerle Ongenae  
Klant: Jan De Nul

Bachelorproef voorgelegd voor het behalen van de graad bachelor in de Bachelor of Science  
in de industriële wetenschappen: informatica

Academiejaar: 2019-2020



## Abstract

Jan De Nul is een wereldwijd baggerbedrijf dat sinds enkele jaren werkt met drones, die zeer veel data genereren en waarvan niet alles bijgehouden of verwerkt wordt. De klant heeft nood aan een droneplanning-tool (een webapplicatie) die hen toelaat om hun data te centraliseren, alsook hun dronevluchten bij te houden en te visualiseren met de ArcGIS JS API. Er werd een databank ontworpen met SQL Server en een webapplicatie gebouwd met het *server-side framework* ASP.NET MVC 5. De klant kan verscheidene projecten, drones, piloten en dronevluchten aanmaken en aanpassen. Verder beschikt de webapplicatie over een inlogsysteem dat afscherming van de buitenwereld garandeert. Er kunnen bestanden geüpload worden die vervolgens met *parsers* verwerkt en weggeschreven worden naar de databank. De applicatie is bovendien in staat om automatisch DAT-bestanden te converteren naar CSV-bestanden en deze te *parsen*. Drone- en pilootdata kunnen geëxporteerd worden naar een logboek dat de klant kan uitprinten, zodat hij ook kan beschikken over een papieren logboek. Deze tool is een volwaardige basis waar Jan De Nul op verder kan werken om hun noden nog meer te verfijnen.

## Inhoudsopgave

Abstract .....	3
Lijst van figuren .....	6
Inleiding .....	7
1. Gebruikersaspecten.....	10
1.1 Databank .....	10
1.2 Webapplicatie.....	10
1.3 Use case-diagrammen .....	11
1.4 Featurelijst.....	15
1.4.1 Sprint 1 .....	15
1.4.2 Sprint 2 .....	16
1.4.3 Sprint 3 .....	17
2. Systeemarchitectuur .....	18
2.1 High-levelsysteemmodel ( <i>deployment diagram</i> ) .....	18
2.2 Databankdiagram .....	19
2.2.1 DroneFlighttabel.....	19
2.2.2 QualityReporttabel .....	22
2.2.3 DroneLogEntrytabel .....	23
2.2.4 Logintabellen .....	24
2.3 MVC .....	25
2.3.1 Model .....	25
2.3.1.1 Simple Factory pattern voor parser-klassen .....	25
2.3.1.2 <i>Helper</i> -klasse .....	28
2.3.2 Controllers .....	29
2.3.2.1 'View'-controllers .....	29
2.3.2.2 'Web API'-controllers .....	31
2.3.3 Views .....	33
2.4 Javascriptklassen .....	34
2.4.1 Uploaden van files .....	34
2.4.2 Visualisatie met ArcGIS JS API .....	42
2.4.3 Data Tables .....	45
2.5 Sequence diagrams .....	46

2.5.3	Bekijken van dronevluchten als anonieme gebruiker .....	46
2.5.4	Inloggen en bestanden uploaden.....	47
3.	Testplan .....	51
3.1	Unit testen.....	51
3.2	Integratietesten.....	52
4.	Evaluaties en discussies.....	53
4.1	Performantie .....	53
4.2	Beveiliging .....	53
4.3	Schaalbaarheid .....	53
4.4	Problemen en geleerde lessen .....	54
5.	Handleidingen .....	55
5.1	Inhoud van de distributie .....	55
5.2	Installatiehandleiding voor ontwikkelaar.....	55
5.2.1	Vereiste software .....	55
5.2.2	Aanmaken van de databank.....	56
5.2.3	Opstarten van de webapplicatie met Visual Studio .....	57
5.2.4	<i>Publishen</i> van de webapplicatie .....	57
5.3	Gebruikershandleiding .....	60
5.4	Problemen die zich kunnen voordoen .....	69
	Besluit.....	72
	Future work .....	73
	Referenties .....	75
	Appendix A: use case-diagrammen stories .....	77
	Appendix B: Integratietesten .....	100

## Lijst van figuren

Figuur 1: het bekijken van entiteiten in het systeem voor de interne en externe gebruiker .....	12
Figuur 2: het toevoegen, wijzigen en verwijderen van entiteiten voor de interne gebruiker en de administrator .....	13
Figuur 3: het wijzigen en verwijderen van entiteiten en het registreren van gebruikers voor de administrator .....	14
Figuur 4: deployment diagram .....	18
Figuur 5: DroneFlighttabel met al haar relaties .....	20
Figuur 6: QualityReporttabel met al haar relaties .....	22
Figuur 7: DroneLogEntrytabel met al haar relaties .....	23
Figuur 8: logintabellen .....	24
Figuur 9: simple factory pattern met parser-klassen .....	26
Figuur 10: pop-up van een afbeelding in de view map .....	28
Figuur 11: de Images-pagina van een dronevlucht .....	28
Figuur 12 werking Web API controllers .....	31
Figuur 13: GET methode van de GCPCController .....	32
Figuur 14: detailpagina van een Drone Flight .....	33
Figuur 15: code snippet: abort-functionaliteit .....	34
Figuur 16: code snippet: controle op bestandsextensie .....	35
Figuur 17: code snippet: verbergen van div-elementen .....	35
Figuur 18: code snippet: beforeSend callback bij uploaden van één of meerdere bestanden .....	36
Figuur 19: code snippet: uploadProgress callback bij het uploaden van één of meerdere bestanden .....	36
Figuur 20: code snippet van de functie startParsing .....	37
Figuur 21: code snippet: helper-functie updatet de progress bar via aan HTTP get call .....	37
Figuur 22: code snippet: instellen en tonen van alle resultaatvelden na afronden parsen .....	38
Figuur 23: code snippet: de verschillende error codes bij het uploaden van bestanden .....	39
Figuur 24: berekenen van totale bestandsgroottes .....	41
Figuur 25: controle bestandsgroottes .....	41
Figuur 26: pop-up voor een vlucht in een map overview .....	42
Figuur 27: de map overview met verschillende datapunten van een dronevlucht .....	42
Figuur 28: LayerList widget .....	43
Figuur 29: search widget .....	43
Figuur 30: legend klasse .....	44
Figuur 31: PopupTemplate .....	44
Figuur 32: Feature klasse .....	45
Figuur 33: color coding menu .....	45
Figuur 34: sequentiediagram over het bekijken van dronevluchten als anonieme gebruiker .....	47
Figuur 35: sequentiediagram over het inloggen in de webapplicatie .....	48
Figuur 36: sequentiediagram over het uploaden van bestanden .....	50
Figuur 37: Unit test voor de index methode van de DroneFlightsController .....	51

## Inleiding

Jan De Nul (de klant) is een grote multinational die zeer veel data genereert, maar waarvan slechts een deel benut wordt om visualisaties te maken. Er worden drones gebruikt om de werf veiliger te maken voor het personeel en om op een eenvoudige manier, vanop een welbepaalde hoogte, een afgebakend oppervlak in kaart te brengen. Dit betekent dat het vaak niet langer nodig is om de werf fysiek te betreden om foto's te nemen. Bovendien kunnen ook moeilijk bereikbare plaatsen toch makkelijk gefotografeerd worden. Andere data, zoals de data die gelogd wordt in de logbestanden van een drone, is beschikbaar, maar wordt op dit moment niet onmiddellijk verwerkt en gebruikt.

Jan De Nul heeft nood aan een droneplanning-tool met een uitgebreide, centrale databank die al deze volumes aan data verwerkt en opslaat. Voorbeelden van data zijn: dronevluchtdata, coördinaten, locatiedata, foto's en logboekdata. Nadien kan Jan De Nul via de visualisatie van deze data een overzicht krijgen van de werf en inzicht verwerven om vervolgens optimalisaties toe te laten. Dit betekent dat de ontwikkelde tool in staat moet zijn om verschillende kenmerken, zoals de diepte van een rivier, te accentueren met een gepaste visualisatietechniek. Hiervoor werkt de klant op dit moment met de *Geographical Information System* (GIS) software *ArcGIS* (ArcGIS API, z.j.). Dit softwarepakket bevat uitgebreide opties aan programmeertaalkeuzes en een ruim aanbod aan visualisaties.

Op basis van de informatie die Jan De Nul verstrekke, werden vijf doelstellingen opgesteld:

- (1) De eerste en onmiddellijk meest cruciale doelstelling bestaat erin alle aanwezige data in kaart te brengen en logisch te groeperen. Vervolgens wordt hieruit in *SQL Server Management Studio* met *SQL Server 2019* een databankmodel aangemaakt dat de relaties tussen de verscheidene datagroepen beschrijft en vastlegt. Dit model kan eenvoudig uitgebreid worden, indien hier een noodzaak voor zou bestaan.
- (2) De databank, beschreven door voorgaand model, moet nu data gaan bevatten. De tweede doelstelling beoogt daarom om echte data los te laten op dit model en het te onderwerpen aan enkele testen. De data van Jan De Nul, die relevant zijn voor dit project, zijn beschikbaar onder verschillende bestandsvarianten, zoals csv, pdf, tfw en xyz. Jan De Nul genereert kwaliteitsrapporten van hun vluchten en wil deze informatie makkelijk kunnen opslaan in de databank. Verder vult elke piloot op dit moment een papieren logboek in voor hun dronevluchten. Dit moet vervangen worden door een eenvoudige interface die hoort bij de databank en deze informatie moet bevatten.

Dit betekent dat er een applicatie ontworpen moet worden die bestanden van deze types automatisch kan verwerken na uploaden. Deze gegevensverwerking of *parsing* van data gebeurt in dit geval met *parser*-klassen die via een *simple factory pattern* beschreven worden.

Er wordt gebruikgemaakt van het *Entity Framework* (EF) in de computertaal C# dat via *Object Relational Mapping* (ORM) objecten aanmaakt van de databanktabellen. Alle ingelezen data wordt weggeschreven in deze objecten en nadien opgeslagen in de databank.

- (3) De derde doelstelling beschrijft hoe Jan De Nul deze data kan raadplegen en op welke manier ermee gewerkt kan worden. Er wordt een webapplicatie ontworpen die toelaat deze data te bekijken en, indien noodzakelijk, aan te passen met een manuele ingreep. Deze applicatie beschrijft in eerste instantie alle dronevluchten, alle drones en alle piloten die in de databank aanwezig zijn. In tweede instantie kan meer gedetailleerde data geraadpleegd worden door de details van deze hoofdcategorieën te bekijken. Voorbeelden hiervan zijn de eerder vermelde kwaliteitsrapporten en dronelogboeken. Zeer specifieke data, die gebruikt worden in de vierde doelstelling, visualisatie, worden niet getoond in de webapplicatie en zijn enkel rechtstreeks aanspreekbaar via de databank. Het gaat hier immers om zeer grote hoeveelheden, moeilijk leesbare data.
- (4) De vierde doelstelling bestaat erin om, zoals eerder werd vermeld, de gigantische volumes aan data te visualiseren met de ArcGIS API. Deze visualisaties kunnen aangesproken worden in de webviewer sectie van de webapplicatie en hebben een belangrijke subdoelstelling: de webviewer moet eenvoudig navigeerbaar zijn voor leken en hen toelaten om op intuïtieve manier een beeld van een visualisatie te delen. Enkele voorbeelden van visualisaties zijn: het tonen van dronepaden; het tonen van *ground control points* (GCP); het visualiseren van het batterijgebruik van de drone; het visualiseren van hoogteverschillen in de gescande oppervlakte; etc. De bibliotheek van ArcGIS heeft een ruim aanbod aan functionaliteiten, wat toelaat om op vraag nadien nog visualisaties toe te voegen aan de webapplicatie.
- (5) Als laatste doelstelling moet het mogelijk zijn voor Jan De Nul om de ingelezen data opnieuw te kunnen exporteren naar bestanden van een ander bestandsformaat.

Het uiteindelijke doel van deze bachelorproef is om deze doelstellingen te verwezenlijken. In het eerste hoofdstuk, **Gebruikersaspecten**, wordt een gedetailleerde beschrijving van de opdracht vanuit het standpunt van de gebruiker (hier: Jan De Nul) gegeven. In dit hoofdstuk



worden verder de gewenste softwarevereisten, use case-diagrammen en volledige featurelijst beschreven.

Het tweede hoofdstuk, **Systeemarchitectuur**, bespreekt hoe het programma deze doelstellingen en vereisten realiseert en hoe deze gestructureerd zijn. Aan de hand van een databankdiagram en klassendiagrammen zal de opbouw van de applicatie beschreven worden en getoond worden op welke manier de verschillende onderdelen samenwerken.

Het derde hoofdstuk, **Testplan**, geeft een overzicht van de voorziene testplannen. Er wordt per type test een voorbeeld gegeven.

Het vierde hoofdstuk, **Evaluaties en discussies**, behandelt de performantie van de applicatie, de beveiliging ervan en zijn schaalbaarheid. Als laatste wordt ook even ingegaan op problemen en geleerde lessen.

Het vijfde en laatste hoofdstuk, **Handleidingen**, is bedoeld voor de klant, Jan De Nul, en haar eindgebruikers. Dit hoofdstuk is van cruciaal belang voor hen, omdat het de installatiehandleiding en gebruikershandleiding bevat.

Doorheen dit verslag zal verwezen worden naar de initiële doelstellingen, zodat het voor de lezer op elk moment duidelijk is wanneer een doelstelling behandeld en geïmplementeerd wordt.

## 1. Gebruikersaspecten

De drones van Jan De Nul verzamelen veel gegevens op hun vluchten, zoals foto's van de site, coördinaten van de drone en data die door de drone zelf gelogd wordt in zijn intern logboek. De gebruiker verwacht een databank waaraan al deze vluchtgegevens eenvoudig toegevoegd en later opnieuw opgehaald kunnen worden.

Naast de gegevens die door de drones verzameld worden, zijn er nog talrijke andere gegevens beschikbaar. Allereerst houdt elke piloot momenteel een papieren logboek bij voor de drone en zichzelf. Het is de bedoeling dat de piloot deze gegevens in de databank kan ingeven en deze logboeken in de databank bijgehouden worden. Verder wordt na elke dronevlucht een kwaliteitsrapport opgesteld dat een analyse van de door de drone verzamelde gegevens bevat. Dit kwaliteitsrapport moet eveneens in de databank komen.

De klant heeft dus nood aan twee hoofdcomponenten: een databank om makkelijk data te kunnen verwerken en opslaan, en een interface om gegevens toe te voegen aan de databank en ook te kunnen opvragen.

### 1.1 Databank

De databank wordt ontworpen in *SQL Server Management Studio (SSMS)* van Microsoft met *SQL Server 2019*, omdat Jan De Nul *in-house* eveneens met SQL Server werkt. Via een databankmodel worden de verscheidene datagroepen beschreven en hun onderlinge relaties vastgelegd. Dit model moet eenvoudig uitbreidbaar zijn indien nieuwe documentatie en data toegevoegd zou moeten worden aan databank.

### 1.2 Webapplicatie

Er moet makkelijk naar een vlucht genavigeerd kunnen worden, opdat zijn gegevens opgevraagd kunnen worden. De interface komt er onder de vorm van een webapplicatie waarin het bovendien mogelijk is om data te visualiseren op basis van gespecificeerde attributen.

De applicatie wordt gebouwd met ASP.NET MVC 5 in Microsoft's Visual Studio 2019. ASP.NET is een *open-source server-side web-application framework* dat toelaat om in C# moderne webapplicaties en -services te bouwen in de .NET omgeving. Het eenvoudige *Model-View-Controller pattern* laat toe om dynamisch krachtige webpagina's aan te maken.

Verder wordt gewerkt met het *Entity Framework*, om met *Object Relational Mapping* (ORM) het databasemodel te *mappen* op automatisch gegenereerde klassen. In dit eindwerk zal met de benaming entiteit verwezen worden naar objecten van deze klassen.

Visualisaties van geografische data gebeuren binnen Jan De Nul met het ArcGIS platform (ArcGIS for Developers, z.j.), waardoor bijgevolg verwacht wordt dat ook de webapplicatie hiervan gebruik zal maken. Hiervoor wordt de ArcGIS API voor JavaScript gebruikt en wordt binnen de webapplicatie een aparte sectie voorzien voor visualisaties.

### 1.3 Use case-diagrammen

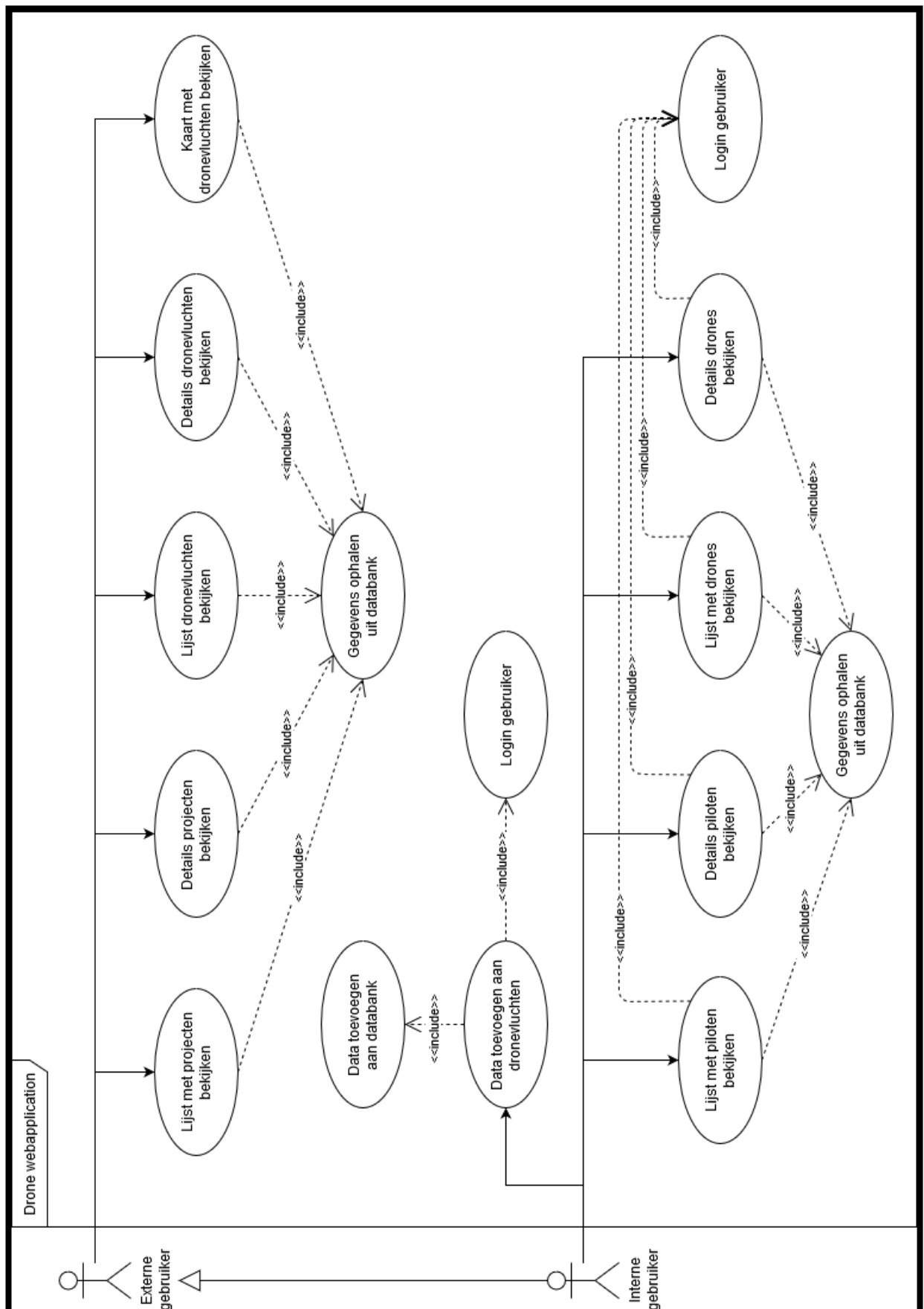
Het gebruik van de webapplicatie wordt verduidelijkt in **Figuren 1, 2 en 3** op pagina's 12, 13 en 14 via drie use case-diagrammen. Hierop is te zien hoe de verschillende actoren, namelijk de externe gebruiker, interne gebruiker en de administrator, kunnen interageren met de verschillende functionaliteiten van het systeem.

De externe gebruiker is een gebruiker die *niet* ingelogd is (**Figuur 1**). Deze kan geen gegevens toevoegen, aanpassen of verwijderen. Hij is bovendien ook beperkt in het bekijken van informatie: hij kan enkel projecten, dronevluchten en de kaart met dronevluchten bekijken. Drone-informatie en pilootinformatie zijn aldus niet toegankelijk voor deze gebruiker.

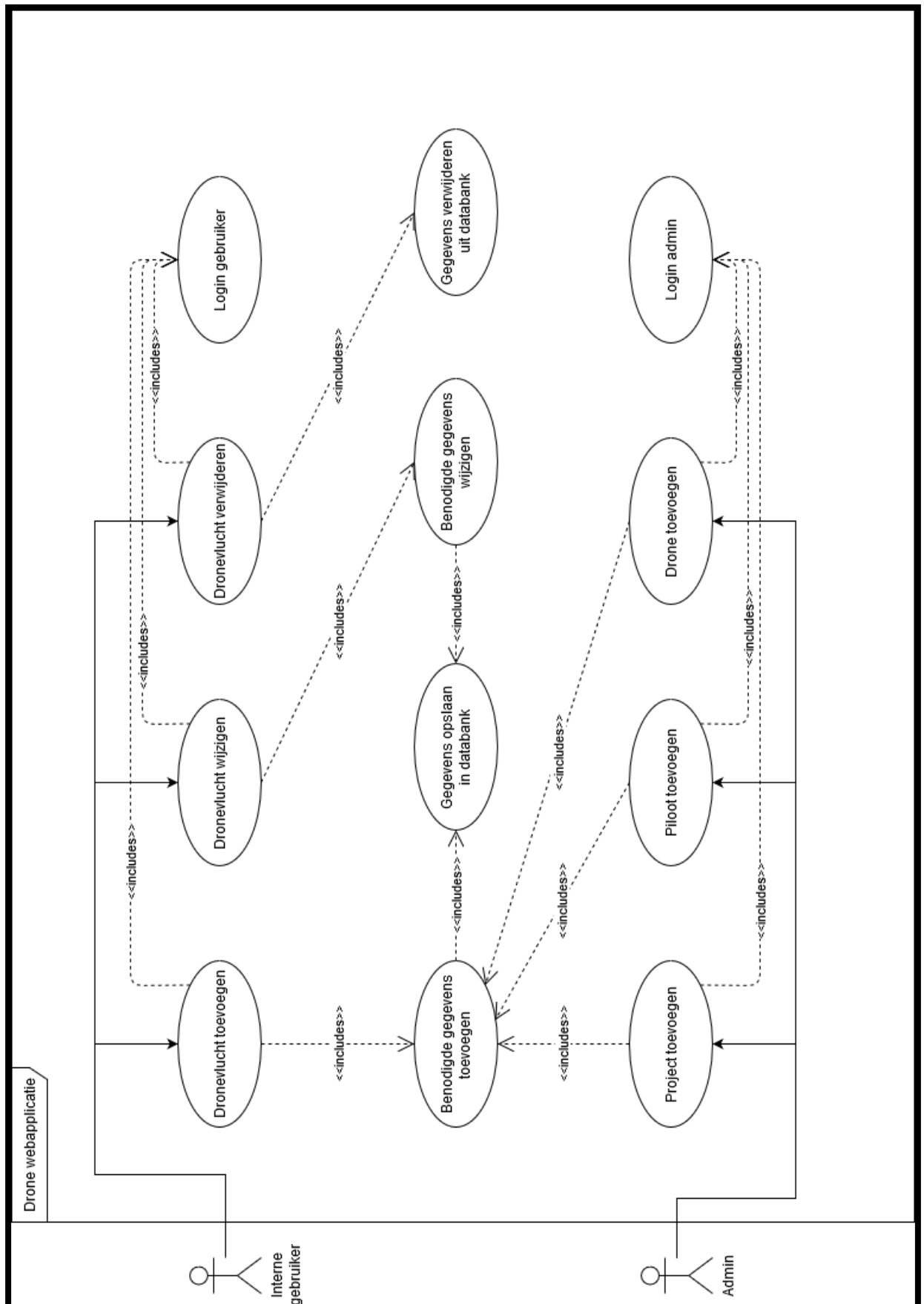
Een ingelogde gebruiker, de interne gebruiker, kan daarentegen *wel* alle informatie bekijken (**Figuren 1 en 2**). Daarnaast kan hij ook dronevluchten toevoegen, wijzigen en verwijderen. Deze acties hebben onmiddellijk effect op de databank aan de server-sidekant. Verder kan de interne gebruiker ook extra data, die bij de dronevluchten horen, in de vorm van bestanden toevoegen aan het systeem (de databank).

De administrator beschikt over CRUD (*Create, Read, Update, Delete*) functionaliteiten voor alle onderdelen van de applicatie (**Figuren 2 en 3**). Naast dronevluchten kan hij dus ook piloten, drones en projecten toevoegen, wijzigen en verwijderen uit het systeem. Naast deze functionaliteiten is het ook de taak van de administrator om nieuwe gebruikers toe te voegen aan het systeem. Deze nieuwe gebruikers worden daarbij direct opgeslagen in de databank.

