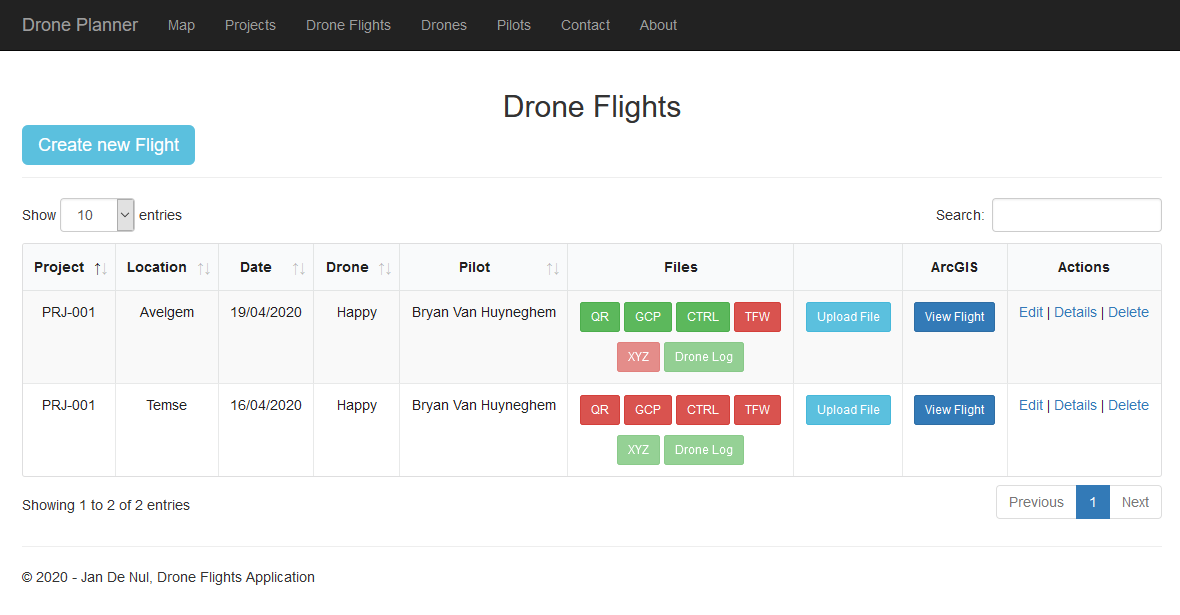


Bachelorproef: eindverslag

Droneplanning-tool



<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

Inhoudsopgave

[Lijst van figuren 4](#_Toc39750519)

[Inleiding 5](#_Toc39750520)

[1. Gebruikersaspecten 8](#_Toc39750521)

[1.1 Databank 8](#_Toc39750522)

[1.2 Webapplicatie 8](#_Toc39750523)

[1.3 Use case-diagrammen 9](#_Toc39750524)

[1.4 Featurelijst 13](#_Toc39750525)

[1.4.1 Sprint 1 13](#_Toc39750526)

[1.4.2 Sprint 2 14](#_Toc39750527)

[1.4.3 Sprint 3 14](#_Toc39750528)

[2. Systeemarchitectuur 15](#_Toc39750529)

[2.1 High-levelsysteemmodel (deployment diagram) 15](#_Toc39750530)

[2.2 Databankdiagram 16](#_Toc39750531)

[2.2.1 DroneFlighttabel 16](#_Toc39750532)

[2.2.2 QualityReporttabel 19](#_Toc39750533)

[2.2.3 DroneLogEntrytabel 20](#_Toc39750534)

[2.3 MVC 21](#_Toc39750535)

[2.4 Klassendiagrammen 22](#_Toc39750536)

[2.4.1 Simple Factory pattern voor parser-klassen 22](#_Toc39750537)

[2.5 Javascriptklassen 23](#_Toc39750538)

[2.5.1 Uploaden van files 23](#_Toc39750539)

[2.5.2 Visualisatie met ArcGIS JS API 29](#_Toc39750540)

[2.4.3 Data Tables 33](#_Toc39750541)

[3. Testplan 33](#_Toc39750542)

[3.1 Web API Controllers 33](#_Toc39750543)

[3.2 “View” Controllers 34](#_Toc39750544)

