

Droneplanning-tool

Sprint 2

[Drone Planner](#) [Map](#) [Projects](#) [Drone Flights](#) [Drones](#) [Pilots](#) [Contact](#) [About](#)

Drone Flights

Create new Flight

Show 10 entries

Project ↑↓	Location ↑↓	Date ↑↓	Drone ↑↓	Pilot ↑↓	Files		ArcGIS	Actions
PRJ-001	Avelgem	19/04/2020	Happy	Bryan Van Huyneghem	<div>QR GCP CTRL TFW</div> <div>XYZ Drone Log</div>	<div>Upload File</div>	<div>View Flight</div>	<div>Edit Details Delete</div>
PRJ-001	Temse	16/04/2020	Happy	Bryan Van Huyneghem	<div>QR GCP CTRL TFW</div> <div>XYZ Drone Log</div>	<div>Upload File</div>	<div>View Flight</div>	<div>Edit Details Delete</div>

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

Inhoudsopgave

Lijst van figuren	4
Inleiding	5
1. Gebruikersaspecten.....	8
1.1 Databank	8
1.2 Webapplicatie.....	8
1.3 Use case-diagrammen	9
1.4 Featurelijst.....	12
1.4.1 Sprint 1	12
1.4.2 Sprint 2	13
1.4.3 Sprint 3	13
2. Systeemarchitectuur	14
2.1 High-levelsysteemmodel (deployment diagram)	14
2.2 Databankdiagram	15
2.2.1 DroneFlight tabel.....	15
2.2.2 QualityReporttabel.....	18
2.2.3 DroneLogEntrytabel	19
2.3 MVC	20
2.4 Klassendiagrammen	21
2.4.1 Simple Factory pattern voor parserklassen.....	21
2.4.2 Javascriptklassen	22
3. Testplan	26
3.1 Web API Controllers	26
3.2 “View” Controllers.....	26
4. Evaluaties en discussies.....	27
5. Handleidingen	28
5.1 Installatiehandleiding.....	28
5.1.1 Vereiste software	28
5.1.2 Aanmaken van de databank	29
5.1.3 Opstarten van de webapplicatie	29
5.2 Gebruikershandleiding	30
Referenties	31

Appendix A: use case-diagrammen stories	32
---	----

Lijst van figuren

Figuur 1: Toevoegen en verwijderen van entiteiten in het systeem	10
Figuur 2: Wijzigen en details bekijken van entiteiten in het systeem.....	11
Figuur 3: Deployment diagram.....	14
Figuur 4: DroneFlight tabel met al haar relaties	16
Figuur 5: QualityReport tabel met al haar relaties.....	18
Figuur 6: DroneLogEntry tabel met al haar relaties	19
Figuur 7: Detailpagina van een Drone Flight	20
Figuur 8: Simple factory pattern voor parsers	21
Figuur 9: De map view.....	22
Figuur 10: LayerList widget.....	23
Figuur 11: Search widget.....	23
Figuur 12: Legend klasse	23
Figuur 13: PopupTemplate	24
Figuur 14: AreaMeasurement2D klasse	24
Figuur 15: Feature klasse.....	25

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 verstrekke, 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 classes* 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 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.