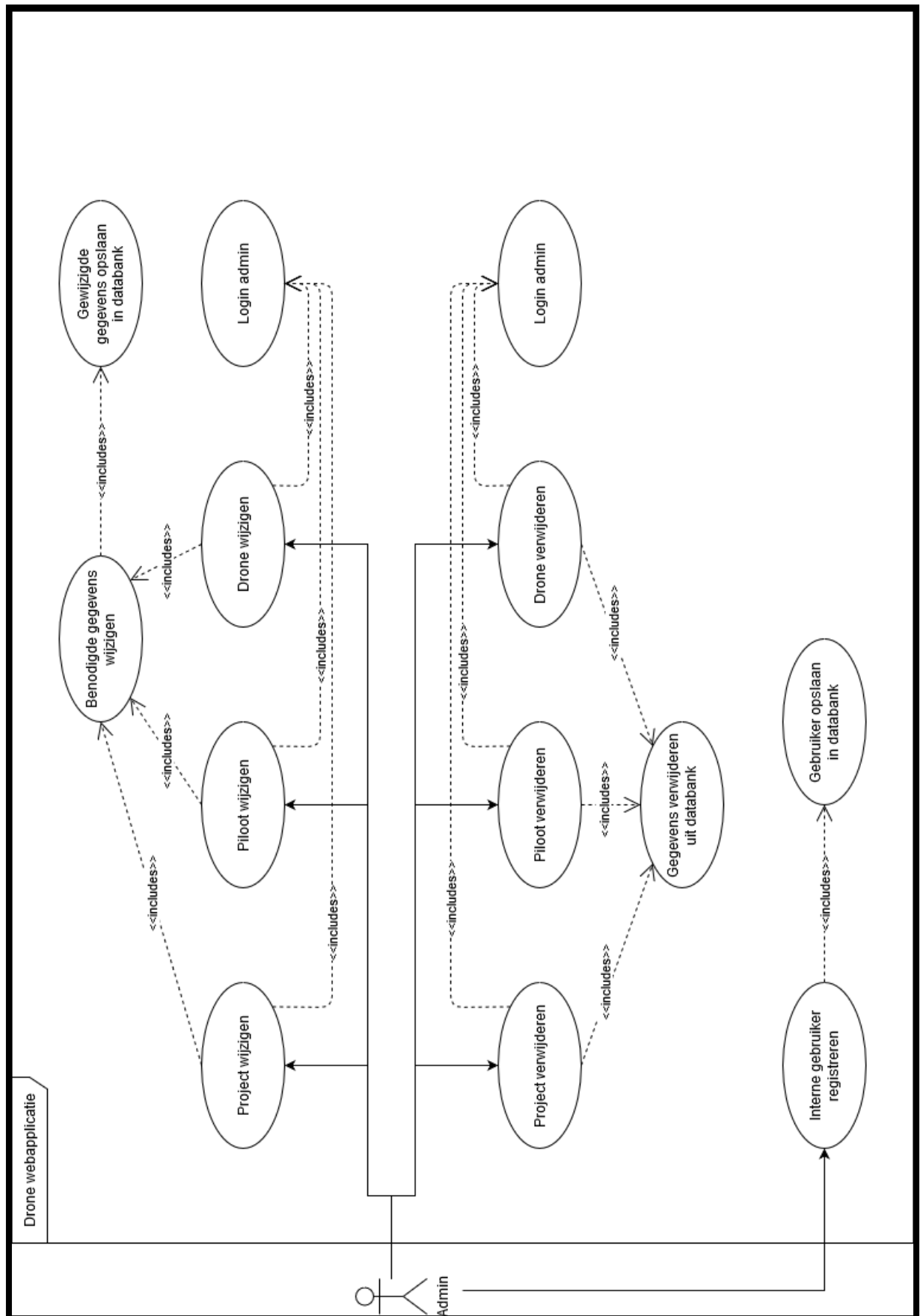
Een complete beschrijving van de verscheidene interacties die plaatsvinden binnen deze use case-diagrammen is te vinden in Appendix A.



Figuur 1: het bekijken van entiteiten in het systeem voor de interne en externe gebruiker



Figuur 2: het toevoegen, wijzigen en verwijderen van entiteiten voor de interne gebruiker en de administrator



Figuur 3: het wijzigen en verwijderen van entiteiten en het registreren van gebruikers voor de administrator

## 1.4 Featurelijst

Hieronder volgt een opsomming van alle features in dit project, inclusief hun complexiteit<sup>1</sup> en prioriteit<sup>2</sup>, per sprint.

### 1.4.1 Sprint 1

- Ontwerp van een databankmodel dat op een logische manier de relaties tussen de verschillende entiteiten beschrijft (gemiddeld, zeer belangrijk);
- Keuze van de databank (SQL Server 2019) en ORM (Entity Framework 6) (makkelijk, redelijk belangrijk);
- Ontwerp en implementatie van het model: de parser-klassen via een *simple factory pattern* (vrij moeilijk, zeer belangrijk);
- Keuze voor MVC *pattern* (makkelijk, redelijk belangrijk);
- Implementeren van de Controllers (vrij moeilijk, zeer belangrijk);
- CRUD-functionaliteit voor dronevluchten, drones en piloten (vrij gemakkelijk, zeer belangrijk);
- Schrijven van de Views (de website) (vrij moeilijk, belangrijk);
- Implementeren van *searchable* en *sortable* tabellen met *paging* (gemiddeld, onbelangrijk);
- *Dependency Injection* met Unity (makkelijk, redelijk belangrijk);
- Uploaden van bestanden die geparset kunnen worden (moeilijk, zeer belangrijk)
- Voorzien van additionele razorpagina's (View) die details geven van de data die ingelezen wordt (vrij makkelijk, redelijk belangrijk);
- Tonen van gepaste foutpagina's op de website aan de gebruiker (makkelijk, belangrijk).

---

<sup>1</sup> Makkelijk, vrij gemakkelijk, gemiddeld, vrij moeilijk, moeilijk

<sup>2</sup> Onbelangrijk, redelijk belangrijk, belangrijk, zeer belangrijk

### 1.4.2 Sprint 2

- Projecttabel in databank met benodigde velden (vrij gemakkelijk, belangrijk)
- Totale vliegtijd van een drone automatisch berekenen en toevoegen aan de databank (vrij gemakkelijk, belangrijk);
- Velden uit logboeken (zoals *type of activity*) toegevoegd aan databank en mogelijkheid voorzien om deze velden in te vullen via de *user interface* (vrij gemakkelijk, belangrijk);
- *Reverse geocoding*, omvormen van latitude- en longitudecoördinaten naar de naam van de locatie (vrij moeilijk, belangrijk);
- Duidelijk weergeven van verplichte velden bij invullen van gegevens in de *user interface* (vrij makkelijk, redelijk belangrijk);
- Dronevluchten per piloot en drone in apart overzicht; te bekijken in *user interface* (gemiddeld, redelijk belangrijk);
- Extra functies GUI, zoals van piloot naar bijhorende vluchten kunnen gaan (gemiddeld, redelijk belangrijk);
- Bij het uploaden van files staat een *progress bar* die weergeeft hoeveel procent van de file reeds geüpload en verwerkt (geparset) is (moeilijk, redelijk belangrijk);
- *Controletool ctrl points* in *point cloud* (vrij moeilijk, belangrijk);
- De *track* van de drone in de map view kunnen visualiseren op basis van een attribuut (hoogte, batterijstand...) met een gepaste *color ramp* (vrij moeilijk, zeer belangrijk);
- Legende met informatie over de vlucht in de *map view* (moeilijk, belangrijk);
- Visualisatie van *point clouds*, gcp's en ctrl's (vrij moeilijk, zeer belangrijk);
- Een *toggle list* bij de *map view* zodat de gebruiker zelf kan kiezen welke *layer* (soort data) getoond wordt van een dronevlucht (gemiddeld, belangrijk);
- *Pop-up templates* als de gebruiker op eender welk punt (gcp, ctrl...) klikt om de bijhorende coördinaten en andere attributen te zien (vrij moeilijk, belangrijk);
- Zoekbalk en *measurement widget* in de *map view* (gemakkelijk, redelijk belangrijk);
- Er kan van attribuut (en bijhorende *color ramp*) gewisseld worden door op een bepaalde toets te drukken om zo de *track* te visualiseren. De legende van een vlucht kan verborgen of getoond worden, eveneens met een toets. (gemakkelijk, vrij belangrijk).



### 1.4.3 Sprint 3

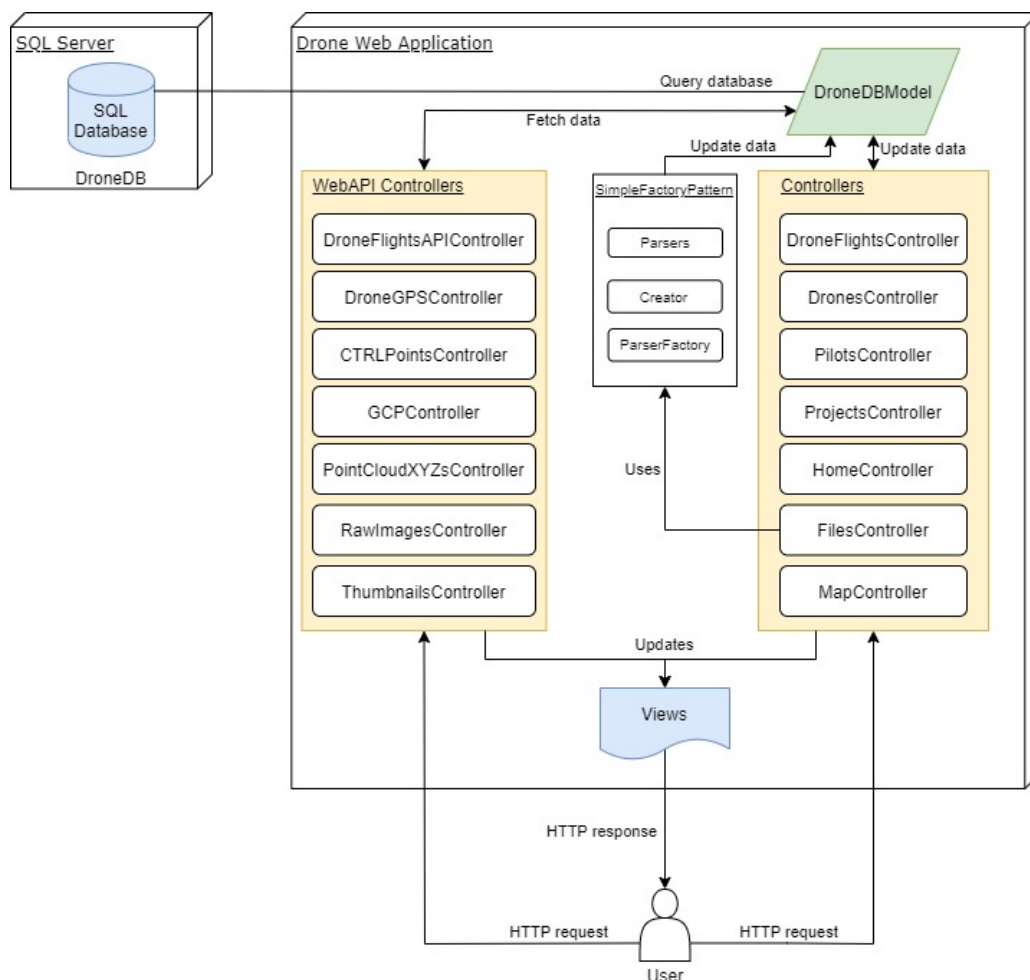
- Meerdere bestanden tegelijk kunnen uploaden (gemakkelijk, belangrijk);
- Bestanden uploaden kan nu veiliger gebeuren. Er werden verscheidene controles op bestandsgroottes en type bestanden toegevoegd. De gebruiker wordt nu ook op de hoogte gebracht van het al dan niet succesvol *parsen* van de bestanden die hij of zij uploadde. (moeilijk, zeer belangrijk);
- De *progress bar* geeft nu een totaal weer voor alle bestanden in plaats van elk afzonderlijk (vrij gemakkelijk, redelijk belangrijk);
- Elke piloot en drone hebben nu een eigen overzicht van hun dronevluchten (gemakkelijk, vrij belangrijk);
- Een drone heeft nu een *threshold*. Dit getal (een tijd) geeft aan wanneer de drone gecontroleerd moet worden. De administrator kan ook zelf een drone markeren voor controle. Na controle kan hij de drone afvinken en wordt een nieuwe *threshold*-tijd berekend. (gemiddeld, belangrijk);
- Notitie-sectie voor een drone (gemakkelijk, onbelangrijk);
- Knoppen voor de map en upload (vrij gemakkelijk, onbelangrijk);
- *Images* die door de drone genomen werden, kunnen geupload en bekeken worden op hun eigen pagina en in de *map view* (gemiddeld, belangrijk);
- De *parsers* voor XYZ- en DAT-bestanden werden herschreven, wat leidt tot grote performantiewinsten (moeilijk, belangrijk);
- Een *map overview* van alle dronevluchten waarbij de gebruiker naar een specifieke vlucht kan navigeren (gemiddeld, belangrijk);
- Verbeteringen in de *map view* GUI, waaronder een aanklikbare menu voor *color coding* (gemiddeld, vrij belangrijk);
- Het exporteren van de logboeken voor de piloten en drones naar PDF- en XYZ-bestanden (gemiddeld, zeer belangrijk);
- Het voorzien van een loginpagina en bepaalde functionaliteiten enkel beschikbaar maken voor bepaalde rollen (vrij moeilijk, belangrijk);
- Bij de verplichte velden wordt een duidelijke boodschap weergegeven wanneer deze niet ingevuld worden (vrij gemakkelijk, onbelangrijk);
- In de database bevindt zich in de *RawImage* tabel alle data die belangrijk is komende van de *raw images* (gemakkelijk, redelijk belangrijk)
- DAT-bestanden worden automatisch via DatCon omgezet wanneer deze geupload worden in de webapplicatie (zeer moeilijk, zeer belangrijk)
- *Breadcrumbing links* om makkelijk naar een vorige pagina terug te kunnen keren (vrij gemakkelijk, redelijk belangrijk)

## 2. Systemarchitectuur

In dit hoofdstuk wordt besproken hoe het programma doelstellingen 1, 2, 3 en 4 realiseert en hoe deze opgebouwd zijn. Allereerst wordt de webapplicatie beschreven vanuit het oogpunt van het *deployment diagram*. Vervolgens wordt dieper ingegaan op de databank a.d.h.v. een databankdiagram. Verder wordt het MVC pattern dat wordt gebruikt in ASP.NET toegelicht m.b.t. dit project. Als laatste worden de frontend en de javascriptbestanden beschreven, en wordt de flow van de webapplicatie aan de hand van twee *sequence diagrams* geïllustreerd.

### 2.1 High-levelsystemmodel (*deployment diagram*)

Het *deployment diagram* in **Figuur 4** bestaat uit twee onderdelen: de SQL Server component en de ASP.NET MVC 5 component. De SQL Server bevat de databank waarin alle data over dronevluchten bijgehouden wordt en waaraan de gebruiker data kan toevoegen door middel van de CRUD-functionaliteiten. De databank wordt in de model component van het MVC pattern voorgesteld als een verzameling van entiteiten aan de hand van het Entity Framework 6.



Figuur 4: deployment diagram

De views zijn niet rechtstreeks gekoppeld aan het model maar communiceren via tussenliggende *controllers*. De gepaste controller wordt gekozen via routing en geeft steeds de veranderingen door aan het model. Op basis van dit model wordt de databank ten slotte aangepast.

## 2.2 Databankdiagram

Het databankdiagram visualiseert de structuur van de informatie die opgeslagen dient te worden in de databank en de relaties tussen deze data. Dit model bestaat uit vier grote onderdelen: DroneFlight, QualityReport en DroneLogEntry. Deze zijn respectievelijk te zien in **Figuren 5, 6, 7 en 8** op pagina's 20, 22, 23 en 24.

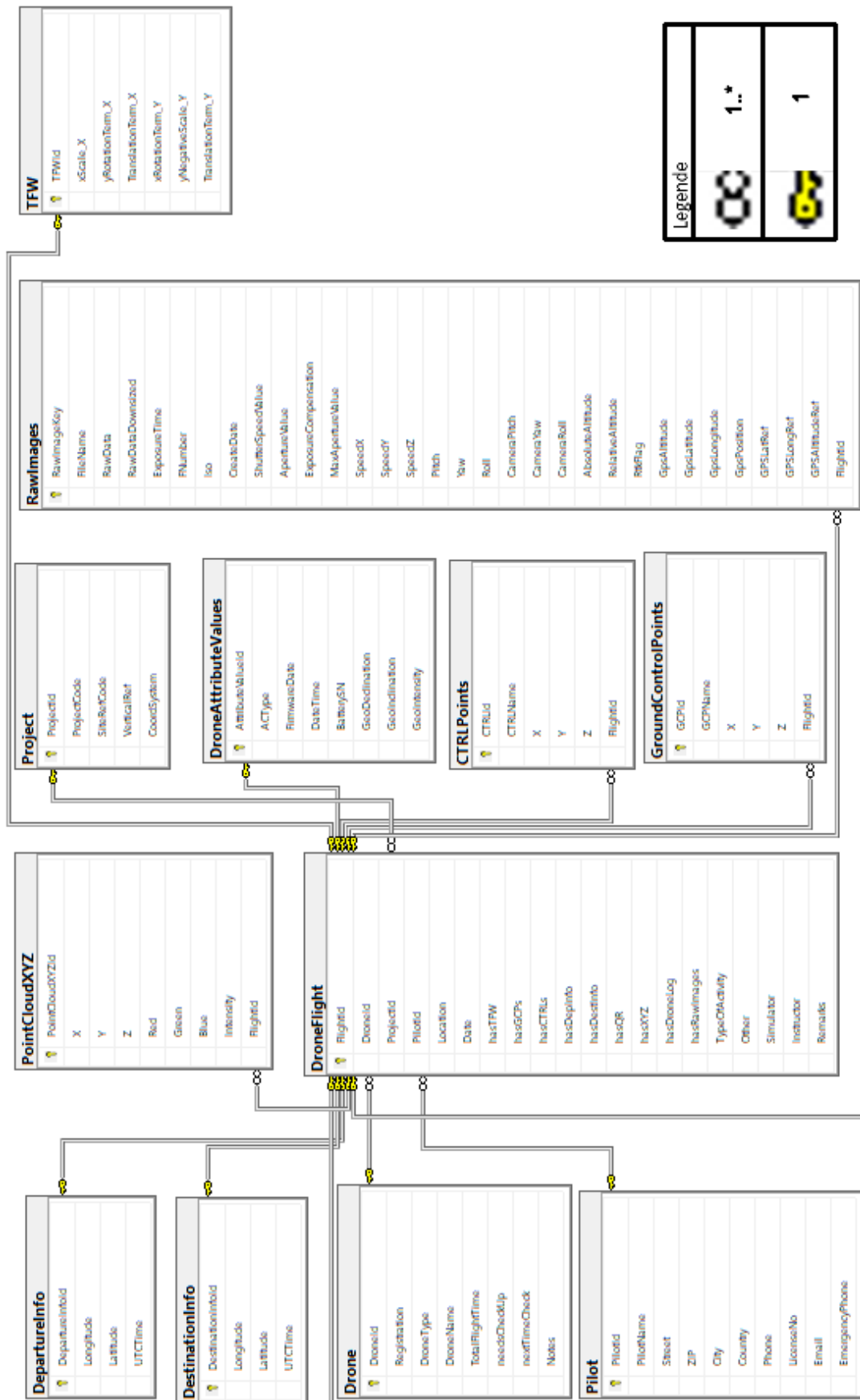
Alle andere tabellen zijn aan deze hoofdtabellen gelinkt met een één-op-één relatie of een één-op-veel relatie. Het linken gebeurt respectievelijk aan de hand van een *mapping* van een *Primary Key* op een andere *Primary Key*, of aan de hand van een *mapping* van een *Primary Key* op een *Foreign Key*. Een verbinding met een sleutel aan beide kanten wijst op een één-op-één relatie en een verbinding met een sleutel en een oneindigheidssteken wijst op een één-op-veel relatie.

Zo is de *Primary Key* van de tabel DepartureInfo, DepartureInfoId, één-op-één gemapt op de *Primary Key* van de tabel DroneFlight, FlightId. Dit betekent dat een DroneFlight slechts één DepartureInfo kan hebben. Een dronevlucht kan immers maar één starttijdstip hebben.

Een voorbeeld van een één-op-veel relatie gebeurt met een mapping van een *Primary Key* op een *Foreign Key*. Zo heeft een dronevlucht (tabel DroneFlight) meerdere Ground Control Points (tabel GroundControlPoints). Deze laatstgenoemde tabel heeft een *Foreign Key*, DroneId, die gemapt wordt op de *Primary Key*, DroneId, van de tabel DroneFlight. De *Foreign Key* kan dus gezien worden als een sleutel die de tabel met de *Primary Key* toegang geeft tot de tabel die de *Foreign Key* bevat, in dit geval de tabel GroundControlPoints.

### 2.2.1 DroneFlighttabel

De eerste en tevens ook belangrijkste tabel in het databankdiagram is de DroneFlight tabel, te zien op **Figuur 5** op pagina 20.



Figuur 5: DroneFlighttabel met al haar relaties

Deze tabel vormt de basis van het model en houdt alle informatie bij over de dronevluchten, zoals de datum en locatie van de vlucht. Verder bevat deze tabel ook een reeks booleans (hasTFW, hasGCPs, hasCTRLs, hasDepInfo, hasDestInfo, hasQR, hasXYZ, hasDroneLog en hasRawImages) die bijhouden of een bepaald bestand of een bepaald type informatie ingelezen is en op dit moment bijgehouden wordt.

Deze tabel heeft bovendien drie *Foreign Keys*: een DroneId, PilotId en ProjectId. Op deze manier kunnen drones en piloten informatie opvragen over hun vluchten en kan elke vlucht gelinkt worden met het project waartoe de vlucht behoort. Merk op dat een drone, een pilot en een project met een dronevlucht een één-op-veel relatie beschrijven. Een drone en een pilot kunnen immers meerdere dronevluchten uitvoeren en binnen een project kunnen er meerdere dronevluchten plaatsvinden.

De DepartureInfo en DestinationInfo tabellen bevatten informatie over een drone zijn tijdstip van vertrek en van aankomst bij een dronevlucht. In deze tabellen wordt ook informatie bijgehouden over de coördinaten van vertrek en aankomst.

De tabel PointCloudXYZ bevat de coördinaten en RGB-waarden die nodig zijn om een *point cloud* aan te maken. Een vlucht heeft vaak miljoenen PointCloudXYZ *entries*.

De tabel GroundControlPoints beschrijft de coördinaten van een *ground control point* en bevat ook een *Foreign Key*, FlightId, dat een vlucht toelaat om deze informatie op te vragen. Deze tabel bevat x-, y- en z-waarden om alle foto's uit de RawImagestabel geografisch juist te positioneren (Understanding world files, z.j.). Zodoende kunnen alle foto's aan elkaar gehangen worden om zo een groot beeld te verkrijgen van de werf. In de voorgenoemde RawImages tabel staat alle informatie horende bij deze *raw Images*.

Het voorgaande geldt ook voor de tabel CTRLPoints, dewelke punten beschrijft die gebruikt worden ter verificatie van de juistheid van ingelezen datapunten.

De tabel TFW bevat rotatie- en translatie informatie, alsook wereldcoördinaten die gebruikt worden bij een *tiff image* bestand in een GIS applicatie. Deze informatie laat toe een andere, externe databank aan te spreken en de juiste resultaten te verkrijgen in de vorm van een kaart.

De laatste tabel DroneAttributeValue houdt informatie over de drone bij die tijdens de dronevlucht wijzigt en informatief kan zijn voor analyse naderhand.

De DroneFlighttabel is met een één-op-één relatie verbonden met de QualityReporttabel.

## 2.2.2 QualityReporttabel

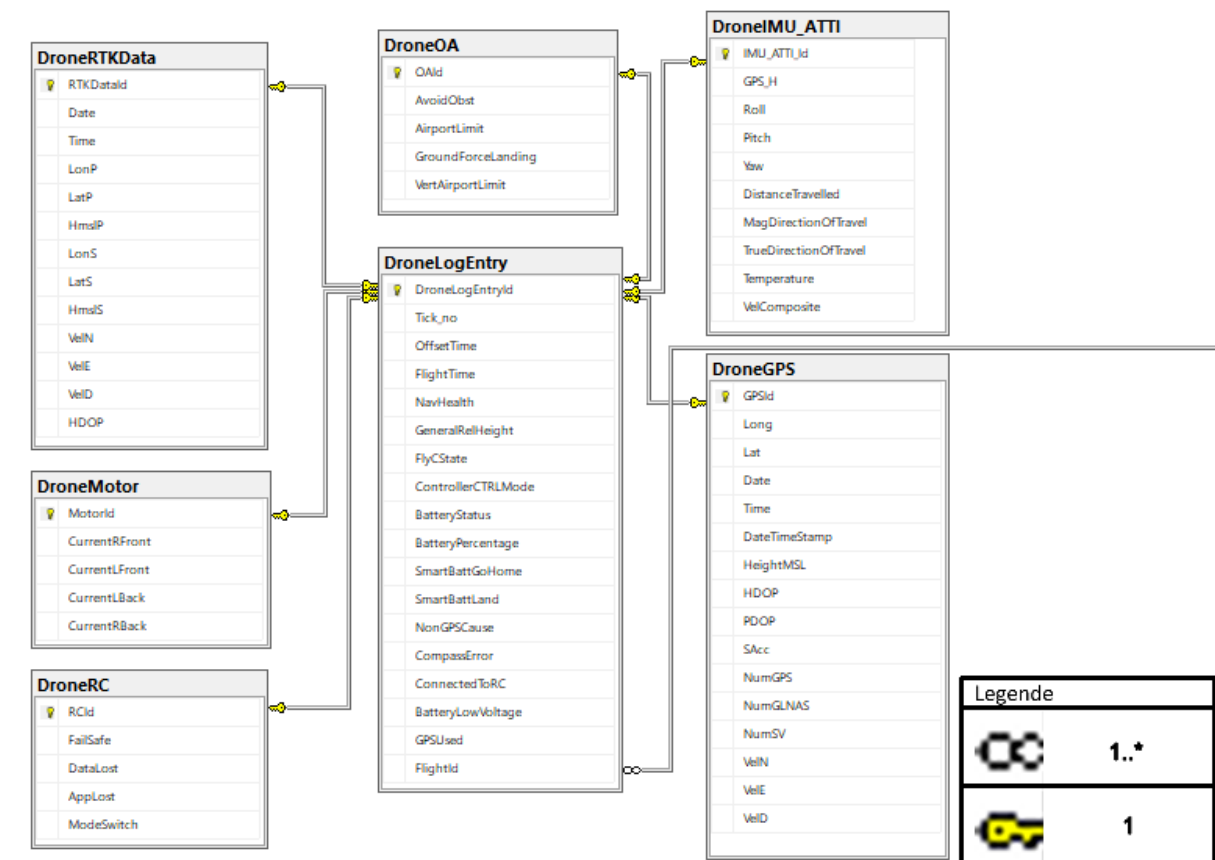
In de QualityReporttabel in **Figuur 6** wordt alle data bijgehouden die ingelezen wordt uit het kwaliteitsrapport. Dit is een PDF-bestand dat de output beschrijft van een analyse in het programma Pix4D. Tabellen die een relationeel verbonden zijn met deze tabel zijn: Uncertainty, AbsoluteGeolocationVariance en GCPError. Op deze manier is de ingelezen informatie uit het kwaliteitsrapport op een logische manier gegroepeerd.



