HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Bachelorarbeit, Abteilung Informatik

Project Helin

Drone Delivery as a Service

Hochschule für Technik Rapperswil

Frühlingssemester 2016
15. Juni 2016

Autoren:	Marcel Amsler, Kirusanth Poopalasingam, Martin Stypinski
Betreuer:	Prof. Dr. Markus Stolze
Arbeitsperiode:	22.02.2015 - 17.06.2015
Arbeitsumfang:	360 Stunden, 12 ECTS pro Student
Link:	http://www.helin.ch/

Danksagung

Zunächst möchten wir uns an dieser Stelle bei Allen bedanken, die uns während dieser Arbeit unterstützt und motiviert haben.

- **Prof. Dr. Markus Stolze** für die Betreuung während des Semesters
- **Babeesan Poopalasingam** für die Hilfe bei der Modellierung der 3D-Teile
- **Michael Burri** für die Unterstützung und die Produktion der Videos
- **Louis Rosenthal** für die Unterstützung bei den Dreharbeiten und den Interviews
- **Verein Coredump** für die schnelle und unkomplizierte Produktion der 3D Teile
- **Marius Huber** für die zahlreichen Anmerkungen und Korrekturen zur Dokumentation
- **Sainuka Poopalasingam** für die zahlreichen Korrekturen zur Dokumentation
- **Christian Spielmann** für die Hilfe beim Beschaffen von Komponenten
- **Prof. Dr. Andreas F. Müller** für die Verifizierung der Fallschirmberechnung
- **Elias Geisser** für die ursprüngliche Idee: Bierlieferung mit Drohnen
- **Caspar Naef** für den Arbeitsplatz während dem Zusammenbau der Drohne (Institut für Mikroelektronik und Embedded Systems)
- **Physik Werkstatt Team** für die Maschinen zum Anpassen der gefertigten Komponenten

Selbstverständlich wollen wir auch Allen nicht persönlich erwähnten danken, wie unseren Familien, die uns während der gesamten Projektdauer moralisch unterstützt haben.

Abstract

Die folgende Arbeit beschreibt einen Prototyp eines automatisierten Drohnen-Liefersystems, den das Team konzipiert, entwickelt und getestet hat. Um die Tests unter realen Bedingungen durchführen zu können, sind zusätzlich zwei Multicopter zusammengestellt und gebaut worden.

Um ein System aufzubauen, das eine Flotte von Drohnen über eine zentrale Applikation verwalten kann, ist jede Drohne über ein Smartphone an eine Message Oriented Middleware (MOM) angebunden. Ausserdem verbindet sich die App über das MAVLink Protokoll mit dem Flight-Controller (Autopilot) der Drohne. Die entwickelte Android-App für die Drohne sendet laufend Telemetrie-Daten an den Server und dient zusätzlich als Benutzer-Interface für die Beladung.

Der zentrale Server berechnet aufgrund der Position des Kunden eine Route, die nur über vordefinierte und sichere Flugzonen führt. Anbieter können die Flugzonen über eine Benutzeroberfläche definieren und damit statische Hindernisse umgehen. Während einer Lieferung kommuniziert der Server mit allen Clients bidirektional um sowohl Kunden, als auch Anbietern eine Echtzeitverfolgung der Drohnen zu ermöglichen.

Während dieser Arbeit konnten zwei Drohnen mit einer selbst konzipierten Halterung für das Smartphone, sowie einer Abwurfvorrichtung ausgestattet werden und damit erfolgreich Lieferungen ausgeführt werden. Das mandantenfähige System steht zu Testzwecken als Software as a Service unter `https://my.helin.ch/` zur freien Verfügung. Der Quellcode wurde unter der MIT-Lizenz auf GitHub veröffentlicht.

Management Summary

Ausgangslage

Drohnen, die autonom fliegen und nicht mehr gesteuert werden müssen, gehört die Zukunft. Sie werden sicherer sein als heutige, von Menschen gesteuerte Drohnen und erlauben ein grösseres Einsatzgebiet. Firmen wie Amazon und die schweizerische Post planen bereits heute den Einsatz von Drohnen zur automatischen Auslieferung von Paketen. Doch vielen Anbietern bleibt diese Möglichkeit verwehrt, obwohl auf dem Markt eine Vielzahl von Komponenten zur Verfügung steht, um autonome Drohnen selbst zu bauen und zu betreiben. Die bestehenden Lösungen steuern aber immer nur eine Drohne gleichzeitig über Funk und bieten nur eine begrenzte Reichweite. Ausserdem existiert noch keine frei verfügbare Software, um automatisierte Dienstleistungen mit Drohnen anzubieten oder eine autonome Flotte zu verwalten.

Vorgehen / Technologien

Um zu zeigen, was mit heutigen Technologien und Standardkomponenten bereits möglich ist, wurde ein Demonstrationssystem mit zwei selbst gebauten Drohnen entwickelt.

Ein Smartphone, welches auf der Drohne montiert ist, ermöglicht die Kommunikation mit der Plattform, aber auch eine Benutzerinteraktion über das Display. Bei einer eingehenden Bestellung von der Bestell-App wird automatisch eine Route zum Kunden berechnet und einer verfügbaren Drohne zugewiesen. Sobald diese beladen ist, fliegt sie autonom zum Kunden, liefert das Produkt aus und kehrt wieder zurück. Dabei bleibt die Drohne in den vordefinierten, sicheren Flugzonen und weicht somit statischen Hindernissen aus. Eine Fernsteuerung wird nicht mehr benötigt, das System ist komplett autonom. Die App dient als Schnittstelle zwischen der cloudbasierten Verwaltungssoftware und der Drohnensteuerung, welche bereits grundsätzliche Funktionen wie GPS, automatische Stabilisierung und sogar einen programmierbaren Autopiloten bietet.

Ergebnisse

Das System zeigt erst einen Bruchteil der Möglichkeiten, die in Zukunft von autonomen Drohnen übernommen werden können. Beispielsweise können Videoaufnahmen, Infrastrukturüberwachung oder Katastrophenhilfe als Angebote integriert werden.

Wir sind überzeugt davon, dass die entwickelte Plattform als Denkanstoss für diese Branche und die Politik dienen kann, um die Technologien und Gesetzeslagen soweit zu verbessern, dass Dienstleistungen von Drohnen bald überall zur Verfügung stehen werden.

Inhaltsverzeichnis

Danksagung i

Abstract ii

Management Summary iii

Inhaltsverzeichnis iv

I. Technischer Bericht 1

1. Einleitung und Übersicht 2
1.1. Einleitung . . . 2
1.2. Ausgangslage . . . 2
1.3. Gesetzliche Rahmenbedingungen . . . 3
1.4. System Kontext . . . 4

2. Anforderungen 5
2.1. Benutzer und Personas . . . 5
2.1.1. Persona Diego: Kunde . . . 5
2.1.2. Persona Stefanie: Administrator . . . 6
2.1.3. Persona Ricardo: Drone-Operator . . . 6
2.2. Szenario . . . 7
2.3. Funktionale Anforderungen (User Stories) . . . 8
2.3.1. Administrator . . . 8
2.3.2. Kunde . . . 8
2.3.3. Drone-Operator . . . 9
2.3.4. Nachträglich ausgeschlossene Anforderungen . . . 10
2.3.5. Nachträglich hinzugefügte Anforderungen . . . 10
2.4. Nichtfunktionale Anforderungen . . . 11
2.4.1. Android Kompatibilität . . . 11
2.4.2. Verbindungsabbruch . . . 11
2.4.3. Flugsicherheit . . . 11
2.4.4. Verbindungswiederherstellung . . . 11
2.4.5. Security . . . 12
2.5. Usability und Accessability . . . 12
2.6. Domain-Model . . . 12

3. Architektur **14**
3.1. Ziele . . . 14
3.2. Einschränkungen . . . 14
3.2.1. Flight-Controller . . . 14
3.2.2. Onboard-App . . . 14
3.3. Übersicht . . . 15
3.4. Logische Architektur . . . 16
3.4.1. Komponenten Übersicht . . . 16
3.5. Server . . . 17
3.5.1. Anforderungen . . . 17
3.5.2. Layer . . . 17
3.6. Onboard-App . . . 19
3.6.1. Anforderungen . . . 19
3.6.2. Layer . . . 19
3.7. Customer-App . . . 20
3.7.1. Anforderungen . . . 20
3.7.2. Layer . . . 20
3.8. Kommunikations-Architektur . . . 21
3.8.1. Verwendete Enterprise Integration Patterns . . . 22
3.9. Bestellprozess . . . 23
3.10. Missionen . . . 25

4. Umsetzung **26**
4.1. Übersicht . . . 26
4.2. Implementierung . . . 26
4.2.1. Message Broker . . . 26
4.2.2. Server . . . 27
4.2.3. Onboard-App . . . 29
4.2.4. Customer-App . . . 32
4.2.5. Datenbank . . . 34
4.2.6. Sicherheit . . . 35
4.3. Projektgrösse . . . 35
4.4. Qualitätssicherung . . . 37
4.4.1. Prozesse . . . 37
4.4.2. Continuous Integration . . . 37
4.4.3. Testing . . . 37
4.5. Flugrouten . . . 39
4.5.1. Das Zonen Model . . . 39
4.5.2. Von der Zone zum Graphen . . . 40
4.5.3. Routing Algorithmus . . . 44
4.5.4. Höhenhandling . . . 44
4.5.5. Wahl der Rückroute . . . 45

5. Versuchsaufbau **46**
5.1. Einleitung . . . 46
5.1.1. Kommunikationshardware . . . 46
5.2. Ablieferungskonzept . . . 47
5.2.1. Landung ohne automatischen Abwurf . . . 47
5.2.2. Landung mit automatischem Abwurf . . . 47
5.2.3. Abwurf mit Fallschirm . . . 48
5.2.4. Entscheidung . . . 48
5.2.5. Fallschirmgrösse . . . 49
5.3. Multicopter Hardware . . . 51
5.3.1. Teile-Liste . . . 51
5.3.2. Frame und Antrieb . . . 52
5.3.3. Flight-Controller . . . 52
5.3.4. Ausbaustufen . . . 53
5.3.5. Tests . . . 57
5.4. Alternative Drohnenarten . . . 58

6. Zusammenfassung und Ausblick **59**
6.1. Impressionen . . . 59
6.2. Zusammenfassung der Ergebnisse . . . 64
6.3. Ausblick . . . 65
6.3.1. Empfohlene Weiterentwicklungen . . . 65
6.3.2. Bekannte Probleme . . . 65
6.4. Schlussfolgerung . . . 66

II. Anhang **67**

A. Aufgabenstellung **68**

B. Projektplan **71**
B.1. Änderungsgeschichte . . . 71
B.2. Einleitung . . . 71
B.2.1. Ziel und Zweck . . . 71
B.2.2. Lieferumfang . . . 71
B.2.3. Annahmen und Einschränkungen . . . 72
B.3. Projektorganisation . . . 72
B.3.1. Organisationsstruktur . . . 72
B.3.2. Verantwortlichkeiten . . . 73
B.3.3. Meetings . . . 74
B.4. Meilensteinplanung . . . 75
B.5. Risikomanagement . . . 76
B.5.1. Risiken . . . 76
B.5.2. Umgang mit Risiken . . . 78

B.5.3. Massnahmen 78
B.6. Qualitätsmassnahmen 80
B.6.1. Dokumentation 80
B.6.2. Projektmanagement 80
B.6.3. Entwicklung 80

C. Infrastruktur **81**
C.1. Server 81

D. Abbildungsverzeichnis **83**

E. Literaturverzeichnis **85**

F. Glossar **87**

Teil I.

Technischer Bericht

1. Einleitung und Übersicht

1.1. Einleitung

Ziel dieser Arbeit ist es, eine mandantefähige Plattform zu konzipieren, zu entwickeln und zu veröffentlichen. Diese soll es Anbietern von Produkten und Services ermöglichen, ihre Drohnen für einen begrenzten Zeitraum oder für bestimmte Aufgaben zur Verfügung zu stellen. Das Ausführen einer Aufgabe soll, bis auf das Beladen der Drohne, komplett automatisiert ablaufen.

Sobald ein Kunde eine Bestellung über eine App tätigt, wird automatisch eine Route zu seiner Position berechnet, die über vordefinierte, sichere Zonen führt. Über ein auf der Drohne angebrachtes Smartphone, werden Anweisungen für die Beladung angezeigt. Sobald die Vorbereitungen abgeschlossen sind, bewegt sich die Drohne, mit Hilfe von GPS, entlang der berechneten Route, erfüllt ihre Aufgabe und kehrt wieder zurück. Während der Mission werden laufend Telemetriedaten an den Server geschickt, welcher diese für den Anbieter visualisiert.

Project Helin wird als Open Source Software veröffentlicht und kann dann von jedem Interessierten gehostet und weiterentwickelt werden.

In der weiteren Dokumentation werden die Anforderungen, die Architektur, die wichtigsten Punkte der Umsetzung sowie der Aufbau eines Anwendungsbeispiels mit zwei echten Drohnen beschrieben.

1.2. Ausgangslage

Alle Multicopter verfügen bereits in der Grundausstattung über einen Flight-Controller, der die Befehle der Fernbedienung in Steuerbefehle für die Motoren übersetzt. Ausserdem nutzen viele Modelle zusätzliche Sensoren (GPS, Beschleunigungsmesser), um autonome Flüge zu ermöglichen und den Copter stabil halten.

Um laufend Telemetriedaten zu erhalten und Funktionen steuern zu können, wird in den meisten Fällen über Funk mit dem Copter kommuniziert. Dafür wird ein mobiles Gerät verwendet (Smartphone, Tablet, Notebook) auf dem eine Bodenstationssoftware läuft. Dies ermöglicht die Steuerung in einem Umkreis von ca. 2 km. Zusätzlich können mit Hilfe einer Bodenstationssoftware fixe Routen auf den Flight-Controller gespeichert werden, die dann im Automatikmodus autonom geflogen werden.

Dieses bewährte System hat allerdings einige Einschränkungen:

- Reichweite ist begrenzt
- Daten der Flüge sind nur für einen Benutzer einsehbar (Lokale Speicherung der Daten)
- Flugrouten müssen von Hand eingegeben werden
- Keine zentrale Komponente, auf die andere Systeme (z.B. Bestell-Apps) zugreifen und Aktionen der Drohnen auslösen können

Project Helin soll nun alle diese Einschränkungen aufheben. Damit können Anbieter neue Dienstleistungen erbringen, die nur mit autonomen Flugdrohnen in einer sinnvollen Effizienz möglich sind. Beispielsweise macht es nur in den seltensten Fällen Sinn, Lieferungen mit Drohnen auszuführen, wenn diese von Hand geflogen werden müssen.

Bis jetzt gibt es kein vergleichbares offenes System, das mit Drohnen über das Internet (4G) live kommuniziert und gleichzeitig die Anbindung von zusätzlichen Systemen erlaubt.

1.3. Gesetzliche Rahmenbedingungen

Für den Flug mit Drohnen in der Schweiz gelten die Vorgaben des Bundesamtes für Zivilluftfahrt (BAZL). Die für uns relevanten Punkte umfassen:

> „Für den Betrieb von Drohnen und Flugmodellen mit einem Gewicht von über 30 Kilogramm braucht es eine Bewilligung des BAZL.“ [8]
>
> „Über Menschenansammlungen bzw. im Umkreis von 100 Metern von Menschenansammlungen dürfen Drohnen grundsätzlich nicht betrieben werden.“ [8]
>
> „Ein automatisierter Flug (autonomer Betrieb) innerhalb des Sichtbereiches des «Piloten» ist erlaubt, sofern dieser bei Bedarf jederzeit in die Steuerung eingreifen kann.“ [8]

In anderen Ländern gelten eher schärfere Regelungen, so ist etwa in den Vereinigten Staaten eine Registrierung für Freizeitdrohnen ab 250g erforderlich [16]. In vielen Ländern werden die gesetzlichen Rahmenbedingungen aktuell angepasst, so dass auch Drohnen ausser Sichtweite betrieben werden können. Zum Beispiel kann in Polen mit einer theoretischen und praktischen Prüfung eine Lizenz erworben werden um Flüge ausser Sichtweite durchzuführen [16].

Diese Entwicklung zeigt, dass die Zulassung von autonomen Drohnen auch in der Schweiz zukünftig denkbar ist.

1.4. System Kontext

Project Helin hat drei verschiedene Nutzerarten: Customer (Kunde), Administrator (Anbieter) und Drone-Operator (Drohnen Operator). Diese werden im Kapitel Anforderungen noch genauer beschrieben.

In Abbildung 1.1 sind die externen Schnittstellen, sowie die Aktoren dargestellt. Google OAuth und PayPal werden auf der Customer-App für die Authentifizierung und das Payment eingesetzt. Die 3DR-Service-App wird auf dem Onboard-App benötigt um die Kommunikation mit dem Flight-Controller der Drohne zu ermöglichen. Für die Kartendarstellungen auf der Administrations-Webseite werden von Open Street Map gehostete Karten verwendet.

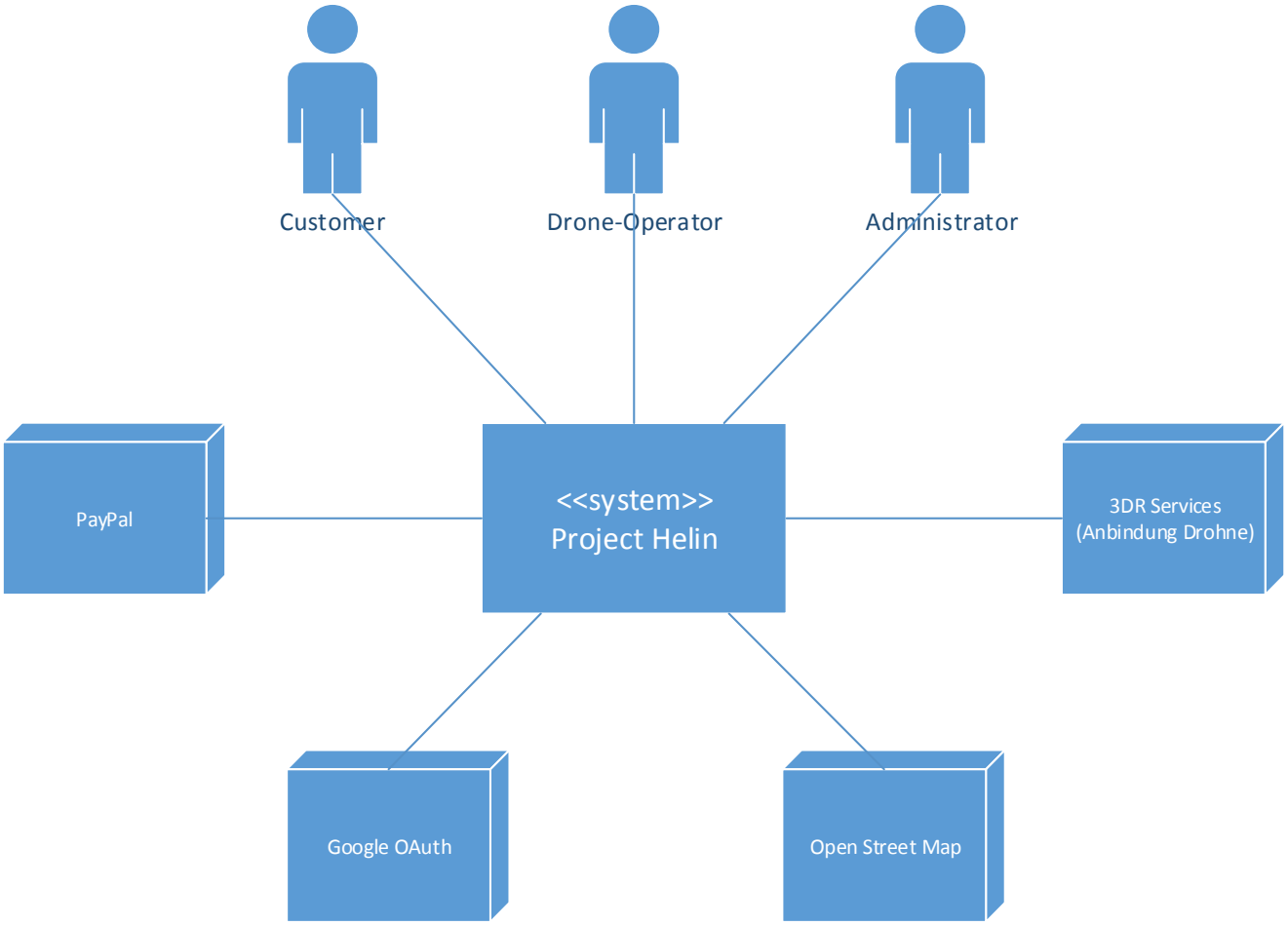


Abbildung 1.1.: System Kontext Diagramm mit externen Schnittstellen

2. Anforderungen

2.1. Benutzer und Personas

Die Benutzer der Project Helin Plattform teilen sich in drei Gruppen auf:

- **Kunde:** Der Kunde möchte Produkte und Dienstleistungen nutzen, welche von Drohnen erbracht werden können. Beispielsweise die Bestellung eines Getränks und sofortige Lieferung an seine Position.
- **Administrator:** Administratoren verwenden die Project Helin Plattform, um eine Flotte von Drohnen zu verwalten. Sie nutzen dazu die Webseite der Plattform und definieren, wo ihre Drohnen fliegen dürfen und welche Produkte und Services wo angeboten werden.
- **Drone-Operator:** Der Drone-Operator ist für die Wartung der Drohne verantwortlich und verwendet dafür die Onboard-App. Er kümmert sich ausserdem um die Beladung der Drohnen.

Bei den Personas handelt es sich um fiktive Personen.

2.1.1. Persona Diego: Kunde

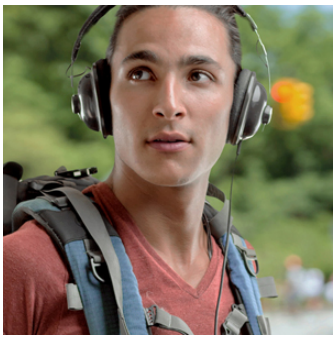


Abbildung 2.1.: Diego[1]

Diego ist **23 Jahre alt** und wohnt in einer WG in Uster. Diego hat eine Informatik-Lehre mit BMS abgeschlossen und ist auf der Suche nach einer neuen beruflichen oder schulischen Herausforderung.

Technisches Verhalten Er arbeitet täglich acht Stunden mit dem PC und nutzt gerne neue Technologien. Er besitzt ein Android-Smartphone der neusten Generation. Die Android Updates macht er immer sofort. Interessiert sich für neue Technologien und sieht sich regelmässig Kickstarter Projekte an.

Ziele Er will sich weiterbilden und neue Herausforderungen finden. Er möchte neue Technologien entdecken und einsetzen.

[1] Freie Lizenz, Quelle:http://blog.placeit.net/free-avatar-pack/

2.1.2. Persona Stefanie: Administrator

Stefanie ist **33 Jahre alt** und lebt alleine in Zürich.
Als diplomierte Eventmanagerin arbeitet Stefanie bei einer Eventagentur.

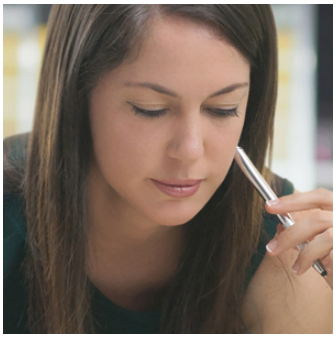


Abbildung 2.2.: Stefanie [1]

Technisches Verhalten Bei der Arbeit nutzt sie vor allem Excel, Word, Outlook und Chrome als Browser. Ausserdem hat sie viel Erfahrung mit diversen mandantenfähigen Systemen (CRM, ERP), die als Web-Applikationen umgesetzt sind. Sie besitzt ein Samsung Galaxy S4, das sie schon seit einigen Jahren verwendet.