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

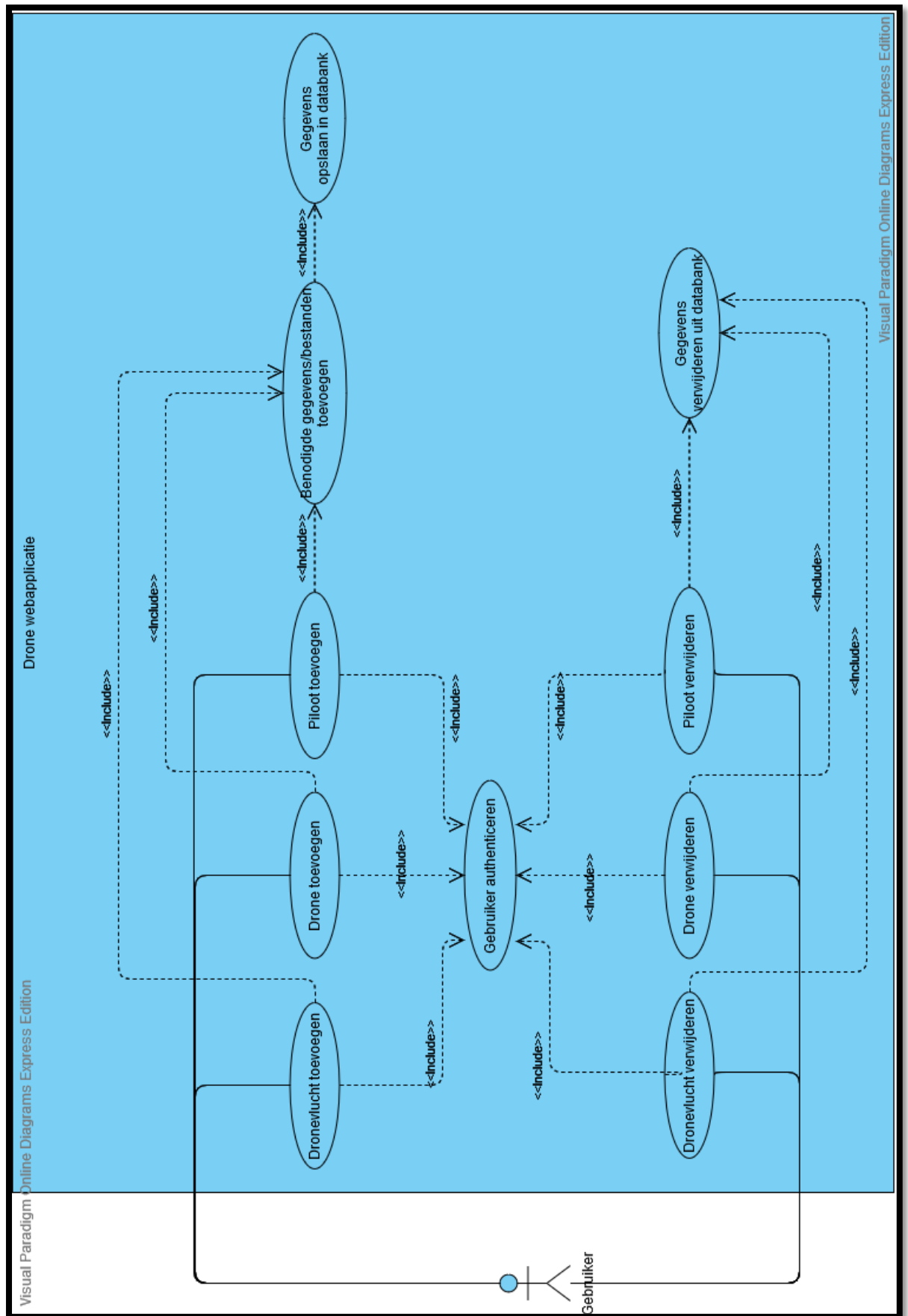
Het gebruik van de webapplicatie wordt verduidelijkt in **Figuren 1 en 2** op pagina's 10 en 11 via twee use case-diagrammen. Hierop is te zien hoe de actor (de gebruiker) kan interageren met de verschillende functionaliteiten van het systeem. Een gebruiker beschikt over CRUD (*Create, Read, Update,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