[4. Evaluaties en discussies 34](#_Toc39750545)

[5. Handleidingen 35](#_Toc39750546)

[5.1 Installatiehandleiding 35](#_Toc39750547)

[5.1.1 Vereiste software 35](#_Toc39750548)

[5.1.2 Aanmaken van de databank 36](#_Toc39750549)

[5.1.3 Opstarten van de webapplicatie 37](#_Toc39750550)

[5.2 Gebruikershandleiding 38](#_Toc39750551)

[Referenties 39](#_Toc39750552)

[Appendix A: use case-diagrammen stories 40](#_Toc39750553)

# Lijst van figuren

[Figuur 1: Toevoegen en verwijderen van entiteiten in het systeem 10](#_Toc38319351)

[Figuur 2: Wijzigen en details bekijken van entiteiten in het systeem 11](#_Toc38319352)

[Figuur 3: Deployment diagram 14](file:///S:\Documents\Universiteit%20Gent\Jaar%203\Semester%20VI\Bachelorproef\2020\drone1\verslag\verslag-sprint-2-drone1.docx#_Toc38319353)

[Figuur 4: DroneFlighttabel met al haar relaties 16](file:///S:\Documents\Universiteit%20Gent\Jaar%203\Semester%20VI\Bachelorproef\2020\drone1\verslag\verslag-sprint-2-drone1.docx#_Toc38319354)

[Figuur 5: QualityReporttabel met al haar relaties 18](file:///S:\Documents\Universiteit%20Gent\Jaar%203\Semester%20VI\Bachelorproef\2020\drone1\verslag\verslag-sprint-2-drone1.docx#_Toc38319355)

[Figuur 6: DroneLogEntrytabel met al haar relaties 19](#_Toc38319356)

[Figuur 7: Detailpagina van een Drone Flight 20](file:///S:\Documents\Universiteit%20Gent\Jaar%203\Semester%20VI\Bachelorproef\2020\drone1\verslag\verslag-sprint-2-drone1.docx#_Toc38319357)

[Figuur 8: Simple factory pattern voor parsers 21](#_Toc38319358)

[Figuur 9: De map view 22](#_Toc38319359)

[Figuur 10: LayerList widget 23](#_Toc38319360)

[Figuur 11: Search widget 23](#_Toc38319361)

[Figuur 12: Legend klasse 23](#_Toc38319362)

[Figuur 13: PopupTemplate 24](#_Toc38319363)

[Figuur 14: AreaMeasurement2D klasse 24](#_Toc38319364)

[Figuur 15: Feature klasse 25](#_Toc38319365)

# 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*. Dit softwarepakket bevat uitgebreide opties aan programmeertaalkeuzes en een ruim aanbod aan visualisaties.

Op basis van de informatie die Jan De Nul verstrekte, werden enkele 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.

1. 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.
2. 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.
3. 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.

# 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 pilot 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.

## 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.

## 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 gespecifieerde 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.

## Use case-diagrammen

Het gebruik van de webapplicatie wordt verduidelijkt in **Figuren 1 en 2** op pagina’s 10 en 11 via twee use case-diagrammen. Hierop is te zien hoe de actor (de gebruiker) kan interageren met de verschillende functionaliteiten van het systeem. Een gebruiker beschikt over CRUD (*Create, Read, Update, Delete)* functionaliteiten.

Een ingelogde gebruiker kan een dronevlucht, drone of piloot toevoegen (**Figuur 1**) aan het systeem (de databank) en onmiddellijk reeds enkele beschrijvende gegevens toevoegen aan deze entiteiten. Deze gegevens worden opgeslagen in de databank.

Een ingelogde gebruiker kan ook een dronevlucht, drone of piloot verwijderen (**Figuur 1**) uit het systeem. Verder kan een gebruiker de bijhorende en beschrijvende informatie van dronevluchten, drones of piloten wijzigen (**Figuur 2**). Deze aanpassingen hebben onmiddellijk effect op de databank aan server-sidekant.

Een gebruiker die *niet* ingelogd is, kan geen gegevens toevoegen, aanpassen of verwijderen. Hij is bovendien ook beperkt in het bekijken van informatie: hij kan enkel dronevluchten bekijken. Drone-informatie en pilootinformatie zijn aldus niet toegankelijk voor deze gebruiker. Een ingelogde gebruiker, daarentegen, kan *wel* alle informatie bekijken (**Figuur 2**).