Kommunikationsverhalten Sie kommuniziert geschäftlich hauptsächlich per E-Mail. Mit Freunden unterhält sie sich oft über WhatsApp oder den Facebook-Messenger. Auch während der Arbeit verbringt sie gerne Zeit auf Facebook.

Ziele Sie möchte mit neuen und innovativen Ideen Events für Besucher spannender gestalten.

2.1.3. Persona Ricardo: Drone-Operator

Ricardo ist **26 Jahre** alt und wohnhaft in Wetzikon. Der ausgebildete Schlosser hat zurzeit keinen festen Job. Er lebt das Leben von Tag zu Tag und geniesst es, keine festen Verpflichtungen zu haben.

Abbildung 2.3.: Ricardo [1]

Technisches Verhalten Er nutzt den PC selten, vor allem aber zum Surfen im Internet. Er besitzt er ein altes Android-Smartphone. Er möchte sich auch kein neues Gerät kaufen, da er ausser WhatsApp und der Taschenlampe das Gerät nicht verwendet.

Kommunikationsverhalten Er verwendet hauptsächlich sein Mobiltelefon zur Kommunikation. Mit Freunden ist er über WhatsApp in Kontakt.

Ziele Ricardo möchte gerne mit kleinen Jobs etwas Geld dazu verdienen und ohne lange Einarbeitungszeit den Anforderungen der Eventagentur genügen.

[1] Freie Lizenz, Quelle:http://blog.placeit.net/free-avatar-pack/

2.2. Szenario

Um eine bessere Übersicht über die Anforderungen zu erhalten wurde folgendes Szenario erstellt. Die rechtliche Situation wird dabei ausser Acht gelassen:

Stefanie hat in einem Facebook-Post von Project Helin erfahren und möchte deshalb am nächsten Openair, das ihre Firma organisiert, einen Getränke-Lieferservice mit Drohnen anbieten. Sie registriert sich und ihre Organisation auf https://my.helin.ch/ und lässt sich von einem lokalen Anbieter zwei Drohnen bauen, die Getränke tragen und abwerfen können. Nach dem Aufbau des Festgeländes zeichnet sie die Flugzonen auf https://my.helin.ch/ auf der Karte ein. Sie engagiert ausserdem Ricardo, der die Drohnen beladen und warten soll. Dieser lädt eine App herunter und installiert sie auf den zwei Smartphones, die auf den Drohnen montiert werden. Um die Drohnen mit dem Server zu verbinden, kann er die Adresse des Servers und einen Code eingeben, den Stefanie auf der Webseite abgeschrieben hat.

Diego möchte an einem Openair ein kühles Getränk für sich bestellen. Er hat auf einem Plakat vor Ort gesehen, dass ein Drohnen-Lieferservice existiert. Er lädt die Project Helin Bestell-App herunter und bestellt ein Rivella. Ihm wird eine Karte angezeigt mit dem voraussichtlichen Lieferort, den er bestätigen muss. Er muss sich nun einloggen und die Bestellung auf dem Mobiltelefon mit seiner Kreditkarte bezahlen.

Ricardo erhält eine Nachricht auf dem Onboard-App, die ihn fragt ob die Drohne einsatzbereit ist. Er bestätigt und ihm wird angezeigt, dass er ein Rivella laden muss. Er belädt die Drohne mit der Flasche und dem Fallschirm und bestätigt die Beladung. Die Drohne zeigt einen Countdown an und fliegt nach zehn Sekunden los.

Diego wundert sich, ob seine Bestellung unterwegs ist und sieht auf einer Karte in der App wie sich die Drohne auf ihn zubewegt. Die Drohne fliegt über ihn und wirft die Ladung ab. Danach fliegt sie auf dem gleichen Weg wieder zurück.

Ricardo sieht, dass die Drohne im Anflug ist. Sie landet an der Position, die ihm Stefanie zuvor gezeigt hatte. Er kann nun prüfen, ob die Batterie noch genug Spannung hat und ob mit der Drohne sonst alles in Ordnung ist.

Stefanie sitzt Zuhause und hat den Flug der Drohne auf https://my.helin.ch/ verfolgt. Sie sieht wie sich der Stand der Batterie während des Flugs geändert hat und dass Ricardo gerade eine Drohne deaktiviert hat, bei der er die Batterie tauschen muss.

2.3. Funktionale Anforderungen (User Stories)

Die funktionalen Anforderungen leiten sich aus der Aufgabenstellung, sowie den mit dem Betreuer diskutierten Ideen ab. Standardoperationen sind mit Teilen von CRUD bezeichnet. Einige Anforderungen sind mit 'zusätzlich' oder 'ausgeschlossen' gekennzeichnet, da sie nachträglich geändert wurden. Eine Erklärung zu jeder geänderten Anforderung findet sich im nächsten Abschnitt.

2.3.1. Administrator

- Als Administrator möchte ich auf der Webseite einen Account erstellen können.
- Als Administrator möchte ich meine Organisation verwalten können (CRU). (zusätzlich, siehe unten)
- Als Administrator möchte ich neue Administratoren hinzufügen und entfernen können. (zusätzlich)
- Als Administrator möchte ich ein Projekt erfassen können.
- Als Administrator möchte ich eine Drohne dem Projekt hinzufügen können.
- Als Administrator möchte ich alle Drohnen verwalten können (RUD).
- Als Administrator möchte ich Produkte verwalten können (CRUD).
- Als Administrator möchte ich Produkte einem Projekt hinzufügen können. (zusätzlich)
- Als Administrator möchte ich Services (z.B. Drone Selfies) verwalten können (CRUD). (optional)
- Als Administrator möchte ich Flug-, Lade- und Abwurfzonen verwalten können (CRUD).
- Als Administrator möchte ich Bestellungen verwalten können (CRUD). (teilweise ausgeschlossen)
- Als Administrator möchte ich Telemetriedaten der Drohne, sowie die berechnete Route vor, nach und während der Auslieferung einer Bestellung ansehen können.
- Als Administrator möchte ich Bestellungen abbrechen können. (ausgeschlossen)

2.3.2. Kunde

- Als Kunde möchte ich eine App aus dem Google Play Store herunterladen können, um diese verwenden zu können.
- Als Kunde möchte ich die App nutzen, ohne mich anmelden zu müssen.
- Als Kunde möchte ich in der App aus einer Liste von Produkten und Services, eine Auswahl treffen können.
- Als Kunde möchte ich nur Produkte und Services sehen, die in meiner Umgebung angeboten werden. (zusätzlich)

- Als Kunde möchte ich eine Bestellung tätigen können.
- Als Kunde möchte ich die bestellte Ware direkt bezahlen können. (optional)
- Als Kunde möchte ich auf der Karte des Smartphones die Bewegung der Drohne verfolgen können um abzuschätzen wann meine Lieferung eintrifft.
- Als Kunde möchte ich eine Bestellung stornieren können. (ausgeschlossen)

2.3.3. Drone-Operator

- Als Drone-Operator möchte ich eine Android-App mit Hilfe der heruntergeladenen APK installieren können.
- Als Drone-Operator muss ich die Drohne beim Server registrieren können.
- Als Drone-Operator muss ich die Drohne zu einer Organisation hinzufügen können.
- Als Drone-Operator muss ich das OTG-fähige Smartphone an einen MAVLink kompatiblen Flight-Controller über USB anschliessen können.
- Als Drone-Operator muss ich eine Verbindung zwischen App und Server über das Internet herstellen können.
- Als Drone-Operator muss ich eine Verbindung zwischen App und Flight-Controller herstellen können.
- Als Drone-Operator möchte ich den aktuellen Zustand der Verbindungen zur Drohne und zum Server sehen.
- Als Drone-Operator möchte ich den aktuellen Status des Flight-Controllers, beispielsweise GPS und Batteriespannung, sehen.
- Als Drone-Operator möchte ich eine Liste von Produkten angezeigt bekommen, die für die aktuelle Mission geladen werden müssen.
- Als Drone-Operator möchte ich eine Mission annehmen oder ablehnen können, um eine Drohne bei Problemen austauschen zu können.
- Als Drone-Operator möchte ich die Beladung einer Drohne bestätigen können.
- Als Drone-Operator erhalte ich ein visuelles und akustisches Countdown-Signal bevor die Drohne startet.
- Als Drone-Operator möchte ich den Start der Drohne während des Countdowns verhindern können.

2.3.4. Nachträglich ausgeschlossene Anforderungen

Bestellung löschen und ändern

Gemäss den anfänglichen Anforderungen sollte der Administrator die Möglichkeit haben eine Bestellung zu bearbeiten (CRUD). Dies wurde reduziert auf das Ansehen von Bestellungen (R). Unserer Meinung nach, sollte nur der Kunde die Möglichkeit haben seine Bestellung zu löschen. Ausserdem muss es auch dort Einschränkungen geben, da eine bezahlte Bestellung unter keinen Umständen gelöscht werden darf.

Mission abbrechen

Das Abbrechen einer Mission wurde aus dem Scope entfernt, da sich einerseits Fragen über den sinnvollen Einsatz eines solchen Features stellten und andererseits wichtigere Tasks wie die Bezahlung im App priorisiert werden konnten.

2.3.5. Nachträglich hinzugefügte Anforderungen

Verwalten von Organisationen

Um die Applikation mandantenfähig zu machen, wurden Organisationen eingefügt. Diese trennen verschiedene Kunden komplett ab und ermöglichen den Einsatz als Software as a Service.

Administratoren hinzufügen und entfernen

Um Organisationen nutzbar zu machen, muss es auch möglich sein, zusätzliche Administratoren zu einer Firma hinzuzufügen und wieder zu entfernen.

Produkte einem Projekt hinzufügen

Dieses Feature war nötig, damit Organisationen ihre Produkte nur einmal erfassen müssen und diese dann für verschiedene Projekte (z.B. Events) verwendet werden können (siehe Abb. 2.4).

Nur verfügbare Produkte anzeigen

Es macht keinen Sinn, dass ein Kunde Produkte sieht, die gar nicht zu ihm geliefert werden können. Deshalb wird die Liste mit Hilfe seiner Position auf verfügbare Produkte gefiltert.

2.4. Nichtfunktionale Anforderungen

2.4.1. Android Kompatibilität

Synopsis	Die Onboard-App funktioniert mit Android 4.4 und die Customer-App mit Android 6.1
Messbarkeit	Die obengenannten Apps können alle funktionalen Anforderungen erfüllen, wenn sie mit Android 4.4 bzw. 6.1 gestartet werden.

2.4.2. Verbindungsabbruch

Synopsis	Verbindungsabbruch der Onboard App zum Server soll keine negativen Auswirkungen auf die Mission haben.
Messbarkeit	Nach dem Start der Mission schliesst die Drohne, auch ohne Verbindung zum Server, die Mission ab.

2.4.3. Flugsicherheit

Synopsis	Eine Drohne führt vor dem Freigeben der Motoren (Arming) einen Check durch, der prüft ob alle nötigen Voraussetzungen für einen Start erfüllt sind. Ausserdem müssen vor einem Start ebenfalls Voraussetzungen der Mission erfüllt sein, beispielsweise muss der Drone-Operator den Start freigeben. Sollte eine dieser Voraussetzungen nicht erfüllt sein, darf die Drohne nicht starten.
Messbarkeit	Drohne startet nicht, falls der Pre-Flight-Check des Autopiloten nicht erfolgreich war oder Voraussetzungen für die aktuelle Mission nicht erfüllt sind.

2.4.4. Verbindungswiederherstellung

Synopsis	Nach einem Verbindungsabbruch zwischen dem Server und der Onboard App soll die Verbindung wiederhergestellt werden sobald wieder eine Internetverbindung verfügbar ist.
Messbarkeit	Die Verbindung zwischen Server und App wird nach dem deaktivieren und wieder aktivieren der Internetverbindung(4G) innert 30 Sekunden wiederhergestellt.

2.4.5. Security

Sichere Messaging-Verbindungen

Synopsis	Es darf nicht möglich sein, dass jemand die Steuerung einer beim Server registrierten Drohne übernehmen kann.
Messbarkeit	Der Übertragungskanal vom Server zur Drohne ist verschlüsselt, sodass sich keine weiteren Message Producer anmelden können.

Sichere HTTP-Verbindungen

Synopsis	Neben der Messaging Verbindung muss auch die HTTP-Verbindung gesichert sein.
Messbarkeit	Die Verbindung über den Webbrowser lässt nur HTTPS zu. Die Verbindung vom Customer-App zum Server läuft über HTTPS und Secure-WebSockets. Die Verbindung vom Onboard-App zum Server läuft über HTTPS.

2.5. Usability und Accessability

Usability-Tests und Anforderungen in der Accessability wurden bewusst und in Absprache mit dem Betreuer aus dem Scope ausgeschlossen.

2.6. Domain-Model

Aus den funktionalen Anforderungen ergibt sich das folgende Domainmodel.

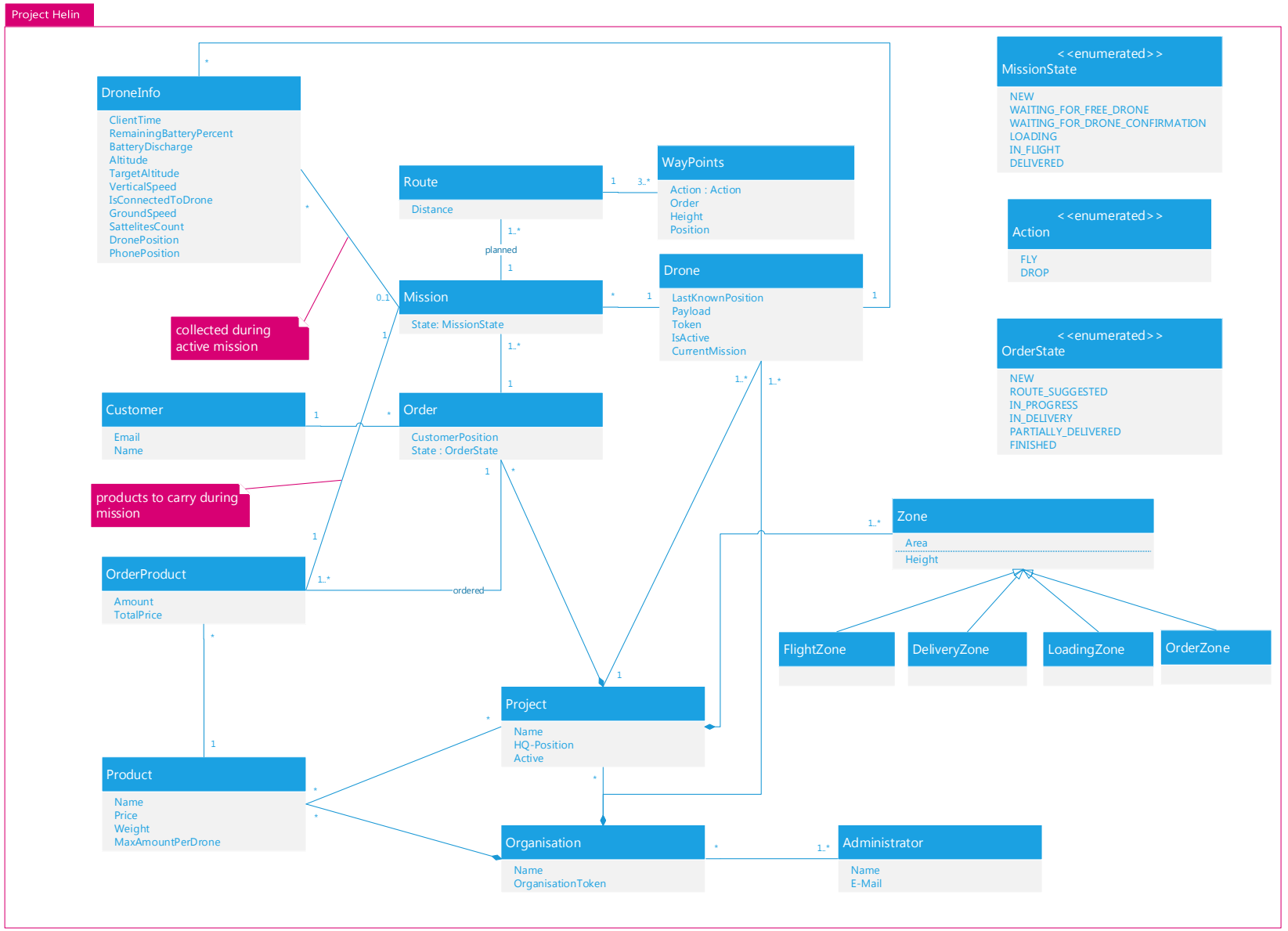


Abbildung 2.4.: Domain-Model

3. Architektur

3.1. Ziele

Die folgende Architektur soll es ermöglichen, eine Flotte von Drohnen automatisiert und zentralisiert zu verwalten. Die Kommunikation zwischen Server und Drohne muss ausserdem von beiden Seiten initiierbar sein (Push-Messages). Zusätzlich muss eine Schnittstelle für Kunden existieren, damit Bestellungen getätigt werden können.

3.2. Einschränkungen

3.2.1. Flight-Controller

Da der Flight-Controller bereits vor der Arbeit evaluiert wurde, wird dieser als vorgegebene Limitierung angesehen.

3.2.2. Onboard-App

Während der gesamten Arbeit wird davon ausgegangen, dass das Onboard-App im Vordergrund verwendet wird.

3.3. Übersicht

Die Abbildung 3.1 zeigt eine Übersicht der verschiedenen Tiers und Server-Komponenten.

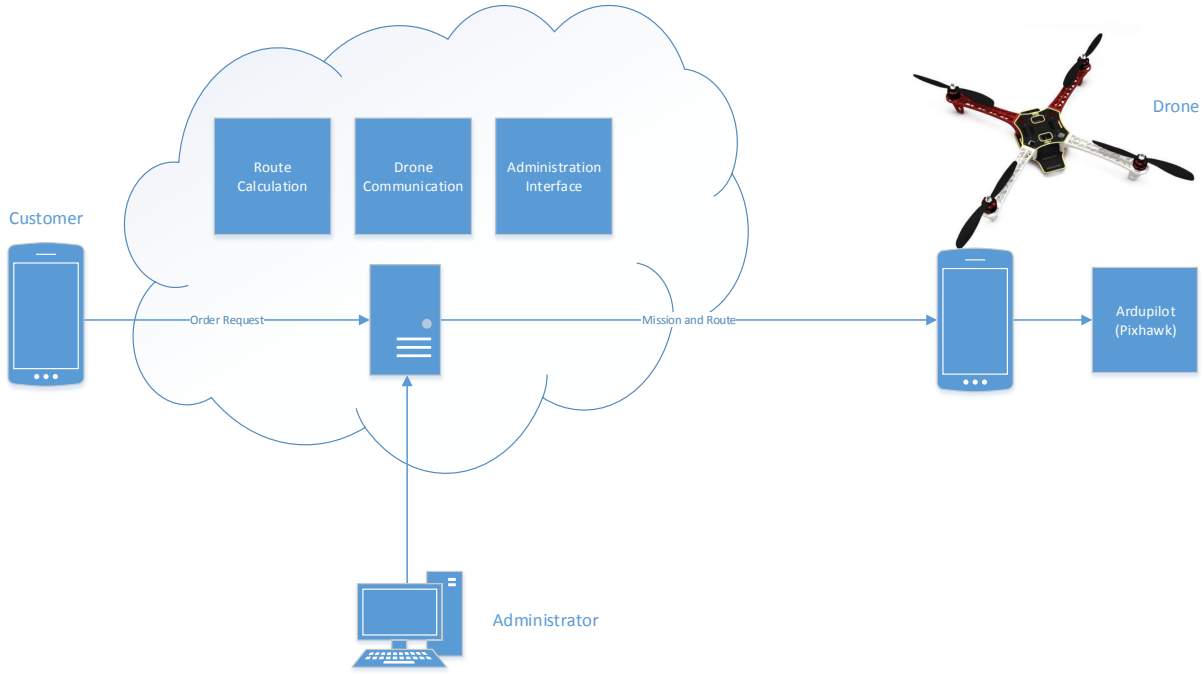


Abbildung 3.1.: Übersicht der Project Helin Architektur

3.4. Logische Architektur

3.4.1. Komponenten Übersicht

Abbildung 3.2 zeigt die Hauptkomponenten, sowie eine Übersicht der enthaltenen Layer und Packages. Ausserdem sind die Abhängigkeiten zu den wichtigsten externen Komponenten dargestellt. Hervorzuheben ist ebenfalls die gemeinsame Verwendung der Commons-Komponente.

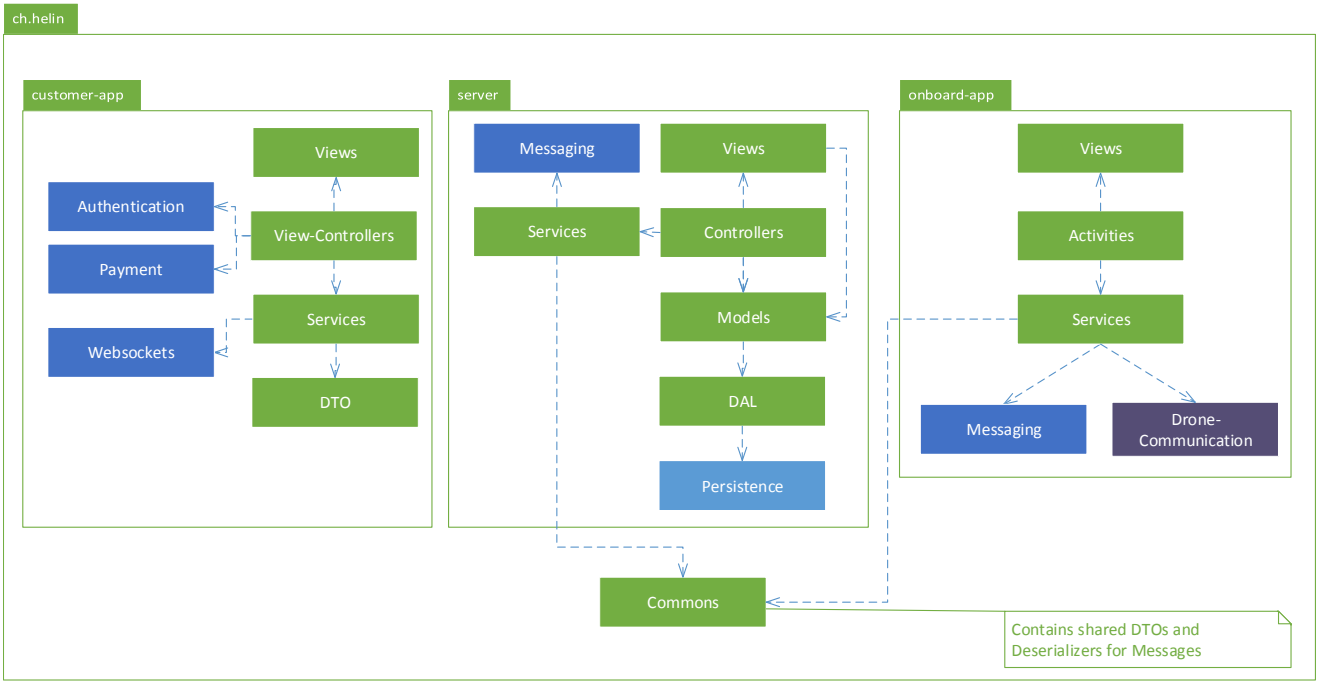


Abbildung 3.2.: Vereinfachte Übersicht der logischen Architektur und deren Komponenten

3.5. Server

3.5.1. Anforderungen

Aus den in den Anforderungen definierten Zielen, ergeben sich für die Server-Komponente folgende Anforderungen:

- Bietet Benutzeroberfläche für Administratoren
- Verwaltet CRUD für alle nötigen Klassen (siehe Domain Model Abb. 2.4)
- Verwaltet Verbindungen zu Onboard-Apps und Customer-Apps
- Berechnet Flugrouten basierend auf den erhaltenen Bestellungs-Koordinaten
- Persistiert alle Daten

3.5.2. Layer

Die Architektur des Servers basiert auf dem MVC Pattern [7], da dieses für CRUD Applikationen mit Benutzeroberfläche besonders geeignet ist und dafür viele geeignete Frameworks zur Verfügung stehen.