Figuur 6: QualityReporttabel met al haar relaties

### 2.2.3 DroneLogEntrytabel

In **Figuur 7** staat de DroneLogEntrytabel centraal. Deze tabel bevat voor elke *tick* data ingelezen uit het dronelogbestand. Alle tabellen (DroneGPS, DroneOA, DroneIMU\_ATT, DroneRTKData, DroneMotor en DroneRC) hebben een één-op-één relatie met de hoofdtabel, DroneLogEntry, en bevatten dus waarden voor elke *dronelog entry*. De tabel DroneLogEntry zelf heeft een één-op-veel relatie met de DroneFlighttabel, aangezien elke dronevlucht meerdere *dronelog entries* heeft.

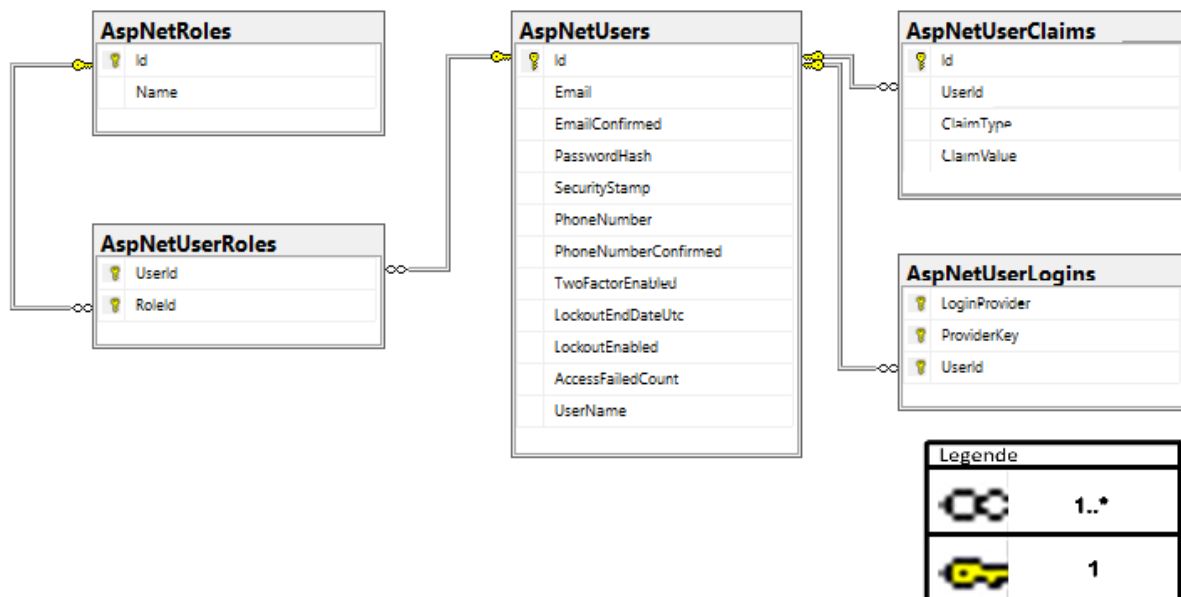


Figuur 7: DroneLogEntrytabel met al haar relaties

## 2.2.4 Logintabellen

De tabellen die gebruikt worden bij het loginsysteem met ASP.NET Identity zijn te zien op **Figuur 8**, hierbij staat de `AspNetUsers` tabel centraal. Deze bevat de gegevens van de geregistreerde gebruikers in het systeem. De `AspNetRoles` tabel bevat de verschillende rollen die in de webapplicatie gebruikt worden. In de `AspNetUserRoles` tabel wordt voor elke gebruiker bijgehouden welke rol aan hem is toegekend. Deze tabel heeft een één-op-veel relatie met de `AspNetUsers` tabel, een gebruiker kan dus meerdere rollen hebben.

De `AspNetUserClaims` tabel en `AspNetUserLogins` worden in onze applicatie niet gebruikt en zijn dus leeg. Ze moeten echter wel aanwezig zijn om het Identity systeem te laten werken en kunnen ook gebruikt worden voor latere uitbreidingen.



Figuur 8: logintabellen



## 2.3 MVC

De ontwikkeling van de webapplicatie gebeurt met ASP.NET in de .NET omgeving van Microsoft met een combinatie tussen de *high-level* computertaal C#, HTML, CSS en JavaScript. Er werd gekozen voor een MVC (Model-View-Controller) *pattern*.

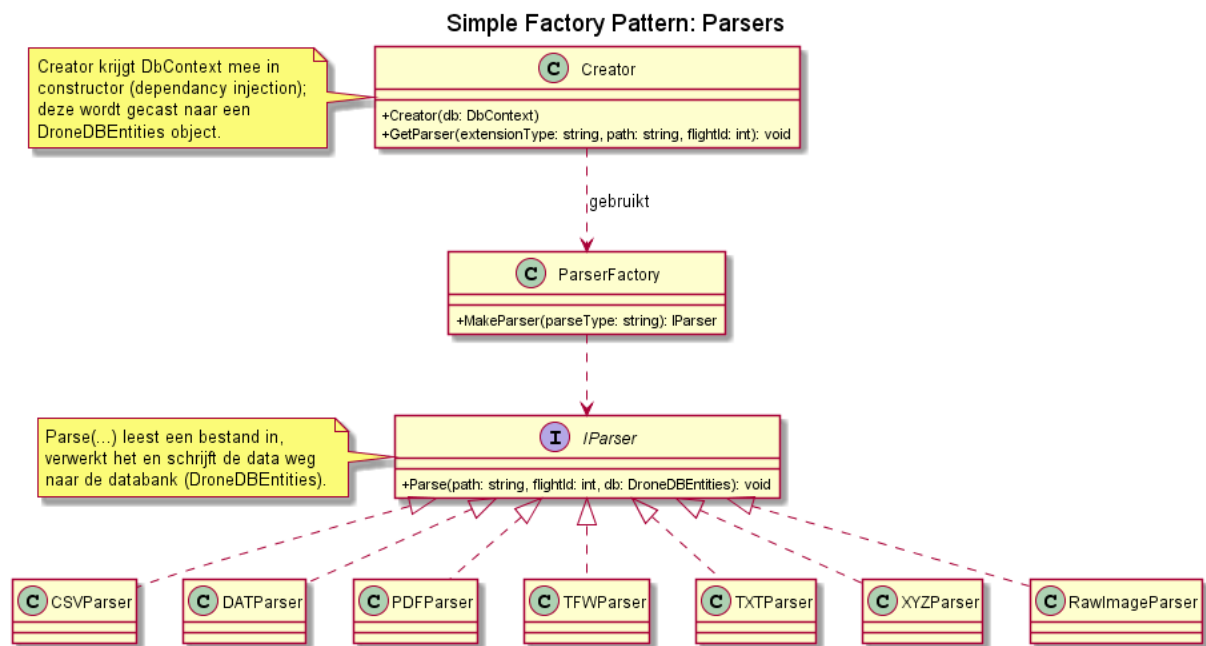
### 2.3.1 Model

Op basis van het databankmodel worden met het Entity Framework 6 via ORM entiteitsklassen aangemaakt. Hier worden databanktabellen voorgesteld als klassen en databankkolommen als velden van de overeenkomstige velden (Hoyos, 2018). Dit laat toe om op eenvoudige wijze te communiceren met de databank zonder echt expliciet SQL te moeten spreken. Dit geheel stelt de *Model* component in MVC voor.

#### 2.3.1.1 Simple Factory pattern voor parser-klassen

Dit onderdeel beschrijft het *simple factory pattern* voor het parsen van de bestanden. Dit parsen van data uit bestanden gebeurt via parser-klassen. Deze klassen worden gegroepeerd in een *simple factory pattern* (**Figuur 9**, pagina 26), waarbij een klasse Creator aangemaakt wordt in de webapplicatie. Deze klasse spreekt vervolgens met de methode GetParser een factory aan, ParserFactory, die vervolgens met de methode MakeParser de juiste parser-klasse aanmaakt en teruggeeft aan de Creator. Daarna vindt het parsen plaats met de methode Parse van de aangemaakte parser-klasse.

De constructor van Creator krijgt via *Dependency Injection* (met *Unity*) een instantie van de databank mee, zodat de parsers de ingelezen en verwerkte data kunnen wegschrijven naar de databank. Dankzij deze injectie wordt het aanmaken van dure databankconnecties beperkt.



Figuur 9: simple factory pattern met parser-klassen

#### ➤ CSVParser

Deze *parser* leest de CSV-bestanden in die de *ground control points* (GCPs) en *controle points* (CTRLs) bevatten voor een dronevlucht.

#### ➤ DATParser

Deze *parser* converteert een *drone log* DAT-bestand automatisch naar een CSV-bestand. Dit bestand wordt vervolgens geparset en alle nuttige informatie wordt weggeschreven naar de juiste dronevlucht in de databank. De *parser* kan ook reeds geconverteerde DAT-bestanden, of dus dronelogbestanden onder de vorm van CSV-bestanden, *parsen*. Het *parsen* van DAT-bestanden gebeurt door het oproepen van DatCon in een apart process in de methode `convertDat`. Hiervoor moet Java geïnstalleerd zijn op de server. De *parser* wacht tot DatCon het bestand geconverteerd heeft en *parset* daarna het gegenereerde CSV-bestand. Het pad naar DatCon wordt geconfigureerd in het bestand `Web.config` in de `<appSettings>`-tag.

Deze *parser* maakt gebruik van ADO.NET. ADO.NET leidt tot grote performantiewinsten in vergelijking met Entity Framework. Dit komt door het *object-oriented* principe van Entity Framework, wat leidt tot het aanmaken van objecten bij elke te *parsen* lijn. Het aanmaken van objecten is een dure operatie tijdens het *parsen*. Het wegschrijven van deze data naar de databank kan bovendien ook niet snel genoeg gebeuren, waardoor de keuze voor ADO.NET snel gemaakt kon worden. Hier worden *commands* gebruikt die bij elke gelezen lijn uitgevoerd worden om de data efficiënt weg te schrijven naar de databank. Elke *command* wordt ingesteld op basis van Parameter objecten om *SQL injections* te voorkomen.

In de DATparser bevindt zich ook de *reverse geocoding*, welke de coördinaten van de beginlocatie van de dronevlucht gebruikt om de naam van de locatie van de dronevlucht automatisch te bepalen. Het ophalen van deze locatiennaam gebeurt door middel van een ArcGIS REST API call, welke als parameters de x- en y-coördinaat van de locatie meekrijgt, alsook een *token*, die nodig is omdat de verkregen locatie in de databank wordt opgeslagen.

De *token* voor de *reverse geocoding* wordt telkens opnieuw opgeroepen door middel van een functie welke, net als voor de *reverse geocoding* een ArcGIS REST API call gebruikt, ditmaal met als parameters een *Client ID* en *Client Secret*, welke te verkrijgen zijn via ArcGIS. De *reverse geocoding* REST API call geeft een JSON terug waaruit de stad behorende bij de meegegeven coördinaten gehaald kan worden. Indien er geen stad hoort bij de opgehaalde locatie, omdat die bijvoorbeeld op zee is, wordt de locatie ingevuld met 'NA' en kan de gebruiker deze later nog zelf aanpassen.

➤ PDFParser

Deze *parser* leest het kwaliteitsrapport van een dronevlucht in en schrijft de nuttige informatie weg naar de databank. Hier wordt er gebruik gemaakt van *Ivy Pdf Parser*. Deze *library* is betalend.

➤ TFWParser

Deze *parser* leest het TFW-bestand van een dronevlucht in en schrijft deze informatie weg naar de databank.

➤ TXTParser

Ongebruikt.

➤ XYZParser

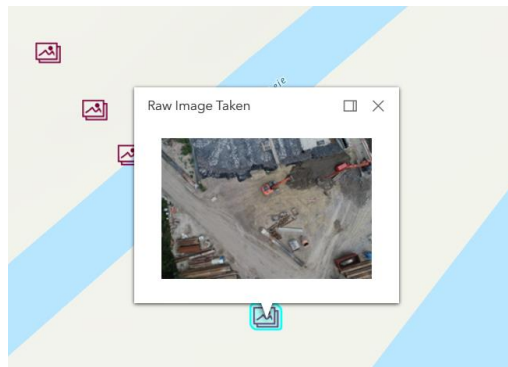
Deze *parser* leest een XYZ-bestand lijn per lijn in en schrijft de XYZ-coördinaten en RGB-waarden weg naar de juiste dronevlucht in de databank. Indien er ook een *intensity* aanwezig is op één of meerdere lijnen in dit bestand, dan wordt dit ook mee ingelezen.

Het gebruik van ADO.NET is analoog aan dat van de DATParser.

➤ RawImageParser

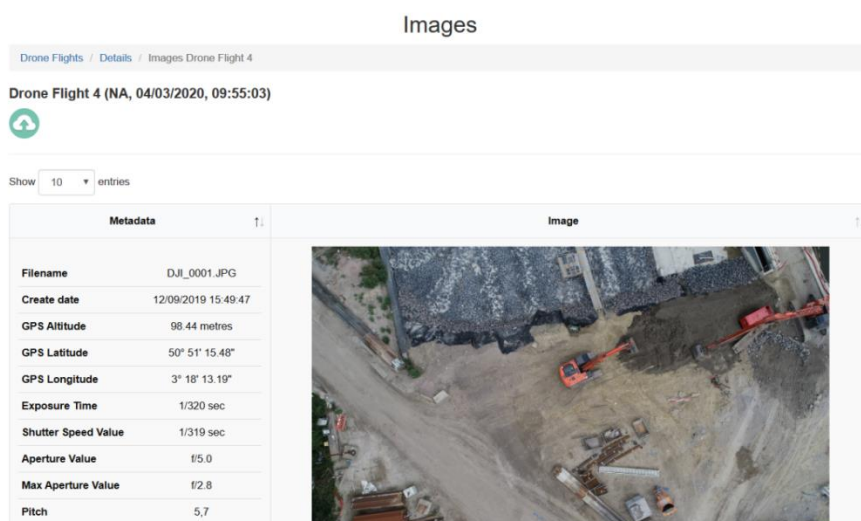
De gebruiker kan voor een vlucht afbeeldingen uploaden. Eerst worden de *images* ingelezen door de gepaste *parser*-klasse RawImageParser. Deze *parser* leest een afbeelding en zijn *meta data* in. Dit wordt gerealiseerd met de *library* 'MetadataExtractor'. Deze *meta data* bevat belangrijke info, zoals de coördinaten waar deze foto genomen werd en bijkomende informatie over de drone GPS en camera. De afbeelding wordt volledig ingelezen met een FileStream en daarna weggeschreven naar de databank onder de vorm van een byte *array*.

Een bijkomende *thumbnail* wordt met een hulpmethode aangemaakt en weggeschreven naar de databank. Deze *thumbnails* worden gebruikt in *pop-ups* in de view map van een individuele vlucht (zie **Figuur 10**). Elke entry in de RawImagestabel van de databank bestaat uit de afbeelding en de *thumbnail* (beide onder de vorm van bytes), en de *meta data*.



Figuur 10: pop-up van een afbeelding in de view map

De klant kan voor een vlucht deze afbeeldingen bekijken in de webapplicatie indien deze geüpload werden. Dit kan door voor een dronevlucht op 'Images' te klikken in het overzicht van de dronevluchten of door naar de detailpagina van een dronevlucht te surfen. De gebruiker kan nu alle geüploade images en hun bijhorende *meta data* zien (**Figuur 11**).



Figuur 11: de Images-pagina van een dronevlucht

### 2.3.1.2 Helper-klasse

Deze klasse beschikt over een *static variable* *progress* die gebruikt wordt tijdens het *parsen* van bestanden. Deze variable houdt de status van het *parsen* bij en laat toe om deze status te communiceren naar de frontend (*client*).

Verder beschikt deze klasse over drie methodes:

➤ **UpdateTotalDroneFlightTime**

Deze methode herberekend bij elke aanpassing van een drone, bv. toevoegen van een dronevlucht, aanpassen van een vlucht, verwijderen van een vlucht en toevoegen van bestanden, de gevlogen tijd van een drone. Op deze manier kan er nooit twijfel zijn over de totale gevlogen tijd, want alle mogelijke manieren waarop de gebruiker aanpassingen zou kunnen uitvoeren om deze tijd te manipuleren, worden gedekt.

Verder controleert deze methode of een drone een *check-up* nodig heeft of niet. Indien wel, dan markeert hij de drone, wat ervoor zal zorgen dat in de frontend de gevlogen tijd rood kleurt. Een administrator beschikt over de mogelijkheid om een drone manueel te markeren voor een *check-up* en kan na de *check-up* de drone markeren als zijnde goedgekeurd. De tijd kleurt dan opnieuw groen en een nieuwe *threshold* wordt berekend voor deze drone. Deze *threshold* is een ingestelde tijd in de DronesController (op dit moment 50 uren) die aangeeft wat de maximaal toegelaten vliegtijd is vooraleer een drone gecontroleerd moet worden.

Het manueel markeren voor een *check-up* of het markeren van een drone die gecontroleerd is, kan door een administrator gebeuren via de 'Edit'-pagina van een drone. Via een checkbox geeft de administrator zijn keuze aan.

➤ **CountFileLines**

Deze methode loopt snel door een bestand en telt het aantal lijnen. Op deze manier kan een accuratere progressie getoond worden aan de gebruiker tijdens het *parsen*.

➤ **SetProgress**

Deze methode wordt gebruikt in de *parser*-klassen om de *static variable* *progress* te updaten.

## 2.3.2 Controllers

### 2.3.2.1 'View'-controllers

Het project bevat verscheidene 'View'-*controllers* (de C in het MVC *pattern*) die instaan voor navigatie (routing):

- de ProjectController;
- de DroneFlightsController;
- de DronesController;
- de PilotsController;
- de FilesController;
- de MapController;
- de AccountController;

- de `ManageController`;
- de `HomeController`.

De vier eerste controllers beschikken allen over CRUD-functionaliteit (Create, Read, Update, Delete), maar de `DroneFlightsController` beschikt ook nog over de mogelijkheid om te routen naar razorpagina's die additionele informatie tonen, zoals de inhoud van het kwaliteitsrapport of een overzicht van afbeeldingen.

In de `DronesController` kan u instellen wat de threshold moet zijn vooraleer een drone gecontroleerd moet worden (*check-up*). Dit staat op dit moment op 50 uren. Via een Helper-methode uit de `Helper` klasse wordt de gevlogen tijd van een drone na elke aanpassing opnieuw berekend om consistentie te behouden. Deze methode controleert eveneens of een drone gecontroleerd moet worden of niet.

De `FilesController` zorgt ervoor dat de bestanden die de gebruiker uploadt voor een vlucht juist worden afgehandeld en maakt hiervoor gebruik van de *parser*-klassen. Deze controller beschikt bovendien voor een methode `Export` dat de informatie van een piloot of een drone (inclusief de toegekende vluchten) exporteert naar een CSV-bestand of PDF-bestand. Als laatste heeft de controller ook een functie `GetStatus` dat wordt gebruikt om met de *client-side* de progressie van het *parsen* van een bestand te communiceren.

Indien er tijdens het uploaden iets fout gaat, dan kan deze *controller* de *client* hiervan op de hoogte brengen door een *error code* te communiceren (cf. 2.2.1 Uploaden van files).

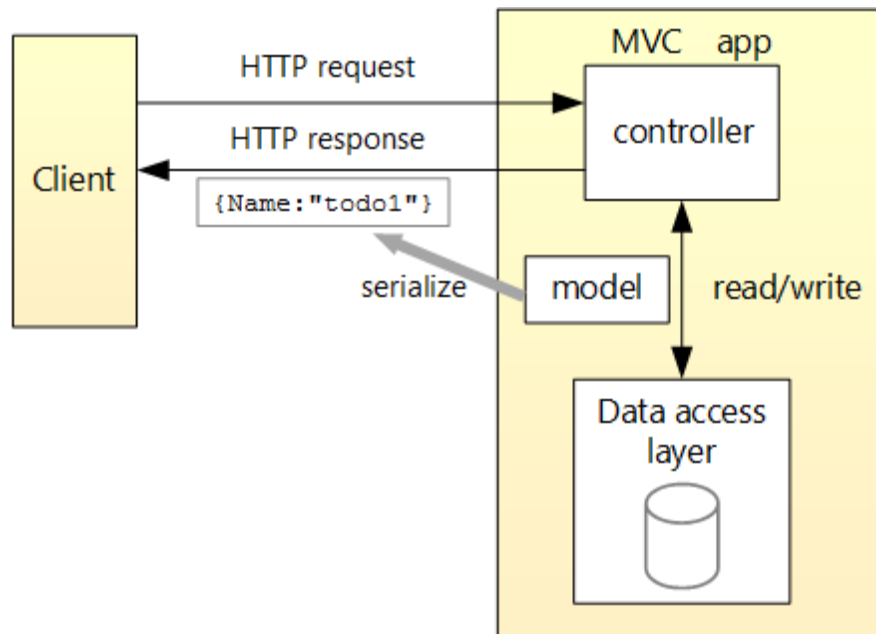
De `MapController` zorgt ervoor dat een `id` wordt meegegeven in de `ViewBag` van een de `ViewMap.cshtml`, zodat in de View de juiste javascript kan worden aangesproken.

- Er wordt een overzicht gebouwd van alle aanwezige dronevluchten met een *drone log* indien er geen `id` wordt meegegeven (`id = NULL`).
- Er wordt een track gebouwd voor een individuele dronevlucht indien er een geldige `id` wordt meegegeven (d.i. een `id` van een dronevlucht dat effectief bestaat). De controle hiervoor wordt in de *controller* uitgevoerd.

De `AccountController` zorgt ervoor dat het mogelijk is om gebruikers te registreren, hierbij worden de logingegevens vastgelegd alsook de rol van deze nieuwe gebruiker. Eenmaal de gebruiker is geregistreerd, zorgt deze controller er ook voor dat de gebruiker zich kan inloggen. De `ManageController` maakt het mogelijk voor bestaande gebruikers om hun account te beheren, zo kunnen ze bijvoorbeeld hun wachtwoord veranderen.

### 2.3.2.2 'Web API'-controllers

Naast 'View'-*controllers* heeft het project verschillende 'Web API'-*controllers*. Deze klassen staan in voor de communicatie tussen de *client* (in dit project de javascriptcode) en de server (**Figuur 12**). Deze zijn:



Figuur 12 werking Web API controllers

- de CTRLPointsController;
- de DroneFlightsApiController;
- de DroneGPsController;
- de GCPCController;
- de PointCloudXYZsController;
- de RawImagesController;
- de ThumbnailsController;

De Web API controllers vragen data op uit de databank. Op basis van de bestaande Model klassen wordt een *HTTP response* gemaakt met de *projected data*.

Als eerste moeten de *controllers* geconfigureerd worden op hun *routing*. De *ASP.NET* Routing-module is verantwoordelijk voor het toewijzen van inkomende browseraanvragen aan bepaalde MVC-controlleracties. In het project wordt er gewerkt met de HTTP-route 'WebAPI/api/{controller}/{id}/{imageid}'. Data wordt opgevraagd op basis van een *flight id* en eventueel een *image id* voor *images*.

De eerste vijf *controllers* zijn verantwoordelijk voor datapunten van de drone. Elk soort data heeft een eigen *controller*. De *controllers* handelen elk een *GET request* af op basis van een *flight id*. Een voorbeeld hiervan is te zien op **Figuur 13**.

Deze klassen werken op een gelijkaardige manier. Als eerste vragen ze de nodige data op uit de databank met behulp van *Entity Framework*. Ten tweede wordt een *anonymous type* object aangemaakt op basis van deze data met behulp van een *lambda expression*. Als laatste wordt het *anonymous type* geconfigureerd tot een *HTTP Response Message* die een JSON-object bevat. Deze JSON wordt teruggegeven door de methode.

```
// GET: WebAPI/api/GCP/5
0 references
public HttpResponseMessage GetGroundControlPointsByFlightID(int id)
{
    var Flight = db.DroneFlights.Find(id); //bijhorende vlucht vinden
    if (Flight == null || !Flight.hasGCPs)
    {
        return new HttpResponseMessage(HttpStatusCode.NotFound);
    }

    //data projection (anonymous type)
    var GroundControlPoints = Flight.GroundControlPoints.Select(
        gcp => new { gcp.GCPId, gcp.GCPName, gcp.X, gcp.Y, gcp.Z, gcp.FlightId }).ToList();

    //config to set to json
    var response = new HttpResponseMessage(HttpStatusCode.OK);
    response.Content = new StringContent(JsonConvert.SerializeObject(GroundControlPoints));
    response.Content.Headers.ContentType = new MediaTypeHeaderValue("application/json");

    return response;
}
```

Figuur 13: GET methode van de GCPController

De laatste twee controllers, RawImagesController en ThumbnailsController, staan in voor *images* en hun *thumbnails*. Als eerste halen ze de juiste image op uit de databank op basis van een *flight id* en een *image id*. Dit gebeurt aan de hand van een *SQL query*. Deze *query* gebruikt Parameter objecten om zich af te schermen tegen *SQL injections*.

Het is *bad practice* is om afbeeldingen als bytes door te sturen naar de frontend. Daarom handelen deze controllers de conversie af van een *byte array* naar een werkelijke afbeelding en sturen ze deze terug.

De RawImagesController heeft daarnaast ook een methode om alle *image* data op te vragen op basis van een *flight id*, gelijkaardig aan de andere controllers.