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



Figuur 1: Toevoegen en verwijderen van entiteiten in het systeem



Figuur 2: Wijzigen en details bekijken van entiteiten in het systeem

## Featurelijst

Hieronder volgt een opsomming van alle features in dit project, inclusief hun complexiteit[[1]](#footnote-1) en prioriteit[[2]](#footnote-2), per sprint.

## 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).

## Sprint 2

* Projecttabelin 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).

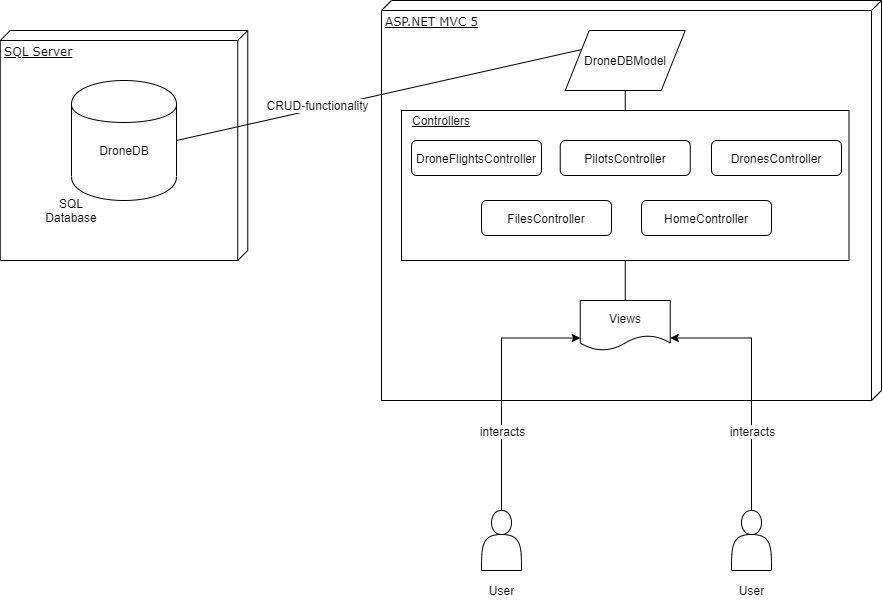
## Sprint 3

# Systeemarchitectuur

In dit hoofdstuk wordt besproken hoe het programma doelstellingen 1, 2, 3 en 4 realiseert en hoe deze opgebouwd zijn. Aan de hand van een databankdiagram en een klassendiagram voor de parser-klassen zal de opbouw van de webapplicatie beschreven worden en getoond worden op welke manier de verschillende onderdelen samenwerken.

## High-levelsysteemmodel (deployment diagram)

Het deployment diagram in **Figuur 3** 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.

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.

Figuur 3: Deployment diagram

## 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 drie grote onderdelen: DroneFlight, QualityReport en DroneLogEntry. Deze zijn respectievelijk te zien in **Figuren 3**, **4** en **5** op pagina’s 15, 17 en 18.

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 oneindigheidsteken 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.

## DroneFlighttabel

De eerste en tevens ook belangrijkste tabel in het databankdiagram is de DroneFlight tabel, te zien op **Figuur 4** op pagina 16.



Figuur 4: 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 en hasDroneLog) 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 RawImages tabel geografisch juist te positioneren (Understanding world files, z.j.). Zodoende kunnen alle foto’s aan elkaar gehangen worden om zo een groot beeld te verkregen van de werf.

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 DroneAttributeValues 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 QualityReport tabel.