Für Komponenten die aus unterschiedlichen Controllern verwendet werden, wurde zusätzlich eine Service Komponente verwendet. Die Services stellen Abstraktionen für wichtige Funktionen, wie das Messaging und die Routenberechnung bereit.

Die Verantwortlichkeiten und Kollaborationen der Layer werden nachfolgend genauer definiert.

View-Layer

Stellt die grafische Benutzeroberfläche zur Verfügung und rendert diese basierend auf den Model-Daten.

Controller-Layer

Verantwortlichkeiten	**Zusammenarbeit**
• Lädt Daten für Views aus dem Data-Access-Layer • Verarbeitet eingehende HTTP-Requests • Verarbeitet eingehende Messages aus den Messaging-Queues • Enthält Business-Logik und steuert Ablauf nach einem Request	• Data-Access-Layer • Service-Layer • Model-Layer • Commons

Model-Layer

Der Model-Layer enthält die Datenmodelle. (siehe Domain Model Abb. 2.4)

Service-Komponente

Verantwortlichkeiten	Zusammenarbeit
• Verwaltet Verbindungen zu den Drohnen • Deserialisiert eingehende Nachrichten • Leitet eingehende Nachrichten von Drohnen an den Controller-Layer weiter • Leitet eingehende Nachrichten vom Kunden-App an den Controller-Layer weiter • Berechnet Flugrouten basierend auf den vorgegebenen Flugzonen	• Controller-Layer • Commons

Data Access Layer (DAL)

Der Data Access Layer bietet die Möglichkeit auf die Persistence Library zuzugreifen und stellt dafür die wichtigsten Funktionen zur Verfügung.

3.6. Onboard-App

3.6.1. Anforderungen

- Kommuniziert mit dem Flight-Controller der Drohne
- Kommuniziert mit dem Server
- Bietet eine Benutzeroberfläche für den Drone-Operator

3.6.2. Layer

Die App enthält die normalen Android-Application-Layer wie Activities und Views. Zusätzlich kommt der Service-Layer hinzu. Dieser enthält keine Android-Services, sondern selbst entwickelte Service-Klassen. Android Services werden nur zwingend benötigt, wenn etwas im Hintergrund weiterlaufen soll. Da die Onboard-App aber immer im Vordergrund läuft, konnte die aufwendige Kommunikation mit Android-Services mit Hilfe von Dependency-Injection [6] umgangen werden.

Service-Komponente

Verantwortlichkeiten	**Zusammenarbeit**
• Verwaltet Verbindung zur Drohne • Verwaltet Verbindung zum Server • Deserialisiert eingehende Nachrichten. • Leitet eingehende Nachrichten vom Server und den Activities-Layer oder andere Services weiter • Ermöglicht das Senden von Nachrichten an den Server	• Activities-Layer • Messaging • Commons

3.7. Customer-App

3.7.1. Anforderungen

- Kommuniziert mit dem Server
- Bietet eine Benutzeroberfläche für den Customer
- Ermöglicht die Bezahlung von Produkten
- Ermöglicht das Login über einen externen Identifikationsprovider

3.7.2. Layer

View-Layer

Ist zuständig für die Darstellung der Benutzeroberfläche und bindet die Schnittstellen zum Zahlungsanbieter (Payment) und Identifikationsprovider (Authentication) an.

Service-Layer

Verantwortlichkeiten	**Zusammenarbeit**
• Verwaltet Verbindung zum Server • Deserialisiert eingehende Nachrichten. • Leitet eingehende Nachrichten vom Server und den Activities-Layer weiter • Ermöglicht das Senden von Nachrichten an den Server	• View-Layer • WebSockets-Library

3.8. Kommunikations-Architektur

Die Abbildung 3.3 zeigt die Kommunikations-Architektur in der Übersicht. Wichtig sind vor allem die verschiedenen verwendeten Protokolle, die benötigt werden um die Anforderungen erfüllen zu können.

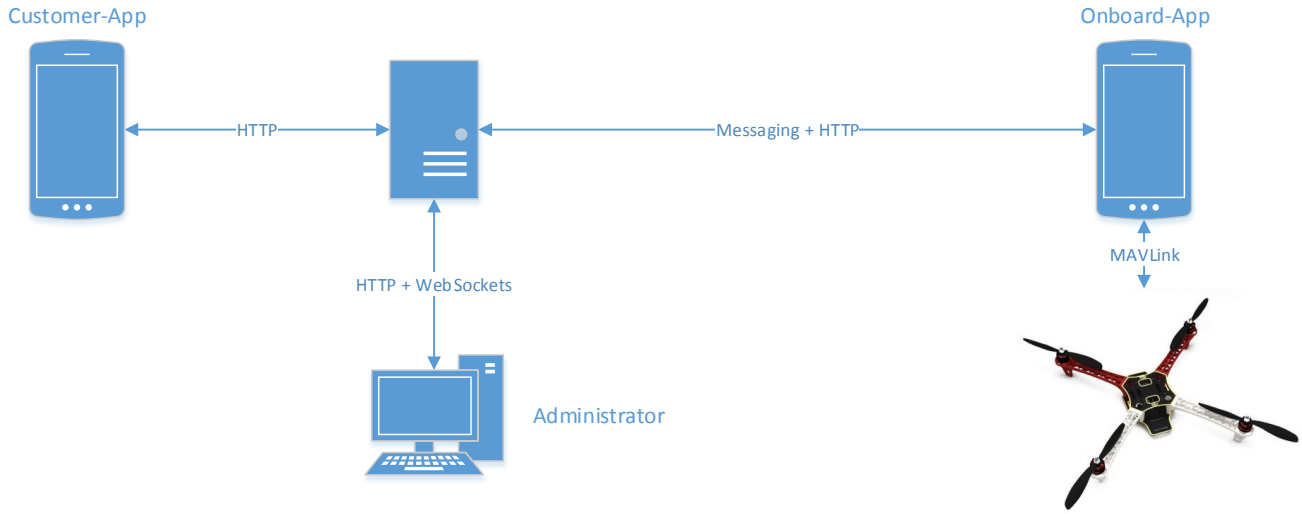


Abbildung 3.3.: Übersicht der Kommunikations-Architektur mit den jeweiligen Protokollen.

Für die Kommunikation mit dem Onboard-App wird Messaging verwendet, um bidirektionale Kommunikation zu ermöglichen und die höchstmögliche Zuverlässigkeit zu erreichen. Eine bidirektionale bzw. beidseitig initiierbare Kommunikation ist bei allen Geräten nötig, um Live-Updates zu ermöglichen (Verfolgung der Drohne). Ausserdem muss die eingesetzte Technologie für die Kommunikation zwischen dem Server und den Onboard-App den Nicht-Funktionalen-Anforderungen gerecht werden, die im Bezug auf Verbindungsabbrüche und Verbindungswiederherstellung bestehen.

Aufgrund der schlechten Erfahrung mit WebSockets auf mobilen Geräten und den Problemen bei Verbindungsabbrüchen, kamen diese nicht in Frage. Messaging bietet ausserdem eine grössere Unabhängigkeit zwischen den Kommunikationspartnern und ist robuster bezüglich den zu erwartenden Verbindungsabbrüchen.

Bei der Anbindung der Customer-App ist die Plattformunabhängigkeit der Schnittstelle höher gewichtet als die Zuverlässigkeit der bidirektionalen Verbindung. Deswegen wird bei dieser App vor allem HTTP verwendet und nur wo nötig WebSockets eingesetzt. Einzig die Liveupdates der Drohnenposition werden über WebSockets gesendet. Damit können in Zukunft auch andere Geräte verwendet werden um das System anzusprechen, ohne dass sie ein Messaging-Protokoll unterstützen müssen.

Die Anbindung an die Administrationsoberfläche erfolgt ebenfalls über HTTP und WebSockets, da dort die Wahrscheinlichkeit eines Verbindungsabbruchs viel geringer ist als bei einem mobilen Gerät. Zusätz-

lich ist eine Verbindungswiederherstellung viel einfacher, da das Neuladen der Seite ausreicht.

3.8.1. Verwendete Enterprise Integration Patterns

Messaging-Systeme und Protokolle bieten eine grosse Auswahl an Patterns, die je nach Anforderungen verwendet werden können [9]. Für dieses Projekt benötigen wir nur einen kleinen Teil davon um den gestellten Anforderungen gerecht zu werden.

Point-to-Point Channel

Um zwischen einer registrierten Drohne und dem Server einen sicheren und zuverlässigen Nachrichtenaustausch zu ermöglichen, wird jede Drohne bzw. jede App über einen separaten Point-to-Point Channel [9, S. 103] angebunden. Der Channel wird nach der Registrierung zugeteilt und wird als Point-to-Point Channel zwischen Drohne und Server genutzt. Dies garantiert dem Server, eine Nachricht an nur eine Drohne zu schicken.

At-most-once

Die Fehlersemantik At-most-once gibt die Sicherheit, dass eine Drohne eine Mission oder einen Befehl nur einmal erhält. Ansonsten müsste das System idempotent gebaut werden. Exactly-once-delivery ist ausserdem in der Praxis eigentlich unmöglich umzusetzen, weshalb wir uns für diese Alternative entschieden haben.

Event-driven Consumer

Event-driven Consumer [9, S. 442] Systeme bieten die Möglichkeit, Aktionen auf Grund von Nachrichten auszuführen. Beispielsweise:

- Drohne erhält neue Mission vom Server und soll dies dem Drone-Operator anzeigen.
- Server erhält neue Position von der Drohne und soll diese Nachricht dem Kunden weiterleiten und dem Administrator auf dem Web-Client anzeigen.
- Smartphone des Kunden erhält neue Position der Drohne, auf der Karte wird die Position der Drohne angezeigt.

3.9. Bestellprozess

Die Abbildung 3.4 beschreibt den Bestellprozess und die damit zusammenhängende Kommunikation.

Der Kunde bestellt mittels der App ein Produkt (Customer-App), der Server bestätigt ihm die Bestellung und schlägt einen Abwurfort vor. Sollte der Abwurfort dem Kunden nicht entsprechen, so kann er den Prozess abbrechen und ihn noch einmal auslösen, sobald er sich an einer passenderen Stelle befindet.

Der Server analysiert die Eignung der verfügbaren Drohnen und bestimmt im Anschluss eine, welche den Auftrag ausführen kann und zeigt die Mission dem Drone-Operator an. Sollte die Drohne bereit sein, so kann er diesen Auftrag bestätigen. Im Falle einer nötigen Wartungsarbeit kann der Auftrag zu diesem Zeitpunkt auch abgelehnt werden. Sobald der Auftrag angenommen wurde, erhält der Drone-Operator genaue Angaben zur Beladung der Drohne. Anschliessend wird die Ladung bestätigt und der Server kann dem Kunden mitteilen, dass die Drohne startklar ist. Während des Fluges erhält der Kunde Benachrichtigungen vom Server mit der aktuellen Position der Drohne. Sobald die Ladung abgeworfen wurde, fliegt die Drohne zur Ladezone zurück und bestätigt dem Server die Ankunft. So kann gewährleistet werden, dass bekannt ist, ab wann die Drohne wieder verfügbar ist.

Der Bezahl- und Anmeldeprozess wird hier aus Übersichtsgründen nicht dargestellt.

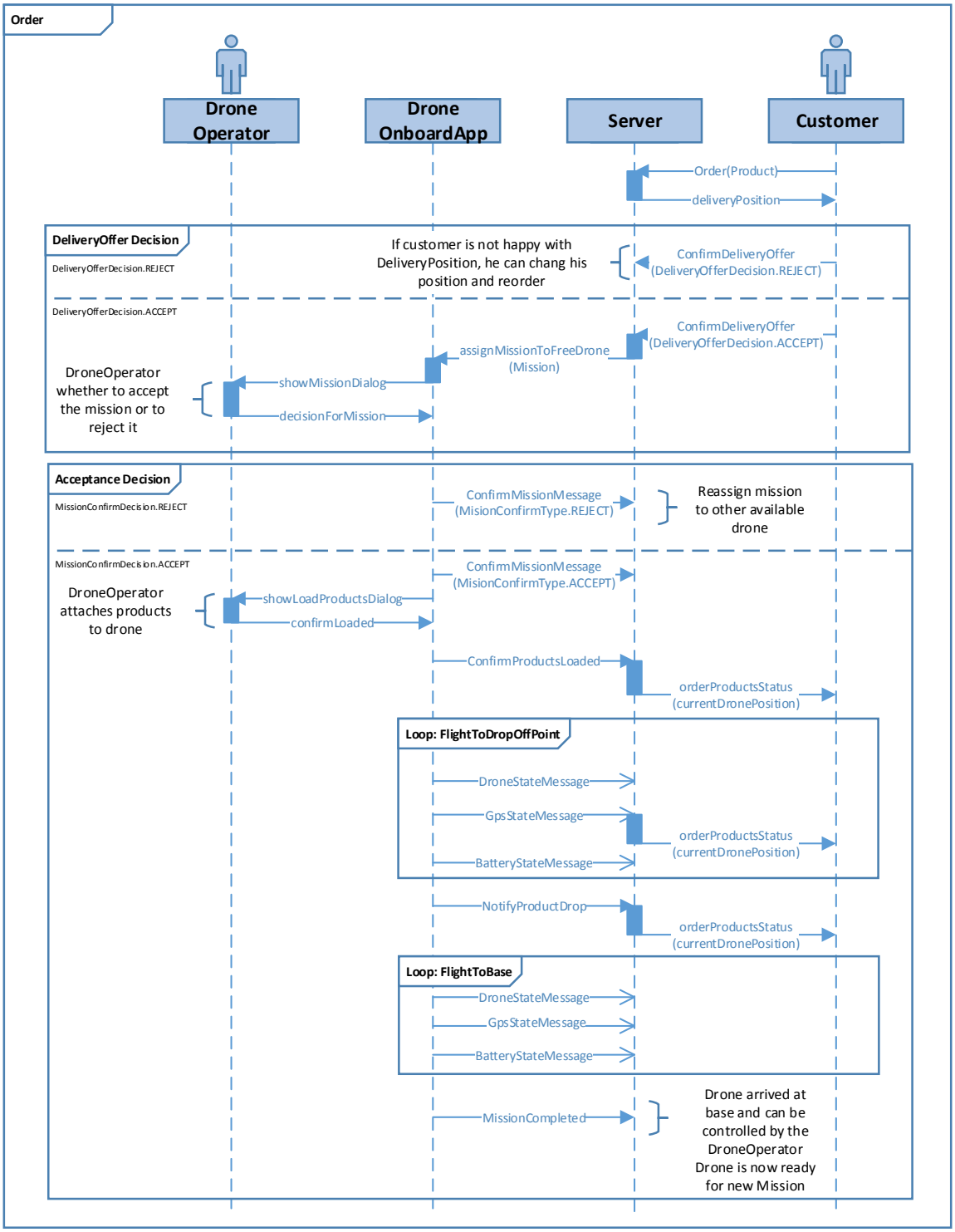


Abbildung 3.4.: Ablauf der Bestellung eines Produkts

3.10. Missionen

Wie im Domain Model (Abb. 2.4) ersichtlich, enthalten alle Bestellungen mindestens eine Mission. Die Mission wiederum enthält alle Informationen, die für die Auslieferung notwendig sind. Das folgende Diagramm (Abb. 3.5) zeigt die möglichen Zustände einer Mission, von der Bestellung bis zur Rückkehr nach der Lieferung.

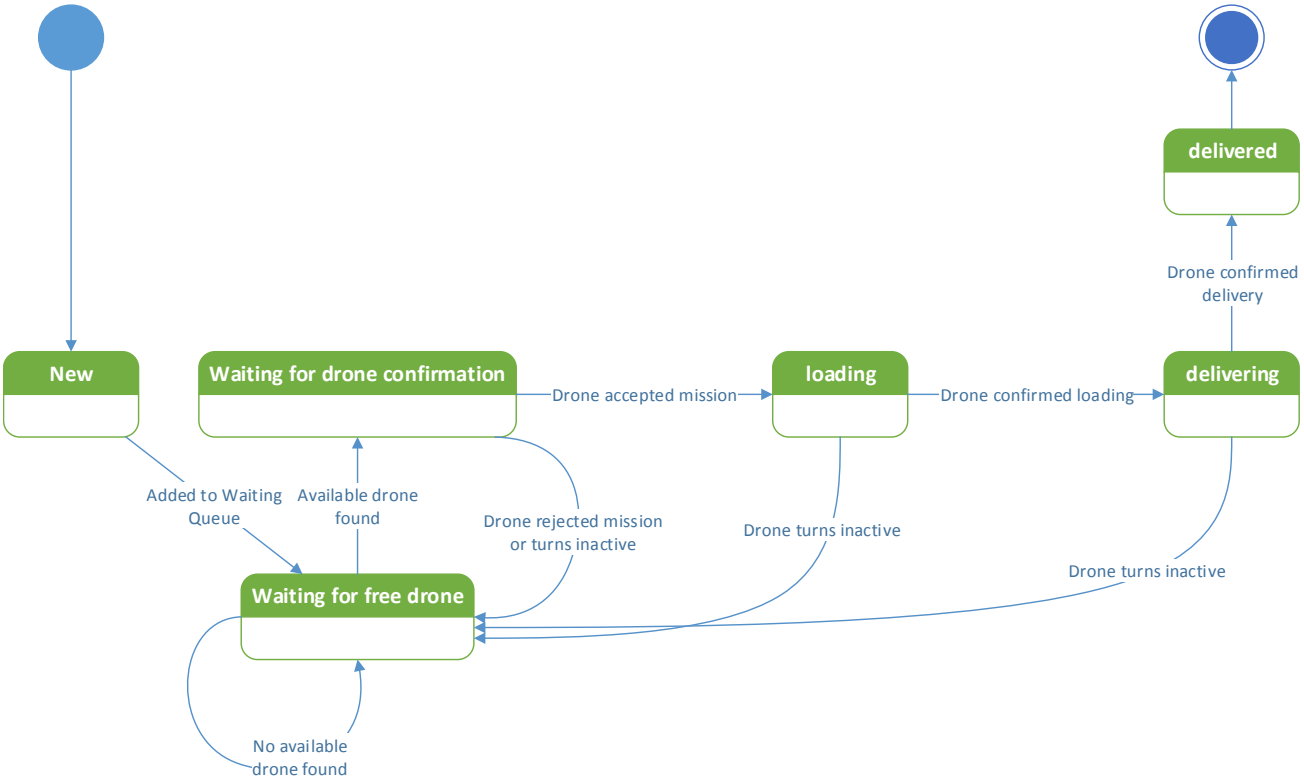


Abbildung 3.5.: Zustände der Mission

4. Umsetzung

4.1. Übersicht

Abbildung 4.1 zeigt eine grobe Übersicht der wichtigsten Komponenten und deren Verbindungen untereinander. Dieses Design orientiert sich an der Kommunikationsarchitektur, die in Kapitel 3.8 beschrieben ist.

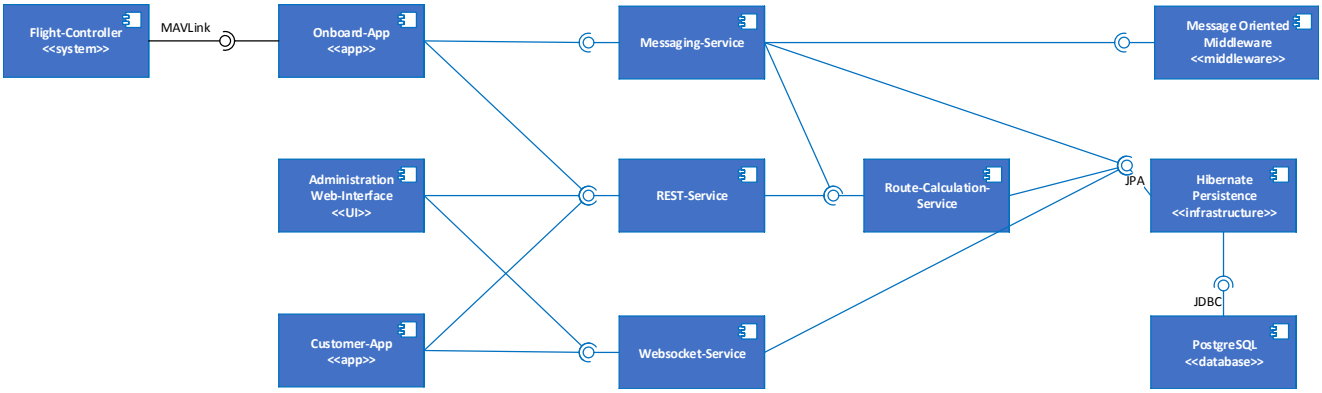


Abbildung 4.1.: Übersicht der wichtigsten Schnittstellen und deren Verbindungen

4.2. Implementierung

4.2.1. Message Broker

In der Architektur wurde festgelegt, dass der Server und die Onboard-App über Messaging kommunizieren. Um dies zu implementieren, entschieden wir einen eigenen Broker auf unserem Server verwenden, um möglichst unabhängig von kostenpflichtigen Cloud-Anbietern zu sein.

Ein geeigneter Message Broker muss sowohl vom Server als auch von der Onboard-App angesprochen werden können. Diese Anforderung verunmöglicht den Einsatz von allen JMS-kompatiblen Message Brokern, da JMS nicht mit der Android Plattform kompatibel ist. Als plattformunabhängige Alternative, kommt somit nur ein AMQP-fähiger Broker in Frage. Dafür stehen zwar mehrere Produkte zur Auswahl, aber nur RabbitMQ kann den Broker und alle Client-Bibliotheken liefern. Aus diesen Gründen haben wir uns für RabbitMQ entschieden

4.2.2. Server

Play Framework

Für eine Anwendung in diesem Umfang ist der De-Facto-Standard SpringMVC. Da Spring aber ein schwergewichtiges Framework ist und wir eine neuere, leichtgewichtigere Alternative in der Praxis evaluieren wollten, haben wir uns für das Play Framework entschieden.

Zusätzliche Gründe die unsere Entscheidung für Play unterstützt haben:

- Fortschrittlicher O/R-Mapper im Framework integriert
- Vorgegebene MVC-Architektur (speziell geeignet für CRUD-Operationen)
- Schnelle Implementation durch Spezialisierung auf diese Anwendungsart
- Alle Entwickler des Projekts sind versiert in Java
- RabbitMQ Anbindung möglich
- WebSocket Anbindung möglich

Kommunikation mit den Drohnen

Um die Möglichkeit zu haben mit einzelnen Drohnen zu kommunizieren, müssen auf dem Server zur Laufzeit alle Verbindungen zu allen Drohnen bekannt sein. Ausserdem muss es möglich sein, eingehende Nachrichten an einem zentralen Punkt abzuarbeiten. Aufgrund der bestehenden MVC-Struktur der Play-Applikation haben wir uns entschieden, auch die Nachrichten aus dem Messaging-System in Controllern zu behandeln, wie es normalerweise nur mit HTTP-Requests geschieht.

Die Abbildung 4.2 zeigt ein Objekt-Diagramm von Instanzen, welche die Kommunikation mit den Drohnen erlauben, sobald die Applikation gestartet ist.