### 2.3.3 Views



Razorpagina's zijn bestanden van het type cshtml en beschrijven de *View* in het MVC pattern. Voor elke methode in een *controller*, die een *ActionResult* teruggeeft als return-type, bestaat een razorpagina met overeenkomstige naam. In onderstaande afbeelding, **Figuur 14**, is de CSHTML-pagina te zien waar de gedetailleerde informatie van een dronevlucht op weergegeven wordt. Een groene of rode knop wijst respectievelijk op het aanwezig of afwezig zijn van het overeenkomstige document.

[Drone Planner](#) [Map](#) [Projects](#) [Drone Flights](#) [Drones](#) [Pilots](#) [Contact](#) [About](#) Hello admin! [Log off](#) [Register](#)

## Drone Flight Details

[Drone Flights](#) / [Details Drone Flight 1](#)

### Drone Flight 1



Information	
Project Code	PRJ-001
Location	Avelgem
Date	06/02/2020, 12:52:25
Drone	<a href="#">Happy</a>
Pilot	<a href="#">Bryan Van Huyneghem</a>
Starting time (UTC)	12:52:25
Ending time (UTC)	12:57:29
Type of Activity	
Other	
Simulator	
Instructor	
Remarks	

Documents	
Quality Report	<a href="#">QR</a>
Ground Control Points	<a href="#">GCP</a>
Control Points	<a href="#">CTRL</a>
TFW	<a href="#">TFW</a>
Images	<a href="#">Images</a>
XYZ	<a href="#">XYZ</a>
Drone Log	<a href="#">Drone Log</a>

[Edit Flight](#) | [Back to Drone Flights List](#) | [Delete](#)

© 2020 - Jan De Nul, Drone Flights Application

Figuur 14: detailpagina van een Drone Flight

## 2.4 Javascriptklassen

In dit onderdeel wordt de functionaliteit van de webapplicatie beschreven met betrekking tot de javascriptklassen in dit project. Er wordt JavaScript gebruikt om de gebruiker welbepaalde bestanden te kunnen laten uploaden (2.4.1), om de vluchten te visualiseren (2.4.2) en functionaliteit te voorzien om zogenaamde *Data Tables* te kunnen aanmaken (2.4.3).

### 2.4.1 Uploaden van files

De gebruiker kan aan een dronevlucht documenten toevoegen. Voorbeelden hiervan zijn: het logboek van een drone, een TFW-bestand, het kwaliteitsrapport, het XYZ-bestand, foto's van de vlucht en CSV-bestanden. De eis van doelstelling 2 (cf. Inleiding) stelt dat de ingelezen bestanden moeten worden opgeslagen in de databank na het verwerken (*parsen*) ervan. Deze verwerking gebeurt, zoals eerder vermeld in 2.4.1, met *parser*-klassen die elk een bepaald type bestand afhandelen.

De controller `FilesController` staat in voor het juist afhandelen van de aanvraag van de gebruiker om bestanden te uploaden. De gebruiker navigeert naar de uploadsectie van de webapplicatie via het pad `/Files/Index/{DroneFlightId}`, waarbij `DroneFlightId` aangeeft voor welke dronevlucht de gebruiker bestanden wil uploaden. Er is gepaste foutenafhandeling bij het opgeven van een onbestaande `DroneFlightId` of het niet opgeven van een `DroneFlightId`. De gebruiker wordt op een gepaste wijze geïnformeerd wat er precies fout gaat.

Aan de *client-side* verzorgt een stuk javascriptcode, het bestand **fileUpload.js**, de communicatie met de *server-side*. Dit bestand vervult meerdere functies:

- Het controleert of de gebruiker een bestand meegeeft bij het uploaden. Het zorgt er met andere woorden voor dat de gebruiker niet zomaar op de uploadknop kan klikken, zonder één of meerdere bestanden toe te voegen. Dit wordt bereikt door een abort uit te voeren op wat men probeert te versturen (**Figuur 15**). Aan server-side wordt er nogmaals gecontroleerd of er weldegelijk een bestand werd meegegeven; dit als dubbele controle.

```
// AjaxForm
$('#myform').ajaxForm({
  beforeSend: function (xhr) {
    // If the user did not submit any files, abort the ajaxForm
    if ($('#file').get(0).files.length === 0) {
      xhr.abort();
    }
  }
});
```

Figuur 15: code snippet: abort-functionaliteit

- Het controleert de toegestane bestandsextensies, zoals weergegeven in **Figuur 16**. Indien een gebruiker een bestand probeert te uploaden dat niet is toegestaan, dan geeft de webapplicatie de gebruiker hier een melding van. Er wordt een lijst getoond met extensies die wel zijn toegelaten. De uploadknop wordt vergrendeld tot wanneer de gebruiker geldige bestanden opgeeft. De gebruiker kan nu opnieuw proberen uploaden. Aan server-side wordt er nogmaals gecontroleerd of de bestanden die werden meegegeven een geldige extensie hebben; dit als dubbele controle.

```

$("#file").change(function () {
    var fileExtension = ['pdf', 'dat', 'txt', 'csv', 'xyz', 'tfw', 'jpg']; // allowed extensions
    if ($.inArray($(this).val().split('.').pop().toLowerCase(), fileExtension) == -1) {
        alert("Only formats are allowed : " + fileExtension.join(', '));
        $("#uploadbtnSubmit").prop('disabled', true); // disable upload button
    }
    else {
        $("#uploadbtnSubmit").prop('disabled', false); // enable upload button
    }
});

```

Figuur 16: code snippet: controle op bestandsextensie

- Het uploaden van de bestanden via jQuery's AjaxForm. Dit bevat 3 *callback*-functies: `beforeSend`, `uploadProgress` en `success`.
  - `beforeSend` voert de controle uit op het aantal bestanden dat werd toegevoegd. Indien dit 0 is, dan wordt er een abort uitgevoerd. In de andere gevallen werden er één of meerdere bestanden meegegeven. Resultaatvelden worden op *hidden* gezet, zoals in **Figuur 17**, aangezien de gebruiker reeds in een tweede rond van uploaden kan zitten, zonder de pagina te *refreshen*.

```

// Helper-function: Reset all output fields to hidden
function initHide() {
    $("#endField").hide(); // hide endField ending message
    $("#successField").hide(); // hide the successField ending
    $("#failedField").hide(); // hide failedField ending message
    $("#progressField").hide(); // hide progress of reading files
    $("#failedFilesList").empty() // empty the unordered list of failed files
    $("#moreUpload").hide(); // hide the moreUpload text
    $("#oneFile").hide(); // hide single failed file text
    $("#multipleFiles").hide(); // hide multiple failed files text
}

```

Figuur 17: code snippet: verbergen van div-elementen

In **Figuur 18** worden de velden met resultaten verborgen en worden variabelen, zoals `firstCheck` en `filesLeftToParse`, gereset naar hun initiële default-waarde of geüpdatet naar een nieuwe waarde, zoals in het geval van de variabele `totalFilesToParse`.

```
// AjaxForm
$('#Myform').ajaxForm({
  beforeSend: function (xhr) {
    // If the user did not submit any files, abort the ajaxForm
    if ($('#file').get(0).files.length === 0) {
      xhr.abort();
    }
    else {
      //init
      $('#progressbar').progressbar({ value: 0 }); // reset to 0%
      $('#progressbar').progressbar("enable");
      initHide();
      totalFilesToParse = $('#file').get(0).files.length; // amount of uploaded files
      filesLeftToParse = 0; // reset
      firstCheck = true;
    }
  },
},
```

Figuur 18: code snippet: `beforeSend` callback bij uploaden van één of meerdere bestanden

- `uploadProgress` verzorgt het eigenlijke uploaden van de bestanden (zie **Figuur 19**). De progress bar wordt periodiek geüpdatet door de `AjaxForm` en van zodra het uploaden voltooid is (`percentComplete == 100`), wordt de gebruiker hiervan op de hoogte gebracht. Vervolgens wordt alles in gereedheid gebracht om de gebruiker de voortgang van het parsen te kunnen tonen en wordt de `<div>` met id `progressField` getoond. Het eigenlijke parsen wordt gestart met de functie `startParsing`.

```
uploadProgress: function (event, position, total, percentComplete) {
  $('#uploadstatus').text("Uploading files... (" + percentComplete + "%)");
  $('#progressbar').progressbar("value", percentComplete);
  // Once uploading is complete...
  if (percentComplete == 100) {
    $('#uploadstatus').text("Upload complete.");
    $('#initialMessage').hide();
    console.log("Attempting to parse: " + totalFilesToParse + " files.");
    currentFile = "";
    $(".amountParsed").text(0);
    $(".totalParsed").text(totalFilesToParse);
    $('#progressField').show();
    // begin parsing
    console.log("startParsing");
    startParsing();
  }
}
```

Figuur 19: code snippet: `uploadProgress` callback bij het uploaden van één of meerdere bestanden

De functie `startParsing` (zie **Figuren 20, 21 en 22** op deze pagina, en pagina 38) reset de waarde van de *progress bar* terug naar 0 en roept met de functie `setTimeout` de functie `parse` op. Deze laatstgenoemde functie wordt blijft opnieuw en opnieuw afgevuurd worden zolang het aantal verwerkte bestanden niet gelijk is aan het aantal verzonden bestanden. Van zodra dit wel het geval is, zal het laatst ontvangen HTTP-bericht ook de lijst met eventueel gefaalde bestanden bevatten. Alles wordt in gereedheid gebracht om de gebruiker de juiste velden te tonen (via de functies `show` of `hide` op de nodige `<div>`-elementen). Er wordt gecontroleerd of er effectief bestanden zijn die niet werden ingelezen; in elk van de twee gevallen (ja of neen) wordt de gepaste informatie (feedback) aan de gebruiker getoond.

```
// Changes the progress bar value at an interval
function startParsing() {
    $("#progressbar").progressbar({ value: 0 });
    intervalID = setTimeout(parse, 500); //every 0.5 sec the progress bar updates
}
```

*Figuur 20: code snippet van de functie `startParsing`*

```
// Helper-function for startParsing: called in startParsing to change the progress bar value ...
// ... through an ajax call
function parse() {
    $.get("/Files/GetStatus/", function (result) {
        currentFile = result.currFileName // set the current file
        console.log("Get:" + result.currFilesLeft);
        filesLeftToParse = result.currFilesLeft;
        if (firstCheck) {
            if (filesLeftToParse > 0) {
                firstCheck = false;
            }
            intervalID = setTimeout(parse, 500);
            return;
        }

        amountParsed = totalFilesToParse - filesLeftToParse;
        // set View
        $(".amountParsed").text(amountParsed);
        // Update the amount of files that still have to be parsed
        console.log("Files left to parse: " + filesLeftToParse);
        //update the progress bar
        let progress = Math.round((amountParsed / totalFilesToParse
                                    + (result.currProgress / 100)
                                    / totalFilesToParse)
                                    * 100);
        $("#uploadstatus").text("Parsing file: " + currentFile + " (" + progress + "%)");
        $("#progressbar").progressbar("value", progress);
    });
}
```

*Figuur 21: code snippet: helper-functie updatet de progress bar via aan HTTP get call*

```

if (amountParsed == totalFilesToParse) { // ending procedure
  console.log("Parsed: " + amountParsed + " (had to parse: " + totalFilesToParse + ")");
  clearTimeout(intervalID); // clear
  $(".amountParsed").text(amountParsed);
  $(".totalParsed").text(totalFilesToParse);
  $(".uploadstatus").text("Parsing complete.");
  $(".progressField").hide();
  console.log("Done.")

  // Check whether any files failed to parse because of duplicates
  if (result.failedFiles.length == 0) { // no failed files
    $("#successField").show();
    $("#endField").addClass("alert-success");
    $("#endField").removeClass("alert-warning");
    $("#endField").show();
  }
  else { // failed files
    console.log("array length else: " + result.failedFiles.length);
    $("#failedField").show();
    $(".totalFailed").text(result.failedFiles.length);
    if (result.failedFiles.length > 1) { // more than 1 failed file
      $("#multipleFiles").show()
    }
    else { // 1 failed file
      $("#oneFile").show()
    }
    $("#endField").removeClass("alert-success");
    $("#endField").addClass("alert-warning");
    document.getElementById("failedFilesList").appendChild(makeUL(result.failedFiles));
    $("#endField").show();
  }
  $(".moreUpload").show();
}
else { // fire another parse
  intervalID = setTimeout(parse, 500);
}

```

*Figuur 22: code snippet: instellen en tonen van alle resultaatvelden na afronden parsen*

- success wordt uitgevoerd nadat het uploaden werd voltooid. In deze *callback*-functie wordt de gepaste *error*-boodschap getoond, indien dit nodig is. Dit is weergegeven in **Figuur 23** op pagina 39.

```

success: function (errorCode) {
    // to do: print message on screen for user
    // except when it is == 1
    if (errorCode != 1) {
        $("#progressField").hide();
        $("#endField").hide();
        let text;
        if (errorCode == 0) {
            text = "No files were submitted.";
        }
        else if (errorCode == 2) {
            text = "Please specify a Drone Flight.";
        }
        else if (errorCode == 3) {
            text = "This Drone Flight does not exist.";
        }
        else if (errorCode == 4) {
            text = "Someone else is already uploading. Please try again in a few minutes.";
        }
        else if (errorCode == 4) {
            text = "An invalid file type was submitted.";
        }
        $("#errorField").show();
        $("#errorMessage").text(text);
    }
}

```

Figuur 23: code snippet: de verschillende error codes bij het uploaden van bestanden

- Het tonen van een *progress bar* voor het uploaden en het parsen van de bestanden. Er wordt periodiek gecommuniceerd met de server-side of een status op te halen van het parsen. Op deze manier kan aan de gebruiker een voortgang worden getoond van het parsen van zijn bestanden.
- Indien er iets misloopt aan server-side, dan wordt de gebruiker hier ook van op de hoogte gebracht. Hiervoor zijn zelfgemaakte errorcodes in het leven geroepen:
  - 1: succes;
  - 0: no files submitted;
  - 2: no drone flight specified;
  - 3: drone flight does not exist;
  - 4: someone else is already uploading;
  - 5: invalid file type.
- Eén limitatie van de *file uploader* is dat deze slechts één gebruiker per keer toelaat om te uploaden. Dit zou verholpen kunnen worden door een unieke id toe te kennen aan een gebruiker. Heden wordt een boodschap weergegeven indien een andere gebruiker reeds aan het uploaden is.
- De gebruiker kan gedurende het parsen van de bestanden zien welk bestand de FilesController aan het verwerken is. Hiervoor wordt periodiek een *call* gemaakt naar de server-side. Deze antwoordt met een *HTTP-response* waarvan de inhoudt gemaakt wordt met de methode GetStatus van de FilesController.

Deze methode maakt een anoniem object dat naar JSON geconverteerd wordt. De inhoud bevat steeds:

- De huidige *progress* van een bestand;
- Het parse-resultaat van een parser;
- De bestandsnaam van het bestand dat op dit moment geparset wordt;
- Het aantal bestanden dat nog geparset moet worden;
- (Een lijst met de namen van de bestanden die niet geparset werden).

De allerlaatste call naar `GetStatus` gebeurt nadat alle bestanden geparset zijn. In dit geval staat de waarde van het aantal bestanden dat nog geparset moet worden op 0 en wordt een lijst gebouwd met de namen van alle bestanden die niet geparset werden (het laatste puntje van bovenstaande lijst).

- Aan client-side wordt dan een *unordered list* gebouwd met bovenvermelde bestandsnamen, zodat de gebruiker kan zien welke bestanden niet geparset werden (doorgaans omwille van duplicaten, d.w.z. dat deze bestanden reeds aanwezig waren voor deze vlucht).
- De gebruiker kan kiezen om nogmaals bestanden te uploaden of om terug te keren naar de lijst van dronevluchten of de detailpagina van de dronevlucht waarvoor hij zojuist bestanden heeft geupload/proberen uploaden.

Het uploaden van bestanden via de webapplicatie is onderhevig aan enkele beperkingen:

- Zoals eerder vermeld: slechts één gebruiker per keer kan uploaden.
- Er kan maximaal 2.147 GB aan bestanden geupload worden. Deze beperking wordt opgelegd door ASP.NET zelf in het bestand `Web.config`. De gebruiker kan daarom op dit moment enkel bestanden kleiner dan (of in totaal) 2.147 GB uploaden.
- Bij het uploaden van zeer veel afbeeldingen kan de server mogelijks een 'not enough memory' fout opgooien. Dit is sterk afhankelijk van wat er op de server qua geheugen beschikbaar is voor de webapplicatie. Indien hij of zij afbeeldingen wil uploaden, dan is deze limiet voorlopig ingesteld op 500 MB. Dit kan mogelijks verhoogd worden, indien meer geheugen kan worden toegekend aan de applicatie. Het geheugen wordt terug vrijgegeven na het *parsen* van de afbeeldingen. Een test gaf aan dat het uploaden van 70 afbeeldingen de applicatie tot zijn ingestelde limiet van 3 GB liet gaan.
- Aan de hand van een lus wordt de totale som gemaakt van de bestandsgroottes, zoals weergegeven in **Figuur 24** op pagina 41. Er wordt vervolgens gecontroleerd op de totale bestandsgrootte (**Figuur 25** op pagina 41) van alle bestanden en de totale bestandsgrootte van de afbeeldingen apart. Indien deze overschreden worden, dan wordt een gepaste boodschap getoond aan de gebruiker.



```
var files = $("#file").get(0).files;
var totalSize = 0;
var imageSize = 0;
for (i = 0; i < files.length; i++) { // limit images upload to 500 MB
    if (files[i].type.toLowerCase() == 'image/jpeg') {
        imageSize += files[i].size;
    }
    totalSize += files[i].size;
}
```

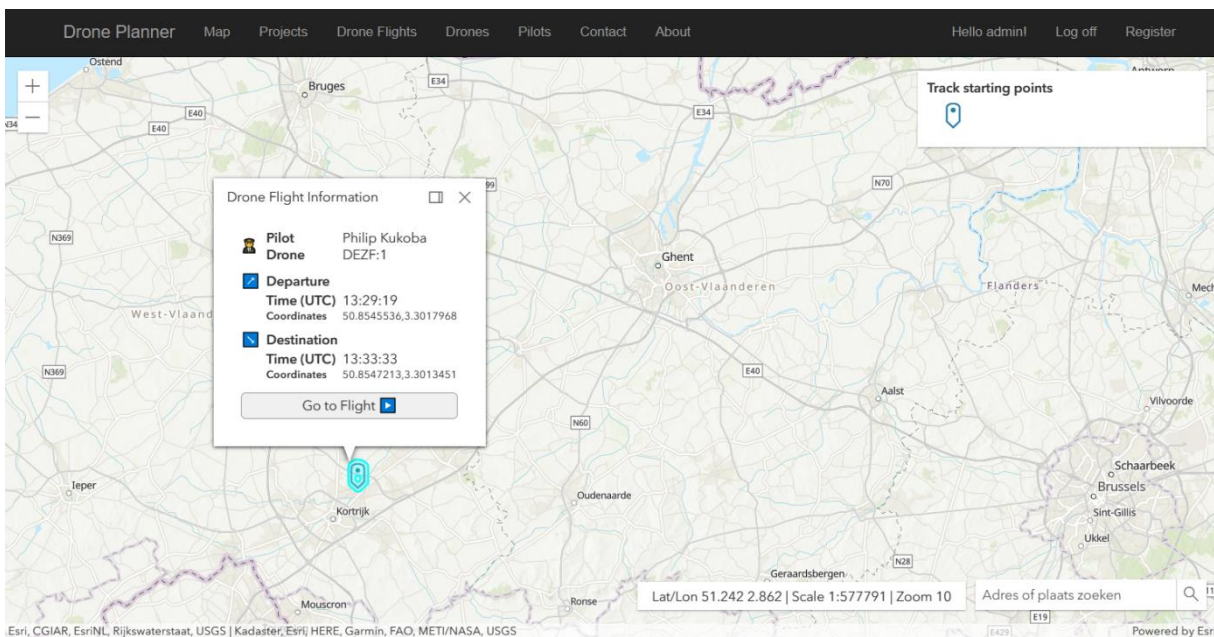
*Figuur 24: berekenen van totale bestandsgroottes*

```
else if (totalSize > 2147483647) {
    alert("Maximum total upload size is: 2.1 GB.");
    $("#uploadbtnSubmit").prop('disabled', true); // disable upload button
    $("#maximum").show();
    $("#maxUploadTextFiles").show();
}
else if (imageSize > 524288000) { // 500 MB
    alert("Maximum total upload size for images is: 500 MB.");
    $("#uploadbtnSubmit").prop('disabled', true); // disable upload button
    $("#maximum").show();
    $("#maxUploadTextImages").show();
}
```

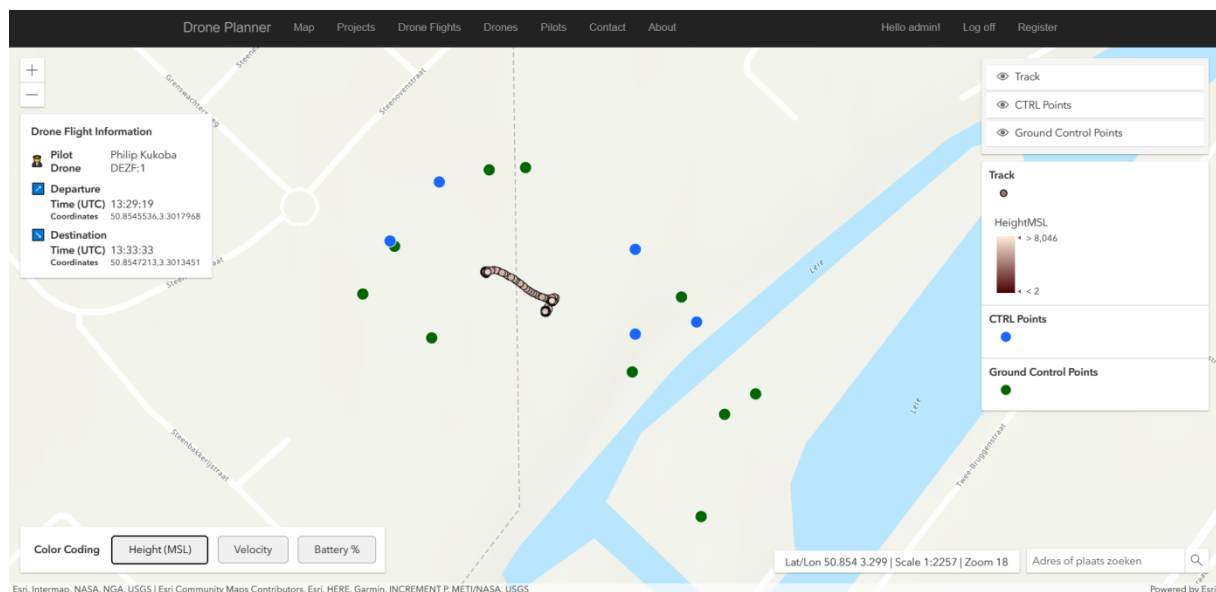
*Figuur 25: controle bestandsgroottes*

## 2.4.2 Visualisatie met ArcGIS JS API

De gebruiker kan via de navigatiebalk navigeren naar 'Map'. Deze *map view* (**Figuur 26**) biedt een overzicht van alle dronevluchten op een kaart. Voor elke vlucht komt er een icoon op de startplaats van deze vlucht. Er kan op het icoon klikken om een *pop-up* te zien met informatie over deze vlucht. Deze informatie bevat de naam van de piloot, de naam van de drone en informatie over start- en eindtijd van de vlucht. Deze *pop-up* bevat ook een knop om deze specifieke vlucht te bekijken in een eigen *map view* (**Figuur 27**).



Figuur 26: pop-up voor een vlucht in een map overview



Figuur 27: de map overview met verschillende datapunten van een dronevlucht

De frontend bestaat uit verschillende klassen uit de ArcGIS Javascript API.

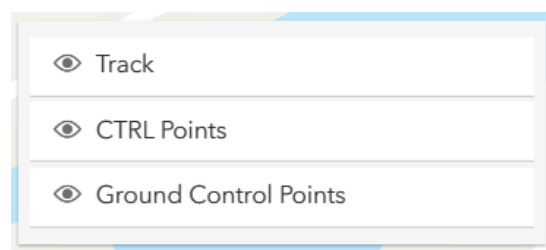
**Map:** De Map klasse is een container voor de *layers* van de applicatie en heeft bijhorende methodes voor deze *layers*.

**MapView:** De MapView toont een 2D view van de Map. De layers worden gerenderd door deze klasse. Om de MapView zichtbaar te maken heeft deze een referentie nodig naar de Map en naar het *Document Object Model* (DOM) element waar deze in hoort.

**Graphic:** Een Graphic stelt een visuele figuur voor op de *map*. Deze heeft typisch de velden *geometry* (*point*, *polyline* of *polygon*), een referentie naar de bijhorende *layer*, een eventuele *PopupTemplate* en een aantal zelfgemaakte attributen (vooral van belang bij de *track* van de vlucht).

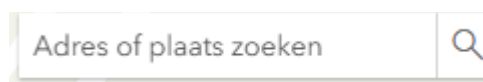
**SpatialReference:** Deze klasse definieert de ruimtelijke referentie van de te visualiseren data. Het geeft aan welk geografisch projectiesysteem wordt gebruikt om geografische punten op de kaart te situeren. Deze ruimtelijke referentie kan het gemakkelijkst gespecificeerd worden met behulp van het *well-known ID* (*WKID*) veld van de SpatialReference klasse. Voor dit project wordt het coördinatensysteem "Belgian Lambert 1972" gebruikt met WKID 31370.

**LayerList:** Een *widget* die een legende toont van alle layers van de map met opties om elke layer te verbergen of te tonen. De LayerList (**Figuur 28**) heeft een referentie nodig naar de MapView.