## QualityReporttabel

In de QualityReporttabel in **Figuur 5** 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 5: QualityReporttabel met al haar relaties

## DroneLogEntrytabel

In **Figuur 6** staat de DroneLogEntrytabel centraal. Deze tabel bevat voor elke *tick* data ingelezen uit het dronelogbestand. Alle tabellen (DroneGPS, DroneOA, DroneIMU\_ATTI, 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 6: DroneLogEntrytabel met al haar relaties

## MVC

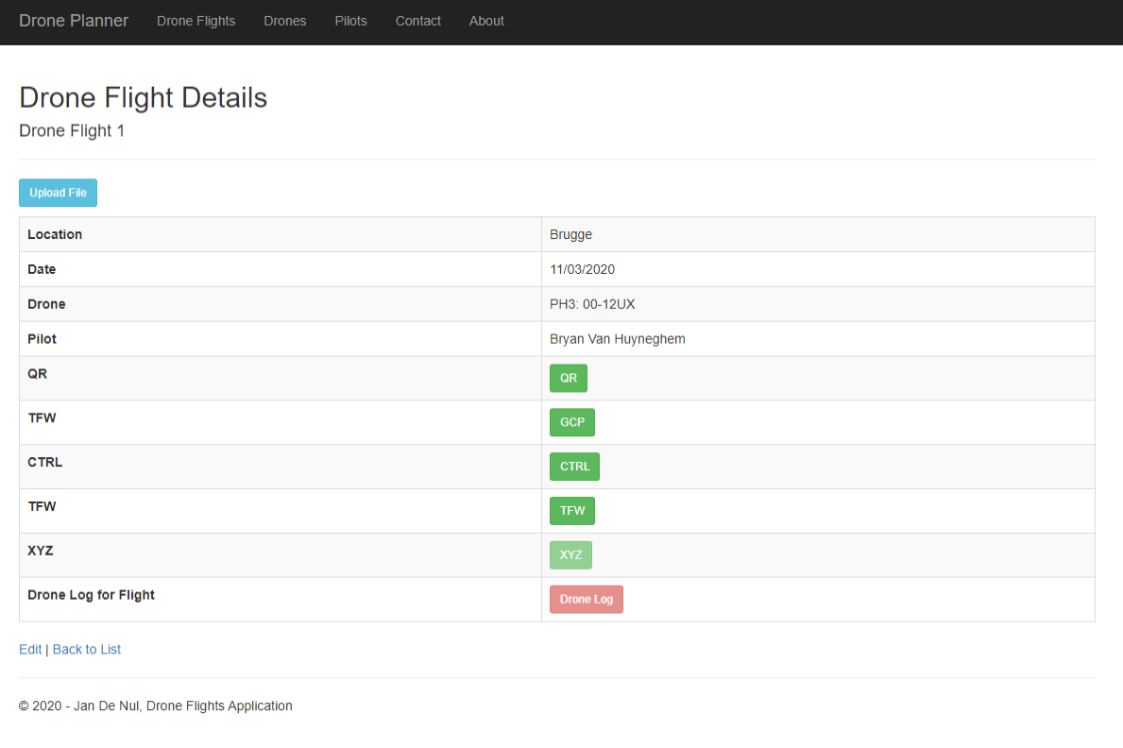
Todo: add controllers

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*.

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. 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.

Vervolgens worden enkele *Controllers* (de C in het MVC pattern) aangemaakt die instaan voor navigatie (routing): de DroneFlightsController, de DronesController, de PilotsController, de FilesController en de HomeController. De drie 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).

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 7** 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.



Figuur 7: Detailpagina van een Drone Flight

## Klassendiagrammen

In dit onderdeel worden de klassendiagrammen beschreven die gebruikt werden in deze applicatie: een *simple factory pattern* voor het parsen van de bestanden. + controllers?

## Simple Factory pattern voor parser-klassen

Het parsen van data uit bestanden gebeurt via parser-klassen. Deze klassen worden gegroepeerd in een *simple factory pattern* (**Figuur 8**), 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 8: Simple factory pattern voor parsers

## 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 (2.5.1), om de vluchten te visualiseren (2.5.2) en functionaliteit te voorzien om zogenaamde *Data Tables* te kunnen aanmaken (2.5.3).

## 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 moet worden opgeslagen in de databank na het verwerken 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 x). Aan server-side wordt er nogmaals gecontroleerd of er weldegelijk een bestand werd meegegeven; dit als dubbele controle.



* Het controleert de toegestane bestandsextensies, zoals weergegeven in Figuur x. 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.



* 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 x, aangezien de gebruiker reeds in een tweede rond van uploaden kan zitten, zonder de pagina te *refreshen*.



In Figuur x 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.