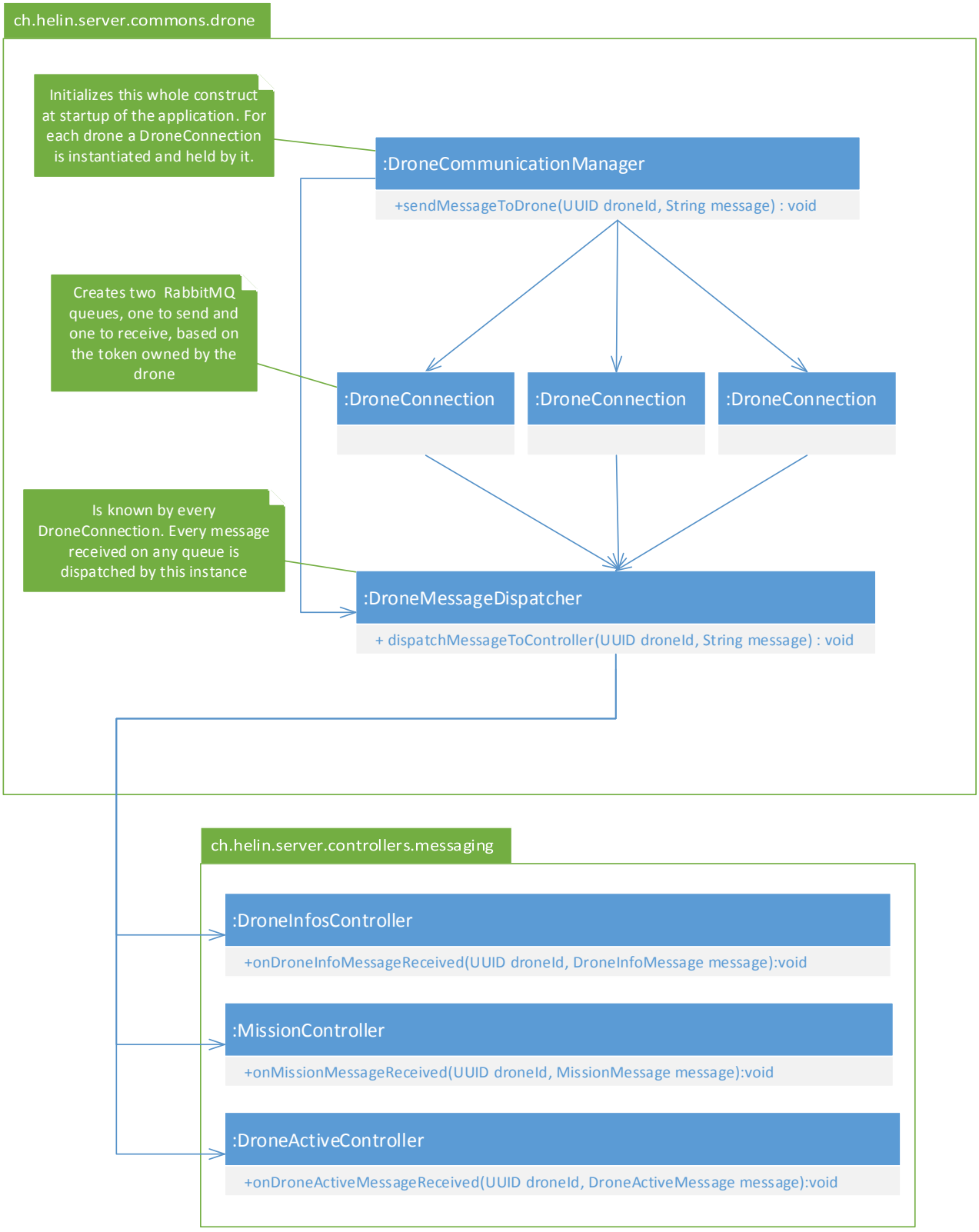


Abbildung 4.2.: Objekt-Diagramm der Kommunikationsstruktur

Verwendete Bibliotheken

Name	Verwendungszweck	Version	Lizenz
Hibernate ORM Mapper	objekt-relationales Mapping zwischen Datenbank und Java Objekten	5.1.0	Apache 2.0
RabbitMQ Client	Client Komponente zur Kommunikation mit dem Rabbit MQ Broker	3.6.0	Mozilla Public License 1.1, GPL 2, Apache 2.0
jGraphT	Graph-Bibliothek für Java, um effizient Operationen auf dem Graph auszuführen	0.9.2	LGPL, EPL

Bemerkungen aus der Evaluation oben stehender Komponenten:

- **Hibernate ORM Mapper:** Hibernate ORM ist der einzige ORM Mapper, der eine Spatial-Extension anbietet. Durch Hibernate Spatial ist es möglich, PostGIS Objekte aus der Datenbank direkt zu Java Objekten zu konvertieren.
- **RabbitMQ Client:** Als Message Broker wurde auf RabbitMQ gesetzt, daher wurde konsequenterweise RabbitMQ Client als Bibliothek verwendet.
- **jGraphT:** Hibernate Spatial arbeitet mit der Java Topology Suite (JTS) zur Abbildung der geografischen Daten. Um aus JTS Objekten einen Graphen zu bilden, kann nur jGraphT verwendet werden.

4.2.3. Onboard-App

Entwicklungsumgebung

Um den Flight-Controller mit MAVLink ansprechen zu können, musste eine Android/Java Library gefunden werden, die dies ermöglicht. Die Firma 3DR, der Hersteller des Pixhawk Flight-Controllers, stellt dafür eine Open Source Android-API bereit. Die API kann im App integriert werden und erlaubt es, dem Flight-Controller direkt Befehle zu erteilen. Ausserdem kann man über diese Schnittstelle auch Telemetriedaten auslesen und Events handeln, z.B. Höhenänderungen. Damit war es möglich, die Drohne vom App und damit auch vom Server aus zu steuern.

Um während der Entwicklung laufend Tests durchzuführen, ohne jedes Mal mit der echten Drohne zu testen, verwendeten wir eine Software, die den Flight-Controller simuliert. Die Simulation verhält sich nahezu identisch wie ein richtiger Controller und hilft herauszufinden, ob die gesendeten Befehle die erwartete Wirkung zeigen.

Die verschiedenen Möglichkeiten, das System zu testen sind im GitHub Repository dokumentiert. In der Abbildung 4.3 wird ein Setup gezeigt, bei dem die Onboard-App im Android Emulator läuft (rechts unten). Gleichzeitig zeigt die Bodenstationssoftware die Echtzeit-Daten des simulierten Fluges an.

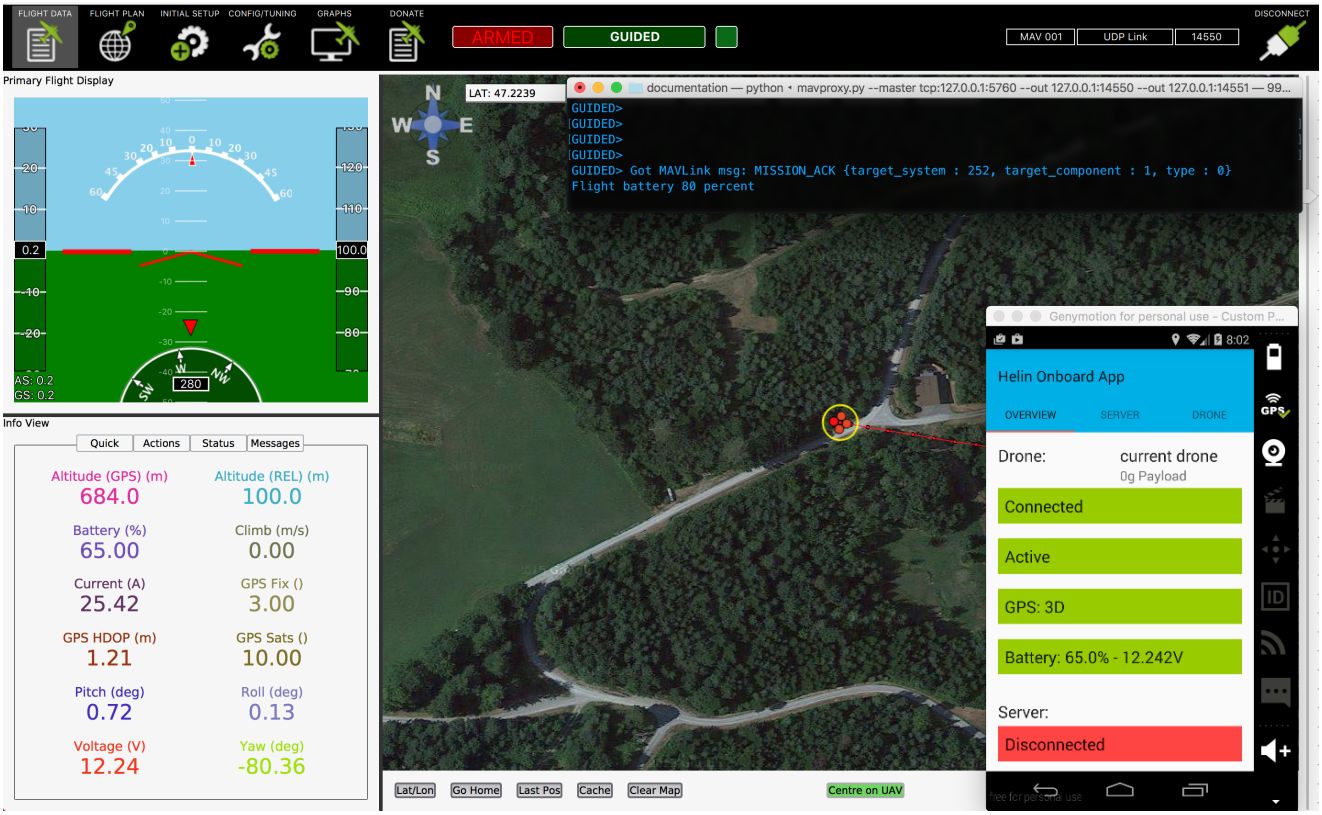


Abbildung 4.3.: APM Planner 2.0 als Bodenstation und Android Emulator mit Onboard App

Verwendete Bibliotheken

Name	Verwendungszweck	Version	Lizenz
DroneKit-Android Client	Android API für MAVLink Protokoll zum Ansteuern der Drohne	1.5.1	Apache 2.0
AMQP Messaging Library	Messaging für Android	3.6.0	Mozilla Public License 1.1, GPL 2, Apache 2.0
Lyra	High availability Messaging	0.4.3	Apache 2.0
Dagger	Dependency Injection Framework für Android	2.0.1	Apache 2.0

Bemerkungen aus der Evaluation oben stehender Komponenten:

- **DroneKit-Android Client:** Es existiert keine gleichwertige Alternative die das MAVLink Protokoll in diesem Umfang abdeckt.
- **Lyra:** Die Bibliothek wird vom Hersteller der Message-Oriented-Middleware (MOM) empfohlen. [15]
- **AMQP Messaging Library:** Ermöglicht den Zugriff auf den RabbitMQ Message Broker.
- **Dagger:** Wird von Google entwickelt und unterstützt Dependency Injection zur Kompilierzeit.

Kommunikation zum Server

Gemäss den nicht-funktionalen Anforderungen soll ein Verbindungsabbruch zum Server keinen Einfluss auf die Ausführung der Lieferung haben. Deshalb wird die Flugroute vor dem Start auf den Flight-Controller übertragen, somit kann die Mission auch ohne Online-Verbindung erfolgreich abgeschlossen werden.

Sollte die Verbindung zwischen Server und Onboard-App unterbrochen sein, sollte verhindert werden, dass Nachrichten verloren gehen. Bei unterbrochener Verbindung wird dies wie folgt erreicht:

- **Nachricht vom Server zum Onboard-App**
  Dieses Szenario wird durch RabbitMQ abgefangen. Der Message Broker persistiert die Nachrichten solange bis die Onboard-App wieder verfügbar ist. Es besteht somit kein zusätzlicher Handlungsbedarf.
- **Nachricht vom Onboard-App an den Server**
  Hier steht der Message Broker nicht zur Verfügung und die Speicherung der Nachrichten musste implementiert werden. Dazu verwenden wir eine Queue in der alle Nachrichten gespeichert werden und erst entfernt werden, wenn sie verschickt wurden.

Mit diesen Massnahmen haben wir eine gute Balance zwischen Zuverlässigkeit und Implementationsaufwand erreicht.

Verbindungswiederherstellung

Gemäss den nicht-funktionalen Anforderungen, muss sichergestellt werden, dass nach einem Unterbruch der Internetverbindung eine Verbindungswiederherstellung stattfindet. Sobald die Verbindung mit dem Internet wieder besteht, soll nach 30 Sekunden wieder eine Verbindung zum Message Broker bestehen.

Bei der Konfiguration der Verbindungswiederherstellung bestand die Wahl zwischen fixen oder inkrementellen Zeitabständen, nach denen eine Wiederherstellung versucht wird. Um das exakte Verhalten zu evaluieren, wurden Versuche gemacht, bei denen die Verbindung für drei Sekunden unterbrochen wurde (siehe Tabelle 4.1).

Da beide Varianten die nicht-funktionalen Anforderungen erfüllen, haben wir uns nach den Versuchen für ein inkrementelles Intervall entschieden, um das Smartphone nicht unnötig zu belasten.

Versuch	Mit inkrementellem Intervall [s]	Mit fixem Intervall [s]
1	17	7
2	7	7
3	9	7
4	11	7
5	10	7

Tabelle 4.1.: Benötigte Zeit für Verbindungswiederherstellung

4.2.4. Customer-App

Gemäss Aufgabenstellung sollte ein Prototyp für eine App, mit welchem Bestellungen am System abgegeben können, entwickelt werden.
Anders als die Onboard-App, welche native mit Java entwickelt wurde und nur auf Android läuft, entschieden wir uns beim Customer-App für eine Cross-Plattform Lösung, welche Android und iOS unterstützt.

Auch wenn gemäss der Aufgabenstellung keine iOS-App gefordert war, stellte sich doch die Frage ob man die iPhone-Kunden ausschliessen soll und spätere Entwickler zwingt, einen grossen Teil des Codes in einer nativen iOS-App duplizieren zu müssen. Mit einem Marktanteil von 42.2% (Zahlen 2015) [13] von iOS Benutzern in der Schweiz, ist anzunehmen, dass zu einem späteren Zeitpunkt eine iPhone App implementiert werden muss.

Deshalb wurde auf Xamarin Forms gesetzt, welches ermöglicht, die ganzen Service-Klassen und einen Teil der Benutzeroberfläche für die verschiedenen Plattformen nur einmal zu implementieren.
Die App wurde in einer minimalen Ausbaustufe erstellt, erfüllt aber bereits alle nötigen funktionalen Anforderungen. In Abbildung 4.4 wird gezeigt, wie ein Benutzer durch die App navigieren kann.
Besonders wichtig sind die Anzeige des berechneten Abwurfpunktes vor der Bezahlung der Bestellung, sowie die Anzeige der Drohnenposition während des Anflugs.

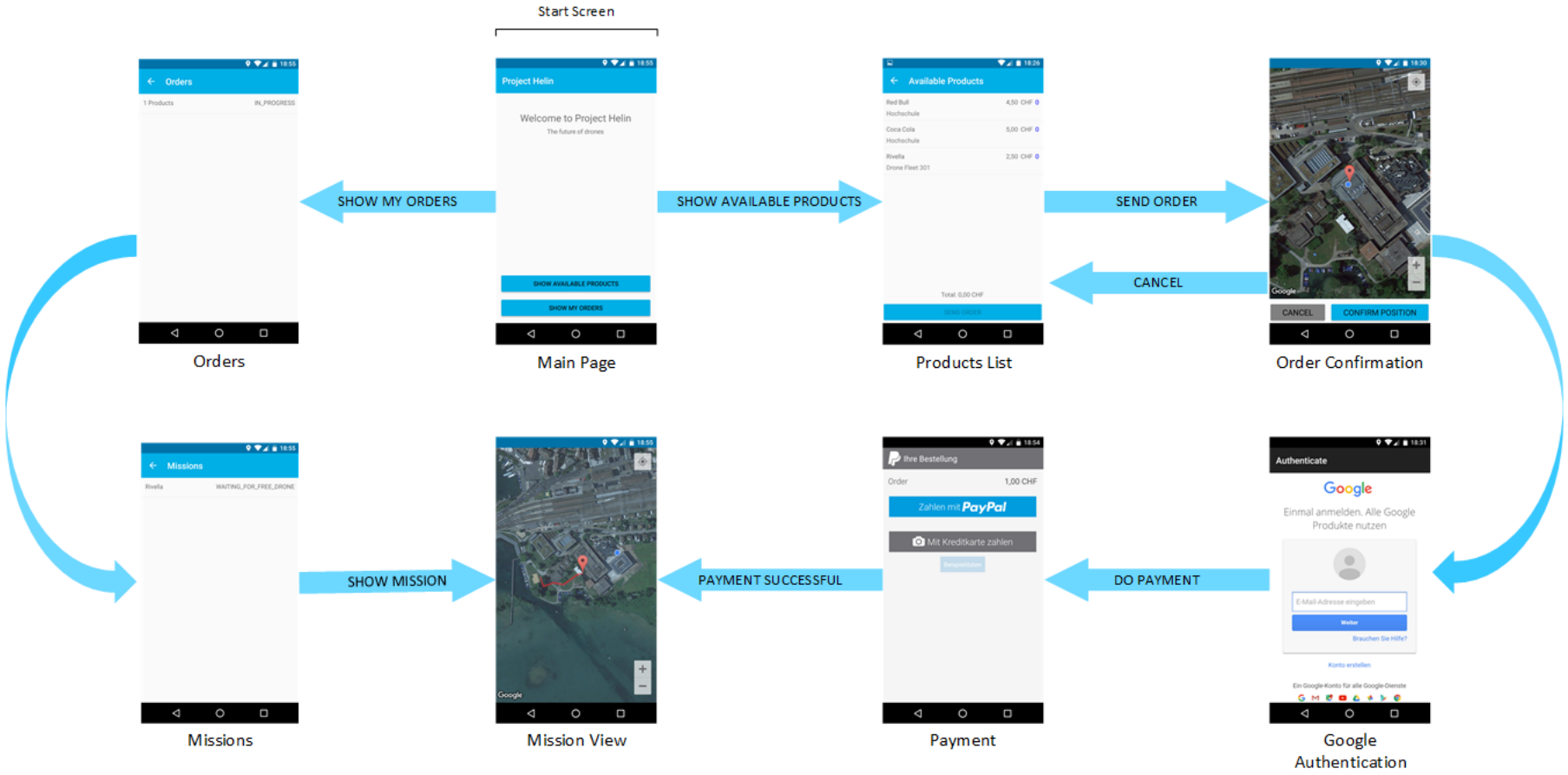


Abbildung 4.4.: Übersicht der Customer-App mit allen Verknüpfungen zwischen den Screens

Verwendete Bibliotheken

Name	Verwendungszweck	Version	Lizenz
PayPal Forms	PayPal Integration für Xamarin.Forms	2.0.4	MIT
Xamarin Forms	Plattformübergreifende Komponente für Xamarin	2.2.0.45	https://www.xamarin.com/license
Websocket.PCL	Plattformübergreifende WebSocket Anbindung	1.1.9	MIT
Newtonsoft.Json	JSON Konverter	8.0.3	MIT

Bemerkungen aus der Evaluation oben stehender Komponenten:

- **PayPal Forms:** Einziges PayPal Plugin für Xamarin.Forms
- **Xamarin.Forms:** Ermöglicht Crossplattform UI
- **Websocket.PCL:** Einzige gut gewartete Implementation von WebSockets in Xamarin
- **Newtonsoft.Json:** Die Bibliothek hat die höchste Verwendungsrate

4.2.5. Datenbank

Die Anforderungen an das Datenbank Management System (DBMS) für dieses Projekt, können wie folgt zusammengefasst werden:

- Geobasierte Objekte abzuspeichern muss möglich sein (Speicherung von Zonen)
- Anbindung an das Play-Framework muss möglich sein

Es gibt zwei ausgereifte Produkte, die diese Anforderungen erfüllen: Oracle mit Oracle-Spatial und PostgreSQL mit PostGIS. Bei Oracle-Spatial und PostGIS handelt es sich um Erweiterungen, die im DMBS eingebunden werden können und es ermöglichen, geobasierte Daten abzuspeichern und spezialisierte Abfragen darauf auszuführen.

Die Entscheidung zu Gunsten von PostgreSQL wurde getroffen, da zwei Teammitglieder bereits Erfahrungen mit dem System vorweisen können und es sich, wie bei unserem Projekt, um eine Open Source Software handelt.

Zusätzlich wurde für PostGIS die Erweiterung SFCGAL verwendet, welche die Geometriebibliothek CGAL in PostGIS verfügbar macht. Dies ermöglicht noch weitere Funktionen in der Datenbank zu nutzen, welche auf geometrischen Objekten ausgeführt werden können.

4.2.6. Sicherheit

Es ist essentiell, dass die Sicherheit bei der Kommunikation zwischen Server und Drohne gewährleistet ist, da sonst die Kontrolle über eine Drohne übernommen werden könnte (siehe NFR 2.4.5). Um diese Sicherheit zu gewährleisten wurde ein Prozess eingeführt, der es ermöglicht die Zugangsdaten für den RabbitMQ Message Broker und die Queue-Namen geheim zu halten:

Die Onboard-App registriert sich über einen HTTPS Request beim Server. Bei erfolgreicher Registrierung der Drohne, werden ein Token sowie der Benutzer und das Kennwort für die Verbindung zu RabbitMQ an die Onboard-App geschickt. Mit diesen Informationen kann sich die App mit dem Messaging Broker über eine verschlüsselte Leitung verbinden.

Für die darauf folgende Kommunikation mit dem Broker, werden mit Hilfe des Tokens zwei Queues erzeugt:

- {token}-Drone-To-Server
- {token}-Server-To-Drone

Über diese Queues können jetzt Nachrichten ausgetauscht werden.

Ein Angreifer müsste mit dieser Methode den Token herausfinden und auch die Zugangsdaten für den Broker kennen, um mit einer Drohne zu kommunizieren. Falls jemand es aber doch schaffen würde, die Benutzerdaten auszulesen, gibt es trotzdem noch 2^{122} mögliche Tokens, die durchprobiert werden müssten.

4.3. Projektgrösse

Folgende Statistiken wurden am Ende der Arbeit erstellt um einen Eindruck der Grösse des Projekts zu erhalten.

Metrik	**Server**	**Onboard-App**	**Customer-App**	**Total**
Lines of Code	12071	2174	1188	15433
Anteil Lines of Code Tests	29.34%	13.58%	0.00%	24.85%
Anteil Lines of Code Kommentare	4.03%	1.33%	0.00%	3.18%
Anzahl Klassen	174	49	36	259
Anzahl Methoden mit hoher essentieller Komplexität	1.73%	1.57%	N/A	1.46%

Tabelle 4.2.: Code Metriken nach Abschluss der Arbeit

In der Tabelle 4.2 ist vor allem der geringe Anteil von Methoden mit hoher essentieller Komplexität[20] hervorzuheben. Alle Methoden mit hoher Komplexität wurden vom Team geprüft, konnten aber nicht mehr verbessert werden. Hauptsächlich sind dies 'equals' Methoden von grösseren Data Transfer Objects (DTOs) oder es handelt sich um MessageConverter, die einfach alle verschiedenen Messagetypen verschieden behandeln müssen. Das folgende Beispiel zeigt die Methode mit der höchsten zyklomatischen Komplexität der Server-Applikation (Essentielle Komplexität = 10, Zyklomatische Komplexität = 10).

```
private Message parseMessageWithoutCare(String messageAsJson) {
   ...
   switch (payloadType) {
      case ConfirmCargoLoaded:
         return gson.fromJson(messageAsJson, ConfirmCargoLoaded.class);
      case NotifyCargoDrop:
         return gson.fromJson(messageAsJson, NotifyCargoDrop.class);
      case DroneInfo:
         return gson.fromJson(messageAsJson, DroneInfoMessage.class);
      case DroneDto:
         return gson.fromJson(messageAsJson, DroneDtoMessage.class);
      case AssignMission:
         return gson.fromJson(messageAsJson, AssignMissionMessage.class);
      case FinalAssignMission:
         return gson.fromJson(messageAsJson, FinalAssignMissionMessage.class);
      case ConfirmMission:
         return gson.fromJson(messageAsJson, ConfirmMissionMessage.class);
      case FinishedMission:
         return gson.fromJson(messageAsJson, FinishedMissionMessage.class);
      case DroneActiveState:
         return gson.fromJson(messageAsJson, DroneActiveStateMessage.class);
   }
}
```

Da diese Methode gut strukturiert ist und auch gut verstanden werden kann, gibt es keinen Grund, diese auf Grund der Metrik anzupassen.