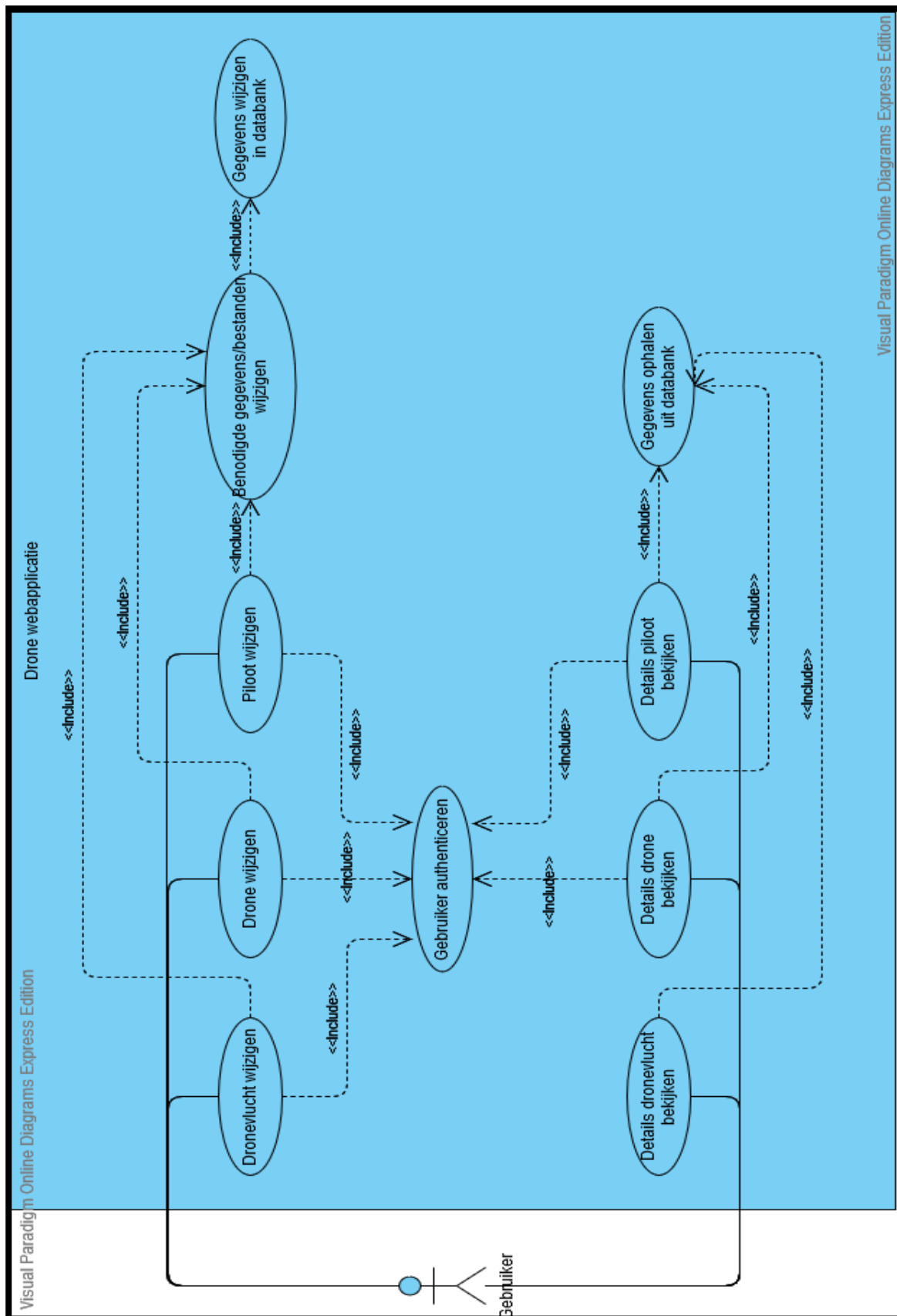
Een ingelogde gebruiker kan ook een dronevlucht, drone of piloot verwijderen (**Figuur 1**) uit het systeem. Verder kan een gebruiker de bijhorende en beschrijvende informatie van dronevluchten, drones of piloten wijzigen (**Figuur 2**). Deze aanpassingen hebben onmiddellijk effect op de databank aan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