* + uploadProgress verzorgt het eigenlijke uploaden van de bestanden (zie Figuur x). 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().

De functie startParsing() (zie Figuur x, x en x) reset de waarde van de progress bar terug naar 0 en roept met 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 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.







* + 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 x.



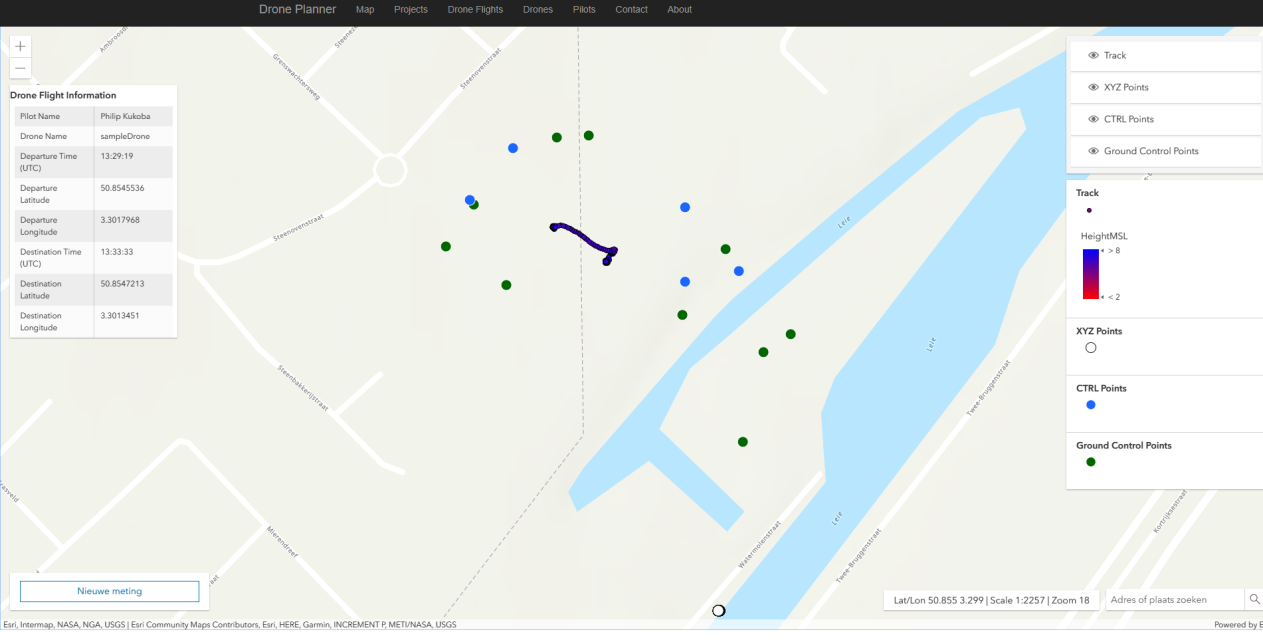
* Het tonen van een *progress bar* voor het uploaden en het parsen van de bestanden. Er wordt periodiek gecommuniceerd met de server-side of een status op te halen van het parsen. Op deze manier kan aan de gebruiker een voortgang worden getoond van het parsen van zijn bestanden.
* Indien er iets misloopt 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*.
* Een 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 geüpload/proberen uploaden.

## Visualisatie met ArcGIS JS API

De front-end bestaat uit verschillende klassen uit de ArcGIS Javascript API.



Figuur 9: De map view

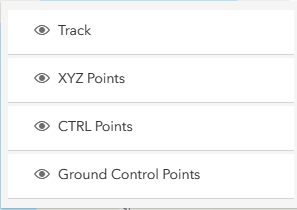
**Map**:De Map klasse is een container voor de *layers* van de applicatie en heeft bijhorende methodes voor deze *layers*.

**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 gespecifieerd worden met behulp van het *well-known ID (WKID)* veld van de SpatialReference klasse. Voor dit project wordt het coordinatensysteem "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 heeft een referentie nodig naar de MapView.