4.4. Qualitätssicherung

4.4.1. Prozesse

Zur Qualitätssicherung wurde im Projektmanagement-Tool neben Todo, In Progress und Done ein neuer Quality-Assurance-Status für die Issues eingeführt. Alle Issues in diesem Status mussten von einem anderen Teammitglied überprüft werden, bevor sie auf Done geschoben werden konnten.

Die Überprüfung eines Issues beinhaltet folgende Aufgaben:

- Akzeptanzkriterien aus der User-Story manuell prüfen
- Code-Review durchführen (Intensität je nach Grösse und Komplexität des geschriebenen Codes)
- Code auf Verstösse gegen Style-Guide prüfen
- Tests prüfen

Damit konnten viele Fehler früh entdeckt werden und die Code-Qualität konnte über die ganze Projektdauer konstant gehalten werden.

4.4.2. Continuous Integration

Als Build-Server haben wir TeamCity verwendet. Dort sind alle Projekte (ausser Customer-App) jeweils mit einem Build für den develop- und master-branch eingerichtet. Alle Builds führten die Tests aus und produzierten ein entsprechendes Artifact für das Deployment. Das Deployment für den Server wurde ebenfalls automatisiert.

4.4.3. Testing

Je nach Plattform wurden andere Formen von Tests durchgeführt:

Server

Auf dem Server wurden vor allem E2E-Tests und Integration-Tests verwendet, um die Funktionalität zu prüfen. Wir haben uns bei den meisten Komponenten bewusst gegen Unit-Tests entschieden, da auf dem Server nur wenig Logik zu finden ist, die nicht von der Datenbank oder der RabbitMQ-Connection abhängt. Unit-Tests hätten deswegen nur einen ganz kleinen Teil der Anwendungen abdecken können und wären in den meisten Fällen sehr aufwendig gewesen. Bei der Routenberechnung hingegen konnte sehr gut mit Unit-Tests gearbeitet werden.

Wir haben darauf geachtet, dass immer nur die minimale Integrationsstufe gewählt wurde. Für die Simulation eines Benutzers (E2E) wurde Selenium verwendet, das wiederum einen Firefox Browser verwendet. Für die Api- und Messaging-Controller wurden Integrationtests verwendet, welche keinen Browser benötigen.

Es wurde eine **durchschnittliche Testabdeckung von 72%** über das ganze Projekt erreicht. In wichtigen Packages liegen diese aber meist über 85%.

Onboard-App

Beim Onboard App wurden nur Unit-Tests ausgeführt. Der Aufbau von E2E Tests ist denkbar, hätte den Rahmen dieser Arbeit aber gesprengt. Deshalb wurden hauptsächlich die Message-Handler und die Daten-Mapper getestet. Diese haben eine **Testabdeckung von 86%**.

Customer-App

Bei der Customer-App wurden keine Tests eingerichtet, da es sich um einen Prototypen handelt.

4.5. Flugrouten

Die zentrale Aufgabe des Projektes ist es Drohnen zu verwalten, die sich in einem geografischen Gebiet autonom und sicher bewegen können. Dieses Gebiet, beispielsweise die HSR (siehe Abb. 4.5) kann Hindernisse wie Gebäude oder Bäume enthalten, welche um- oder überflogen werden müssen.

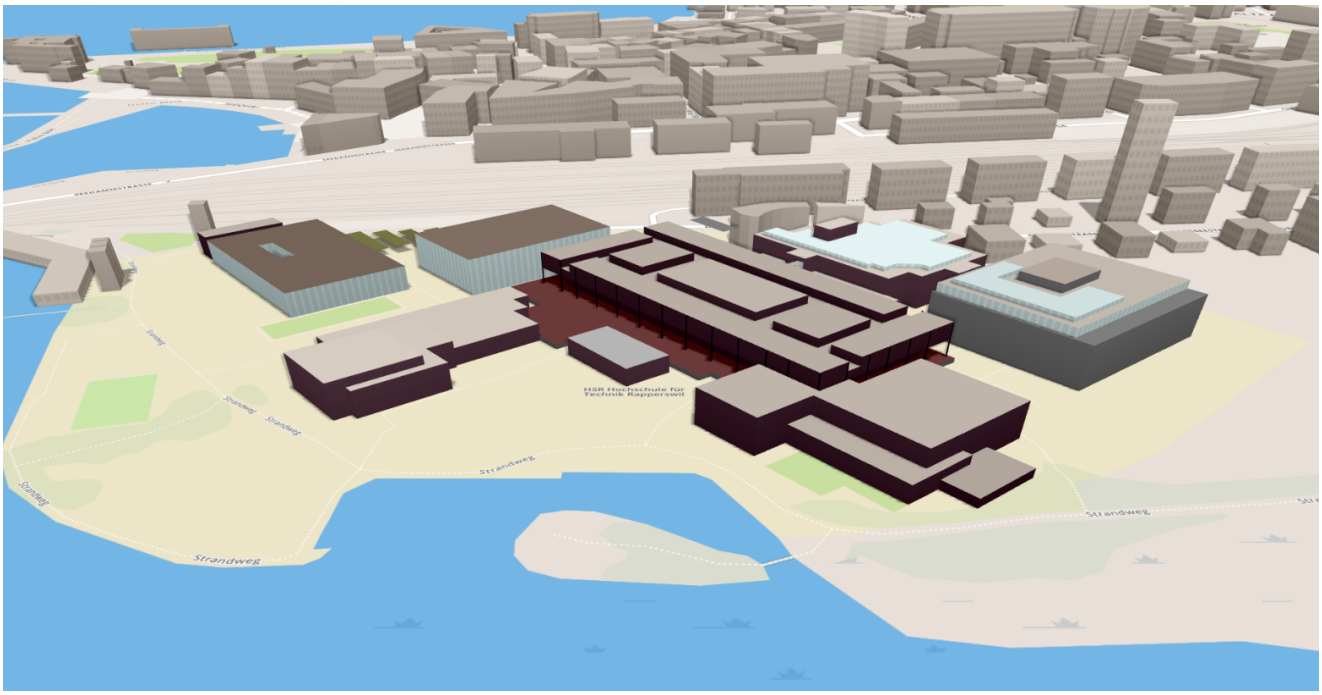


Abbildung 4.5.: Topologische Ansicht des Campus HSR

4.5.1. Das Zonen Model

Um Kollisionen mit statischen Objekten zu vermeiden und sichere Flugwege zu erhalten, wurde ein Zonen Model eingeführt. Dieses ermöglicht dem Anbieter zu definieren, wo geflogen werden kann und wie hoch dort geflogen werden muss. Die Abbildung 4.6 zeigt die Benutzeroberfläche für die Zonendefinition anhand des HSR-Campus.

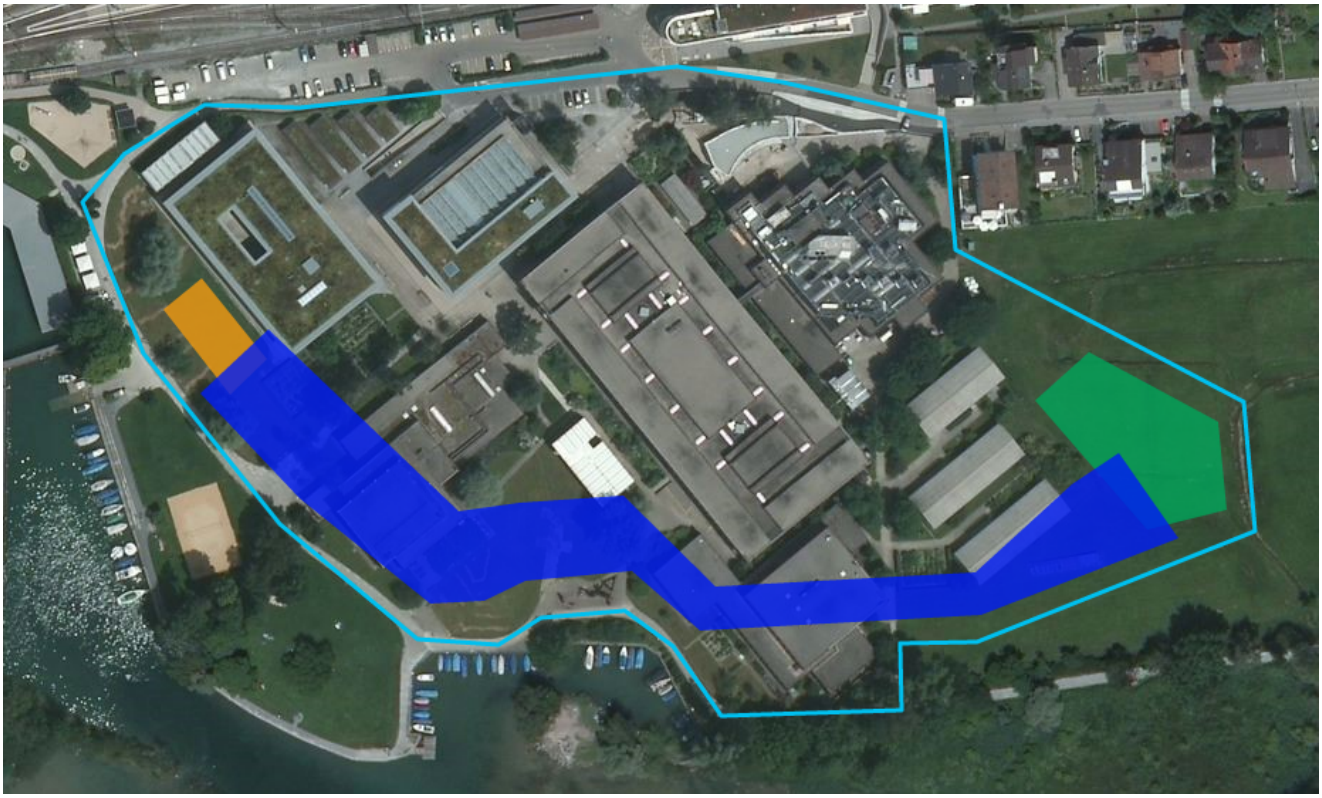


Abbildung 4.6.: Verschiedene Flugzonen am Beispiel der HSR

Jedes Polygon bildet eine Zone ab. Jede Zone enthält eine Flughöhe sowie einen Typ:

- **Order Zone:** Zone in welchem die Produkte des Projekts bestellt werden können. (*hell blauer Rahmen*)
- **Loading Zone:** Zone in welcher die Drohne beladen wird. (*orange*)
- **Delivery Zone:** Zone in der geliefert und geflogen werden darf. (*grün*)
- **Flight Zone:** Zone in der nur geflogen aber nicht geliefert werden darf. (*blau*)

4.5.2. Von der Zone zum Graphen

Normalerweise wird eine Routenberechnung mit Hilfe eines Graphen durchgeführt, der durch die möglichen Wege (z.B. Strassen) definiert ist [18]. Da die Zonen aber eine Fläche bilden, mussten diese zuerst in einen Graphen umgewandelt werden.

Folgende Lösungsmöglichkeiten wurden evaluiert:

Erstes Konzept: Direkte Route

Dieses Konzept umfasste den Ansatz den Start und den Endpunkt direkt zu verbinden, und bei einem Verlassen des Polygons dem Rand zu folgen (siehe Abb. 4.7).

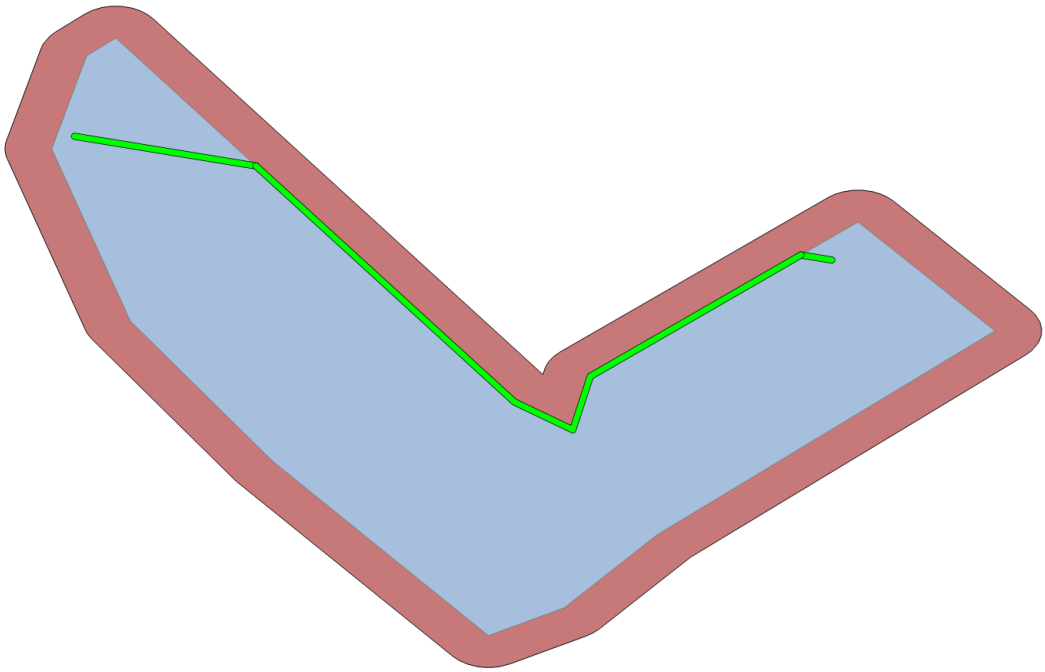


Abbildung 4.7.: Erstes Konzept

Es wurde ausserdem ein Rand (*Rot*) hinzugefügt. Diese Massnahme wurde getroffen, um allfällige GPS Ungenauigkeiten zu berücksichtigen.

Der Nachteil dieser Lösung zeigt sich bei grösseren Polygonen, wo unnötigerweise dem Rand entlang geflogen wird, obwohl ein Flug durch die Mitte viel sicherer wäre.

Zweites Konzept: Visibility Graph

Das nächste Konzept orientiert sich an einem 'Visibility Graphen'. [10]

> Using visibility graphs for determining the shortest path is very practical and intuitive. The visibility graph of a set of nonintersecting polygonal obstacles in the plane is an undirected graph whose vertices are the vertices of the obstacles and whose edges are pairs of vertices such that the open line segment between each two vertices does not intersect any of the obstacles.

In unserem Fall haben wir den Gedanken umgedreht. Wir haben den Graphen in einem Polygon mit potentiellen Löchern berechnet. Diese Löcher stellen Enklaven für Bäume oder Gebäude dar. Das Konzept ist aber identisch, angewendet auf unser Beispiel Polygon ergibt sich folgende Abstraktion:

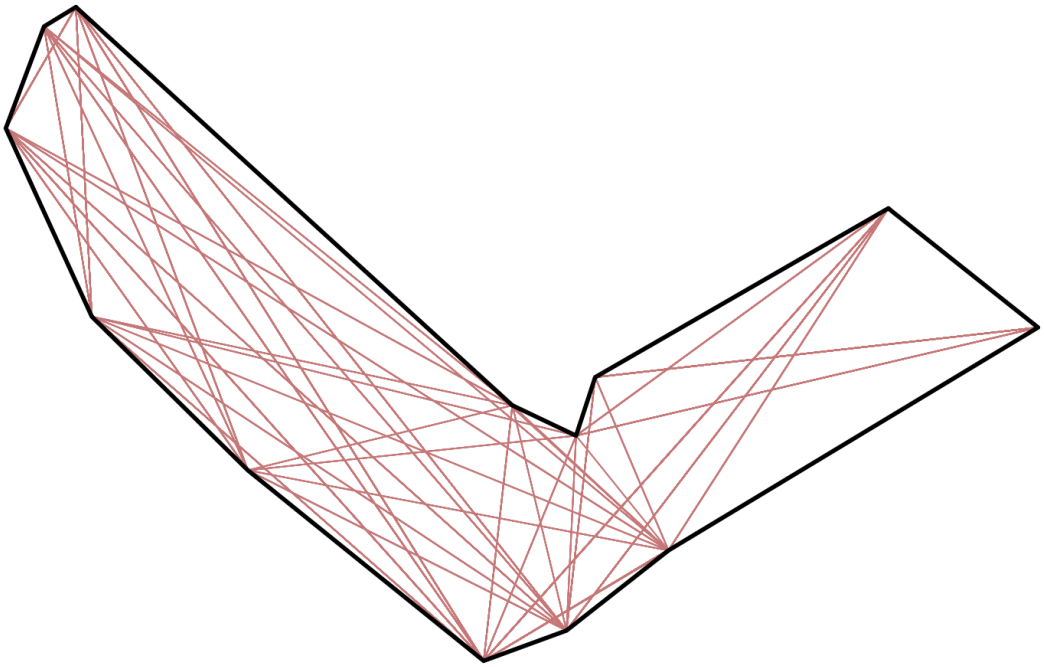


Abbildung 4.8.: Visibility Graph in einer Flugzone

Wie in der Abbildung 4.8 gezeigt wird, verlaufen deutlich weniger Flugrouten entlang der Kante des Polygons.

Dies ist eine deutliche Verbesserung gegenüber dem ersten Konzept. Der Nachteil ist, dass die innere Ecke des L-förmigen Polygons sicher angeflogen wird, da es die vermeintlich kürzeste Route in einem nächsten Schritt darstellen würde. Aus diesem Grund haben wir uns gegen den Einsatz dieses Konzepts entschieden.

Drittes Konzept: Straight Skeleton

Da die Pfadfindung von UAV bereits ein weit erarbeitetes und erforschtes Gebiet ist, waren wir der Überzeugung, dass bereits verwendete Konzepte vereinfacht und abstrahiert werden konnten, um unser Problem zu lösen. Während vollständig autonome Drohnen an Hindernissen vorbei navigieren müssen, können wir uns auf einen bestehenden Flugkorridor verlassen.

> The idea behind this method is to assign a function similar to the electrostatic potential to each obstacle and then derive the topological structure of the free space in the form of minimum potential valleys. The robot is positioned at the start point, and then the goal generates a strong attractive force. By the action of the force, the robot moves along the steepest descent of the potential to the goal, at the same time the obstacle generates a repulsive force to keep robot away from colliding with them

Die Kernaussage des zitierten Papers [11] ist, dass ein Pfad den grösstmöglichen Abstand zu den Hindernissen aufweisen soll. Auf unseren Fall übertragen, sind die Polygonränder die Hindernisse. Somit muss gewährleistet werden, dass die Drohne den grössten Abstand zu den Rändern aufweist. Aus diesem Grund erscheint diese Lösung für uns sehr geeignet, da sie mittenbetonte Routen vorschlägt.

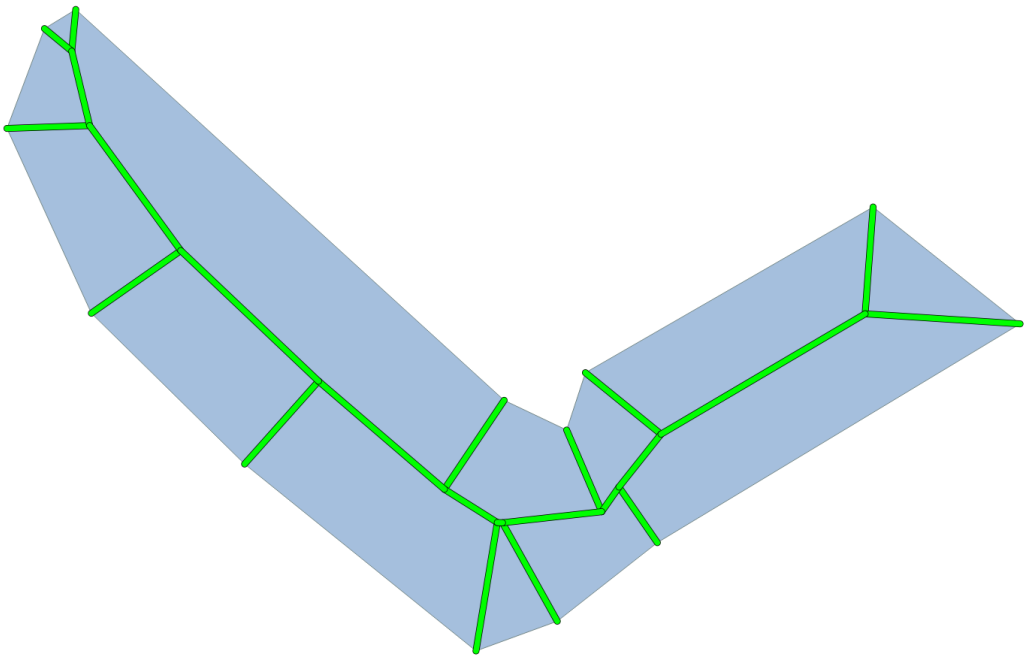


Abbildung 4.9.: Skeleton in Polygon

Die Abbildung 4.9 illustriert die Lösung mithilfe eines Skeletons. Dieses Konzept kann nachvollzogen werden, indem man sich das Polygon 3-dimensional vorstellt und dann von der Mitte aus, parallel zu den Kanten schneidet. Der entstandene Grat ist hier als Linie sichtbar und wird als Skeleton bezeichnet.

Der Felkel-Algorithmus, welcher diesen Skeleton aus einem 2-dimensionalen Polygon berechnet ist bereits in CGAL implementiert [1] und war uns im weiteren Verlauf des Projektes von grossem Nutzen.

4.5.3. Routing Algorithmus

Nach der Herleitung des Graphen aus den Zonen, musste nun der passende Algorithmus gefunden werden, um einen möglichst effizienten Weg von A nach B zu finden. Es kamen vier Algorithmen in Frage, wobei ein Algorithmus sich als besonders geeignet herausstellte.

- **BFS:** (*Breadth-first search*) Die Breitensuche durchsucht einen Baum in der Breite. Sie bringt garantiert eine Lösung, wenn eine Lösung vorhanden ist und kann problemlos mit Zyklen im Graph umgehen. Bei der Lösung handelt es sich aber nicht um die 'optimale Lösung' (kürzester Pfad). Die Lösung liefert den Pfad mit den wenigsten Knoten. [3]
- **DFS:** (*Depth-first search*) Die Tiefensuche fängt an einem Knoten an und arbeitet sich dann in die Tiefe. Das Problem ist, dass sie sehr anfällig auf Zyklen ist und ohne weitere Hilfsmittel (loop detection oder pruning) nicht terminieren kann. [3]
- **A-Star:** Der A-Star Algorithmus arbeitet mit einer Heuristik und funktioniert effizienter als die zwei bereits genannten Algorithmen. Allerdings ist der Aufwand für die Umsetzung deutlich höher. [3]
- **Dijkstra:** Der Dijkstra-Algorithmus ist einer der bekanntesten Algorithmen in diesem Bereich. Der Vorteil ist, dass er grundsätzlich ohne Heuristik auskommt und robust mit Zyklen umgehen kann. Ausserdem können die Kanten mit einem Gewicht versehen werden und dadurch, der kürzeste Weg berechnet werden.

Wir haben uns für den Dijkstra Algorithmus entschieden. Er liefert ohne Metrik eine ähnliche Lösung wie die BFS-Suche, kann aber im Nachhinein erweitert werden um effizientere Routen zu liefern.