1.4 Featurelijst

Hieronder volgt een opsomming van alle features in dit project, inclusief hun complexiteit¹ en prioriteit², 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 parserklassen 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).

¹ Makkelijk, vrij gemakkelijk, gemiddeld, vrij moeilijk, moeilijk

² 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 bestanden ziet de gebruiker een *progress bar* die weergeeft hoeveel procent van het bestand reeds geüpload is (moeilijk, redelijk belangrijk);
- *Controletool ctrl points* in *point cloud* (vrij moeilijk, belangrijk);
- Mogelijkheid om op basis van verschillende attributen *tracks* van drones te visualiseren (vrij moeilijk, zeer belangrijk);
- Automatisch veranderende legende bij visualisatie van de *tracks* (gemiddeld, belangrijk);
- Visualisatie van *point clouds*, *gcp's* en *ctrl's* (vrij moeilijk, zeer belangrijk);
- Verschillende *layers* bij visualisatie zodat de gebruiker zelf kan kiezen welke informatie getoond wordt van een dronevlucht (gemiddeld, belangrijk).

1.4.3 Sprint 3

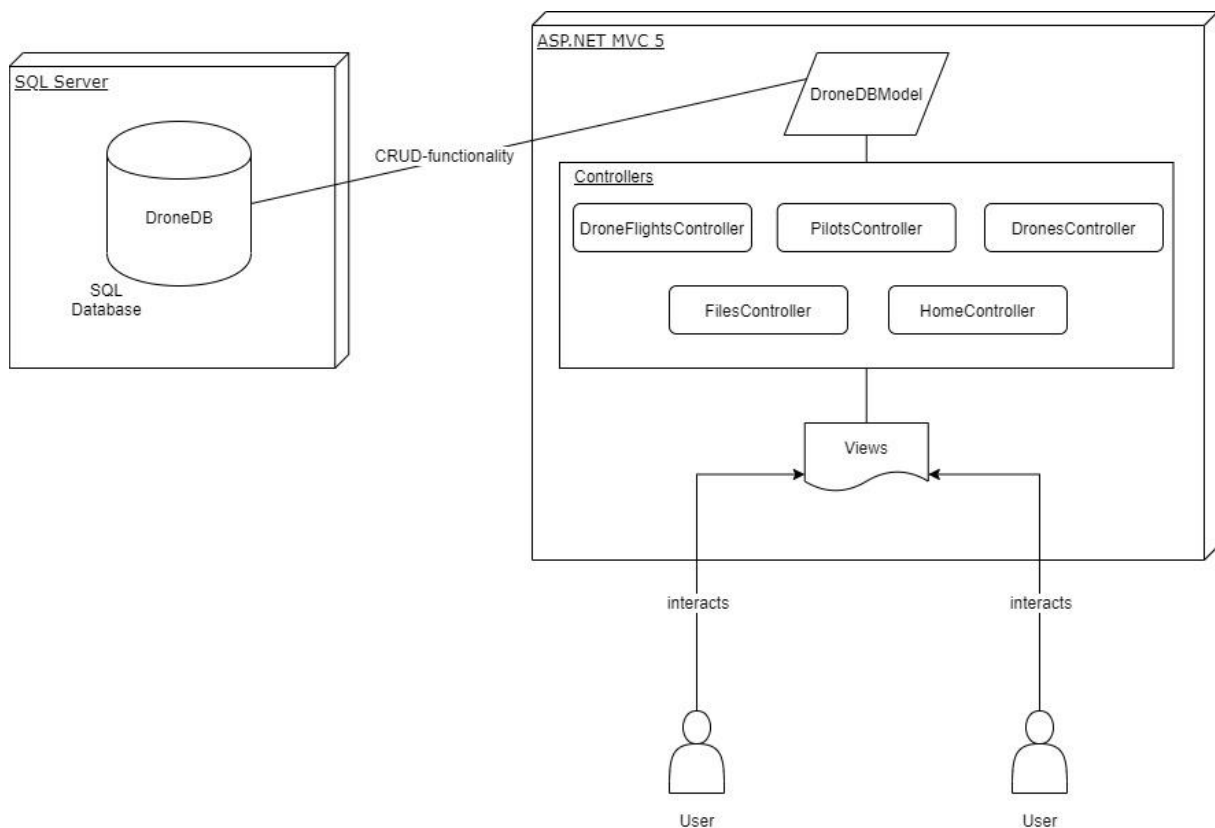
2. Systemarchitectuur

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 classes* zal de opbouw van de webapplicatie beschreven worden en getoond worden op welke manier de verschillende onderdelen samenwerken.

2.1 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

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 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 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 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 verkrijgen 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.

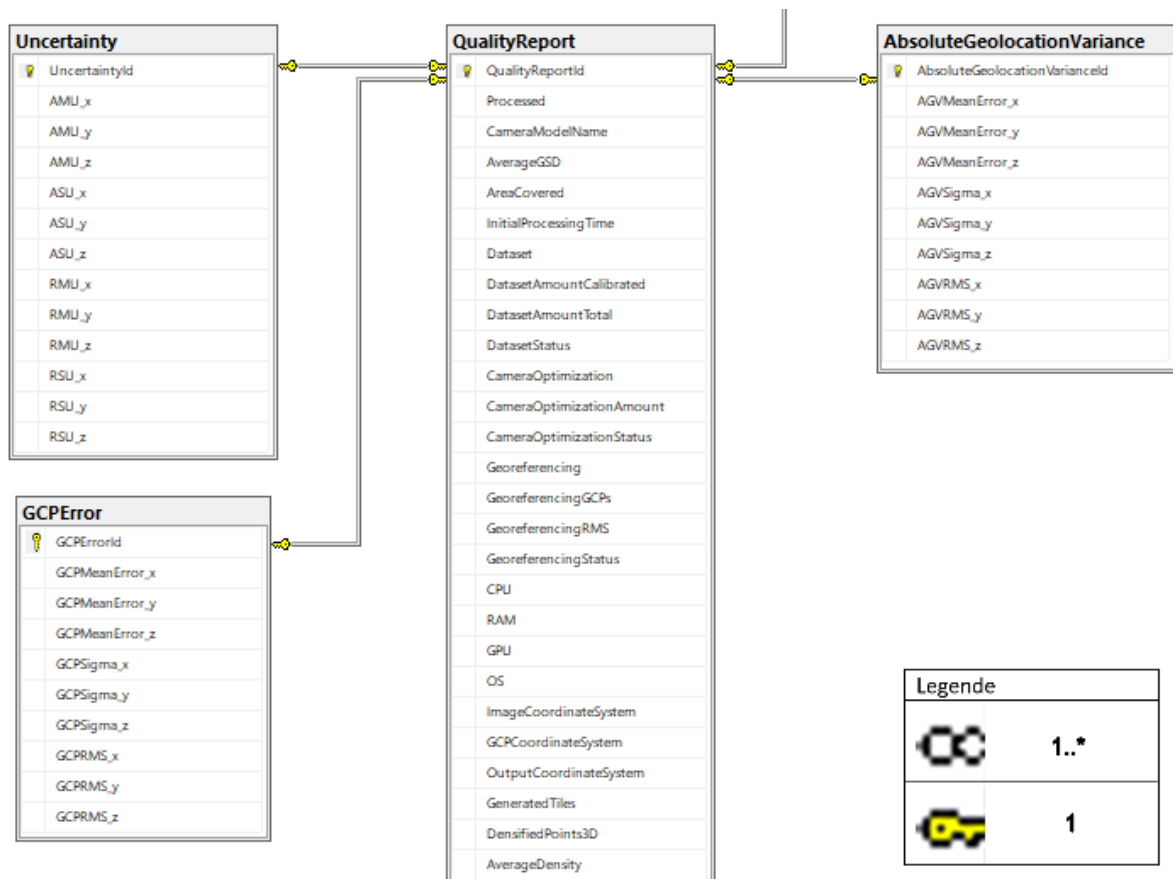
De tabel TFW bevat rotatie- en translatie informatie, alsook wereldcoördinaten die gebruikt worden bij een *tiff image* bestand in een GIS applicatie. Deze informatie laat toe een andere, externe databank aan te spreken en de juiste resultaten te verkrijgen in de vorm van een kaart.

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.