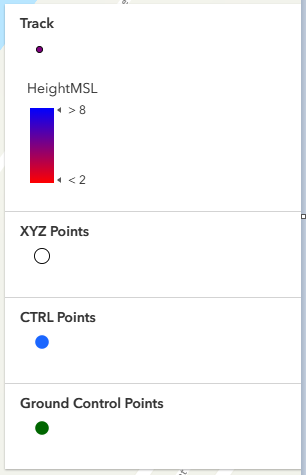
Figuur 10: LayerList widget

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

****

Figuur 11: Search widget

**Legend**: Toont een legende 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 12: Legend klasse

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



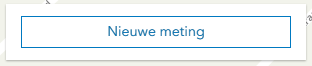
Figuur 13: PopupTemplate

**FeatureLayer:** Een FeatureLayer stelt éen 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).*

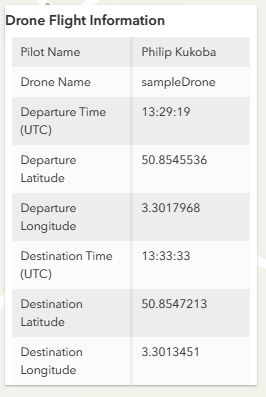
Als laatste moet het *objectIdField* ingesteld worden op het unieke ID attribuut van de Graphic objecten.

**AreaMeasurement2D:** Een widget waarbij een oppervlakte van eender welke vorm opgemeten kan worden met opties voor verschillende eenheden.

**

Figuur 14: AreaMeasurement2D klasse

**Feature:** Deze *widget* 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 15: Feature klasse

## Data Tables

Deze tabellen laten *paging*, *searching* en *sorting* toe op gewone HTML-tabellen.

# Testplan

Een mock-databank moeten worden gemaakt om Unit en Integration tests te kunnen uitvoeren, zodat de controllers getest kunnen worden. Dit werd geprobeerd met Moq (een *mocking framework* in C#), maar dit lukte, helaas, niet.

## Web API Controllers

De Web API Controllers, op basis van een Flight ID, geven een JSON object terug met de opgevraagde data uit de databank. Hierbij moeten drie voorwaarden gecontroleerd worden:

1) Als een vlucht niet bestaat in de databank, moet de controller HttpResponseMessage 404 teruggeven.

2) De *data projection* van een C#-klasse naar JSON moet juist gebeuren. Er moet gecontroleerd worden op de juiste waarden van de attributen van de teruggegeven data.

3) De JSON moet alle opgevraagde data bevatten.

## “View” Controllers

1) Bij de gewone controllers moet de Index() methode getest worden op het teruggeven van een View met een dependancy 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.

# Evaluaties en discussies

(sprint 3)

# Handleidingen

## 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;
* Dit **verslag** met handleidingen en documentatie.

## 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.

## 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](https://www.microsoft.com/en-us/sql-server/sql-server-downloads) downloaden op de website van Microsoft.
   1. Scroll naar beneden en kies de versie die u verkiest.
   2. Klik “Download now”. Het programma downloadt.
   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](https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?redirectedfrom=MSDN&view=sql-server-ver15) downloaden.
   1. Klik op “Download SQL Server Management Studio (SSMS)”. Het programma downloadt.
   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](https://visualstudio.microsoft.com/vs/) downloaden.
   1. Klik op “Download Visual Studio” en kiest de Community 2019 of de Professional 2019 versie naargelang uw eigen voorkeur. Het programma downloadt.
   2. Volg na het uitvoeren van het bestand de instructies op het scherm.
   3. Kies tijdens de installatie om de volgende “Workloads*”* te installeren: “ASP.NET and web development*”* en “Data storage and processing*”*.
4. Surf naar [deze](http://www.ivytools.net/downloads.html) website.
   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”.
   2. Kopieer deze sleutel.
   3. Navigeer in het ZIP-bestand dat u downloadde in 5.1.1 naar de map drone1/**IvyPdf\_1.62**.
   4. Voer het bestand “**IvyTemplateEditor.exe***”* uit.
   5. Navigeer via de balk bovenaan het programma naar “Help>About>Apply License Code”.
   6. Plak de gekopieerde sleutel in het veld en drukt op OK.
   7. Sluit *“*IvyTemplateEditor*”* af.
   8. U heeft nu toegang tot de IvyParser dll-bestanden in de webapplicatie. Deze worden gebruikt bij het inlezen van een PDF-bestand.