4.5.4. Höhenhandling

Damit die Route 2-Dimensional berechnet werden kann, werden die Höhen erst zum Schluss berücksichtigt. Das Höhenhandling bildet den abschliessenden Schritt bei der Berechnung einer Route und wird in zwei Schritten durchgeführt:

- **Eindeutigkeit der Höhe:** In einem ersten Schritt werden die Zonen nach Höhe absteigend sortiert und Überlagerungen zugunsten der höheren Zone entfernt.
- **Einfügen von Schnittpunkten** An jedem Schnittpunkt des Pfads mit der Kante eines Polygons, wird ein Punkt eingefügt und mit der jeweiligen Zonenhöhe versehen.

Wie gemäss der Abbildung 4.10 ersichtlich ist, wird auf dem Übergang zwischen zwei Zonen ein Punkt eingefügt der für die Höhenänderung verwendet wird.

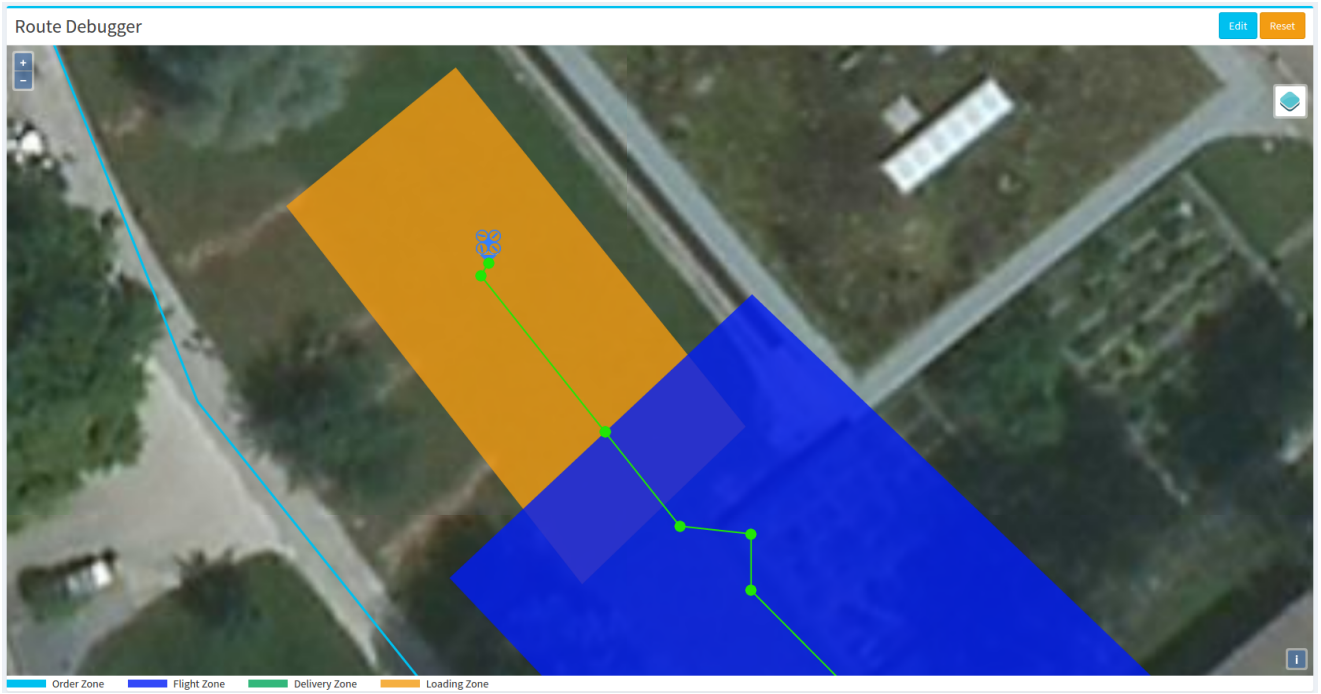


Abbildung 4.10.: Beispiel der Höhe am Übergang zweier Polygone

4.5.5. Wahl der Rückroute

Der berechnete Pfad wird invertiert und bis auf den Abwurfpunkt am Ende der Route eingefügt. So ist garantiert, dass die Drohne auf dem gleichen Weg zurückfliegt.

5. Versuchsaufbau

5.1. Einleitung

Um die vom Team erstellte Plattform auch unter realen Bedingungen testen zu können, wurde ein Versuchsaufbau mit zwei Drohnen erstellt, welche es ermöglichen Produkte zu transportieren und über dem Kunden mit einem Fallschirm abzuwerfen. Der Aufbau der Drohnen beinhaltete die Zusammenstellung der passenden Komponenten, den Zusammenbau der Drohnen und die Planung und Erstellung von zusätzlichen Halterungen für die obengenannten Anforderungen.

5.1.1. Kommunikationshardware

Da die Drohne eine Verbindung zum Server benötigt, muss geeignete Kommunikationshardware installiert werden. Aus folgenden Gründen haben wir uns für ein Android-Smartphone entschieden:

- Der Status der Drohne und Anweisungen zur Beladung können angezeigt werden
- Die Anbindung an den Flight-Controller ist über eine bestehende API möglich
- Das Android-Betriebssystem stellt die Kommunikation über WLAN oder GSM zur Verfügung

Falls die Beladung automatisiert wird, oder keine Beladung notwendig ist, benötigt man nicht zwingend ein Smartphone. Um Gewicht und Platz auf der Drohne zu sparen gibt es folgende Alternativen:

- **Raspberry PI**
  Bereits ab dem leichtgewichtigen Raspberry PI Zero sind die nötigen Funktionen vorhanden. Zusätzlich muss hier auf ein GSM Modem zurückgegriffen werden um eine permanente Internetverbindung aufzubauen. Bei Verwendung eines Raspberry PI 3 kann sogar auf ein eingebautes WLAN Interface verwendet werden.
- **Arduino**
  Es könnte eine Arduino Plattform verwendet werden mit einem GSM Shield 2, das eine mobile Internetverbindung ermöglicht. Eine WLAN Verbindung ist nur mit einem zusätzlichen Modul möglich.
- **Pixhawk Controller mit GSM Modul**
  Es existieren GSM Module, wie etwa DroneCell [14], die am Pixhawk angeschlossen werden können. Diese ersetzen die Telemetrie-Funkverbindung und ermöglichen die Kommunikation über 4G.

Die ersten zwei Alternativen benötigen zusätzlich eine externe Stromversorgung. Somit muss ein BEC (Battery Eliminator Circuit) verwendet werden, um eine stabile Spannung für die jeweilige Komponente zu liefern.

5.2. Ablieferungskonzept

Um ein möglichst sicheres und einfaches Ablieferungsverfahren zu finden, wurden folgende Optionen in Betracht gezogen.

5.2.1. Landung ohne automatischen Abwurf

Bei diesem Konzept landet die Drohne an der Position des Kunden und dieser kann dann selbst das Paket von der Drohne lösen. Er wird dann mit Hilfe der Onboard-App aufgefordert den Erhalt der Bestellung zu bestätigen. Nach der Bestätigung startet die Drohne einen Countdown, hebt danach ab und fliegt zur Ausgangsposition zurück. Der Kunde kann den Countdown unterbrechen und erneut starten.

Vorteile:

- Kunde bestätigt Lieferung
- Ware kann keinen Schaden nehmen
- Kunde kann entscheiden, wann es sicher genug ist, damit die Drohne starten kann

Nachteile:

- Drohne kann Personen verletzen, während sie sich in Bodennähe befindet
- Aufwendiges Handling auf der Onboard-App
- Landung kann, je nach Gelände, schwierig sein
- Fremde Personen haben physischen Zugriff auf die Drohne

5.2.2. Landung mit automatischem Abwurf

Dieses System ermöglicht es, die Ware abzuwerfen ohne einen Fallschirm zu verwenden. Dabei landet die Drohne, löst die Ladung und startet sofort wieder. Es gibt keine Interaktion mit dem Kunden.
Vorteile:

- Ware kann keinen Schaden nehmen

Nachteile:

- Drohne kann Personen verletzen, während sie sich in Bodennähe befindet
- Landung kann, je nach Gelände, schwierig sein
- Fremde Personen haben physischen Zugriff auf die Drohne

5.2.3. Abwurf mit Fallschirm

Dabei wird ein Fallschirm an der Ware befestigt. Die Drohne löst die Ladung in einer vorgegebenen Höhe und die Ladung schwebt zu Boden.

Vorteile:

- Drohne kann keine Personen verletzen
- Drohne ist ausser Reichweite von Personen
- Einfaches Handling auf der Onboard-App
- Mechanismus erlaubt auch Landung mit automatischem Abwurf

Nachteile:

- Ware kann Schaden nehmen
- Ware kann Personen verletzen bei Versagen des Fallschirms
- Einweg-Fallschirme (Kosten und Abfall)
- Windlage kann Fallschirm-Flugrichtung beeinflussen

5.2.4. Entscheidung

Kriterium	Gewicht	Landung ohne Abwurf		Landung mit Abwurf		Abwurf mit Fallschirm	
		Bewertung	Gesamt	Bewertung	Gesamt	Bewertung	Gesamt
Personenschäden	40%	1	0.4	1	0.4	3	1.2
Schaden an der Ware	20%	3	0.6	3	0.6	1	0.2
Schäden an der Drohne	10%	1	0.1	1	0.1	3	0.3
Einfachheit Prozess	10%	1	0.1	2	0.2	3	0.3
Einfachheit mech. System	10%	3	0.3	2	0.2	2	0.2
Erweiterbarkeit	10%	1	0.1	2	0.2	3	0.3
Gesamt	100%		1.6		1.7		**2.5**

Abbildung 5.1.: Nutzwertanalyse der verschiedenen Liefervarianten

Die Nutzwertanalyse in Abbildung 5.1 zeigt klar, dass die letzte Variante die meisten Vorteile bringt. Wichtig ist auch, dass ein Abwurf im gelandeten Zustand weiterhin möglich bleibt, ohne die mechanischen Komponenten zu verändern. Deshalb haben wir uns für einen Abwurf mit dem Fallschirm entschieden.

5.2.5. Fallschirmgrösse

Wir nehmen an, dass eine 0.5 Liter PET-Flasche die von einem ein Meter hohen Tisch (h = Höhe) fällt, unbeschädigt bleibt. Basierend auf dieser Annahme haben wir die Geschwindigkeit für diesen Aufprall errechnet.

$$\begin{aligned} v &= \sqrt{2gh} \\ v &= \sqrt{2 \cdot 9.81 \frac{\mathrm{m}}{\mathrm{s}^2} \cdot 1.0\mathrm{m}} \\ v &= 4.429 \frac{\mathrm{m}}{\mathrm{s}} \approx 4.5 \frac{\mathrm{m}}{\mathrm{s}} \end{aligned} \tag{5.1}$$

Nun musste der Fallschirm dimensioniert werden.

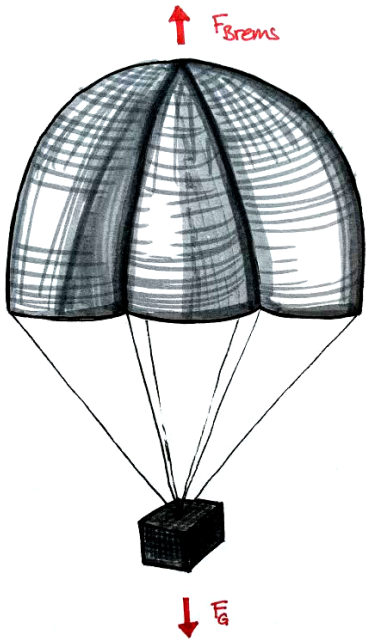


Abbildung 5.2.: Skizze des physikalischen Modells eines Fallschirms

Einfachheitshalber wird angenommen, dass der Fallschirm die Form einer Kugelkalotte (Abb. 5.2) hat und ein Gewicht von 500g trägt. Die Fläche des Fallschirms kann anhand der folgenden Gleichung ausgedrückt werden:

$$\begin{aligned} F_{\mathrm{Brems}} &= F_{\mathrm{G}} \\ c_w \cdot A \cdot \frac{\rho}{2} \cdot v_{\mathrm{sink}}^2 &= m_{\mathrm{Pet}} \cdot g \\ A &= \frac{2 \cdot m_{\mathrm{Pet}} \cdot g}{\rho \cdot c_w \cdot v_{\mathrm{sink}}^2} \end{aligned} \tag{5.2}$$

Nun lässt sich der Durchmesser (d) über die Öffnungsfläche $A = \pi \cdot r^2$ berechnen. Wir nähern an, dass der Öffnungsdurchmesser dem Fallschirmdurchmesser entspricht.

$$\begin{aligned} \pi \cdot r^2 &= \frac{2 \cdot m_{\text{Pet}} \cdot g}{\rho \cdot c_w \cdot v_{\text{sink}}^2} \\ r &= \sqrt{\frac{2 \cdot m_{\text{Pet}} \cdot g}{\rho \cdot c_w \cdot v_{\text{sink}}^2 \cdot \pi}} \\ d &= 2 \cdot \sqrt{\frac{2 \cdot m_{\text{Pet}} \cdot g}{\rho \cdot c_w \cdot v_{\text{sink}}^2 \cdot \pi}} \end{aligned} \tag{5.3}$$

Für die Luftdichte ρ wurde der Standardwert von $1.225 \frac{\text{kg}}{\text{m}^3}$ genommen. Für den Strömungswiderstandskoeffizenten c_w wird vom Literaturwert für die konkave Seite der Halbkugelschale abgesehen und der Erfahrungswert[1] 1.80 verwendet.

$$\begin{aligned} d &= 2 \cdot \sqrt{\frac{2 \cdot 0.5\text{kg} \cdot 9.81 \frac{\text{m}}{\text{s}^2}}{1.225 \frac{\text{kg}}{\text{m}^3} \cdot 1.80 \cdot 4.5 \frac{\text{m}}{\text{s}} \cdot \pi}} \\ d &= 1.12\text{m} \end{aligned} \tag{5.4}$$

Der Durchmesser des Prototypenfallschirms muss somit etwa 1.12m betragen.

[1] Herr Prof. Dr. F. Müller hat uns darauf hingewiesen, dass aus seinen Raketenversuchen ein c_w-Wert von 1.80 als gute Annäherung verwendet werden kann.

5.3. Multicopter Hardware

Bei der Zusammenstellung der Komponenten haben wir vor allem auf die gute Verfügbarkeit von Ersatzteilen und einen hohen Marktanteil der Komponenten geachtet. Besonders wichtig erschien uns, dass ein Anbieter nicht an einen Hersteller gebunden ist, sondern die Drohne an seine Bedürfnisse (Zuladung, Reichweite, Geschwindigkeit) anpassen kann. Es sollten also möglichst wenige Voraussetzungen bestehen, um die Plattform nutzen zu können. Folgende Anforderungen sind allerdings unabdingbar:

- Flight-Controller muss MAVLink Protokoll unterstützen.
- Onboard-App muss über MAVLink mit dem Flight-Controller verbunden sein. (Über USB oder eine Funk-Telemetrieverbindung.)

5.3.1. Teile-Liste

Bezeichnung	**Bezugsquelle**		**Preis**
DJI F450 ARF Kit	conrad.ch	CHF	229.95
DJI F450/F550 Landegestell	conrad.ch	CHF	24.95
3DR Pixhawk	3dr.com	USD	199.99
3DR uBlox GPS with Compass Kit	3dr.com	USD	89.99
Pixhawk external LED and USB	3dr.com	USD	20.00
FrSky X8+ Radio Transmitter	3dr.com	USD	89.99
FrSky X8R Receiver	hebu-shop.ch	CHF	39.95
Delock Micro USB OTG Kabel	digitec.ch	CHF	15.90
Multistar High Capacity 4S Akku (2x)	hobbyking.com	USD	50.50
APM Flight Controller Damping Platform	hobbyking.com	USD	2.69
Smartphone und GPS Halterung	eigener 3D-Print	CHF	27.50
Abwurfvorrichtung	eigener 3D-Print	CHF	25.50
Hitec Super Servo S-Bec (Stromversorgung Servo)	brack.ch	CHF	13.90
Tactic TSX10 (Servo für Abwurf)	brack.ch	CHF	14.90
Kleinmaterial	diverse	CHF	50.00
Total		**CHF/USD**	**≈ 900.00**

Es wurde ein Wechselkurs von 1:1 angenommen.

5.3.2. Frame und Antrieb

Der Frame, die Motoren und ESCs wurden als Kit gekauft. Es handelt sich dabei um ein DJI Flamewheel 450 Frame mit DJI 2312 960kV Motoren und ESeries 420 20A ESCs. Dieses Kit ist weltweit gut verfügbar und deshalb ideal geeignet um einen Versuchsaufbau zu erstellen.

Abbildung 5.3.: DJI F450 Flamewheel Kit

5.3.3. Flight-Controller

Der Flight-Controller ist das Herzstück eines Multicopters. Im Unterschied zu anderen ferngesteuerten Fahr- und Flugzeugen kann ein Multicopter nur über ein Fly-by-Wire System kontrolliert werden. Das heisst, alle Befehle, die von der Fernbedienung gesendet werden, müssen interpretiert und umgewandelt werden, damit die Motoren eine Bewegung in die gewünschte Richtung erzeugen können. In Kombination mit einem GPS Modul (Abb. 5.5) ermöglicht der Controller verschiedene Flugmodi, wie beispielsweise das Schweben an einem Punkt oder automatisches Abfliegen von Wegpunkten.

Als Flight-Controller setzen wir ein Pixhawk ein. Es ist sehr vielseitig und kann gut mit zusätzlichen Sensoren erweitert werden, ausserdem unterstützt es gängige Firmwares, die auch auf günstigeren Controllern laufen. Als Firmware für das Pixhawk setzen wir ArduCopter ein, da sie komplett Open Source ist und auch bei vielen anderen Projekten eingesetzt wird. Die Firmware unterstützt ausserdem das MAVLink Protokoll.

Abbildung 5.4.: Pixhawk Flight-Controller

Abbildung 5.5.: GPS-Modul für Pixhawk

5.3.4. Ausbaustufen

Während des Projekts wurde die Hardware laufend den Bedürfnissen angepasst. Daher sind mehrere Versionen der Drohne entstanden, die für die Versuche genutzt wurden und halfen, Risiken früh auszuschliessen.

Version 1

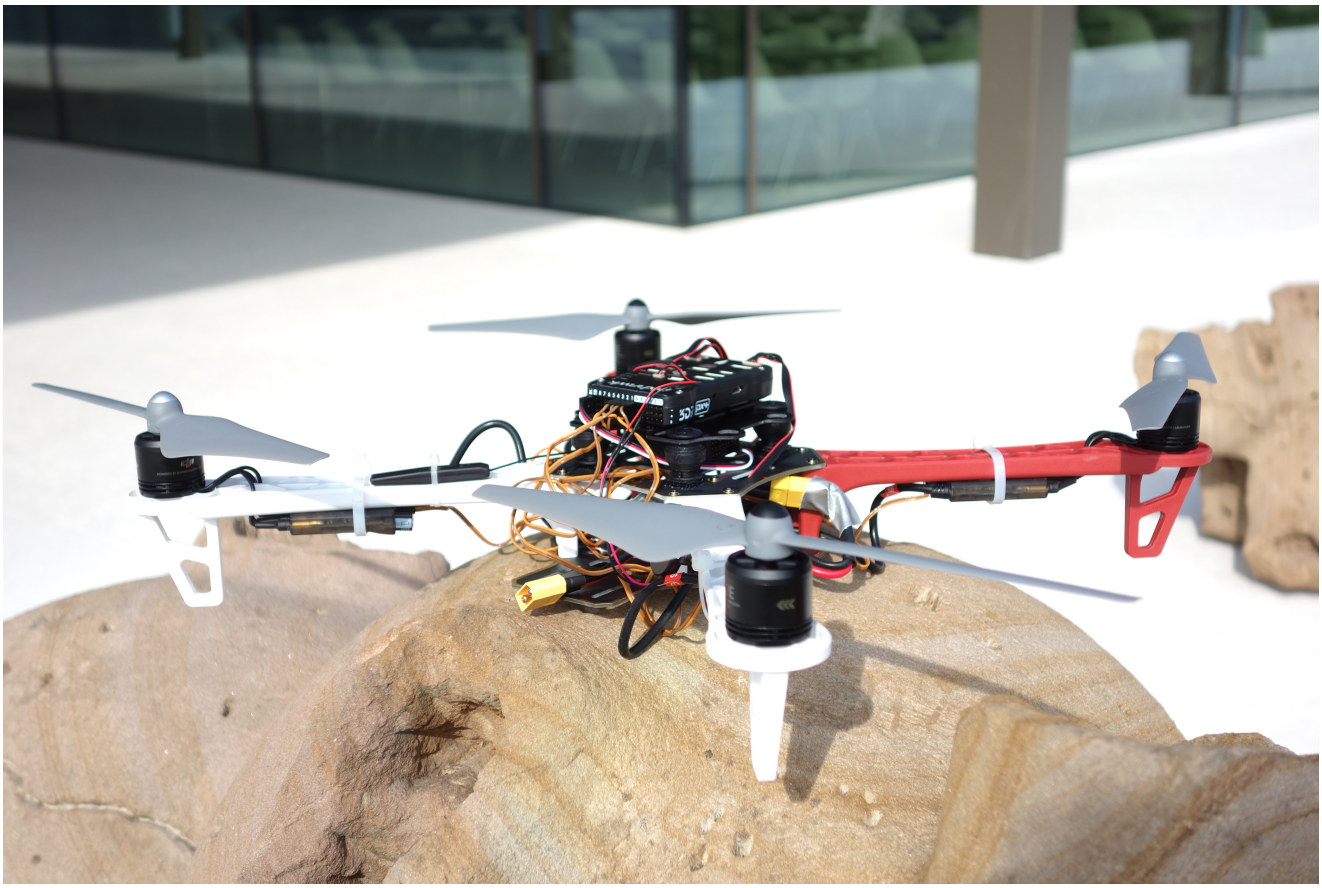


Abbildung 5.6.: Erster Prototyp ohne Landegestell und ohne Smartphone

Um das Zusammenspiel der Hardwarekomponenten zu testen und erste Erfahrungen mit dem GPS und den verschiedenen Flugmodi zu sammeln, wurden nur die nötigsten Teile installiert.

Version 2

Abbildung 5.7.: Drohnen Aufbau mit Smartphone

Um die Risiken R08 (Ardupilot Handhabung) und R09 (Ardupilot API) frühzeitig auszuschliessen (siehe Tabelle B.1), wurde das Smartphone provisorisch auf die Drohne montiert, und mit einem ersten Prototypen der Onboard-App getestet.

Nach dem Test wurde uns klar, dass eine Befestigung für das Smartphone erarbeitet werden musste, um es sicher zu transportieren und bedienbar zu positionieren.

Version 3

Abbildung 5.8.: Drohne mit Handy Halterung

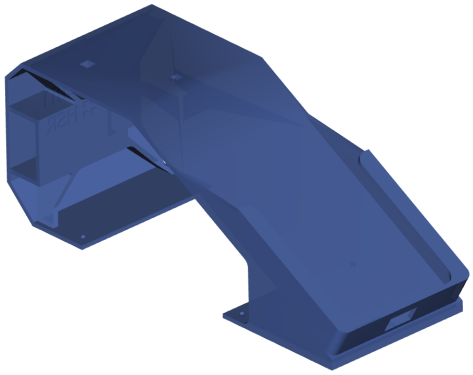


Abbildung 5.9.: Halterung Modell

Es wurde eine Halterung konzipiert (siehe Abb. 5.9), welche wir modellieren und im 3D-Druck Verfahren produzieren liessen. Die Handy-Halterung konnte auch gleich genutzt werden, um das GPS-Modul an einer passenden Stelle zu positionieren.