2.2.2 QualityReporttabel

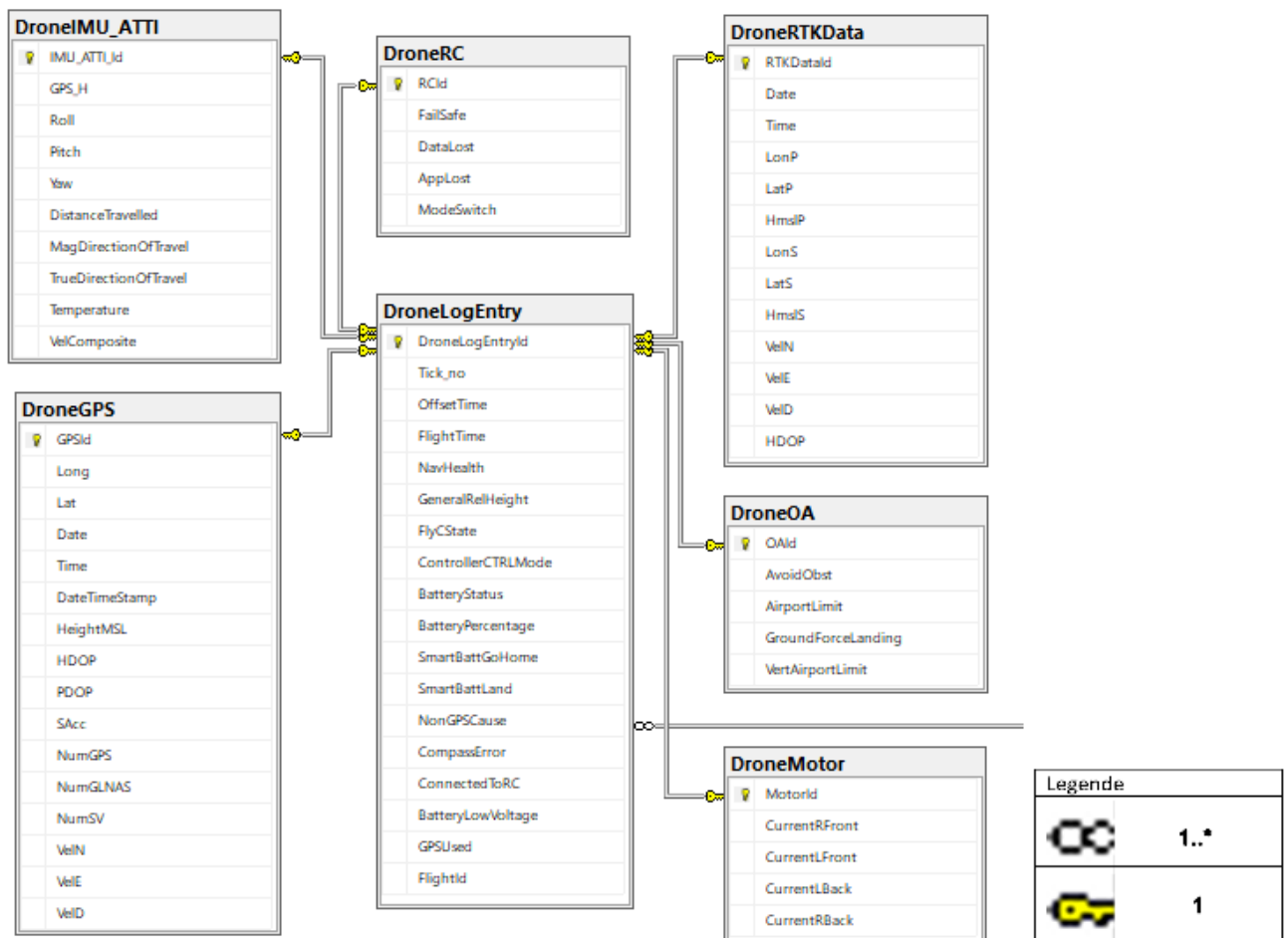
In de QualityReporttabel in **Figuur 5** wordt alle data bijgehouden die ingelezen wordt uit het kwaliteitsrapport. Dit is een pdf-bestand dat de output beschrijft van een analyse in het programma Pix4D. Tabellen die een relationeel verbonden zijn met deze tabel zijn: Uncertainty, AbsoluteGeolocationVariance en GCPErrord. Op deze manier is de ingelezen informatie uit het kwaliteitsrapport op een logische manier gegroepeerd.



Figuur 5: QualityReporttabel met al haar relaties

2.2.3 DroneLogEntrytabel

In **Figuur 6** 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 6: DroneLogEntrytabel met al haar relaties

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*.

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.

Drone Planner | Drone Flights | Drones | Pilots | Contact | About

Drone Flight Details

Drone Flight 1

[Upload File](#)

Location	Brugge
Date	11/03/2020
Drone	PH3: 00-12LUX
Pilot	Bryan Van Huyneghem
QR	QR
TFW	GCP
CTRL	CTRL
TFW	TFW
XYZ	XYZ
Drone Log for Flight	Drone Log

[Edit](#) | [Back to List](#)