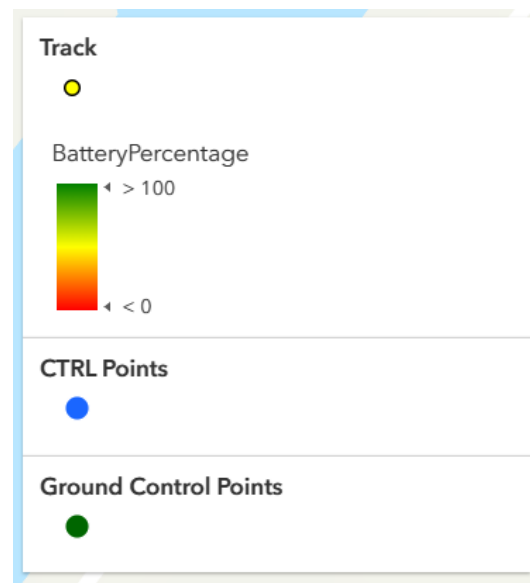
Figuur 28: LayerList widget

**Search:** Een zoekbalk met suggesties (zoals bij Google Maps); **Figuur 29**.



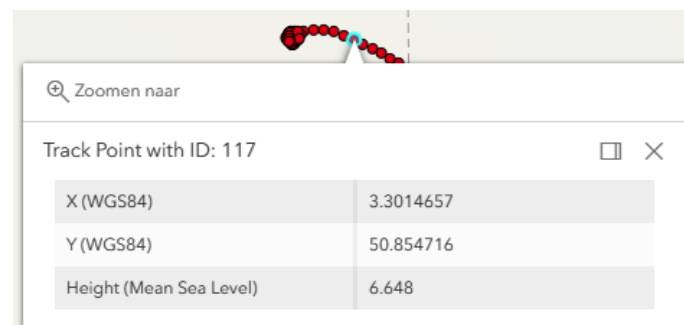
Figuur 29: search widget

**Legend:** Toont een legende (**Figuur 30**) van alle *layers* van de MapView met een voorbeeld van de visuele figuren in elke *layer*. De Legend updatet automatisch als een *layer* verborgen of getoond wordt, of als de highlighte attribuut van de track verandert.



Figuur 30: legend klasse

**PopupTemplate:** Deze klasse definieert de inhoud van de *pop-up* van een specifieke Graphic. Deze popup (**Figuur 31**) wordt gevuld met de coördinaten en alle andere attributen van de Graphic.



Figuur 31: PopupTemplate

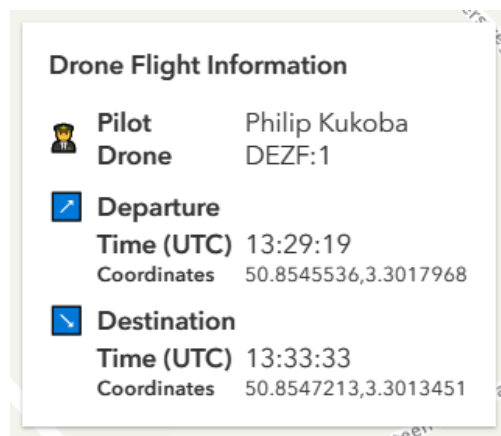
**FeatureLayer:** Een FeatureLayer stelt één enkele *layer* voor. Deze heeft verschillende velden die ingesteld moeten worden. Het *source* veld krijgt een *array* van Graphic objecten mee die bij deze layer horen. Het *fields* veld bevat een array van alle mogelijk attributen bij de Graphic objecten, nodig voor de PopUpTemplate. Waarop volgt: het *popUpTemplate* veld moet een referentie naar het gepaste PopUpTemplate object bevatten.

Het *renderer* veld bepaalt hoe de Graphic objecten gevisualiseerd moeten worden (zoals kleur, grootte en *opacity*). Bij de *layer* van de *track* krijgt de FeatureLayer een *custom*

*renderer* mee die een attribuut van de *track* punten, bv. hoogte of batterijstand, visualiseert met een *visual variable* object. Deze *visual variable* heeft een gepaste *color ramp* voor een specifiek attribuut. Bijvoorbeeld: als de *track* gevisualiseerd wordt op basis van de batterijstand van de drone kleurt een punt rood als de batterijstand dicht bij 0% is, kleurt het geel bij 50% en kleurt het groen bij 100%. Dit kleurenverloop is een continu spectrum (*color ramp*).

Als laatste moet het *objectIdField* ingesteld worden op het unieke ID attribuut van de *Graphic* objecten.

**Feature:** Deze *widget* (**Figuur 32**) toont data volgens zijn *popUpTemplate* veld. Deze klasse wordt gebruikt wanneer informatie constant moet getoond worden, in tegenstelling tot pop-ups. De *Feature* bevat een tabel met informatie over de dronevlucht.



Figuur 32: Feature klasse

Daarnaast is er een *color coding* menu (**Figuur 33**) aanwezig links onderaan om de *track* te visualiseren op basis van hoogte, snelheid of batterijstand. Deze menu is gebaseerd op de *feature* klasse. De acties worden geïmplementeerd met jQuery.



Figuur 33: color coding menu

### 2.4.3 Data Tables

Het script *datatabel\_script.js* voorziet de nodige functionaliteit om gemarkeerde HTML-tabellen om te zetten naar zogenaamde *data tables*. Deze tabellen laten *paging*, *searching* en *sorting* toe op HTML-tabellen.

In totaal worden 10 tabellen omgezet naar dit genre tabellen. Het gaat om:

- de index van de *drone flights*;
- de index van de *projects*;
- de index van de *drones*;
- de index van de *pilots*;
- de index van een *pilot* zijn of haar *drone flights*;
- de index van een *drone* zijn *drone flights*;
- de index van een *project* zijn *drone flights*;
- de tabel voor de *CTRLs*;
- de tabel voor de *GCPs*;
- de tabel voor de *images*.

## 2.5 Sequence diagrams

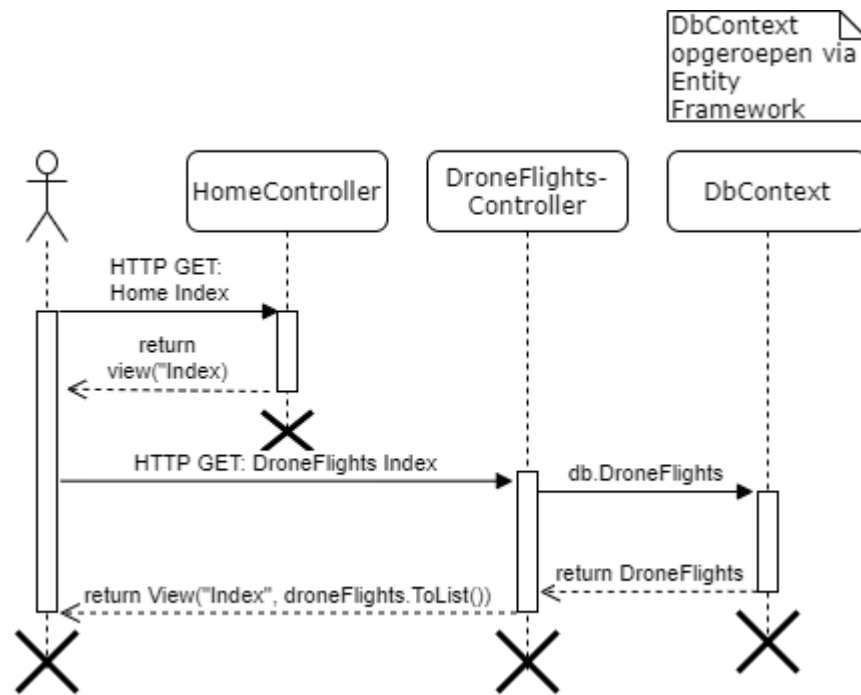
Dit hoofdstuk behandelt drie sequentiediagrammen die twee belangrijke scenario's van de droneplanning-tool beschrijven, namelijk eerst het bekijken van dronevluchten als anonieme gebruiker, en vervolgens het inloggen als *admin* en het uploaden van bestanden.

Sequentiediagrammen beschrijven de *flow* doorheen de applicatie, meer bepaald wanneer methodes opgeroepen en objecten aangemaakt worden.

### 2.5.3 Bekijken van dronevluchten als anonieme gebruiker

Het eerste en tevens eenvoudigste sequentiediagram behandelt het bekijken van dronevluchten als anonieme gebruiker en is te zien in **Figuur 34**, op pagina 47. De gebruiker begint met de applicatie op te starten, een *HTTP GET request* naar de HomeController zorgt er vervolgens voor dat de gebruiker de juiste *view* te zien krijgt, namelijk de *Index view* komende van de HomeController.

In een volgende stap klikt de gebruiker de knop 'Drone Flights' aan, welke weer een *HTTP GET request* start, deze keer echter naar de DroneFlightsController. Deze *controller* communiceert eerst met de DbContext, welke opgeroepen wordt via het *Entity Framework*, om een lijst te verkrijgen met alle dronevluchten. De laatste stap van dit sequentiediagram is het weergeven van de dronevluchten in een tabel. De DroneFlightsController zorgt hier voor door het teruggeven van een *Index view* met een lijst van alle dronevluchten.



Figuur 34: sequentiediagram over het bekijken van dronevluchten als anonieme gebruiker

#### 2.5.4 Inloggen en bestanden uploaden

Het inloggen in de webapplicatie en het uploaden van bestanden komt aan bod in de sequentiediagrammen in **Figuren 35** en **36**, op pagina 48 en 50. In deze figuren staan twee aparte diagrammen die echter als een geheel gezien kunnen worden; **Figuur 36** is namelijk een rechtstreeks vervolg op **Figuur 35**.

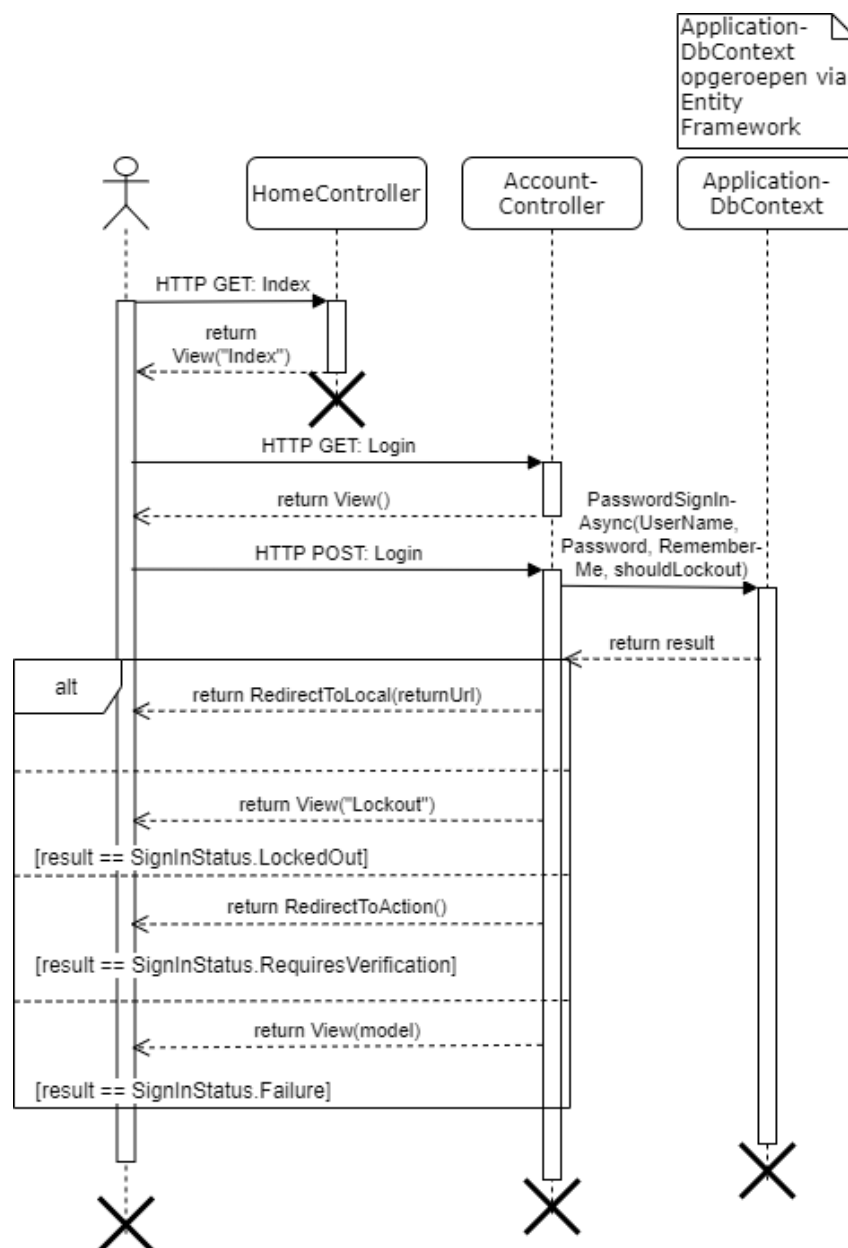
Op het eerste diagram is te zien dat de eerste stappen analoog zijn aan de eerste stappen uit het vorig diagram, namelijk het opstarten van de applicatie en het verkrijgen van de *Index view* van de HomeController. Na het opstarten van de applicatie klikt de gebruiker op de knop 'Login', welke een *HTTP GET request* uitvoert naar de AccountsController en het scherm (de *View*) teruggeeft waarop de gebruiker kan inloggen door middel van het ingeven van een *username* en wachtwoord. Na het ingeven van de inloggegevens volgt er een Login *HTTP POST*, welke met behulp van de methode *PasswordSignInAsync* zal controleren of de gebruiker correcte inloggegevens ingegeven heeft.

Het antwoord van de methode *PasswordSignInAsync* heeft verschillende mogelijkheden, afhankelijk van de waarden van de inloggegevens. De eerste mogelijkheid is wanneer de gebruiker correcte inloggegevens ingeeft. De gebruiker wordt aan de hand van de *requestUrl* teruggestuurd naar de pagina die voor het overgaan naar de loginpagina actief was; in dit geval dus de *Index*-pagina verkregen van de HomeController.

Wanneer de gebruiker te veel mislukte pogingen voor een juiste *username* en wachtwoord ondernomen heeft, komt hij in de *Lockout*-situatie terecht, waar de mogelijkheid om in te loggen op de webpagina er niet meer is.

In het derde scenario moet de gebruiker nog geverifieerd worden, hier zal de gebruiker teruggestuurd worden naar de pagina van de *requestUrl*, analoog aan de eerste situatie - alleen is de gebruiker nu nog niet ingelogd.

In het vierde en laatste scenario geeft de gebruiker een verkeerde combinatie van *UserName* en wachtwoord in, waardoor hij een *Error* te zien krijgt met als boodschap "*Invalid login attempt.*".



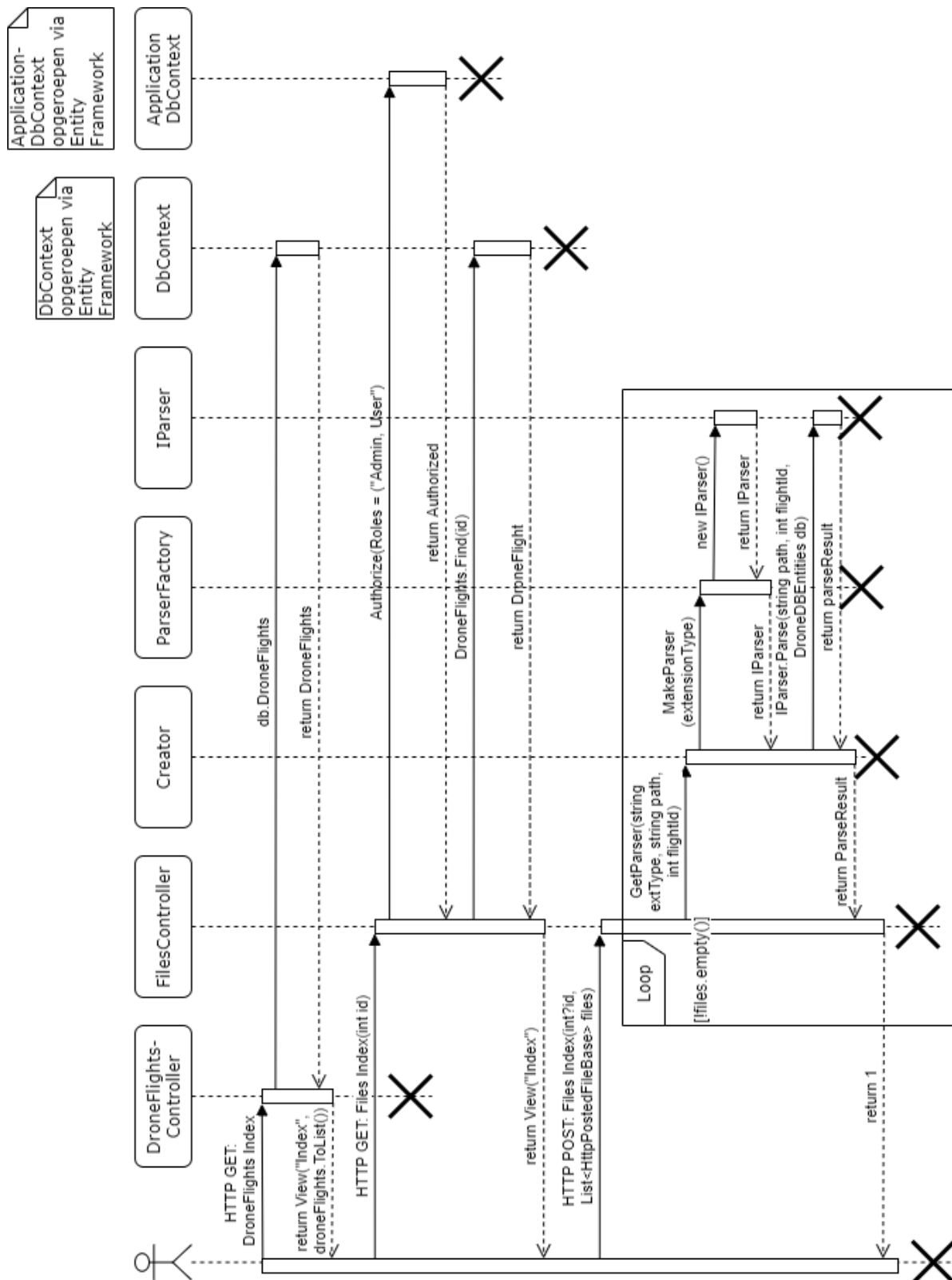
Figuur 35: sequentiediagram over het inloggen in de webapplicatie



Na het inloggen in de applicatie wil de gebruiker nu enkele bestanden uploaden, wat beschreven staat in het sequentiediagram in **Figuur 36** op pagina 50. Deze figuur neemt de eindsituatie van het vorige sequentiediagram als beginsituatie over.

De gebruiker gaat eerst naar de webpagina met betrekking tot de dronevluchten, welke, net zoals bij het eerste sequentiediagram, eerst een *HTTP GET request* start en, met een lijst verkregen uit de databank, een tabel weergeeft met alle dronevluchten. Bij elke dronevlucht in de tabel staat een knop om bestanden te kunnen uploaden, welke de gebruiker aanklikt. Een *HTTP GET request* naar de *FilesController* start een eerste contact met de *ApplicationDbContext*, net zoals de gewone *DbContext* opgeroepen via het *Entity Framework*, voor de authenticatie van de gebruiker, die in dit geval ingelogd is en dus files mag uploaden. Na de authenticatie haalt de methode de juiste dronevlucht uit de databank aan de hand van de *DbContext*, om zo een *view* terug te geven waarop bestanden kunnen geupload worden naar de juiste dronevlucht.

Vervolgens selecteert de gebruiker de bestanden die hij wil uploaden en drukt hij op de knop 'Uploaden', wat een *HTTP POST* start met als parameters de *flight id* van de vlucht en een lijst van alle te uploaden bestanden. De *HTTP POST* start een lus die voor elk van de bestanden, aan de hand van de *getParser*-methode van de *Creator*-klasse van de *ParserFactory* de juiste *parser* zal krijgen, die dan met de *Parse*-methode de bestanden zal verwerken en opslaan in de databank. Na het uploaden van alle bestanden, toont de *view* een gepaste boodschap naargelang het slagen of falen van alle of sommige bestanden. De gebruiker beëindigt de applicatie na het uploaden van de bestanden door de *view* te sluiten.



Figuur 36: sequentiediagram over het uploaden van bestanden

### 3. Testplan

#### 3.1 Unit testen

Bij de unit testen worden de *view controllers* getest (**Figuur 37**). Er moet gebruikgemaakt worden van een *mock*-databank om deze *controllers* afzonderlijk te kunnen. Hiervoor werd voor Moq gekozen (een *mocking framework* in C#). Vooraleer deze testen gestart kunnen worden, moet de *mock*-databank eerst opgevuld worden met test data (Jason, 2017). Dit gebeurde aan de hand van *mock*-sets die een bepaalde tabel van de databank voorstellen. Deze sets bevatten de nodige gegevens om de testen te kunnen uitvoeren.

Bij de *controllers* werden de volgende functionaliteiten getest.

1. Bij de gewone *controllers* moet de Index-methode getest worden op het teruggeven van een View met een *dependency injection*.
2. Gelijkaardige *testing* voor QualityReport, CTRLPoints etc. waarbij een View teruggegeven wordt met bijhorende data (een lijst of een object).
3. CRUD functionaliteit uittesten van de controllers.

```
[TestMethod()]
0 references
public void IndexTest_ActionExecutes_ReturnsViewForIndex()
{
    // Create mock context
    Mock<DroneDBEntities> mockContext = new Mock<DroneDBEntities>();
    DroneFlightsController controller = new DroneFlightsController(mockContext.Object);

    // Create a mock DbSet
    List<DroneFlight> flights = GetFlights();
    var mockSet = CreateMockSet(flights);
    // Set up the DroneFlights property so it returns the mocked DbSet
    mockContext.Setup(o => o.DroneFlights).Returns(() => mockSet.Object);

    var result = controller.Index() as ViewResult;
    Assert.AreEqual("Index", result.ViewName);
}
```

Figuur 37: Unit test voor de index methode van de DroneFlightsController

### 3.2 Integratietesten

Bij de integratietesten wordt de applicatie als geheel getest en is er dus geen nood aan een *mock*-databank. Er is gekozen om de integratietesten manueel uit te voeren. Voor de testscenario's wordt er van de verschillende use cases vertrokken. Er wordt beschreven wat de scenario's precies inhouden en de verschillende stappen die doorgaan in deze scenario's. Uiteindelijk wordt ook de uitkomst van de testen beschreven. In tegenstelling tot de MVC-controllers, werden de Web API controllers niet apart getest aan de hand van unit tests. Deze zullen nu in een aantal scenario's getest worden bij de integratietesten.

Er wordt één integratietest beschreven om hier geen al te lange opsomming van de verschillende scenario's te moeten geven. De andere testen zijn te vinden in Appendix B.

- Kaart met dronevluchten bekijken

Als de gebruiker een bepaalde dronevlucht op kaart wil zien, moet eerst naar de 'Drone Flights'-pagina genavigeerd worden. Op deze pagina krijgt de gebruiker een overzicht van alle dronevluchten. Er kan nu, bij een bepaalde dronevlucht, op het kaarticoon in de kolom Map geklikt worden. Hierna is de kaart met de dronevlucht en bijbehorende data te zien. In deze test worden de Web API-controllers getest. Zo zal bijvoorbeeld de `CTRLPointsController` de nodige data van de controlepunten beschikbaar stellen als *JSON* aan de frontend. Hierna wordt deze data opgehaald door een *AJAX-call* en is deze zichtbaar op de kaart. Dit gebeurt analoog voor de andere Web API-controllers.

## 4. Evaluaties en discussies

### 4.1 Performantie

Het interageren met de databank om piloten of drones toe te voegen, verwijderen of aan te passen gaat snel.

In de frontend van de applicatie kan het even duren voordat bepaalde *layers* geladen worden. Dit komt omdat bepaalde ontvangen JSON-objecten zeer groot kunnen zijn door de hoge frequentie van opgemeten punten van de drone. Deze grootte van de JSON kan beperkt worden door minder punten door te sturen of door gebruik te maken van een *differential view* voor XYZ punten (zie *Future Work*).

### 4.2 Beveiliging

Het loginsysteem biedt een basisbeveiliging aan voor de webapplicatie. Het schermt belangrijke handelingen af van niet-geautoriseerde gebruikers. Dit loginsysteem moet wel verder uitgebreid worden (zie *Future Work*).

Verder is de applicatie beschermd tegen *SQL injections* bij elke gebruikte query.

### 4.3 Schaalbaarheid

Door het gebruik van MVC is de webapplicatie eenvoudig uit te breiden. Daarnaast is het *simple factory pattern* van de parsers ideaal voor potentiële uitbreidingen met betrekking tot bestanden die aan een dronevlucht toegevoegd kunnen worden. Dit werd ook ondervonden tijdens het project, waarbij de *RawImagesParser* pas in sprint 3 werd geïmplementeerd.

Het databankmodel is minder makkelijk aan te passen wegens de relatief oude technologie van SQL. Er werd in het project wel rekening gehouden met schaalbaarheid in het model door lege velden te voorzien die de klant misschien toch cruciaal zou vinden.

Momenteel kan maar één persoon bestanden uploaden. De klant zal de bestaande code moeten uitbreiden (eventueel met *websockets* of op z'n minst een unieke id per gebruiker die wil uploaden) om upload-functionaliteit voor meerdere personen tegelijk mogelijk te maken.

#### 4.4 Problemen en geleerde lessen

Enkele geleerde lessen zijn:

- Het belang van een goeie keuze voor *design patterns* voor goede functionaliteit en schaalbaarheid;
- De voor- en nadelen van de gegevenstechnologieën *Entity Framework* en *ADO.NET*;
- Het belang van beveiliging tegen *SQL injections*;
- De asynchrone werking van JavaScript kan problemen veroorzaken met de flow van de webapplicatie; hier moet rekening mee gehouden worden.
- De werking van de communicatie tussen front- en backend (Web API Controllers).

## 5. Handleidingen

### 5.1 Inhoud van de distributie

In de distributie van dit project bevinden zich volgende bestanden:

- de **DroneWebApp**-map met alle code voor de webapplicatie;
- het SQL-script **DroneDB.sql** om de databank aan te maken;
- **ChangeDataSourceName\_Script.exe**, een Perl-executable;
- dit **verslag** met handleidingen en documentatie.

### 5.2 Installatiehandleiding voor ontwikkelaar

De installatiehandleiding is opgedeeld in drie delen:

- Deel 1: Vereiste software
- Deel 2: Aanmaken van de databank
- Deel 3: Opstarten van de webapplicatie
- Deel 4: *Deployen* van de webapplicatie

#### 5.2.1 Vereiste software

In dit deel installeert u alle benodigde software om de webapplicatie te laten werken op **Windows 7 en hoger**.