Version 4

Nachdem mit der Drohne zahlreiche autonome Testflüge unternommen wurden, musste eine Möglichkeit erarbeitet werden, wie die Lieferung an den Kunden gebracht werden kann (Abschnitt 5.2).

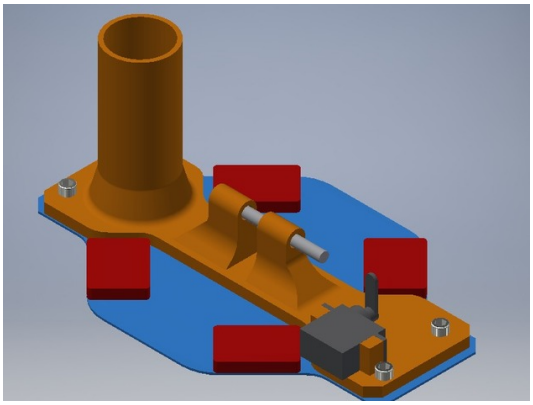


Abbildung 5.10.: Halterung

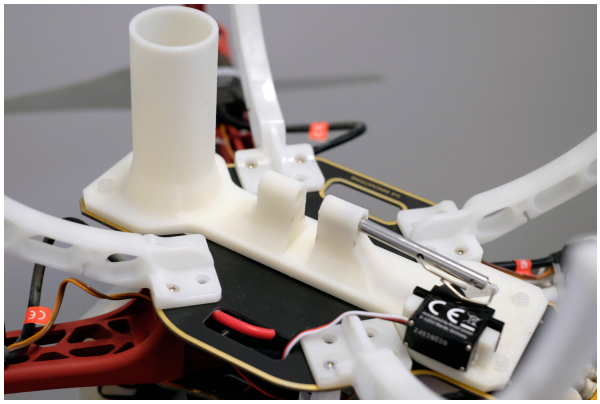


Abbildung 5.11.: Halterung montiert

Es wurde ein einfacher Mechanismus mit einem handelsüblichen Modelbau-Servo konstruiert. Die Abbildung 5.11 zeigt, den Mechanismus mit dem Stift, der vom Servo herausgezogen werden kann. Das Rohr auf der Vorrichtung dient zur Aufbewahrung des Fallschirms während des Flugs. Nach dem Lösen des Stifts zieht das Gewicht der Ladung den Fallschirm aus dem Rohr.

5.3.5. Tests

Um die Leistungsfähigkeit und die Einschränkungen eines solchen Multicopters zu testen, wurden diverse Experimente durchgeführt.

Akku Laufzeittest

Um die maximale Akku Laufzeit zu testen, wurden folgende Tests durchgeführt:

Akku	Zuladung	Laufzeit
3S	keine	16min 29s
2x 4S	500g	19min

Tragfähigkeitstests

Um das maximale Gewicht zu prüfen, welches auf unsere Drohnen geladen werden kann, wurden Tests mit dem Zielgewicht von 500g durchgeführt. Dies entspricht dem Gewicht einer 1/2-Liter Flasche oder einem leichten Defibrillator. Ausserdem war das Mobiltelefon (ca. 150g) während des Tests auf der Drohne angebracht.

Nutzlast	Akku Typ	Nötige Leistung	Erwartete Flugzeit	Subjektives Flugverhalten
500g	3S	ca. 75%	n.A.	Ziemlich Träge, mehr Gewicht wäre kritisch
500g	4S	ca. 45%	n.A.	Gewicht kaum Spürbar

Auch mit 3S Akkus ist es also möglich eine PET-Flasche zu transportieren. Allerdings empfehlen wir für Gewichte über 300g 4S-Akkus zu verwenden.

Aus den Tragfähigkeitstests schliessen wir, dass auch ein Defibrillator (siehe Aufgabenstellung) mit einer von uns erstellten Drohne transportierbar ist. Folgende Textpassage [5, S3] bestätigt, dass in einem ähnlichen Gewichtsbereich bereits Produkte existieren.

> I identified the two lightest weight defibrillators on the market in the U.S. at the time of the exploration (March 2013): the Schiller FRED EasyPort (600 grams) and the HeartSine samaritan PAD 300P (1100 grams). The former model is not designed for layperson use, but is far and away the lightest defibrillator available.

5.4. Alternative Drohnenarten

Grundsätzlich können, über das von uns verwendete MAVLink Protokoll, alle Arten von Drohnen angesteuert werden. Wie in Abbildung 5.12 ersichtlich, bietet die Autopilot-Software 'ArduPilot' Varianten für Multicopter, Flugzeuge und Fahrzeuge an.

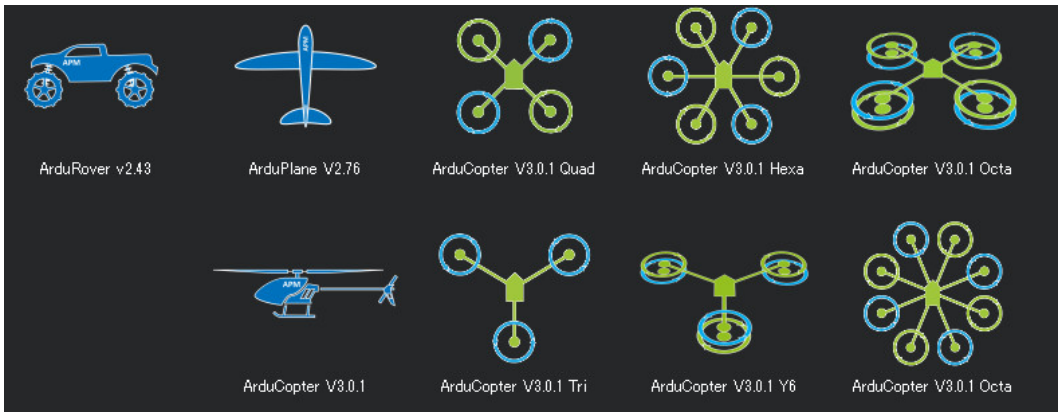


Abbildung 5.12.: Screenshot der verschiedenen Firmwarevarianten.

Auch andere Projekte könnten von unserem System profitieren. Uplift Aero beispielsweise, ist ein Startup, das Hilfsgüter mit Hilfe von Drohnen von der Türkei aus nach Syrien transportieren wollte. Sie haben für ihre Flugzeuge dieselben Flugcontroller wie wir verwendet, mussten lediglich eine andere Firmware aufspielen. Dies beweist wie flexibel und vielseitig die von uns eingesetzte Hardware verwendet werden kann.

6. Zusammenfassung und Ausblick

6.1. Impressionen

Um den Einstieg für die Zielgruppe zu erleichtern, wurde eine Webseite erstellt, die die wichtigsten Funktionen erklärt und eine Übersicht über die Funktionsweise der Plattform bietet.

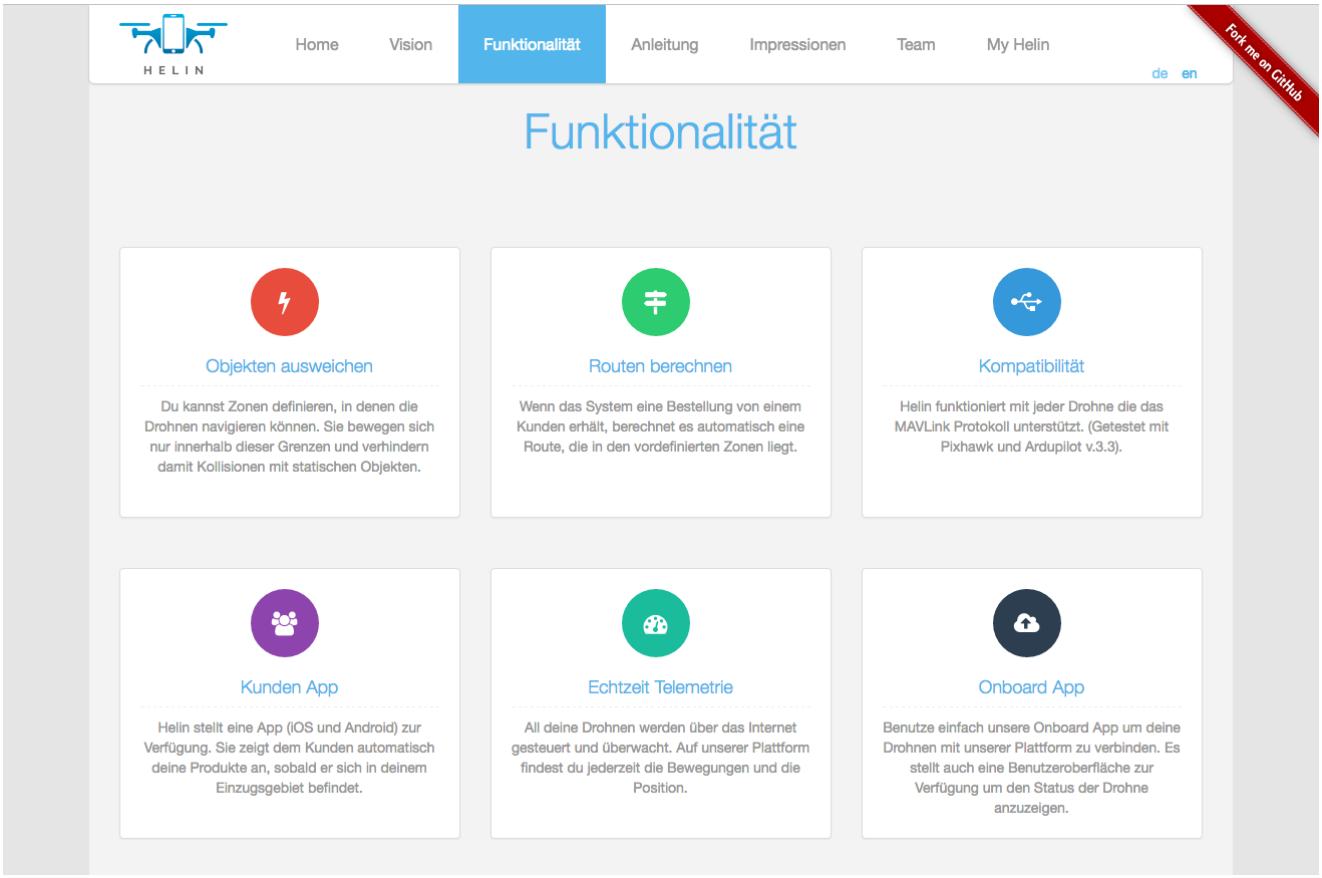


Abbildung 6.1.: Ausschnitt der Webseite https://my.helin.ch/

Ausserdem wurde die Plattform auf `https://my.helin.ch/` freigeschaltet und kann frei genutzt werden.

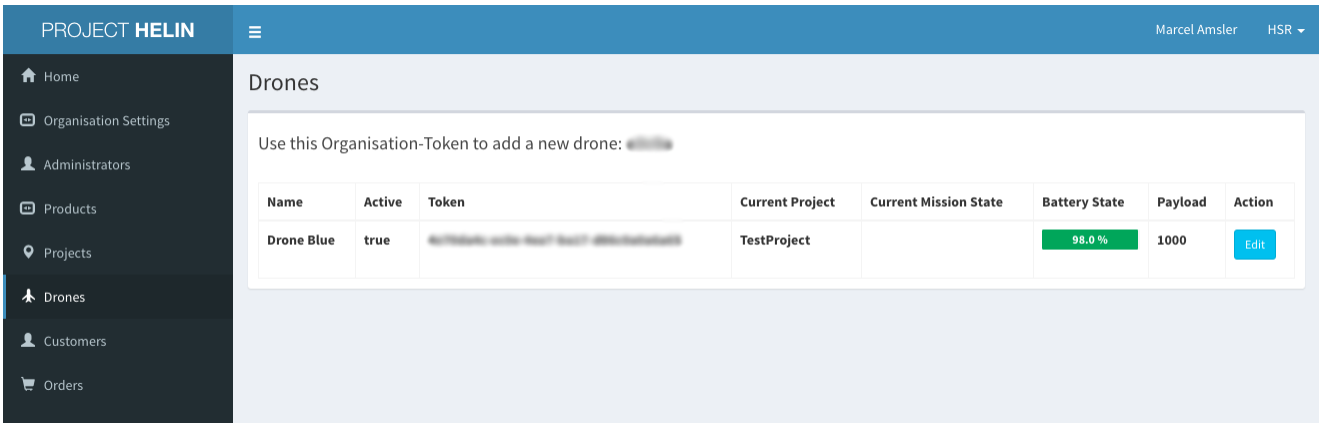


Abbildung 6.2.: Drohnenliste von MyHelin

Die Single-Page-Applications für das Zeichnen von Zonen, das Testen eines Zonensetups und die Überwachung der Lieferungen stehen dann sofort zur Verfügung.

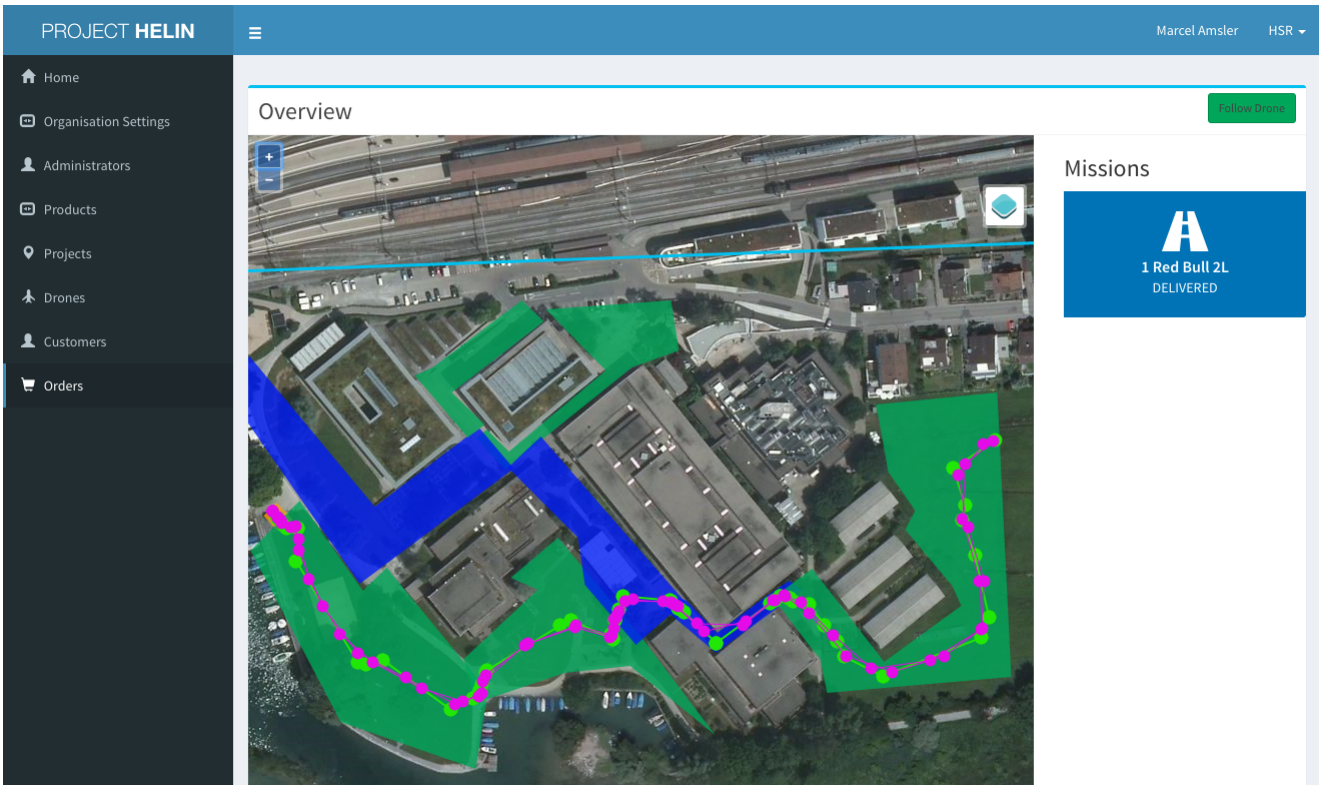
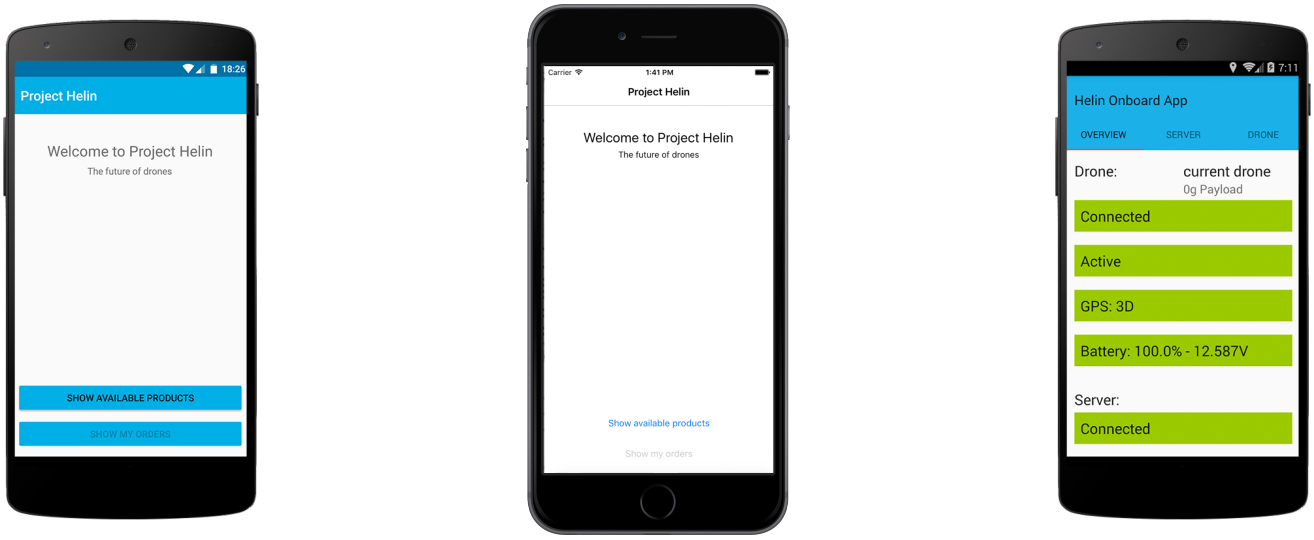


Abbildung 6.3.: Kartenansicht bei der Verfolgung einer Drohne während der Mission

Das Customer-App für Android (links), das Customer-App für iOS (mitte) und das Onboard-App (rechts) können verwendet werden.



Die Drohne zusammen mit dem Onboard-App konnte auf einen Stand gebracht werden, mit dem sie mehrere Lieferungen hintereinander ohne Schwierigkeiten ausführen kann.

Abbildung 6.4.: Diverse erfolgreiche Tests mit dem autonomen Abwurf der Ladung

Abbildung 6.5.: Liefertest mit einem Verbandskasten

Abbildung 6.6.: Liefertest nach dem Öffnen des Fallschirms

Abbildung 6.7.: Drohne mit Onboard-App in der finalen Ausbaustufe

Abbildung 6.8.: Zweite Drohne in der finalen Ausbaustufe

6.2. Zusammenfassung der Ergebnisse

Während dieser Arbeit hat das Team eine Plattform konzipiert und entwickelt, die es jedem Anbieter ermöglicht, einen autonomen Drohnen-Lieferservice für ein von ihm definiertes Gebiet aufzubauen. Die Plattform wurde auf einem Server als Software as a Service zur Verfügung gestellt. Ausserdem wurde der gesamte Code des Projekts auf GitHub veröffentlicht und steht unter einer MIT-Lizenz als Open Source Software zur Verfügung. Die Tests mit den zwei eigens aufgebauten Drohnen zeigen, dass das System als Ganzes funktioniert.

Folgende Ergebnisse sind besonders hervorzuheben:

- Webseite für Administratoren zur Verwaltung von Organisationen, Drohnen, Produkten und Bestellungen
- Cross-Plattform Bestell-App (iOS, Android) für Kunden
- Android App zur Steuerung der Drohne über das Internet
- Verfolgung der Drohnen-Telemetrie während, vor und nach der Lieferung einer Bestellung
- Automatisierte dynamische Routenberechnung über vordefinierte Flugzonen, abhängig von der Position des Kunden
- Automatisierte Verteilung von Lieferaufträgen an die Drohnen
- Getestete Vorlage für den Aufbau einer Lieferdrohne
- Vorgefertigte 3D-Modelle für den 3D-Druck der Smartphone-Halterung und der Abwurfvorrichtung

6.3. Ausblick

Die entwickelte Plattform zeigt erst einen Bruchteil der Möglichkeiten, die in Zukunft von autonomen Drohnen übernommen werden können. Beispielsweise können Videoaufnahmen, Infrastruktur-Überwachung oder Katastrophenhilfe als Angebote integriert werden.

6.3.1. Empfohlene Weiterentwicklungen

Während des Projekts sind weitere Ideen entstanden:

Plattform

- Services hinzufügen (Drone-Selfie, Follow-Me Video, geographische Vermessung)
- Aktion am Zielort wählbar oder vom Produkt abhängig machen
- Test, ob es auch mit Radio-Telemetry am Onboard-App funktioniert
- Flight-Controller ohne MAVLink auch unterstützen (z.B. DJI-Onboard-SDK [4])
- Globale Flugzonen die in allen Projekten sichtbar sind
- Kollisionsvermeidung von Drohnen auf denselben Routen
- Objekte mit Kollisionspotential (z.B. Gebäude, Bäume) automatisiert aus Flugzonen ausschliessen
- Zugang zum Broker besser absichern (Zertifikate)

Drohnen Hardware

- Testen in Kombination mit Obstacle Avoidance (z.B. Intel RealSense[12]) Hardware
- Tests mit anderen Arten von Drohnen (siehe Abschnitt 5.4)
- Smartphone ersetzen durch Embedded-System (siehe Alternativen im Abschnitt 3.8)
- Automatisches Beladungssystem für Drohnen
- Automatisches Batterieaustauschsystem für Drohnen

6.3.2. Bekannte Probleme

Alle bekannten kleineren Probleme und zusätzlichen Features, die vor einem produktiven Einsatz des Systems umgesetzt werden sollten, werden im Server-GitHub-Repository als Issues erfasst. Dies ermöglicht es auch anderen Entwicklern an dem Projekt weiterzuarbeiten und dessen Limitierungen zu kennen.

6.4. Schlussfolgerung

Während dieser Arbeit haben wir aus unserer Vision ein Produkt entwickelt und fertiggestellt. Damit konnten wir beweisen, dass man mit verfügbaren Technologien Liefersysteme mit autonomen Drohnen schon heute realisieren kann. Die entstandene Plattform kann nun zu Testzwecken von allen Interessierten genutzt und weiterentwickelt werden.

Die erarbeiteten funktionalen- und nicht-funktionalen Anforderungen konnten alle erfüllt und teilweise übertroffen werden, trotz der vielfältigen und teilweise interdisziplinären Aufgaben, die dieser Arbeit innewohnten.

Unserer Meinung nach ist unsere Plattform ein Beispiel für die Drohnen-Projekte der Zukunft. Vor allem in Bezug auf die Verwendung einer zentralen Instanz, zur Steuerung und Überwachung einer Drohnenflotte. Dadurch gewinnen viele Projekte, wie das Überwachen von Haien [19], die Lieferung von Defibrillatoren [2] oder das Finden von Überlebenden in einem Katastrophengebiet [17], deutlich an praktischer Relevanz und können flächendeckend eingesetzt werden. Wir hoffen, dass die entwickelte Plattform als Anstoss für die Stakeholder in dieser Branche dienen kann und sich die Technologien und Gesetzeslagen insoweit verbessern, dass Dienstleistungen von autonomen Drohnen bald für eine grössere Anzahl von Kunden zur Verfügung stehen werden.

Teil II.