© 2020 - Jan De Nul, Drone Flights Application

Figuur 7: Detailpagina van een Drone Flight

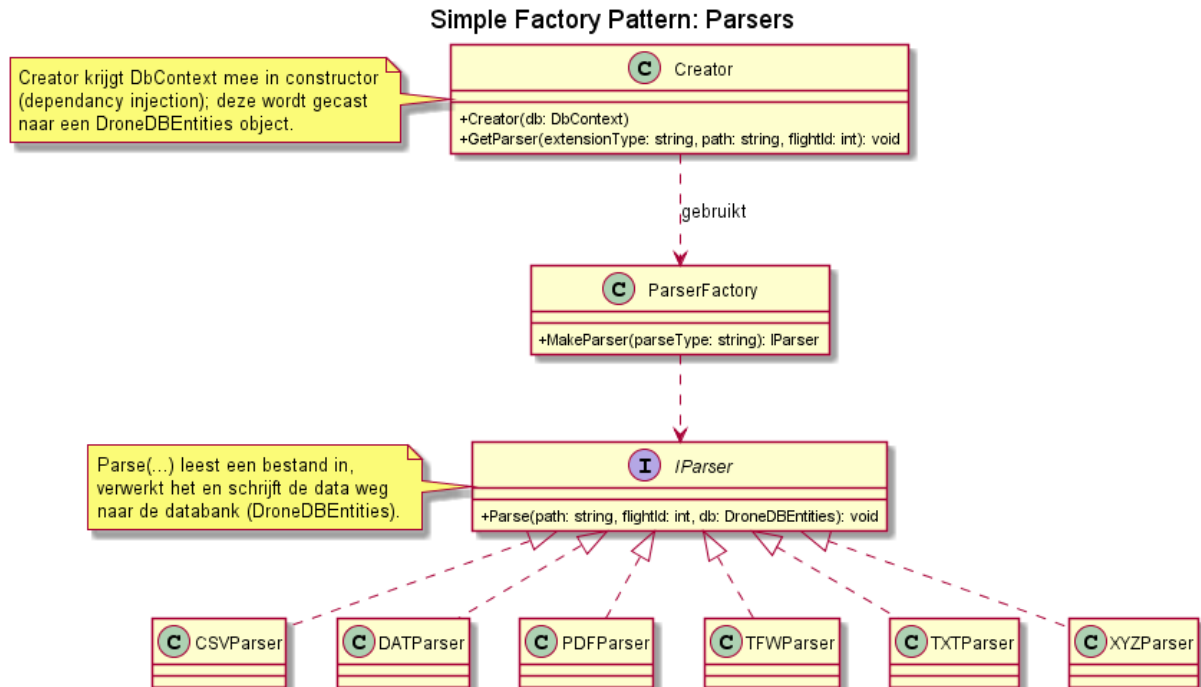
2.4 Klassendiagrammen

In dit onderdeel worden de klassendiagrammen beschreven die gebruikt werden in deze applicatie: een simple factory pattern voor het parsen van de bestanden, alsook javascriptklassen voor het visualiseren van data.

2.4.1 Simple Factory pattern voor parserklassen

Het parsen van data uit bestanden gebeurt via parserklassen. 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 parserklasse aanmaakt en teruggeeft aan de Creator. Daarna vindt het parsen plaats met de methode `Parse()` van de aangemaakte parserklasse.

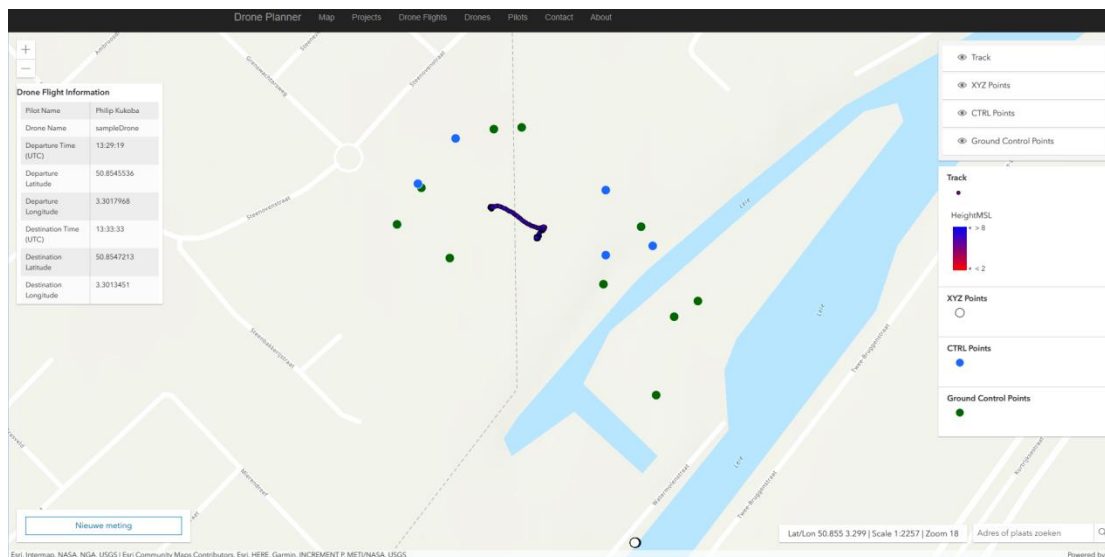
De constructor van `Creator` krijgt via *Dependency Injection* (met *Unity*) een instantie van de databank mee, zodat de *parsers* de ingelezen en verwerkte data kunnen wegschrijven naar de databank. Dankzij deze injectie wordt het aanmaken van dure databankconnecties beperkt.