- 1 Installeer **SQL Server 2019** op uw machine. U kan deze software [hier](#) downloaden op de website van Microsoft.
  - 1.1 Scrol naar beneden en kies de versie die u verkiest (Developer of Express).
  - 1.2 Klik 'Download now'. Het programma downloadt.
  - 1.3 Volg na het uitvoeren van het gedownloade bestand de instructies op het scherm.
- 2 Installeer **SQL Server Management Studio (18.4) (SSMS)** op uw machine. U kan deze software [hier](#) downloaden.
  - 2.1 Klik op 'Download SQL Server Management Studio (SSMS)'. Het programma downloadt.
  - 2.2 Volg na het uitvoeren van het gedownloade bestand de instructies op het scherm.
- 3 Installeer **Visual Studio 2019** op uw machine. U kan deze software [hier](#) downloaden.
  - 3.1 Klik op 'Download Visual Studio' en kies de Community 2019 of de Professional 2019 versie naargelang uw eigen voorkeur. Het programma downloadt.

- 3.2 Volg na het uitvoeren van het bestand de instructies op het scherm. Kies tijdens de installatie om de volgende *workloads* te installeren: 'ASP.NET and web development' en 'Data storage and processing'.
- 4 Surf naar de website van [IvyTools](#).
  - 4.1 Op deze webpagina kunt u de gratis *personal license key* verkrijgen die u zult nodig hebben om de IvyTools software te activeren. Klik hiervoor op 'Click here to get your free personal license key'.
  - 4.2 Kopieer deze sleutel.
  - 4.3 Navigeer in de distributie naar de map 'drone1\IvyPdf\_1.62'.
  - 4.4 Voer het bestand **IvyTemplateEditor.exe** uit.
  - 4.5 Navigeer via de balk bovenaan het programma naar 'Help > About > Apply License Code'.
  - 4.6 Plak de eerder gekopieerde sleutel in het veld en druk op OK.
  - 4.7 Sluit IvyTemplateEditor af.
  - 4.8 U heeft nu toegang tot de IvyParser dll-bestanden in de webapplicatie. Deze worden gebruikt bij het inlezen van een pdf.
- 5 Surf naar de website van [Java](#).
  - 5.1 Klik op deze webpagina op 'Gratis Java-download'.
  - 5.2 Scrol op de webpagina die verschijnt naar beneden tot bij de knop 'Ga akkoord met de licentiebepalingen en start de download', en klik op deze knop om een *executable* van Java te downloaden.
  - 5.3 Start de *executable*.
  - 5.4 Er verschijnt een scherm. Onderaan staat de knop 'Install'. Deze zal de installatie automatisch starten.
  - 5.5 Indien u al een of meerdere oudere versies van Java geïnstalleerd had op uw machine, komt nu de mogelijkheid om de verouderde versie te verwijderen. Indien u dit wil, kan u op 'Uninstall' klikken. Indien u de oude versies wil behouden kan u gewoon op 'Next' klikken.
  - 5.6 Indien u koos voor het verwijderen van de oudere versies, komt er een scherm die samenvat welke versies verwijderd zijn. Hier kunt u gewoon op 'Next' klikken.
  - 5.7 Na al deze stappen komt er een scherm dat bevestigt dat Java geïnstalleerd is. Klik op 'Close'.
  - 5.8 U kan nu aan de slag met Java op uw toestel.

### 5.2.2 Aanmaken van de databank

In dit deel maakt u de SQL-Serverdatabank aan.

- 1 Start **SQL Server Management Studio** op en verbind met uw machine.
  - 1.1 Het veld 'Server name' wordt automatisch ingevuld.



- 1.2 Noteer deze naam, want u heeft deze later nodig in een volgend deel (opstarten van de webapplicatie).
- 1.3 Klik op 'Connect'.
- 2 Ga naar het menu 'File' bovenaan links.
- 3 Kies 'Open' en ga naar 'File'.
- 4 Navigeer in de distributie naar het script **DroneDB.sql** en open dit.
- 5 Klik op 'Execute' om het script uit te voeren.
- 6 In het 'Messages'-venster verschijnt "Commands completed successfully". U kan verifiëren dat de databank is aangemaakt met volgende stappen:
  - 6.1 Klik in het 'Object Explorer'-venster op 'refresh'.
  - 6.2 Vouw de Machinenaammap en Databasesmap open. Hierin bevindt zich nu de nieuwe database **DroneDB**. Dit is tevens de naam die in het bestand Web.config gebruikt wordt om de brug te leggen naar de databank. Deze naam wordt automatisch ingevuld door het Perlscript in punt 9 (zie later).
- 7 Een lege databank is nu aangemaakt en klaar voor gebruik.
- 8 Sluit SQL Server Management Studio.
- 9 In de distributie bevindt zich op het pad 'DroneWebApp\DroneWebApp\Programs\Perl' een bestand genaamd **ChangeDataSourceName\_Script.exe**. Voer dit bestand uit om de juiste *connection strings* in te vullen in **Web.config**. Deze leggen de verbinding tussen de databank en de webapplicatie.

### 5.2.3 Opstarten van de webapplicatie met Visual Studio

- 1 Voer de webapplicatie vanuit **Visual Studio** uit met **F5**.
- 2 De allereerste keer kan een venster verschijnen dat u vraagt om het 'IIS Express SSL certificate' te vertrouwen.
  - 2.1 Klik 'yes'.
- 3 Er verschijnt een 'security warning'.
  - 3.1 Klik 'yes'.
- 4 U kunt nu aan de slag met de dronewebapplicatie.

### 5.2.4 Publiken van de webapplicatie

Opmerking: Na het publiceren van de webapplicatie is het niet mogelijk om .DAT bestanden te parsen in de applicatie. Er loopt iets mis bij het uitvoeren van DatCon.exe. Het is wel mogelijk om reeds geconverteerde .DAT bestanden up te loaden.

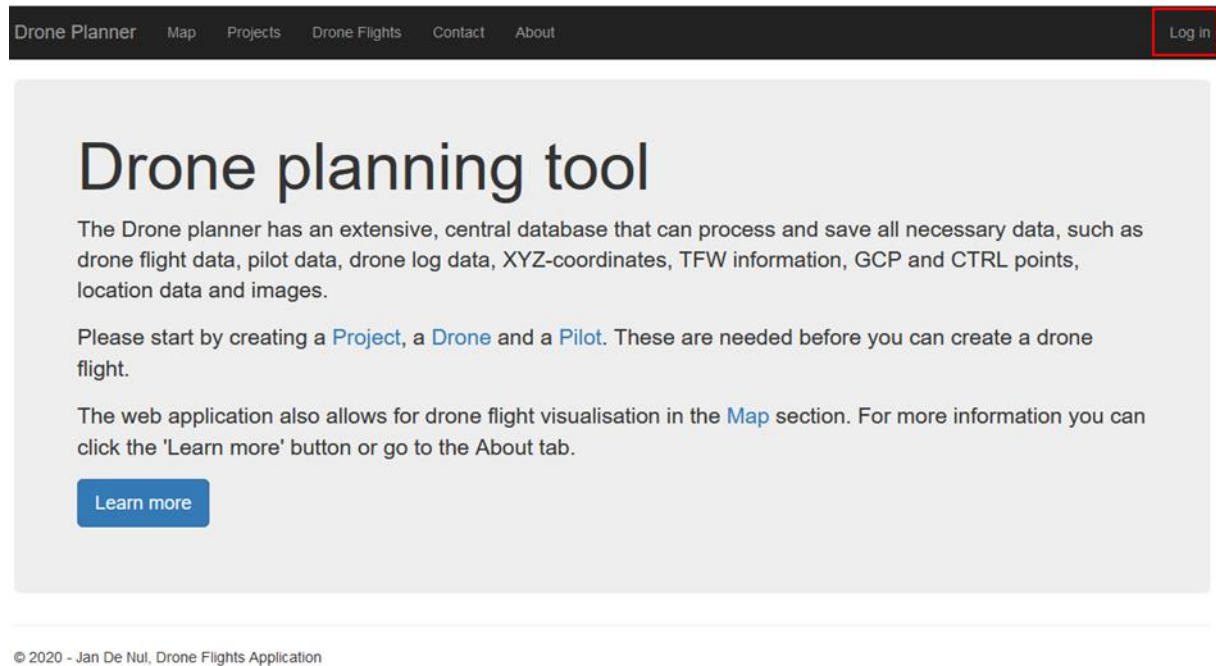
- 1 Installeer IIS op uw machine.
  - 1.1 Druk op de Windows + R toets om het 'Run' venster te openen.

- 1.2 Typ 'appwiz.cpl' in het tekstveld en druk op enter.
  - 1.3 Nu opent het 'Programma's en Onderdelen'-venster. Klik hier op Windows-onderdelen in- of uitschakelen.
  - 1.4 Vink de 'Internet Information Services' *check box* aan en klik op OK.
- 2 Installeer .NET versie 4, dit kan u [hier](#) downloaden.
  - 2.1 Scrol naar beneden.
  - 2.2 Klik op 'Downloaden' en volg de instructies op het scherm.
- 3 Start **SQL Server Management Studio** op en verbind met uw machine.
- 4 Start Visual Studio op als administrator en open het 'DroneWebApp'-project.
- 5 Klik rechts bij Solution Explorer op het project en kies 'Publish'.
- 6 Klik in het 'Publish'-venster op Start.
- 7 Kies bij 'Pick a publish target' voor 'IIS, FTP, etc' en klik op 'Create Profile'.
  - 7.1 Kies bij 'Publish method' voor File System
  - 7.2 Navigeer bij 'Target location' naar de 'wwwroot' *folder* van IIS (bv C:\inetpub\wwwroot).
  - 7.3 Klik op 'Next'.
  - 7.4 Selecteer bij 'Configuration' 'Release' en bij 'File Publish Options' 'Delete all existing files prior to publish'.
  - 7.5 Klik op 'Save'.
- 8 Klik nu op 'Publish'.
- 9 Open nu IIS Manager.
  - 9.1 Druk op de Windows + R toets om het 'Run' venster te openen.
  - 9.2 Typ 'inetmgr' in het tekstveld en klik ok OK.
- 10 Vouw de serverknoop open bij 'Verbindingen' en ga bij 'Sites' naar 'Default Web Site'.
- 11 Klik op 'Default Web Site' en ga bij 'Acties' naar 'Geavanceerde instellingen'.
  - 11.1 Selecteer bij 'Groep van toepassingen' 'DefaultAppPool'.
- 12 Ga naar 'Toepassingsgroepen'
  - 12.1 Rechtsklik op 'DefaultAppPool' en selecteer 'Basisinstellingen'.
  - 12.2 Selecteer bij '.NET CLR-versie' '.NET CLR-versie v4'
  - 12.3 Selecteer bij 'Beheerde pipeline-modus' 'Geïntegreerd'.
  - 12.4 Klik op 'OK'.
- 13 Ga naar **SQL Server Management Studio**.
  - 13.1 Open het 'Security'-tabblad.
  - 13.2 Klik rechts op 'Logins' en selecteer 'New login'.
  - 13.3 Geef als 'Login name' 'IIS APPPOOL\DefaultAppPool' in.
  - 13.4 Ga daarna naar 'User Mapping' en selecteer 'DroneDB'.
  - 13.5 Selecteer als rol 'db datareader', 'db datawriter' en 'db owner'.
  - 13.6 Klik op 'OK'.
- 14 Ga terug naar de 'inetpub' folder.
  - 14.1 Rechtsklik op de 'wwwroot' folder en klik op eigenschappen.

- 14.2 Ga naar het tablad beveiliging en klik op 'Bewerken'.
- 14.3 Klik op toevoegen.
- 14.4 Typ 'IIS APPPOOL\DefaultAppPool' in het tekstveld 'Geef de objectnamen op'.
- 14.5 Klik op 'namen controleren' en klik daarna op OK.
- 14.6 Selecteer de gebruiker 'DefaultAppPool' en vink 'Volledig beheer' aan.
- 14.7 Klik op toepassen en klik daarna op OK.
- 14.8 Klik nog eens op OK.
- 15 Ga terug naar 'IIS Manager'.
  - 15.1 Klik rechts op 'DroneWebApp'.
  - 15.2 Klik bij 'toepassing beheren' op 'bladeren'.
- 16 De website wordt nu opgestart.

## 5.3 Gebruikershandleiding

U kunt de webapplicatie gebruiken als één van drie rollen: bezoeker, gebruiker of administrator. Inloggen als administrator kan u door bovenaan rechts op de pagina te navigeren naar 'Log in', zoals weergegeven op onderstaande afbeelding.



Er wordt voor u een administrator aangemaakt bij het opstarten van de webapplicatie. Een administrator beschikt over de rechten om voor gebruikers accounts aan te maken, zodat u als administrator zelf kan beslissen wie een account heeft.

- Geef als *username* '**Admin**' op en als *password* '**password**'. Dit kan u nadien wijzigen. U bent nu ingelogd als administrator.
- U kan een gebruiker aanmaken door te navigeren naar de pagina 'Register' bovenaan rechts op de navigatiebalk.
- Kies een gebruikersnaam, een e-mailadres, een paswoord (kan nadien gewijzigd worden) en een rol voor deze persoon, zijnde administrator (*Admin*) of gebruiker (*User*).
- Deze nieuwe administrator of gebruiker kan nu op een gelijkaardige manier inloggen.

Een administrator en gebruiker kunnen hun profiel bekijken door bovenaan rechts op 'Hello \*\*\*\*!' te klikken, met '\*\*\*\*' de naam van het profiel. Op deze pagina kan u bovendien uw wachtwoord aanpassen.

U beschikt als administrator over alle rechten. U kan projecten, drones, piloten en dronevluchten aanmaken, wijzigen en verwijderen. Indien dit de eerste keer is dat u de webapplicatie gebruikt, zal u eerst en vooral drones en piloten moeten aanmaken. Een dronevlucht moet immers verplicht over een drone en piloot beschikken en kan dus niet zonder deze twee componenten aangemaakt worden.

### Drone aanmaken

- Navigeer via de navigatiebalk bovenaan de webapplicatie naar 'Drones'.
- Klik op de knop 'Create new Drone' om een nieuwe drone aan te maken.
- De rode sterretjes bij de in te vullen velden geven aan of het om een verplicht veld gaat of niet.
- Geef een *drone type* en een *registration* op. U kan een drone ook een eigen naam toekennen; indien u dit niet doet, dan zal een naam voor u gegenereerd worden op basis van de twee voorgaande, verplichte velden.
- Klik op de groene knop om de drone aan te maken.
- In het overzicht van de drones ziet u nu uw aangemaakte drone.

### Piloot aanmaken

- Navigeer via de navigatiebalk bovenaan de webapplicatie naar 'Pilots'.
- Klik op de knop 'Create new Pilot' om een nieuwe piloot aan te maken.
- Vul een naam in voor deze piloot. Vervolledig het profiel van deze piloot met eventuele gegevens.
- Klik op de groene knop om de piloot aan te maken.
- In het overzicht van de piloten ziet u nu uw aangemaakte piloot.

U hebt nu een drone en een piloot aangemaakt. De volgende stap is om een project aan te maken.

### Project aanmaken

- Navigeer via de navigatiebalk bovenaan de webapplicatie naar 'Projects'.
- Klik op de knop 'Create new Project' om een nieuw project aan te maken.
- Geef de *project code* voor dit project op. De velden *site reference code* en *vertical reference* zijn geen verplichte velden en kan u nadien ook nog aanvullen.
- Klik op de groene knop om het project aan te maken.
- In het overzicht van de projecten ziet u nu uw aangemaakt project.

In de overzichten van de projecten, piloten en drones kan u klikken op 'View Flights' om de vluchten te bekijken die eraan zijn toegekend. Op dit moment zijn er nog geen dronevluchten aangemaakt.

### Dronevlucht aanmaken

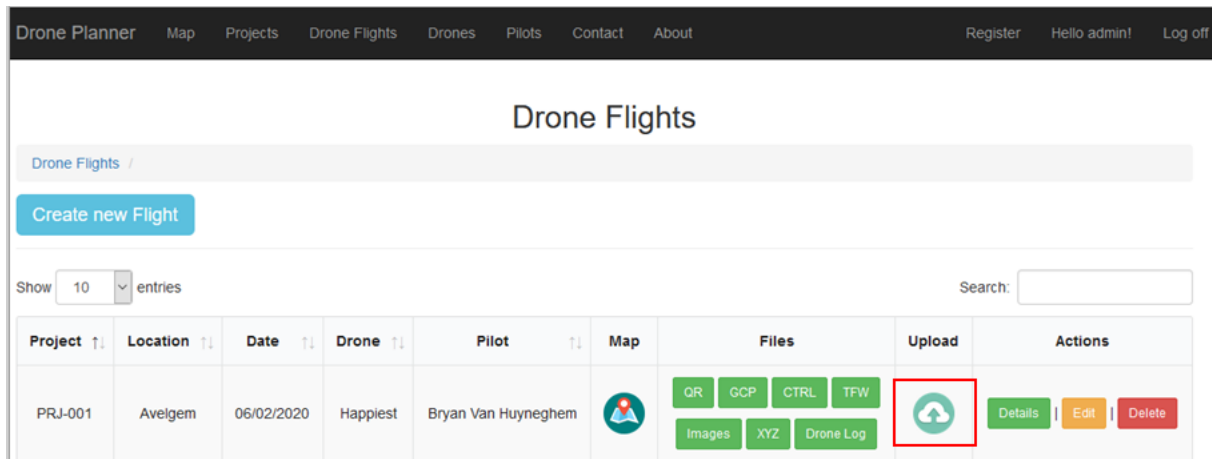
- U kan een dronevlucht aanmaken vanuit één van de overzichten van de projecten, piloten of drones door op de knop 'Create new Flight for this \*\*\*\*\*' te klikken, met '\*\*\*\*\*' een project, een drone of een piloot. Tijdens het aanmaken van deze dronevlucht zal respectievelijk de *project code*, de id van de drone of de id piloot reeds ingevuld worden voor u.
- U kan een dronevlucht ook aanmaken door te navigeren via de navigatiebalk bovenaan naar 'Drone Flights' en op de knop 'Create new Flight' te klikken.
- U bent verplicht om de dronevlucht toe te voegen aan een project, en om een drone en piloot te selecteren voor deze vlucht. Deze kunnen nadien nog aangepast worden indien een fout zou begaan worden in het toekennen van de vlucht.
- U kan ook een locatie en een datum invullen. Indien u dit niet doet, dan kan u dit automatisch laten invullen door nadien een dronelogbestand te uploaden voor deze vlucht.
- U kan ook de kiezen om de velden *type of activity*, *other*, *simulator*, *instructor* en *remarks* in te vullen. Deze velden kunnen nadien ook nog aangepast of ingevuld worden.
- Klik op de groene knop om de dronevlucht aan te maken.
- In het overzicht van de dronevluchten ziet u nu uw aangemaakte vlucht. Indien u geen datum en/of locatie hebt opgegeven, dan werd voor u in deze cellen 'TBD' (*to be determined*) ingevuld als *placeholder*.
- U kan ook navigeren naar de 'Details'-pagina van een dronevlucht om alle informatie over deze vlucht in detail te bekijken.
- Op deze pagina kan u, net zoals in het overzicht van alle dronevluchten, iconen zien voor de verschillende soorten documenten die geüpload kunnen worden.
  - Indien een icoon groen is, betekent dit dat dit document voor deze vlucht reeds geüpload is.
  - Indien een icoon rood is, betekent dit dat dit document voor deze vlucht nog niet geüpload is.

### Dronevluchten, projecten, drones en piloten verwijderen en/of aanpassen

- U kan als administrator dronevluchten, projecten, drones en piloten op een gelijkaardige manier verwijderen of aanpassen door te navigeren naar een overzicht van wat u wil verwijderen.
- Een gebruiker kan enkel dronevluchten verwijderen en aanpassen. Hij kan projecten, drones en piloten bekijken, maar niet verwijderen of aanpassen.

## Bestanden uploaden voor een dronevlucht

- U kan bestanden toevoegen aan een vlucht door naar één van overzichten van de dronevluchten te navigeren en op het icoon te klikken in de kolom 'Upload', zoals te zien is op volgende afbeelding.
- Bestanden worden bijgehouden in een centrale databank.



- U kan deze knop ook vinden in de 'Details'-pagina van de dronevlucht.
- Klik op deze knop om te navigeren naar de uploadpagina.
- Op deze pagina kan u verschillende bestanden selecteren om toe te voegen aan uw dronevlucht.
  - Een kwaliteitsrapport (PDF-bestand);
  - Een XYZ-bestand;
  - Een TFW-bestand;
  - Een CSV-bestand voor de GCP's en CTRL's;
  - Afbeeldingen die door de drone genomen werden (JPG-bestanden);
  - Een dronelogbestand (DAT-bestand);
  - Een geconverteerd dronelogbestand (CSV-bestand).
- U kan één of meerdere bestanden tegelijk uploaden. U ziet eerst de uploadprogressie op de progressiebalk verschijnen. Daarna ziet u de progressie van het *parsen* van deze bestanden.
- U kan geen duplicaten uploaden. Bestanden die niet afgehandeld kunnen worden door de server zullen eveneens niet aanvaard worden. U zal een melding krijgen met een lijst van de bestanden die niet werden toegevoegd aan de databank.

## Opmerkingen

- Slechts één gebruiker per keer kan bestanden uploaden. Indien een andere gebruiker op hetzelfde moment bestanden probeert te uploaden, dan zal hij of zij een melding krijgen dat er iemand anders aan het uploaden is en even te wachten.
- U kan maximaal 2.1 GB aan bestand in één keer uploaden.
- U kan maximaal 500 MB aan afbeeldingen in één keer uploaden.

## De informatie van een vlucht bekijken

U kan de informatie van een dronevlucht bekijken door te navigeren naar de 'Details'-pagina van een dronevlucht via één van de overzichten van de dronevluchten. U kan alle informatie voor deze dronevlucht bekijken alsook doorklikken op een drone of een piloot om te navigeren naar hun 'Details'-pagina.

Op deze pagina kan u bovendien zien welke bestanden er al dan niet reeds zijn toegevoegd aan deze vlucht door te scrollen naar de sectie 'Documents'.

U kan klikken op deze iconen om te navigeren naar een pagina over dit document. Merk op dat een icoon groen of rood is en een indicatie is voor het al dan niet geüpload zijn van dit bestand voor deze dronevlucht.

De pagina's 'XYZ' en 'Drone Log' zijn niet bereikbaar, omdat deze bijzonder veel informatie bevatten en aldus geen bijdrage leveren om in een webpagina weer te geven.

## De informatie van een project, piloot of drone bekijken

U kan de informatie van een project, piloot of drone bekijken door te navigeren naar de 'Details'-pagina via één van de overzichten. U kan deze overzichten bereiken via de navigatiebalk bovenaan de webapplicatie.

## Drone *check-up*

Na enige tijd moet een drone gecontroleerd worden om falen tijdens het vliegen te voorkomen. Dit gebeurt automatisch na een welbepaald gevlogen tijd (nu ingesteld op 50 uren, in de DronesController) en wordt visueel aangegeven in het overzicht van de drones (de 'Index'-pagina'), zoals te zien is op onderstaande afbeelding.

Show  entries Search:

Name ↑↓	Drone Type ↑↓	Registration ↑↓	Total Flights ↑↓	Total Time Flown ↑↓	Export Logbook	Actions
Happiest	XYZ	Bot	10 <a href="#">[View]</a>	02h 08m 44s	<a href="#">CSV</a> <a href="#">PDF</a>	<a href="#">Details</a> <a href="#">Edit</a> <a href="#">Delete</a>
Happy	Auto	Bot	11 <a href="#">[View]</a>	02h 53m 34s	<a href="#">CSV</a> <a href="#">PDF</a>	<a href="#">Details</a> <a href="#">Edit</a> <a href="#">Delete</a>
Inferno	ABC	Bot	0 <a href="#">[View]</a>	00h 00m 00s	<a href="#">CSV</a> <a href="#">PDF</a>	<a href="#">Details</a> <a href="#">Edit</a> <a href="#">Delete</a>

Showing 1 to 3 of 3 entries [Previous](#) [1](#) [Next](#)

Een administrator kan een drone ook manueel markeren indien er nood is aan een *check-up*.

- Dit kan door te navigeren naar de 'Edit'-pagina van een drone en de *checkbox* aan te vinken.
- Klik vervolgens op 'Save' om uw aanpassingen op te slaan.



Indien de drone vervolgens gecontroleerd en goedgekeurd werd, dan kan op een gelijkaardige manier deze *checkbox* aangepast worden via de 'Edit'-pagina. De tijd verschijnt opnieuw in het groen en er werd een nieuwe, volgende *threshold*-tijd berekend, die bekeken kan worden in de 'Details'-pagina van de drone (zie onderstaande afbeelding).