## 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. Het veld “*Server name”* wordt automatisch ingevuld.
   2. Noteer deze naam, want u heeft deze later nodig in een volgend deel (opstarten van de webapplicatie).
   3. Klik op “*Connect”*.
2. Klik op het “File*”* menu bovenaan links.
3. Kies “Open>File”.
4. Navigeer 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*”.
7. Klik in het “Object Explorer”-venster op “refresh*”*.
8. Vouw de Machinenaammap en Databasesmap open. Hierin bevindt zich nu de nieuwe database “**DroneDB**”. Merk op dat de Machinenaammap dezelfde naam heeft als de eerder genoteerde “*Server name”*.
9. Een lege databank is nu aangemaakt en klaar voor gebruik.
10. Sluit SQL Server Management Studio.

## Opstarten van de webapplicatie

In dit deel start u de webapplicatie op.

1. Start **Visual Studio 2019**.
2. Navigeer naar de DroneWebApp*-*solution **DroneWebApp.sln** en open deze.
3. Navigeer in het project naar de map “Models”.
4. Klik rechts op de map “Models” en selecteer “Add>New Item”.
5. Zoek in de zoekbalk bovenaan rechts naar “**model**”.
6. Er verschijnt een *project item* genaamd ADO.NET Entity Data Model. Klik deze eenmalig om hem te selecteren.
7. Geef dit ADO.NET Entity Data Model de naam DroneDBModel en klik op “Add*”*.
8. Een nieuw venster verschijnt. Selecteer “**EF Designer from database***”* en klik op “Next>”.
9. Er verschijnt een nieuw venster. Klik op “New Connection”.
10. Controleer of “Data source*”* van het type “Microsoft SQL Server (SqlClient)” is.
    1. Indien dit niet het geval is, klik dan op “Change*”* en selecteer daar “Microsoft SQL Server (SqlClient)”.
    2. Klik op “OK”.
11. Klik op “Refresh” en wacht tot de lijst van servers opgehaald is.
12. De naam van uw computer verschijnt. Dit is dezelfde naam als de “*Server name”* die u noteerde tijdens het aanmaken van de databank.
    1. Indien er geen naam verschijnt, dan kan u op het pijltje naar beneden klikken om uw computer uit de lijst te selecteren.
       1. Indien deze lijst leeg is, geef dan uw computernaam (*server name*) manueel in.
13. In de sectie “Connect to a database*”*, onder “Select or enter a database name*”*, selecteert u “**DroneDB**”. Klik op OK.
14. Controleer of “Save connection settings in Web.Config as*”* aanvinkt is en “**DroneDBEntities”** heet.
15. Klik op “Next>”.
16. Selecteer “Tables” in het veld “Which database objects do you want to include in your model?.
17. Vink “Pluralize or singularize generated object names*”* aan.
18. Vink “Include foreign key columns in the model” aan.
19. Klik op “Finish*”*.
20. Na enige tijd zijn de modelklassen van de databank aangemaakt.
21. Voer de webapplicatie uit met **F5**.
22. De allereerste keer kan een venster verschijnen dat u vraagt om het “IIS Express SSL certificate” te vertrouwen.
    1. Klik “yes”.
23. Er verschijnt een “security warning*”*.
    1. Klik “yes*”*.
24. U kunt nu aan de slag met de dronewebapplicatie.

## Gebruikershandleiding

Deze sectie wordt pas gepubliceerd op het einde van sprint 3.

## Problemen die zich kunnen voorzien

(klad)

Problemen; connectiestring:

1. Navigeer in de “Solution Explorer” van het project naar het bestand **Web.config**, helemaal onderaan de mappenstructuur.
   1. Dubbelklik op dit bestand om het te openen.
2. Indien er een <connectionStrings> tag in het bestand helemaal onderaan zit, dan moet u deze tag en zijn inhoud verwijderen.
3. Sla dit bestand op (*Save*) en sluit het.

Problemen: edmx:

* 1. Indien er een DroneDBModel.edmx aanwezig is, dan verwijdert u deze.
     1. U klikt hiervoor rechts op DroneDBModel.edmx > Delete > OK.

Problemen: IvyTools

1. U heeft nu toegang tot de IvyParser dll-bestanden in de webapplicatie. Deze worden gebruikt bij het inlezen van een PDF-bestand.