Figuur 8: Simple factory pattern voor parsers

2.4.2 Javascriptklassen

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



Figuur 9: De map view

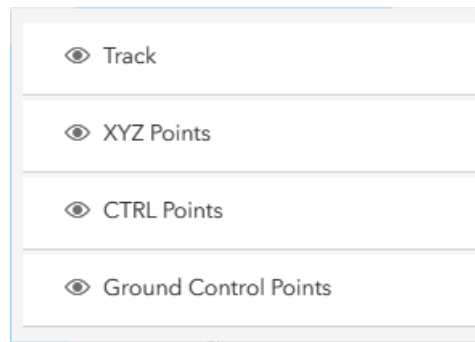
Map: De Map klasse is een container voor de *layers* van de applicatie en heeft bijhorende methodes voor deze *layers*.

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).

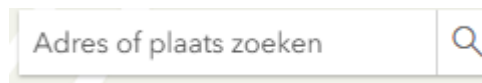
SpatialReference: Deze klasse definieert de ruimtelijke referentie van de te visualiseren data. Het geeft aan welk geografisch projectiesysteem wordt gebruikt om geografische punten op de kaart te situeren. Deze ruimtelijke referentie kan het gemakkelijkst gespecificeerd worden met behulp van het *well-known 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 heeft een referentie nodig naar de MapView.



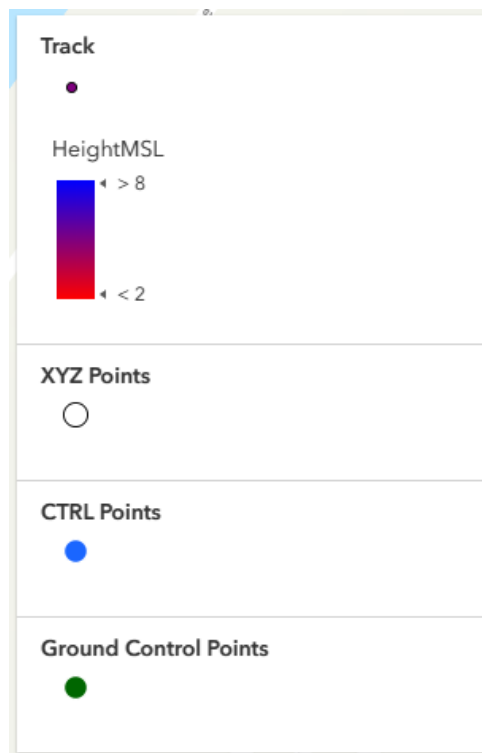
Figuur 10: LayerList widget

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



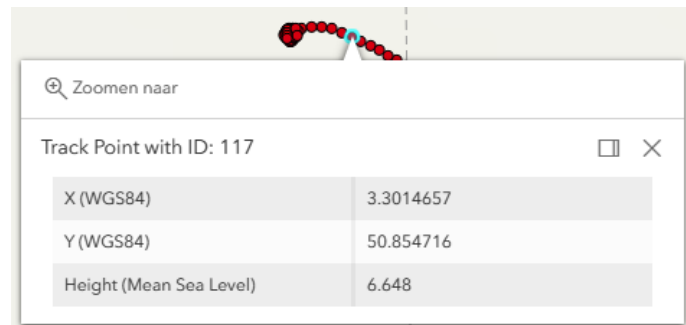
Figuur 11: Search widget

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 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*).

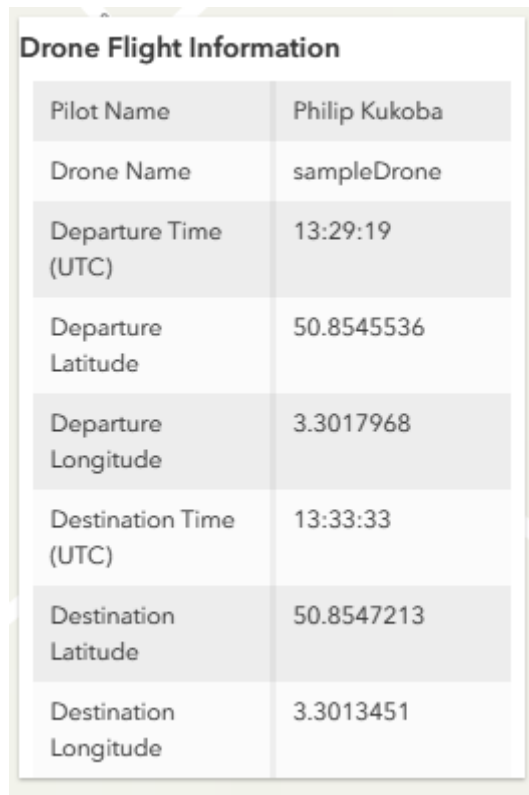
Als laatste moet het *objectIdField* ingesteld worden op het unieke ID attribuut van de Graphic objecten.

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.



The image shows a screenshot of a web application interface. At the top, there is a title 'Drone Flight Information'. Below the title is a table with two columns. The first column contains labels for various flight parameters, and the second column contains their corresponding values. The table is styled with alternating light and dark gray rows. The labels are: Pilot Name, Drone Name, Departure Time (UTC), Departure Latitude, Departure Longitude, Destination Time (UTC), Destination Latitude, and Destination Longitude. The values are: Philip Kukoba, sampleDrone, 13:29:19, 50.8545536, 3.3017968, 13:33:33, 50.8547213, and 3.3013451.