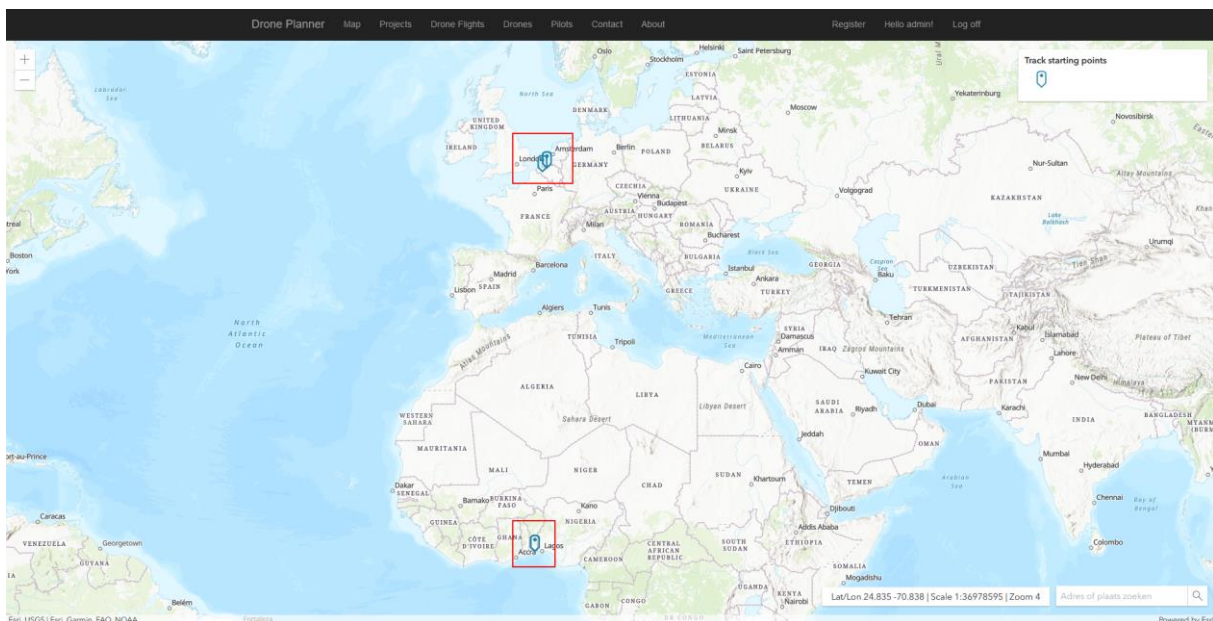
## Happiest

Export options: [CSV](#) [PDF](#)

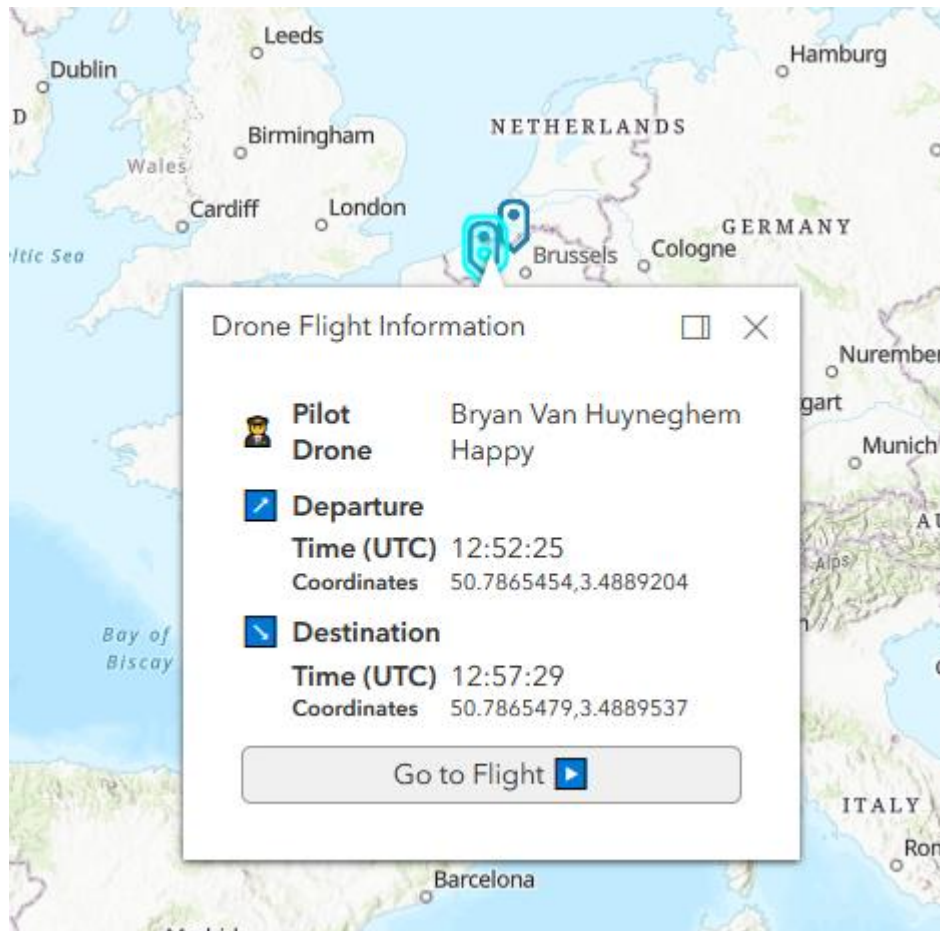
Information	
Name	Happiest
Total Flights	10   <a href="#">View Flights</a>
Total Time Flown	02h 08m 44s
Registration	Bot
DroneType	XYZ
Needs check up?	No; next check-up at: 52h 08m 44s

## Een overzicht van alle dronevluchten bekijken op een kaart

Navigeer via de navigatiebalk bovenaan de webapplicatie naar 'Map'. Dit opent een wereldkaart waarop elke dronevlucht op locatie met een icoon weergegeven is (zie onderstaande afbeelding), indien er voor deze dronevlucht een dronelogbestand geüpload werd.

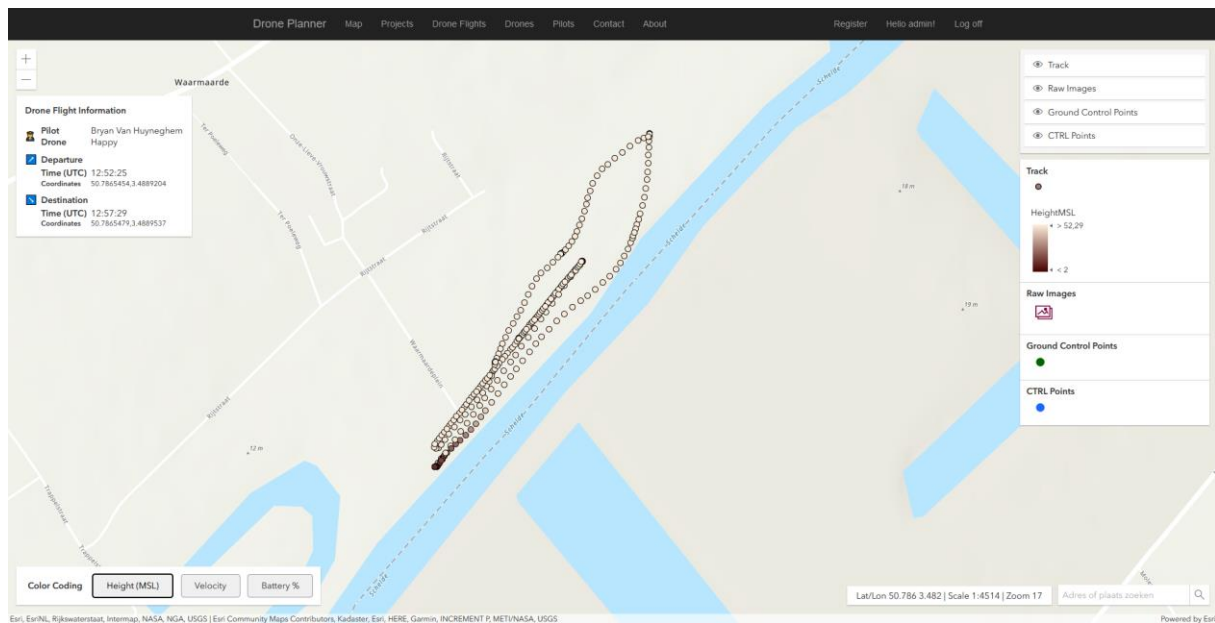


- U kan klikken op een dronevlucht om een *popup*-venster te laten verschijnen met informatie over deze vlucht, zoals weergegeven op onderstaande afbeelding.

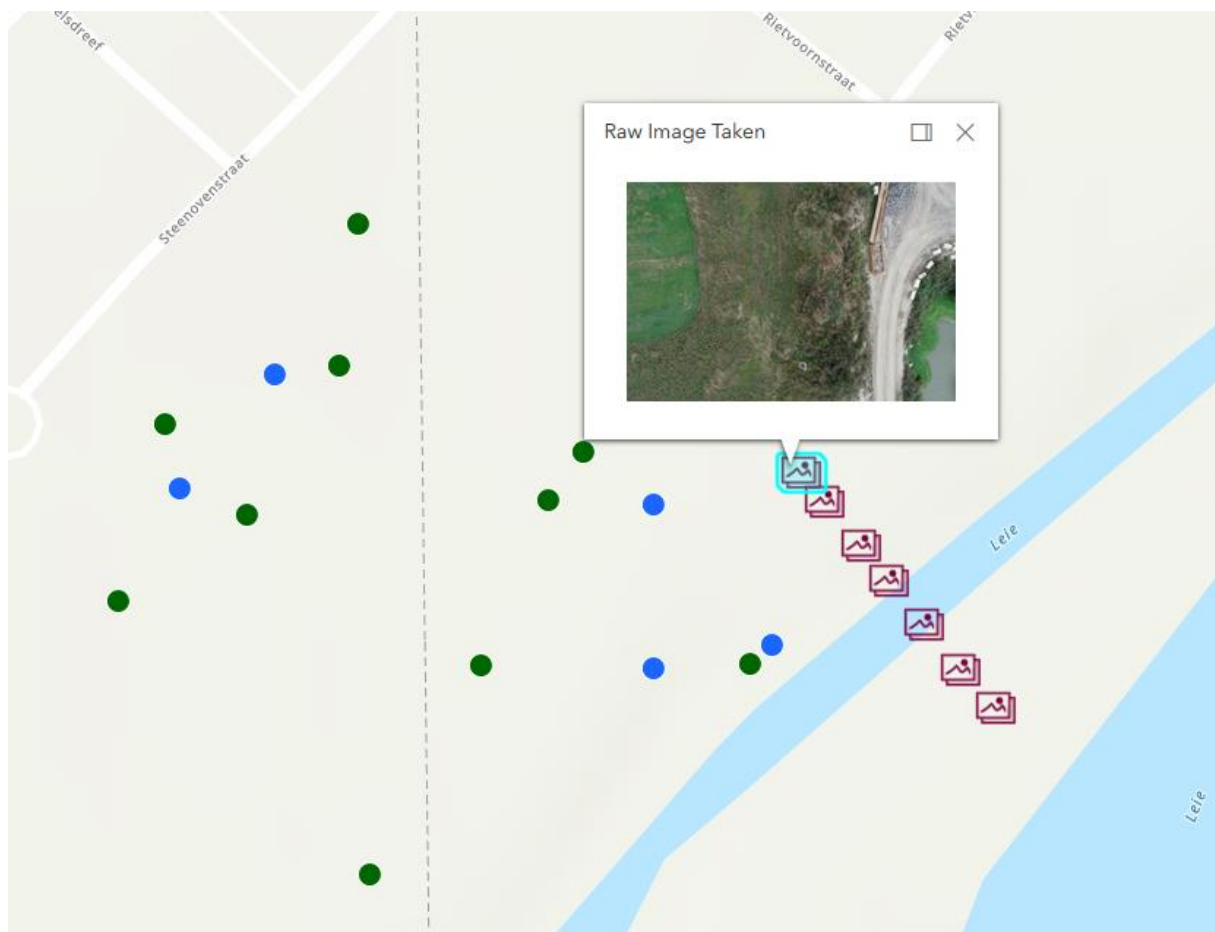


- Klik op 'Go to Flight' om te navigeren naar de kaart van deze vlucht. Op deze kaart worden alle gegevens die gevisualiseerd kunnen worden weergegeven, zoals de *track* van een dronevlucht, de *ground control points*, de *control points* en de *images*.
- De gebruiker kan kiezen om de *color coding* van de *track* in te stellen op de hoogte, de snelheid of het batterijpercentage, naargelang deze data aanwezig was in het dronelogbestand.
- De verschillende componenten kunnen verborgen worden door te klikken op de ogen in de rechterbovenhoek.

Op de volgende pagina kan u een voorbeeld zien van een *track* op basis van de hoogte.



- Punten zijn aanklikbaar en openen een *widget* met informatie, zoals weergegeven op onderstaande afbeelding.



## Een pilotenlogboek of dronelogoek exporteren

U kan een logboek voor een piloot of een drone aanmaken door te navigeren naar het overzicht van een drone of een piloot via de navigatiebalk bovenaan de webapplicatie.

- In de kolom 'Export Logbook' kiest u voor het bestandstype waarnaar u dit logboek wil exporteren. U kan kiezen voor een CSV-bestand of een PDF-bestand.
- U wordt gevraagd om dit bestand te openen of op te slaan.

## Zoeken in een tabel en sorteren van een kolom

U kan het zoekveld boven elk overzicht van een project, dronevlucht, piloot of drone gebruiken om te zoeken naar specifieke informatie aanwezig in deze tabel.

- Zoek in een tabel door te typen in het zoekveld, zoals weergegeven op onderstaande afbeelding.
- U kan in de aangeduide kolommen zoeken.
- U kan deze kolommen ook sorteren met de pijlen.

### Drone Flights

Drone Flights /

Create new Flight

Show 10 entries

Search: Philip

Project ↑↓	Location ↑↓	Date ↑↓	Drone ↑↓	Pilot ↑↓	Map	Files	Upload	Actions
PRJ-001	Temse	25/10/2019	Happy	Philip Kukoba		QR GCP CTRL TFW Images XYZ Drone Log		Details Edit Delete
PRJ-001	NA	04/03/2020	Happiest	Philip Kukoba		QR GCP CTRL TFW Images XYZ Drone Log		Details Edit Delete
PRJ-001	NA	04/03/2020	Happiest	Philip Kukoba		QR GCP CTRL TFW Images XYZ Drone Log		Details Edit Delete
PRJ-003	TBD	TBD	Happiest	Philip Kukoba		QR GCP CTRL TFW Images XYZ Drone Log		Details Edit Delete
PRJ-003	NA	04/03/2020	Happiest	Philip Kukoba		QR GCP CTRL TFW Images XYZ Drone Log		Details Edit Delete

Showing 1 to 5 of 5 entries (filtered from 21 total entries)

Previous 1 Next

## 5.4 Problemen die zich kunnen voordoen

- De webapplicatie start op, maar hij kan niet connecteren met de databank

Er is hoogstwaarschijnlijk een probleem met de *connection string* die de brug legt tussen de webapplicatie en de databank.

- 1 Noteer de naam waarmee u in SQL Server Management Studio ingelogd hebt. Dit is doorgaans de computernaam.
- 2 In de distributie bevindt zich onder het pad 'drone1\DroneWebApp\DroneWebApp' een bestand **Web.config**. Open dit bestand met een editor naar keuze, zoals Visual Studio, Sublime Text of Notepad.
- 3 Gebruik de toetsencombinatie **CTRL + F** om te zoeken in dit bestand. Zoek naar 'data source'. U vindt twee gevallen.
- 4 Vervang de *string* tussen het gelijkheidsteken en de puntkomma door uw computernaam. Dit is tevens de naam waarmee u in SQL Server Management Studio ingelogd hebt. Onderstaande afbeelding geeft u een voorbeeld van de *strings* die u moet vervangen, met als voorbeeld de computernaam 'Predator'.

```
<connectionStrings>
  <!--new connection string for ADO.NET -->
  <add name="DroneDB_ADONET" connectionString="Data Source=Predator;Initialia
  <add name="DroneDBEntities" connectionString="metadata=res://*/Models.Dr
    data source=Predator;initial catalog=DroneDB;integrated security=True;
</connectionStrings>
```

- De databank werd aangepast en nu klopt het model niet meer.

U zal de webapplicatie moeten openen in Visual Studio en volgende stappen ondernemen:

- 1 Start **Visual Studio**.
- 2 Indien er een **DroneDBModel.edmx** aanwezig is in het project, dan verwijdert u deze.  
2.1 Klik hiervoor rechts op 'DroneDBModel.edmx > Delete > OK'.
- 3 Navigeer naar het bestand **Web.config**. Gebruik **CTRL + F** om te zoeken op 'connectionStrings'. Verwijder de tag <add name="DroneDBEntities" ... />.
- 4 Sla dit bestand op.
- 5 Navigeer in het project naar de map '**Models**'.
- 6 Klik rechts op deze map en selecteer 'Add > New Item'.
- 7 Zoek in de zoekbalk bovenaan rechts naar "model".
- 8 Er verschijnt een project item genaamd **ADO.NET Entity Data Model**. Klik deze eenmalig aan om hem te selecteren.
- 9 Geef dit ADO.NET Entity Data Model de naam **DroneDBModel** en klik op 'Add'.

- 10 Een nieuw venster verschijnt. Selecteer 'EF Designer from database' en klik op 'Next'.
- 11 Er verschijnt een nieuw venster. Klik op 'New Connection'.
- 12 Controleer of 'Data source' van het type 'Microsoft SQL Server (SqlClient)' is.
  - 12.1 Indien dit niet het geval is, klik dan op 'Change' en selecteer daar 'Microsoft SQL Server (SqlClient)'.
- 13 Klik op 'OK'.
- 14 Klik op 'Refresh' en wacht tot de lijst van servers opgehaald is.
- 15 De naam van uw computer verschijnt. Dit is dezelfde naam als de 'Server name' die u noteerde tijdens het aanmaken van de databank.
  - 15.1 Indien er geen naam verschijnt, dan kan u op het pijltje naar beneden klikken om uw computer uit de lijst te selecteren.
- 16 Indien deze lijst leeg is, geef dan uw computernaam ('server name') manueel in.
- 17 In de sectie 'Connect to a database', onder 'Select or enter a database name', selecteert u '**DroneDB**'. Klik op OK.
- 18 Controleer of 'Save connection settings in Web.Config as' aangevinkt is en 'DroneDBEntities' heet.
- 19 Klik op 'Next'.
- 20 Selecteer 'Tables' in het veld 'Which database objects do you want to include in your model?'.
- 21 Vink 'Pluralize or singularize generated object names' aan.
- 22 Vink 'Include foreign key columns in the model' aan.
- 23 Klik op 'Finish'.
- 24 Na enige tijd zijn de modelklassen van de databank aangemaakt.
  - Waarom heb ik IvyTools nodig? / Ik krijg een fout omtrent IvyTools in de webapplicatie.

IvyTools is een hulp-parser die gebruikt wordt om PDF-bestanden in te lezen. Het ingeven van een sleutel geeft u gedurende twee maanden toegang tot de dll-bestanden die in de webapplicatie aanwezig zijn.

Indien deze sleutel verloopt, dan kan u zoals in 5.2.1, punt 4, naar de website surfen en een nieuwe sleutel generen. U volgt de stappen die daar vermeldt staan om deze sleutel te activeren.

- Er wordt maar geen progressie gemaakt tijdens het uploaden van één of meerdere afbeeldingen.

Uit een aantal testen met de virtuele server werd duidelijk dat er soms problemen zijn bij het uploaden van afbeeldingen bij het *runnen* van de webapplicatie met Visual Studio. Op de laptops en computers van de ontwikkelaars deden zich nochtans geen enkele problemen voor bij het uploaden van afbeeldingen.

Volgende *exceptions* deden zich op de server voor:

- Exception thrown: 'Microsoft.CSharp.RuntimeBinder.RuntimeBinderException' in Microsoft.CSharp.dll
- Exception thrown: 'System.Data.SqlClient.SqlException' in System.Data.dll
- Exception thrown: 'System.Data.Entity.Core.UpdateException' in EntityFramework.dll
- Exception thrown: 'System.Data.Entity.Infrastructure.DbUpdateException' in EntityFramework.dll
- Exception thrown: 'System.Data.Entity.Infrastructure.DbUpdateException' in System.Web.Mvc.dll

Nochtans zijn de dll's dezelfde als op de computers van de ontwikkelaars. Ze zijn tevens allemaal up to date. Mogelijks heeft dit dus eerder met de server te maken.

- Het converteren van een DAT-bestand tijdens het uploaden ervan lukt niet.

Mogelijks heeft de DatCon.exe geen toestemming om te worden uitgevoerd op de computer waarop de webapplicatie draait. Controleer of u Java hebt geïnstalleerd en of de *executable* toestemming heeft te *runnen* op de computer (firewall of andere beveiliging).

## Besluit

Jan De Nul had nood aan een droneplanning-tool met een uitgebreide, centrale databank waarin zij hun dronevluchtdata kunnen opslaan. Op basis van de informatie die Jan De Nul verstrekke, werden vijf doelstellingen opgesteld die gedurende het project de leidraad vormden.

- (1) Alle aanwezige informatie in kaart brengen en een databankmodel creëren voor onze centrale databank;
- (2) Deze informatie parsen en wegschrijven naar deze databank;
- (3) Een webapplicatie en interface ontwikkelen zodat de gebruiker kan interageren met de data;
- (4) De data visualiseren met ArcGIS in een webviewer;
- (5) De mogelijkheid voorzien om sommige ingelezen data opnieuw te kunnen exporteren naar een gewenst formaat, zoals pdf of csv.

Deze doelstellingen werden zo goed mogelijk uitgewerkt naar de noden van de klant toe. De droneplanning-tool bestaat uit een webapplicatie gebouwd in het *server-side framework* ASP.NET MVC 5. De databank werd ontworpen met SQL Server van Microsoft en bevat alle informatie van de webapplicatie, de gebruikers en de informatie geüpload door deze gebruikers. ORM zorgt voor de brug tussen de databank en de webapplicatie.

De klant kan verscheidene projecten, drones, piloten en dronevluchten aanmaken en aanpassen. Verder beschikt de webapplicatie over een inlogsysteem dat afscherming van de buitenwereld garandeert.

Er kunnen ook bestanden toegevoegd worden aan dronevluchten. Deze bestanden worden vervolgens door de *parsers* verwerkt. De data van deze bestanden worden dan met behulp van Entity Framework 6 en ADO.NET weggeschreven naar de databank. De webapplicatie is in staat om het programma DatCon te gebruiken om automatisch een DAT-bestand om te zetten naar een CSV-bestand en dit vervolgens te *parsen* en weg te schrijven naar de databank.

Verder kunnen bezoekers en gebruikers dankzij een webviewer, gebouwd met de ArcGIS JavaScript API, dronevluchtdata, zoals de GCP's, de CTRL's, de dronevlucht *track*, *afbeeldingen* en andere informatie visueel zien op een kaart.

Als laatste werden exportopties voorzien voor drone- en pilotenlogboeken. Er kan gekozen worden tussen exporteren naar een CSV-bestand of een PDF-bestand.



## Future work

In dit onderdeel worden beperkingen van de drone-planningtool belicht, alsook enkele functionaliteiten die tijdens een *future work* toegevoegd zouden kunnen worden.

- Beperkingen betreffende het uploaden van bestanden

De eerste (en onmiddellijk de grootste) beperking van de tool betreft het feit dat slechts één gebruiker per keer bestanden kan uploaden. Dit zou opgelost kunnen worden door aan iedere gebruiker een unieke id toe te kennen tijdens het uploaden, zodat de *server-side* weet welke *progress* hij naar welke gebruiker moet sturen. Op dit moment krijgen de andere gebruikers die trachten om bestanden te uploaden een melding indien er reeds een gebruiker bestanden aan het uploaden is. De andere gebruikers krijgen dan onmiddellijk de *progress bar* van de gebruiker die aan het uploaden is te zien. Op deze manier kunnen zij ongeveer inschatten wanneer zij kunnen om beurt kunnen uploaden.

Een tweede beperking betreft het feit dat de som van alle bestandsgroottes van de bestanden die een gebruikers probeert te uploaden niet groter kan zijn dan 2.147 GB (een *signed int*). Deze beperking wordt opgelegd in het bestand **Web.config**. Dit bestand laat geen grotere bestandsgroottes toe. Dit kan een probleem geven voor zeer grote XYZ-bestanden. Mogelijks moeten deze in twee stukken gehakt worden en na elkaar geüpload worden. In het geval dat de totaalsom van een reeks bestanden groter is dan 2.147 GB, dan moet de gebruiker zijn bestanden in twee of meerdere keren uploaden.

Een derde beperking betreft het feit dat de huidige implementatie (met Entity Framework 6) van de RawImageParser tijdens het aanmaken van de *thumbnails* van de afbeeldingen plots zeer veel geheugen na een tijd meer en meer geheugen gaat gebruiken indien zeer veel afbeeldingen tegelijk geüpload worden. Dit kan mogelijks te maken hebben met de *memory cap* van de programmeeromgeving (Visual Studio 2019) die een *out of memory* geeft indien meer dan 3 GB aan memory gebruikt wordt. Dit geheugengebruik is van vrij korte duur en kan dus mogelijks op een server helemaal geen probleem geven. Het geheugen wordt immers snel weer vrijgegeven na het *parsen* van de geüploade afbeeldingen. Hoe dan ook zou het interessant kunnen zijn om de RawImageParser te herschrijven zodat hij gebruikmaakt van ADO.NET (analoog aan de DATParser en XYZParser). Dit zou het *parsen* van de afbeeldingen ook sneller laten verlopen.

- Verwijderen van bestanden behorende bij een dronevlucht

In een *future work* zou het ook mogelijk moeten kunnen zijn voor een *admin* om specifieke data per dronevlucht te verwijderen, indien bijvoorbeeld een foutief bestand geüpload zou zijn. In de betreffende tabellen zou gezocht kunnen worden met de *FlightId* van een dronevlucht, om zo alle informatie die te maken heeft met die dronevlucht uit de databank te verwijderen. Dit zou voor de data van elk bestand moeten kunnen.

- Visualisatie
  - De mogelijkheid voorzien om kleuren van *track* dynamisch te laten aanpassen naargelang de keuze van de gebruiker.
  - De *flow* van de *map view* kan verbeterd worden op basis van *callback* functies en asynchrone events.
  - Een *differential view* implementeren om snel een situatie met een vorige situatie te kunnen vergelijken. Dit zou een nieuw raster maken waarbij de waarde gelijk is aan het verschil tussen de vorige en de laatste meting.
  - Via een KML een polygoon tekenen voor de dronevluchten (d.i. een *flight plan*).
  - De grenzen (uiterste limieten) van alle afbeeldingen tonen op de kaart.
  - De TIFF-afbeelding van een dronevlucht projecteren op de kaart.
  - De track van een dronevlucht aanpassen naar een polyline in plaats van verscheidene, aparte punten.
  - De puntenwolk visualiseren met LAS/LASZ. XYZ punten projecteren op de *map* is op dit moment niet haalbaar. Er zijn teveel individuen XYZ-punten die opgehaald moeten worden met een HTTP GET call en vervolgens getekend moeten worden op de kaart. Mogelijks kan een LAS-bestand hier soelaas brengen.
  - Een *screenshot*-knop om een *screenshot* te nemen van de kaart zoals ze op dit moment weergegeven wordt.