Problemen aanmaken db:

1. In het “Messages*”*-venster verschijnt “*Commands completed successfully*”.

**TODO Vragen prof**:

* deployen
* ik moet ook nog een connection string aanpassen voor ado.net in web.config, hoe doen als dll ofzo?

# Referenties

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

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

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

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

Understanding world files. (z.j.). Geraadpleegd op 13 februari 2020 via <http://webhelp.esri.com/arcims/9.3/General/topics/author_world_files.htm>

# Appendix A: use case-diagrammen stories

**Naam**

Dronevlucht toevoegen

**Doelstelling**

Een gebruiker voegt een dronevlucht toe aan de webapplicatie.

**Actor**

Gebruiker

**Precondities**

De gebruiker is ingelogd.

**Postcondities**

De dronevlucht is opgeslagen in de databank

**Successcenario**

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 de dronevluchten.

**Alternatieve scenario’s**

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

**Naam**

Drone toevoegen

**Doelstelling**

Een gebruiker voegt een drone toe aan de webapplicatie.

**Actor**

Gebruiker

**Precondities**

De gebruiker is ingelogd.

**Postcondities**

De drone is opgeslagen in de databank.

**Successcenario**

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 de 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**

Gebruiker

**Precondities**

De gebruiker is ingelogd.

**Postcondities**

De piloot is opgeslagen in de databank.

**Successcenario**

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 de piloten.

**Alternatieve scenario’s**

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

**Naam**

Dronevlucht verwijderen

**Doelstelling**

Een gebruiker verwijdert een dronevlucht uit de webapplicatie.

**Actor**

Gebruiker

**Precondities**

De gebruiker is ingelogd.

**Postcondities**

De dronevlucht is verwijderd uit de databank.

**Successcenario**

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**

Drone verwijderen

**Doelstelling**

Een gebruiker verwijdert een drone uit de webapplicatie.

**Actor**

Gebruiker

**Precondities**

De gebruiker is ingelogd.

**Postcondities**

De drone is verwijderd uit de databank.

**Successcenario**

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**

Gebruiker

**Precondities**

De gebruiker is ingelogd.

**Postcondities**

De piloot is verwijderd uit de databank.

**Successcenario**

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**

Dronevlucht wijzigen

**Doelstelling**

Een gebruiker wijzigt een dronevlucht uit de webapplicatie.

**Actor**

Gebruiker

**Precondities**

De gebruiker is ingelogd.

**Postcondities**

De dronevlucht is gewijzigd in de databank.

**Successcenario**

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 de dronevluchten.

**Alternatieve scenario’s**

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

**Naam**

Drone wijzigen

**Doelstelling**

Een gebruiker wijzigt een drone uit de webapplicatie.

**Actor**

Gebruiker

**Precondities**

De gebruiker is ingelogd.

**Postcondities**

De drone is gewijzigd in de databank.

**Successcenario**

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 de 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**

Gebruiker

**Precondities**

De gebruiker is ingelogd.

**Postcondities**

De piloot is gewijzigd in de databank.

**Successcenario**

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 de piloten.

**Alternatieve scenario’s**

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

**Naam**

Details dronevlucht bekijken

**Doelstelling**

Een gebruiker bekijkt de details van een dronevlucht uit de webapplicatie.

**Actor**

Gebruiker

**Successcenario**

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 de overige dronevluchten.

**Naam**

Details drone bekijken

**Doelstelling**

Een gebruiker bekijkt de details van een drone uit de webapplicatie.

**Actor**

Gebruiker

**Precondities**

De gebruiker is ingelogd.

**Successcenario**

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 de overige drones.

**Naam**

Details piloot bekijken

**Doelstelling**

Een gebruiker bekijkt de details van een piloot uit de webapplicatie.

**Actor**

Gebruiker

**Precondities**

De gebruiker is ingelogd.

**Successcenario**

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 de overige piloten.

1. Makkelijk, vrij gemakkelijk, gemiddeld, vrij moeilijk, moeilijk [↑](#footnote-ref-1)
2. Onbelangrijk, redelijk belangrijk, belangrijk, zeer belangrijk [↑](#footnote-ref-2)