Drone Flight Information	
Pilot Name	Philip Kukoba
Drone Name	sampleDrone
Departure Time (UTC)	13:29:19
Departure Latitude	50.8545536
Departure Longitude	3.3017968
Destination Time (UTC)	13:33:33
Destination Latitude	50.8547213
Destination Longitude	3.3013451

Figuur 15: Feature klasse

3. 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.

3.1 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.

3.2 “View” Controllers

- 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.

4. Evaluaties en discussies

(sprint 3)

5. Handleidingen

5.1 Installatiehandleiding

De installatiehandleiding is opgedeeld in drie delen:

- Deel 1: vereiste software;
- Deel 2: aanmaken van de databank;
- Deel 3: opstarten van de webapplicatie.

5.1.1 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** (Developer editie) op uw machine. U kan deze software [hier](#) downloaden op de website van Microsoft.
 - a. U scrolt naar beneden tot u de Developer versie ziet en klikt “Download now”. Het programma downloadt.
 - b. Volg na het uitvoeren van het bestand de instructies op het scherm.
2. Installeer **SQL Server Management Studio (18.4) (SSMS)** op uw machine. U kan deze software [hier](#) downloaden.
 - a. U klikt op “Download SQL Server Management Studio (SSMS)”. Het programma downloadt.
 - b. Volg na het uitvoeren van het bestand de instructies op het scherm.
3. Installeer **Visual Studio 2019** op uw machine. U kan deze software [hier](#) downloaden.
 - a. U klikt op “Download Visual Studio” en kiest de Community 2019 of de Professional 2019 versie naargelang uw eigen voorkeur. Het programma downloadt.
 - b. Volg na het uitvoeren van het bestand de instructies op het scherm.
 - c. Bij installatie kiest u om de volgende “*Workloads*” te installeren: “*ASP.NET and web development*” en “*Data storage and processing*”.
4. Installeer **IvyTools** op uw machine. U kan deze software [hier](#) downloaden.
 - a. Navigeer naar de “*Downloads*” sectie van de deze webpagina. Een nieuwe webpagina verschijnt.
 - b. Download de “*30-day evaluation version*” of koop een licentie indien u deze tool langer dan twee maanden wenst te gebruiken door op “*Contact us*” te klikken.
 - c. Op deze webpagina kunt u ook de gratis *personal license key* verkrijgen die u zult nodig hebben om deze software te activeren. Klik hiervoor op “*Click here to get your free personal license key*”.

- d. Kopieer deze sleutel.
- e. Nadat de software gedownload is, opent u "*IvyTemplateEditor.exe*".
- f. U navigeert via de balk bovenaan het programma naar "*Help>About>Apply License Code*".
- g. U plakt de gekopieerde sleutel in het veld en drukt op OK.
- h. Controleer of de licentie geactiveerd is.
- i. Sluit "*IvyTemplateEditor*" af.
- j. U heeft nu toegang tot de IvyParser dll-bestanden in de webapplicatie. Deze *parser* wordt gebruikt bij het inlezen van het pdfbestand.

5.1.2 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.
 - a. Het veld "*Server name*" wordt automatisch ingevuld.
 - b. Noteer deze naam, want u heeft deze later nodig in een volgend deel (opstarten van de webapplicatie).
 - c. Klik op "*Connect*".
2. Klik op het "*File*" menu bovenaan links.
3. Kies "*Open>File*".
4. Navigeer 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*".
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.

5.1.3 Opstarten van de webapplicatie

In dit deel start u de webapplicatie op.

1. Start **Visual Studio 2019**.
2. Navigeer naar de *DroneWebApp solution* en open deze.
3. Navigeer in de *Solution Explorer* van het project naar het bestand **Web.config**, helemaal onderaan de mappenstructuur.
 - a. Dubbelklik om dit bestand te openen.

4. In de <connectionStrings> tag verandert u in de tag <add> het attribuut *data source* naar: `data source=UW_SERVER_NAME`.
 - a. UW_SERVER_NAME is de *server name* die u eerder tijdens het aanmaken van de databank noteerde.
5. Sla dit bestand op (*Save*) en sluit het.
6. Verander de mode van “*Debug*” naar “*Release*” en voer de webapplicatie uit met **F5**.
7. De allereerste keer kan een venster verschijnen dat u vraagt om het “*IIS Express SSL certificate*” te vertrouwen.
 - a. Klik *yes*.
8. Er verschijnt een “*security warning*”.
 - a. Klik *yes*.
9. U kunt nu aan de slag met de dronewebapplicatie.

5.2 Gebruikershandleiding

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

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

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 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.

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 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.

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 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.

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

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.

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 verwijdt een piloot uit de webapplicatie.

Actor

Gebruiker

Precondities

De gebruiker is ingelogd.

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 verwijdt 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.

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 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.

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 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.

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 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

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 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.

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 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.

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 de overige piloten.