- Logboeken

Idealiter zouden de PDF-bestanden die worden gemaakt voor drone en piloot via de exportfunctionaliteit er wat kleurrijker mogen uitzien.

- Huisstijl van Jan de Nul

Verder zou de huisstijl van Jan de Nul toegepast moeten worden. Dit wil zeggen hun lettertypes, lay-out... toepassen op de webapplicatie.

## Referenties

ArcGIS API. (z.j.) Geraadpleegd op 15 maart, 2020 via

<https://developers.arcgis.com/javascript/latest/guide/get-api/>

ArcGIS for Developers. (z.j.) Geraadpleegd op 13 februari, 2020 via

<https://developers.arcgis.com/labs/>

ASP.NET MVC. (2016, mrt. 20). Create Database, Retrieve Database table item in MVC [Video]. YouTube. <https://youtu.be/TwUwOm8DvOE>

ASP.NET. (z.j.) Geraadpleegd op 15 februari, 2020 via

<https://dotnet.microsoft.com/apps/aspnet>

Connecting to SQL Server using Visual Studio. (z.j.) Geraadpleegd op 15 februari, 2020 via

<https://blogs.gre.ac.uk/cmssupport/application-development/programming/asp-net/connecting-to-sql-server-using-visual-studio/>

Corey Schafer. (2014, jul. 22). ArcGIS API for JavaScript Part 1: Our First Web Map [Video].

YouTube. <https://youtu.be/-tsFnoKNtNc>

DatCon V3 .CSV (z.j.) Geraadpleegd op 24 maart, 2020 via

<https://datfile.net/DatCon/fieldsV3.html>

File upload in asp.net core mvc. (z.j.) Geraadpleegd op 28 februari, 2020 via <https://csharp-video-tutorials.blogspot.com/2019/05/file-upload-in-aspnet-core-mvc.html>

Hoyos, Mario. (2018, dec. 24). What is an ORM and Why You Should Use it [Blog post].

Geraadpleegd via <https://blog.bitsrc.io/what-is-an-orm-and-why-you-should-use-it-b2b6f75f5e2a>

IAmTimCorey. (2018, nov 5). SQL Data Tools In C# - Database Creation, Management, and Deployment in Visual Studio [Video]. YouTube. <https://youtu.be/ijDcHGxyqE4>

Jason. (2017, oct. 22). How to Mock an Entity Framework DbContext and DbSet with Moq. Geraadpleegd op 19 april, 2020 via

<https://codingoncaffeineblog.wordpress.com/2017/10/23/how-to-mock-an-entity-framework-dbcontext-and-dbset-with-moq/>

Rick-Anderson. (z.j.). Creating Model Classes with the Entity Framework (C#). Geraadpleegd op 13 februari, 2020, via <https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/models-data/creating-model-classes-with-the-entity-framework-cs>

Rick-Anderson. (z.j.). Tutorial: Create the Web Application and Data Models for EF Database First with ASP.NET MVC. Geraadpleegd op 13 februari, 2020 via <https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/database-first-development/creating-the-web-application>

SeeSharpCode. (2015, nov. 2). C# Entity Framework 6 | Database First | Part 2 - Foreign Keys & Relationships [Video]. YouTube. <https://youtu.be/dXvSSnQJWPo>

Software Dev. (2017, jan. 23). C# Entity Framework 6 Database First Complete Tutorial [Video]. YouTube. <https://youtu.be/o3ROFXgvXsM>

Understanding world files. (z.j.). Geraadpleegd op 13 februari, 2020 via [http://webhelp.esri.com/arcims/9.3/General/topics/author\\_world\\_files.htm](http://webhelp.esri.com/arcims/9.3/General/topics/author_world_files.htm)

## Appendix A: use case-diagrammen stories

**Naam**

Lijst met projecten bekijken

**Doelstelling**

Een gebruiker bekijkt een lijst van alle projecten uit de webapplicatie

**Actor**

Externe gebruiker

**Successscenario**

1. De gebruiker duidt aan dat hij alle projecten wil bekijken
2. Het systeem haalt de projecten op uit de databank
3. Het systeem toont een lijst met alle projecten

**Naam**

Details project bekijken

**Doelstelling**

Een gebruiker bekijkt de details van een project uit de webapplicatie

**Actor**

Externe gebruiker

**Successscenario**

1. De gebruiker duidt aan welk project hij wil bekijken
2. Het systeem haalt het project op uit de databank
3. Het systeem toont de details van de dronevlucht
4. De gebruiker geeft aan dat hij het bekijken van de details wil afronden
5. Het systeem geeft een overzicht van alle dronevluchten

**Naam**

Lijst met dronevluchten bekijken

**Doelstelling**

Een gebruiker bekijkt een lijst van alle dronevluchten uit de webapplicatie

**Actor**

Externe gebruiker

**Successscenario**

1. De gebruiker duidt aan dat hij alle dronevluchten wil bekijken
2. Het systeem haalt de dronevluchten op uit de databank
3. Het systeem toont een lijst met alle dronevluchten

**Naam**

Details dronevlucht bekijken

**Doelstelling**

Een gebruiker bekijkt de details van een dronevlucht uit de webapplicatie

**Actor**

Externe gebruiker

**Successscenario**

1. De gebruiker duidt aan welke dronevlucht hij wil bekijken
2. Het systeem haalt de dronevlucht op uit de databank
3. Het systeem toont de details van de dronevlucht
4. De gebruiker geeft aan dat hij het bekijken van de details wil afronden
5. Het systeem geeft een overzicht van alle dronevluchten



**Naam**

Kaart met dronevluchten bekijken

**Doelstelling**

Een gebruiker bekijkt de kaart met de visualisatie van een dronevlucht uit de webapplicatie

**Actor**

Externe gebruiker

**Successscenario**

1. De gebruiker duidt aan welke dronevlucht hij wil visualiseren
2. Het systeem haalt de dronevlucht op uit de databank
3. Het systeem toont de kaart met de dronevlucht

**Naam**

Lijst met piloten bekijken

**Doelstelling**

Een gebruiker bekijkt een lijst van alle piloten uit de webapplicatie

**Actor**

Interne gebruiker

**Precondities**

De gebruiker is ingelogd

**Successscenario**

1. De gebruiker duidt aan dat hij alle piloten wil bekijken
2. Het systeem haalt de piloten op uit de databank
3. Het systeem toont een lijst met alle piloten

**Naam**

Details piloot bekijken

**Doelstelling**

Een gebruiker bekijkt de details van een piloot uit de webapplicatie

**Actor**

Interne gebruiker

**Precondities**

De gebruiker is ingelogd

**Successscenario**

1. De gebruiker duidt aan welke piloot hij wil bekijken
2. Het systeem haalt de piloot op uit de databank
3. Het systeem toont de details van de piloot
4. De gebruiker geeft aan dat hij het bekijken van de details wil afronden
5. Het systeem geeft een overzicht van alle piloten

**Naam**

Lijst met drones bekijken

**Doelstelling**

Een gebruiker bekijkt een lijst van alle drones uit de webapplicatie

**Actor**

Interne gebruiker

**Precondities**

De gebruiker is ingelogd

**Successscenario**

1. De gebruiker duidt aan dat hij alle drone wil bekijken
2. Het systeem haalt de drones op uit de databank
3. Het systeem toont een lijst met alle drones

**Naam**

Details drone bekijken

**Doelstelling**

Een gebruiker bekijkt de details van een drone uit de webapplicatie

**Actor**

Interne gebruiker

**Precondities**

De gebruiker is ingelogd

**Successscenario**

1. De gebruiker duidt aan welke drone hij wil bekijken
2. Het systeem haalt de drone op uit de databank
3. Het systeem toont de details van de drone
4. De gebruiker geeft aan dat hij het bekijken van de details wil afronden
5. Het systeem geeft een overzicht van alle drones

**Naam**

Data toevoegen aan dronevluchten

**Doelstelling**

Een gebruiker voegt bestanden toe aan een dronevlucht uit de webapplicatie

**Actor**

Interne gebruiker

**Precondities**

De gebruiker is ingelogd

**Postcondities**

De data is opgeslagen in de databank

**Successscenario**

1. De gebruiker selecteert de dronevlucht waaraan hij bestanden wil toevoegen
2. De gebruiker selecteert de bestanden die hij wil toevoegen
3. Het systeem voegt de gegevens uit de bestanden toe aan de databank
4. De gebruiker geeft aan dat hij het toevoegen van bestanden wil afronden
5. Het systeem geeft een overzicht van alle dronevluchten

**Alternatieve scenario's**

2.a. De gebruiker geeft aan dat hij het toevoegen van bestanden wil afbreken, ga naar stap 5

**Naam**

Dronevlucht toevoegen

**Doelstelling**

Een gebruiker voegt een dronevlucht toe aan de webapplicatie

**Actor**

Interne gebruiker

**Precondities**

De gebruiker is ingelogd

**Postcondities**

De dronevlucht is opgeslagen in de databank

**Successscenario**

1. De gebruiker geeft aan dat hij een dronevlucht wil toevoegen
2. De gebruiker voert de gegevens van de dronevlucht in
3. De gebruiker geeft aan dat hij het toevoegen van de dronevlucht wil afronden
4. Het systeem voegt de dronevlucht toe aan de databank
5. Het systeem geeft een overzicht van alle dronevluchten

**Alternatieve scenario's**

3.a. De gebruiker geeft aan dat hij het toevoegen van de dronevlucht wil afbreken, ga naar stap 5

**Naam**

Dronevlucht wijzigen

**Doelstelling**

Een gebruiker wijzigt een dronevlucht uit de webapplicatie

**Actor**

Interne gebruiker

**Precondities**

De gebruiker is ingelogd

**Postcondities**

De dronevlucht is gewijzigd in de databank

**Successscenario**

1. De gebruiker duidt aan welke dronevlucht hij wil wijzigen
2. De gebruiker wijzigt de gegevens van de dronevlucht
3. De gebruiker geeft aan dat hij het wijzigen van de dronevlucht wil afronden
4. Het systeem wijzigt de dronevlucht in de databank
5. Het systeem geeft een overzicht van alle dronevluchten

**Alternatieve scenario's**

3.a. De gebruiker geeft aan dat hij het wijzigen van de dronevlucht wil afbreken, ga naar stap 5



**Naam**

Dronevlucht verwijderen

**Doelstelling**

Een gebruiker verwijdert een dronevlucht uit de webapplicatie

**Actor**

Interne gebruiker

**Precondities**

De gebruiker is ingelogd

**Postcondities**

De dronevlucht is verwijderd uit de databank

**Successscenario**

1. De gebruiker duidt aan welke dronevlucht hij wil verwijderen
2. De gebruiker bevestigt dat hij deze dronevlucht wil verwijderen
3. Het systeem verwijdert de dronevlucht uit de databank
4. Het systeem geeft een overzicht van de overige dronevluchten

**Alternatieve scenario's**

2.a. De gebruiker geeft aan dat hij het verwijderen van de dronevlucht wil afbreken, ga naar stap 4

**Naam**

Project toevoegen

**Doelstelling**

Een gebruiker voegt een project toe aan de webapplicatie

**Actor**

Administrator

**Precondities**

De gebruiker is ingelogd als administrator

**Postcondities**

Het project is opgeslagen in de databank

**Successscenario**

1. De gebruiker geeft aan dat hij een project wil toevoegen
2. De gebruiker voert de gegevens van het project in
3. De gebruiker geeft aan dat hij het toevoegen van het project wil afronden
4. Het systeem voegt het project toe aan de databank
5. Het systeem geeft een overzicht van alle projecten

**Alternatieve scenario's**

3.a. De gebruiker geeft aan dat hij het toevoegen van het project wil afbreken, ga naar stap 5

**Naam**

Drone toevoegen

**Doelstelling**

Een gebruiker voegt een drone toe aan de webapplicatie

**Actor**

Administrator

**Precondities**

De gebruiker is ingelogd als administrator

**Postcondities**

De drone is opgeslagen in de databank

**Successscenario**

1. De gebruiker geeft aan dat hij een drone wil toevoegen
2. De gebruiker voert de gegevens van de drone in
3. De gebruiker geeft aan dat hij het toevoegen van de drone wil afronden
4. Het systeem voegt de drone toe aan de databank
5. Het systeem geeft een overzicht van alle drones

**Alternatieve scenario's**

3.a. De gebruiker geeft aan dat hij het toevoegen van de drone wil afbreken, ga naar stap 5

**Naam**

Piloot toevoegen

**Doelstelling**

Een gebruiker voegt een piloot toe aan de webapplicatie

**Actor**

Administrator

**Precondities**

De gebruiker is ingelogd als administrator

**Postcondities**

De piloot is opgeslagen in de databank

**Successscenario**

1. De gebruiker geeft aan dat hij een piloot wil toevoegen
2. De gebruiker voert de gegevens van de piloot in
3. De gebruiker geeft aan dat hij het toevoegen van de piloot wil afronden
4. Het systeem voegt de piloot toe aan de databank
5. Het systeem geeft een overzicht van alle piloten

**Alternatieve scenario's**

3.a. De gebruiker geeft aan dat hij het toevoegen van de piloot wil afbreken, ga naar stap 5

**Naam**

Project wijzigen

**Doelstelling**

Een gebruiker wijzigt een project uit de webapplicatie

**Actor**

Administrator

**Precondities**

De gebruiker is ingelogd als administrator

**Postcondities**

Het project is gewijzigd in de databank

**Successscenario**

1. De gebruiker duidt aan welke project hij wil wijzigen
2. De gebruiker wijzigt de gegevens van het project
3. De gebruiker geeft aan dat hij het wijzigen van het project wil afronden
4. Het systeem wijzigt het project in de databank
5. Het systeem geeft een overzicht van alle projecten

**Alternatieve scenario's**

3.a. De gebruiker geeft aan dat hij het wijzigen van het project wil afbreken, ga naar stap 5

**Naam**

Drone wijzigen

**Doelstelling**

Een gebruiker wijzigt een drone uit de webapplicatie

**Actor**

Administrator

**Precondities**

De gebruiker is ingelogd als administrator

**Postcondities**

De drone is gewijzigd in de databank

**Successscenario**

1. De gebruiker duidt aan welke drone hij wil wijzigen
2. De gebruiker wijzigt de gegevens van de drone
3. De gebruiker geeft aan dat hij het wijzigen van de drone wil afronden
4. Het systeem wijzigt de drone in de databank
5. Het systeem geeft een overzicht van alle drones

**Alternatieve scenario's**

3.a. De gebruiker geeft aan dat hij het wijzigen van de drone wil afbreken, ga naar stap 5

**Naam**

Piloot wijzigen

**Doelstelling**

Een gebruiker wijzigt een piloot uit de webapplicatie

**Actor**

Administrator

**Precondities**

De gebruiker is ingelogd als administrator

**Postcondities**

De piloot is gewijzigd in de databank

**Successscenario**

1. De gebruiker duidt aan welke piloot hij wil wijzigen
2. De gebruiker wijzigt de gegevens van de piloot
3. De gebruiker geeft aan dat hij het wijzigen van de piloot wil afronden
4. Het systeem wijzigt de piloot in de databank
5. Het systeem geeft een overzicht van alle piloten

**Alternatieve scenario's**

3.a. De gebruiker geeft aan dat hij het wijzigen van de piloot wil afbreken, ga naar stap 5

**Naam**

Project verwijderen

**Doelstelling**

Een gebruiker verwijdt een project uit de webapplicatie

**Actor**

Administrator

**Precondities**

De gebruiker is ingelogd als administrator

**Postcondities**

Het project is verwijderd uit de databank

**Successscenario**

1. De gebruiker duidt aan welke project hij wil verwijderen
2. De gebruiker bevestigt dat hij dit project wil verwijderen
3. Het systeem verwijdt het project uit de databank
4. Het systeem geeft een overzicht van de overige projecten

**Alternatieve scenario's**

2.a. De gebruiker geeft aan dat hij het verwijderen van het project wil afbreken, ga naar stap

4



**Naam**

Drone verwijderen

**Doelstelling**

Een gebruiker verwijdert een drone uit de webapplicatie

**Actor**

Administrator

**Precondities**

De gebruiker is ingelogd als administrator

**Postcondities**

De drone is verwijderd uit de databank

**Successscenario**

1. De gebruiker duidt aan welke drone hij wil verwijderen
2. De gebruiker bevestigt dat hij deze drone wil verwijderen
3. Het systeem verwijdert de drone uit de databank
4. Het systeem geeft een overzicht van de overige drones

**Alternatieve scenario's**

2.a. De gebruiker geeft aan dat hij het verwijderen van de drone wil afbreken, ga naar stap 4

**Naam**

Piloot verwijderen

**Doelstelling**

Een gebruiker verwijdert een piloot uit de webapplicatie

**Actor**

Administrator

**Precondities**

De gebruiker is ingelogd als administrator

**Postcondities**

De piloot is verwijderd uit de databank

**Successscenario**

1. De gebruiker duidt aan welke piloot hij wil verwijderen
2. De gebruiker bevestigt dat hij deze piloot wil verwijderen
3. Het systeem verwijdert de piloot uit de databank
4. Het systeem geeft een overzicht van de overige piloten

**Alternatieve scenario's**

2.a. De gebruiker geeft aan dat hij het verwijderen van de piloot wil afbreken, ga naar stap 4

**Naam**

Interne gebruiker registreren

**Doelstelling**

Een administrator voegt een nieuwe gebruiker toe aan het systeem

**Actor**

Administrator

**Precondities**

De gebruiker is ingelogd als administrator

**Postcondities**

De gebruiker is toegevoegd aan de databank

**Successscenario**

1. De administrator geeft aan dat hij een gebruiker wil toevoegen
2. De administrator voert de gegevens van de gebruiker in
3. De administrator geeft aan dat hij het toevoegen van de gebruiker wil afronden
4. Het systeem voegt de gebruiker toe aan de databank
5. Het systeem toont de startpagina

**Alternatieve scenario's**

3.a. De gebruiker geeft aan dat hij het toevoegen van een gebruiker wil afbreken, ga naar stap 5

## Appendix B: Integratietesten

- Bekijken van lijst met projecten

Dit kan simpelweg door naar de 'Projects'-pagina te gaan. Alle aangemaakte projecten worden nu opgehaald uit de databank en in een lijst getoond in de applicatie.

- Details van een project bekijken

Hiervoor moet eerst naar de 'Projects'-pagina gegaan worden, waar een lijst van de aangemaakte projecten getoond wordt. Hier kan de gebruiker bij een bepaald project op de knop 'Details' drukken en hierna worden de details van dat project opgehaald uit de databank en getoond in de *view*.

- Bekijken van lijst met dronevluchten

Dit kan simpelweg door naar de 'Drone Flights'-pagina te gaan. Alle aangemaakte dronevluchten worden nu opgehaald uit de databank en in een lijst getoond in de applicatie.

- Details van een dronevlucht bekijken

Hiervoor moet eerst naar de 'Drone Flights'-pagina gegaan worden, waar een lijst van de aangemaakte dronevluchten getoond wordt. Hier kan de gebruiker bij een bepaalde dronevlucht op de knop 'Details' drukken en hierna worden de details van deze dronevlucht opgehaald uit de databank en getoond in de *view*.

- Kaart met dronevluchten bekijken

Om een bepaalde dronevlucht op kaart te zien moet eerst naar de 'Drone Flights'-pagina gegaan worden. Hier kan bij een bepaalde dronevlucht op het kaart icoontje in de kolom Map geklikt worden. Hierna is de kaart met de dronevlucht en bijbehorende data te zien. In deze test worden de Web API-*controllers* getest. Zo zal bijvoorbeeld de `CTRLPointsController` de nodige data van de controlepunten beschikbaar stellen als *JSON* aan de frontend. Hierna wordt deze data opgehaald door een *AJAX-call* en is deze zichtbaar op de kaart. Dit gebeurt analoog voor de andere Web API-*controllers*.

- Data toevoegen aan dronevluchten

Voor de gebruiker bestanden kan toevoegen aan een dronevlucht, moet deze ingelogd zijn. Hierna gaat de gebruiker naar de 'Drone Flights'-pagina, waar er op het *upload files* icoontje in de Upload kolom kan geklikt worden. Hierna komt de gebruiker op de 'File Upload'-pagina, waar bestanden kunnen geselecteerd worden om toe te voegen. Na het

toevoegen is bij de dronevlucht te zien welke bestanden zijn toegevoegd, deze zijn namelijk groen aangeduid in de Files kolom. In de databank is ook te zien dat de gegevens van deze bestanden correct zijn toegevoegd.

- Bekijken van lijst met piloten

Hiervoor moet de gebruiker eerst ingelogd zijn, hierna kan deze naar de 'Pilots'-pagina gaan. Alle aangemaakte piloten worden nu opgehaald uit de databank en in een lijst getoond in de applicatie.

- Details van een piloot bekijken

Hiervoor moet de gebruiker eerst ingelogd zijn, hierna kan deze naar de 'Pilots'-pagina gaan. Hier wordt nu een lijst van alle aangemaakte piloten getoond. Na het drukken op de 'Details' knop bij een bepaalde piloot, worden de details van die piloot opgehaald uit de databank en getoond in de *view*.

- Bekijken van lijst met drones

Hiervoor moet de gebruiker eerst ingelogd zijn, hierna kan deze naar de 'Drones'-pagina gaan. Alle aangemaakte drones worden nu opgehaald uit de databank en in een lijst getoond in de applicatie.

- Details van een drone bekijken

Hiervoor moet de gebruiker eerst ingelogd zijn, hierna kan deze naar de 'Drones'-pagina gaan. Hier wordt nu een lijst van alle aangemaakte drones getoond. Na het drukken op de 'Details' knop bij een bepaalde drone, worden de details van die drone opgehaald uit de databank en getoond in de *view*.

- Project toevoegen

Dit kan alleen gedaan worden door iemand die als admin is ingelogd. Op de 'Projects'-pagina kan er op *create new project* geklikt worden. Eenmaal alle benodigde info is toegevoegd, is het nieuwe project te zien in het overzicht van projecten.

- Piloot toevoegen

Dit kan alleen gedaan worden door iemand die als admin is ingelogd. Op de 'Pilots'-pagina kan er op *create new pilot* geklikt worden. Eenmaal alle benodigde info is toegevoegd, is de nieuwe piloot te zien in het overzicht van piloten.

- Drone toevoegen

Dit kan alleen gedaan worden door iemand die als admin is ingelogd. Op de 'Drones'-pagina kan er op *create new drone* geklikt worden. Eenmaal alle benodigde info is toegevoegd, is de nieuwe drone te zien in het overzicht van drones.

- Dronevlucht toevoegen

Dit kan alleen gedaan worden door iemand die is ingelogd. Vooraleer er een dronevlucht aangemaakt kan worden, moet er eerst een project, piloot en drone aangemaakt zijn. Op de 'Drone Flights'-pagina kan er op *create new drone flight* geklikt worden. Eenmaal alle benodigde info is toegevoegd, is de nieuwe dronevlucht te zien in het overzicht van dronevluchten.

- Project verwijderen

Dit kan alleen gedaan worden door iemand die als admin is ingelogd. Op de 'Projects'-pagina wordt op *delete* bij het te verwijderen project geklikt. Hierna wordt bevestiging om te verwijderen gevraagd. Nu is het project en de bijhorende dronevluchten verwijderd uit de databank.

- Piloot verwijderen

Dit kan alleen gedaan worden door iemand die als admin is ingelogd. Vooraleer een piloot verwijderd kan worden, moeten eerst al zijn toegewezen dronevluchten verwijderd worden. Hierna kan op de 'Pilots'-pagina op *delete* geklikt worden bij de te verwijderen piloot. Na de bevestiging om de piloot te verwijderen, wordt deze effectief verwijderd uit de databank.

- Drone verwijderen

Dit kan alleen gedaan worden door iemand die als admin is ingelogd. Vooraleer een drone verwijderd kan worden, moeten eerst al zijn toegewezen dronevluchten verwijderd worden. Hierna kan op de 'Drones'-pagina op *delete* geklikt worden bij de te verwijderen drone. Na de bevestiging om de drone te verwijderen, wordt deze effectief verwijderd uit de databank.

- Dronevlucht verwijderen

Dit kan alleen gedaan worden door iemand die is ingelogd. Op de 'Drone Flights'-pagina klikt de admin op *delete* bij de te verwijderen dronevlucht. Hierna wordt bevestiging om te verwijderen gevraagd. Nu is de dronevlucht verwijderd uit de databank.

- Project wijzigen

Dit kan alleen gedaan worden door iemand die als admin is ingelogd. Op de 'Projects'-pagina klikt de admin op *edit* bij het te wijzigen project. Na de gegevens te wijzigen wordt op *save* geklikt en is het project aangepast. Bij de details is te zien dat de gegevens ook effectief aangepast werden.

- Piloot wijzigen

Dit kan alleen gedaan worden door iemand die als admin is ingelogd. Op de 'Pilots'-pagina klikt de admin op *edit* bij de te wijzigen piloot. Na de gegevens te wijzigen wordt op *save* geklikt en is de piloot aangepast. Bij de details is te zien dat de gegevens ook effectief aangepast werden.

- Drone wijzigen

Dit kan alleen gedaan worden door iemand die als admin is ingelogd. Op de 'Drones'-pagina klikt de admin op *edit* bij de te wijzigen drone. Na de gegevens te wijzigen wordt op *save* geklikt en is de drone aangepast. Bij de details is te zien dat de gegevens ook effectief aangepast werden.

- Dronevlucht wijzigen

Dit kan alleen gedaan worden door iemand die is ingelogd. Op de 'Drone Flights'-pagina klikt de admin op *edit* bij de te wijzigen dronevlucht. Na de gegevens te wijzigen wordt op *save* geklikt en is de dronevlucht aangepast. Bij de details is te zien dat de gegevens ook effectief aangepast werden.

- Gebruiker registreren

Dit kan alleen gedaan worden door iemand die als admin is ingelogd. Hierna gaat de administrator naar de 'Register'-pagina waar de gegevens en de rol voor de nieuwe gebruiker kunnen ingevuld worden. Eenmaal de gebruiker geregistreerd is kan deze zich inloggen in de applicatie. In de databank is ook te zien dat er een nieuwe gebruiker is toegevoegd en dat er een bepaalde rol is toegekend aan deze gebruiker.