Anhang

Aufgabenstellung Bachelorarbeit Abteilung I, FS 2016
Marcel Amsler, Kirusanth Poopalasingam, Martin Stypinski

Helin: Multicopter Software-Platform

1. Betreuer & Anwendungspartner

Betreuer dieser Arbeit ist
Prof. Dr. Markus Stolze mstolze@hsr.ch

Co-Referent für diese Arbeit ist
Beat Stettler

Externer Experte für diese Arbeit ist
Thomas Kälin

Anwendungspartner dieser Arbeit ist
Abteilung Informatik

2. Ausgangslage

Multicopter haben in den letzten Jahren grosse technologische Fortschritte erlebt. Sie sind leichter, einfacher zu bedienen, leistungsfähiger und preiswerter geworden, so dass sie auch für kleine Geschäfte und Privatpersonen erschwinglich geworden sind.
Ein möglicher Anwendungsbereich von Multicoptern ist die schnelle Auslieferung von Produkten. So ist es im Prinzip schon heute möglich mit einem Multicopter einen Defibrillator punktgenau zu einer bedürftigen Person an einem Open-Air-Konzert zu bringen. Allerdings erlaubt die aktuelle Gesetzgebung keine automatisch gesteuerten Flüge von Multicoptern. Zudem ist auch manuell gesteuerte Flüge „auf Sicht“ in der Nähe von grösseren Menschenansammlungen nicht erlaubt. Da die gesetzliche Lage aktuell im Fluss ist, kann es sein dass diese Limitationen schon bald nicht mehr gelten und es somit interessant wäre eine Software-Platform zu erstellen mit der automatisch gesteuerte Zustellung von Produkten mit Multicoptern abgewickelt werden können.

3. Ziele der Arbeit

In dieser Arbeit sollen Komponenten eine erweiterbare open-source Software-Plattform mit den folgenden vier Komponenten erstellt werden

(1) Ein Prototyp einer Android App mit der Produkt-Bestellungen beim System abgegeben werden können. Die Android App soll den Standort des Bestellers mittels GPS bestimmen und zusammen mit der Produktwahl an die zentrale Management Anwendung übertragen. Diese App soll bewusst einfach gehalten sein und soll Themen wie Bezahlung ausser Acht lassen.

(2) Eine prototypische Konfigurationsanwendung mit der sich Multicopter, Multicopter-Startplätze, Flugkorridore und Produktlisten erstellen und verwalten lassen.

(3) Eine gut getestete, wartbare und optimierte Management-Anwendung welche Aufträge der Bestell-App entgegennimmt, automatisch einen freien Multicopter für die Durchführung der Zulieferung bestimmt, den Auftrag an die Multicopter-App übermittelt und die Durchführung des Auftrags überwacht. Dabei soll auch das Abbrechen eines Auftrags mitten im Flug möglich sein.

(4) Eine gut getestete und wartbare Multicopter App die für ein Android-Phone welches auf dem Multicopter montiert ist optimiert ist. Die Android App ist über das Handy-Netz in steter Kommunikation mit der Management Anwendung. Die App meldet die Position des Multicopters in regelmässigen Abständen und ist in der Lage im Flugbefehle der Management-Anwendung an den Steuerungs-Controller auf dem Multicopter über eine für Multicopter standardisierte Schnittstelle zu übertragen.

Die Aufgabe umfasst neben der Erstellung der oben genannten Software-Komponenten auch die Zusammenstellung der Bestellliste für den Aufbau eines Demonstrators mit zwei flugfähigen Drohnen, sowie die Demonstration eines realistischen Bestell- und Auslieferungsszenarios. Hierzu sind zum Beispiel Lösungsvorschläge für die für Besteller gefahrlose Ablieferung zu erarbeiten. Neben der Software und der Multicopter-Hardware ist ein wichtiges Produkt dieser Arbeit eine Video-Dokumentation die zeigt inwieweit sich mit den erstellten Software und Hardwarekompoenten ein realistischen Bestell- und Auslieferungsszenarios realisieren liess.

4. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen. Die Dokumentation ist vollständig auf CD/DVD in einem Exemplar abzugeben (Exemplar für das Sekretariat Informatik), sowie ein Download-Link für Prof. Stolze und weitere Exemplare nach Absprache mit dem Co-Referenten (B. Stettler) und dem Experten (T. Kälin).

Zudem ist eine kurze Projektresultatdokumentation im öffentlichen Wiki von Prof. M. Stolze zu erstellen.

5. Weitere Regeln und Termine

Im Weiteren gelten die allgemeinen Regeln zu Bachelor und Studienarbeiten
„Abläufe und Regelungen Studien- und Bachelorarbeiten im Studiengang Informatik“ (HSR Intranet)
https://www.hsr.ch/Ablaeufe-und-Regelungen-Studie.7479.0.html)

Der Terminplan ist hier ersichtlich (HSR Intranet)
https://www.hsr.ch/Termine-Bachelor-und-Studiena.5142.0.html

6. Rechte

Die resultierende Software und Dokumentation soll als open-source Software (MIT Lizenz) publiziert werden. Die Multicopter Hardware wird von der HSR beschafft und bleibt Eigentum der HSR. Es wird durch alle Parteien sicher gestellt, im Source-Code und im Impressum der Anwendung die originale Urheberschaft durch die HSR Studenten weiterhin sichtbar bleibt.
Der Bericht der Bachelorarbeit (ohne geheime Anhänge) wird von der HSR im E-Prints Respository der HSR (eprints.hsr.ch) elektronisch veröffentlich. Titel und Abstract der Arbeit dürfen von der HSR und den Studierenden schon während der Arbeit kommuniziert werden.

7. Beurteilung

Eine erfolgreiche Bachelorarbeit zählt 12 ECTS-Punkte pro Studierenden. Für 1 ECTS Punkt ist eine Arbeitsleistung von ca. 25 bis 30 Stunden budgetiert. Entsprechend sollten ca. 350h Arbeit für die Bachelorarbeit aufgewendet werden. Dies entspricht ungefähr 25h pro Woche (auf 14 Wochen) und damit ca. 3 Tage Arbeit pro Woche pro Student.

Für die Beurteilung ist der HSR-Betreuer verantwortlich. Die Bewertung der Arbeit erfolgt entsprechend der verteilten Kriterienliste.

Die definitive Aufgabenstellung wurde am 8.4.2016 beschlossen.

Rapperswil, 8.4.2016

Prof. Dr. Markus Stolze, Institut für Software, Hochschule für Technik Rapperswil

B. Projektplan

B.1. Änderungsgeschichte

Datum	Version	Änderung	Autor
26.02.2016	1.0	Erstellen des Projektplans	Martin Stypinski
11.03.2016	1.1	Risiken nach Sprint 1	Marcel Amsler
26.03.2016	1.2	Risiken nach Sprint 2	Kirusanth Poopalasingam
12.04.2016	1.3	Risiken nach Sprint 3	Marcel Amsler
28.04.2016	1.4	Risiken nach Sprint 4	Kirusanth Poopalasingam
04.06.2016	1.5	Risiken nach Sprint 5	Martin Stypinski
11.06.2016	1.6	Verantwortlichkeiten dokumentiert	Marcel Amsler
11.06.2016	2.0	Finalisieren des Projektplans	Martin Stypinski
13.06.2016	2.0.1	Qualitätskontrolle und kleinere Korrekturen	Marcel Amsler

B.2. Einleitung

B.2.1. Ziel und Zweck

Ziel dieser Arbeit ist es, eine Web-Applikation zu entwickeln, die es ermöglicht Drohnenflotten zu verwalten und damit vollautomatisierte Lieferungen auszuführen. Nutzer dieser Applikation können dazu Flugzonen definieren, in denen ein sicherer Flug von Drohnen möglich ist. Kunden hingegen sollen durch eine App die Möglichkeit erhalten, Güter zu bestellen, welche an ihre Position ausgeliefert werden.

B.2.2. Lieferumfang

Der Lieferumfang dieser Arbeit entspricht den Vorgaben der HSR:

- Zu Handen des Betreuers:
  - Dokumentation, sämtliche Dokumente und Sourcen per E-Mail
- Poster - Enthält Zusammenfassung der Arbeit
- Abstract für die Bachelorarbeitsbroschüre

Zusätzlich ist es uns ein Anliegen, dass der Sourcecode und die Erfahrungen über den Zeitraum der Bachelorarbeit hinaus bestehen bleiben. Daher wurde ein GitHub-Repository eingerichtet, dass nach Beendigung der Arbeit sämtlichen Code enthält: https://github.com/Project-Helin

B.2.3. Annahmen und Einschränkungen

Es kann angenommen werden, dass der Zeitplan im Rahmen der regulären Bachelorarbeit Zeit gültig ist. Es wird dabei berücksichtigt, dass ein Zeithorizont von 17 Wochen zur Verfügung steht und die maximale Arbeitszeit von 360 Stunden pro Person nicht überschritten werden soll.

B.3. Projektorganisation

B.3.1. Organisationsstruktur

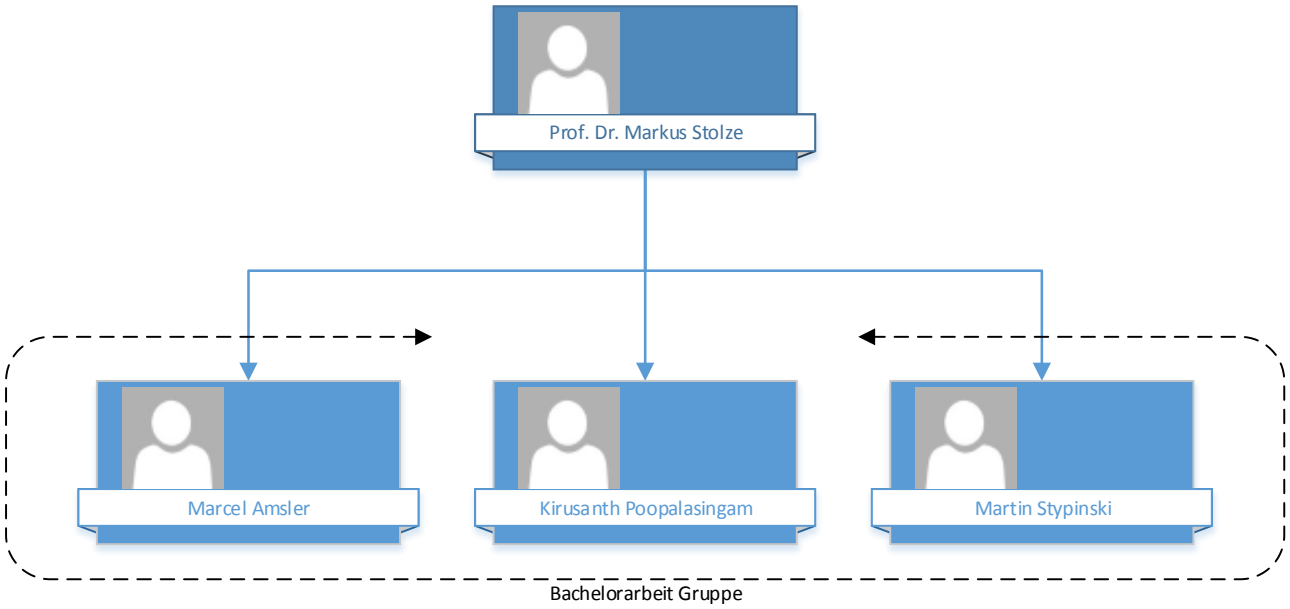


Abbildung B.1.: Grobübersicht über den Projektplan

Name		E-Mail	Verantwortung
Prof. Dr. Markus Stolze	MST	-	Betreuer der Arbeit
Marcel Amsler	ma	-	
Kirusanth Poopalasingam	kp	-	
Martin Stypinski	ms	-	

B.3.2. Verantwortlichkeiten

Die Verantwortung und eine Unterstützungsfunktion für alle Hauptkomponenten haben wird zu Beginn der Arbeit folgendermassen festgelegt.

	Server CRUD	**Server JavaScript**	**Onboard-App**	**Customer App**
Hauptverantwortung	kp	ms	ma	ma
Unterstützung	ms	ma	ms	kp

B.3.3. Meetings

In der Regel wird ein Meeting jeweils am Donnerstag um 10.30 abgehalten. In Ausnahmefällen können die Termine von der Planung abweichen.

SW	Datum	Zeit	Art der Sitzung
1	25.02.2016	10:00 - 11:00	Kick-off Meeting
2	03.03.2016	15:00 - 16:00	Meeting mit Betreuer
3	10.03.2016	10:30 - 11:30	Meeting mit Betreuer
4	17.03.2016	11:00 - 11:45	Meeting mit Betreuer
5	24.03.2016	10:30 - 11:30	Meeting mit Betreuer
6	08.04.2016	13:30 - 14:30	Meeting mit Betreuer
7	14.04.2016	10:30 - 11:30	Meeting mit Betreuer
8	21.04.2016	10:30 - 11:30	Meeting mit Betreuer
9	28.04.2016	11:00 - 11:45	Meeting mit Betreuer
9	29.04.2016	15:00 - 16:00	Zwischenpräsentation bei Prof. Beat Stettler
11	11.05.2016	18:15 - 19:00	Zwischenpräsentation bei Thomas Kälin und Prof. Dr. M. Stolze
11	12.05.2016	10:30 - 11:30	Meeting mit Betreuer
12	19.05.2016	10:00 - 11:00	Meeting mit Betreuer
13	26.05.2016	10:30 - 11:30	Meeting mit Betreuer
14	02.06.2016	10:30 - 11:30	Meeting mit Betreuer
15	09.06.2016	10:30 - 11:30	Meeting mit Betreuer
16	13.06.2016	10:00 - 11:00	Meeting mit Betreuer

B.4. Meilensteinplanung

Grundsätzlich wird während der gesamten Arbeit Agiles-Projektmanagement mit Scrum als Methode eingesetzt. Ein Scrum-Sprint dauert in der Regel 2 Wochen.

ID	Milestone	Anfang	Abschluss	Dauer
1	Einarbeitung	22/02/16	11/03/16	3w
2	Proof of Concept	14/03/16	25/03/16	2w
3	Implementierung	28/03/16	03/06/16	10w
4	Finalisierung	06/06/16	17/06/16	2w

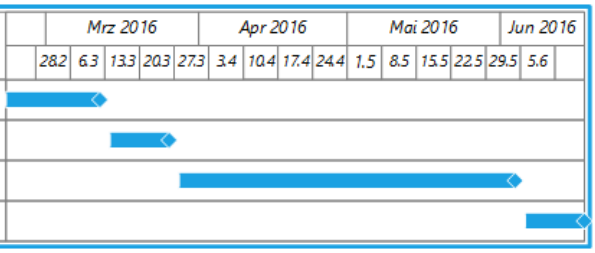


Abbildung B.2.: Grobübersicht über den Projektplan

- **Ende Einarbeitung: (MS-1)**
  - **Datum:** 11.03.2016
  - **Ziel:** Entwicklungsumgebung und Prozesse stehen
  - **Erfüllungskriterium:** Sämtliche prozessbegleitenden Massnahmen sind lauffähig. Projektmanagement-Tool, IDEs, CI
- **Ende Proof of Concept: (MS-2)**
  - **Datum:** 25.03.2016
  - **Ziel:** Die Machbarkeit ist bewiesen
  - **Erfüllungskriterium:** Prototyp ist lauffähig, zeigt Machbarkeit und allfällige Einschränkungen.
- **Ende Implementierung: (MS-3)**
  - **Datum:** 03.06.2016
  - **Ziel:** Code Freeze
  - **Erfüllungskriterium:** Alle zwingenden funktionalen und nicht-funktionalen Anforderungen sind erfüllt.
- **Abgabe: (MS-4)**
  - **Datum:** 17.06.2016
  - **Ziel:** Abgabe der Arbeit
  - **Erfüllungskriterium:** Alle im Lieferumfang geforderten Dokumente sind fertiggestellt

B.5. Risikomanagement

B.5.1. Risiken

	Name	Wahrscheinlichkeit	Kosten	Risiko	Beschreibung	Stand nach Sprint				
	NFR					1	2	3	4	5
R01	Messaging auf Mobilgerät	3 [1]	4 [2]		Die Anbindung zum Message-Broker auf dem Mobilgerät könnte Komplikationen mit sich bringen.					
R02	Play Framework	2	1		Das Play Framework könnte eine zeitaufwendigere Handhabung mit sich bringen.					
R03	Internet auf Mobilegerät	5	4		Die Internetverbindung auf dem Mobilgerät könnte instabil sein.					
R04	Infrastruktur Probleme	5	1		Für das Backend wird ein Server mit JMS Broker benötigt, dies könnte auf Grund von geschlossenen Ports zu Problemen führen.					
	Hardware					1	2	3	4	5
R05	Kapazität der Drohne	4	2		Die Drohne kann das Gewicht des zu transportierenden Gutes nicht heben.					
R06	Absturz und Schäden	5	5		Die Drohne kann Abstürzen und Reparaturarbeiten könnten lange dauern, wenn Ersatzteile nicht verfügbar sind.					

[1] Skala von 1-5, 1 sehr unwahrscheinlich bis 5 sicher
[2] Skala von 1-5, 1 nahezu konsequenzfrei bis 5 katastrophale Folgen

R07	Postitionsungenauigkeiten	5	2	☐	GPS könnte auf der Drohne ungenau sein.	☐	☐	☐	☐	☐
R08	Ardupilot Handhabung	3	4	☐	Ardupilot könnte Updates während dem Betrieb verweigern. Konkret heisst das, dass während des Fluges keine neuen Positionsdaten an den Autopiloten übergeben werden können.	☐	☐	☐	☐	☐
R09	Ardupilot API	4	2	☐	Die vorhandene Dokumentation und das API der Ardupilot Komponente könnte unausreichend sein.	☐	☐	☐	☐	☐
	Workflow					1	2	3	4	5
R10	Entwicklungsprozesse	5	2	☐	Der Entwicklungsprozess kann sehr umständlich sein. Die Drohne muss im Gebäude programmiert werden, der Testflug aber auf offenem Gelände. Dies könnte viel Zeit in Anspruch nehmen.	☐	☐	☐	☐	☐
R11	Kommunikation im Team	1	1	☐	Die Teamgrösse von 3 Personen könnte zu Komplikationen in der Kommunikation führen.	☐	☐	☐	☐	☐
	GUI					1	2	3	4	5
R12	Ablademanagement	3	3	☐	Das Ablademanagement könnte sich schwieriger gestalten als angenommen, es müssen viele Sicherheitsfunktionen für den Verbraucher gewährleistet werden, die schwierig zu evaluieren sind.	☐	☐	☐	☐	☐
R13	Zonedefinition	2	2	☐	Die Gestaltung von Zonen und deren Handhabung könnte einige Fragen aufwerfen.	☐	☐	☐	☐	☐
R14	Höhenproblematik	2	2	☐	Mehrere Drohnen in derselben Flugzone, könnten zu Kollisionen führen, sofern diese auf derselben Höhe fliegen.	☐	☐	☐	☐	☐

Tabelle B.1.: Risiken

B.5.2. Umgang mit Risiken

Die Risiken wurden in einer Risiko Matrix aufgegliedert um besser zu verstehen, welche Risiken eine grosse Bedrohung darstellen.

Eintrittswahrscheinlichkeit	Schadenausmass 1	2	3	4	5
5	R04	R07 R10		R03	R06
4		R05 R09			
3			R12	R01 R08	
2	R02	R13 R14			
1	R11				

Abbildung B.3.: Risiko Matrix

Als Konsequenz der Matrix kann folgende Aufteilung getroffen werden:

- **Risiko hoch:** R03, R06
- **Risiko mittel:** R01, R04, R05, R07, R08, R09, R10, R12
- **Risiko klein:** R02, R11, R13, R14

B.5.3. Massnahmen

Für die Risiken der Kategorie mittel und hoch wurden folgende Massnahmen festgelegt:

- **R03 - Internet auf Mobilgerät:**
  Massnahmen: Es muss sichergestellt werden, dass die Drohne nicht auf eine permanente Serververbindung angewiesen ist. Monitor und Datenlogging dürfen Unterbrechungen aufweisen. Die Mission darf aber zu keinem Zeitpunkt gefährdet werden, die Drohne muss selbstständig ihre Aufgabe erfüllen können und danach wieder zurückkehren.
- **R06 - Absturz und Schäden:**
  Massnahmen: Bei der Wahl der Drohne wurde auf die Verfügbarkeit der Ersatzteile geachtet, sofern dies möglich war.

- **R01 - Messaging auf Mobilgerät:**
  Massnahmen: Bei der Evaluierung der Komponenten wird ein Message-Broker gewählt, der auf einem Android Betriebssystem lauffähig ist. Dies wird in einem Proof of Concept überprüft.

- **R04 - Infrastruktur Probleme:**
  Massnahmen: Aufgrund von Erfahrungen aus früheren Projekten, wird auf die Serverinfrastruktur der HSR verzichtet, dies garantiert eine volle Kontrolle über den Server. Es muss jedoch berücksichtig werden, dass der administrative Aufwand höher ist und somit mehr Zeit in Anspruch nehmen wird.

- **R05 - Kapazität der Drohne:**
  Massnahmen: Es werden Güter verwendet, deren Gewicht von der Drohne transportiert werden kann.

- **R07 - Positionsungenauigkeit:**
  Massnahmen: Bei Flugkorridoren und Landepunkten wird genügend Sicherheitsmarge eingerechnet um die Ungenauigkeit zu relativieren.

- **R08 - Ardupilot Handhabung:**
  Massnahmen: Im Proof of Concept wird überprüft, wann und wie Updates gemacht werden können. Falls Updates während des Flugs nicht möglich sind, wird von Anfang an die gesamte Route an Ardupilot übertragen.

- **R09 - Ardupilot API:**
  Massnahmen: Früh in einem Proof of Concept die Möglichkeiten und Grenzen des APIs nachvollziehen.

- **R10 - Entwicklungsprozesse:**
  Massnahmen: Es wird während des Proof of Concepts versucht einen Simulator zu verwenden, um Zeit zu sparen. Gegebenenfalls Arbeitsplatz im Freien, um Zeit während des Deployments auf die Drohne zu sparen.

- **R12 - Abblademanagement:**
  Massnahmen: Prüfung einer Abwurfmöglichkeit. Gegebenenfalls Benutzer mit GUI auf Mobile begleiten um einen sicheren und unfallfreien Abladevorgang zu garantieren.

B.6. Qualitätsmassnahmen

B.6.1. Dokumentation

Die Dokumentation wird vollständig in Latex geschrieben und befindet sich zu jedem Zeitpunkt auf dem Project-Helin GitHub-Repository: https://github.com/Project-Helin. Alle grösseren Änderungen werden immer von einem anderen Teammitglied gelesen und überprüft.

B.6.2. Projektmanagement

Für das Projektmanagement wird Jira von Atlassian verwendet. Als Projektmanagement Methode wird Scrum verwendet.

B.6.3. Entwicklung

Die Qualität der Entwicklung wird durch folgende Massnahmen sichergestellt:

- **Code Review:** Bei kritischen Komponenten werden Code Reviews durchgeführt.
- **Feature Review:** Bei allen Features bzw. umgesetzten User-Stories führt ein anderes Teammitglied eine Qualitätskontrolle durch. Diese kontrolliert hauptsächlich die Erfüllung der Acceptance-Criterias.
- **Testing:** Das gesamte Projekt wird in Java entwickelt. Als Unit-Test Framework wird JUnit 4 verwendet.
- **Versionierung:** Der gesamte Quellcode wird mit Hilfe von GitHub versioniert.
- **Deployment:** Die Serverkomponenten werden mithilfe eines Build Systems deployt. Die Komponenten auf den Mobiltelefonen werden manuell deployt. Jedoch wird auf dem Build-System für alle Komponenten die Ausführung von Unit-Tests garantiert.

C. Infrastruktur

C.1. Server

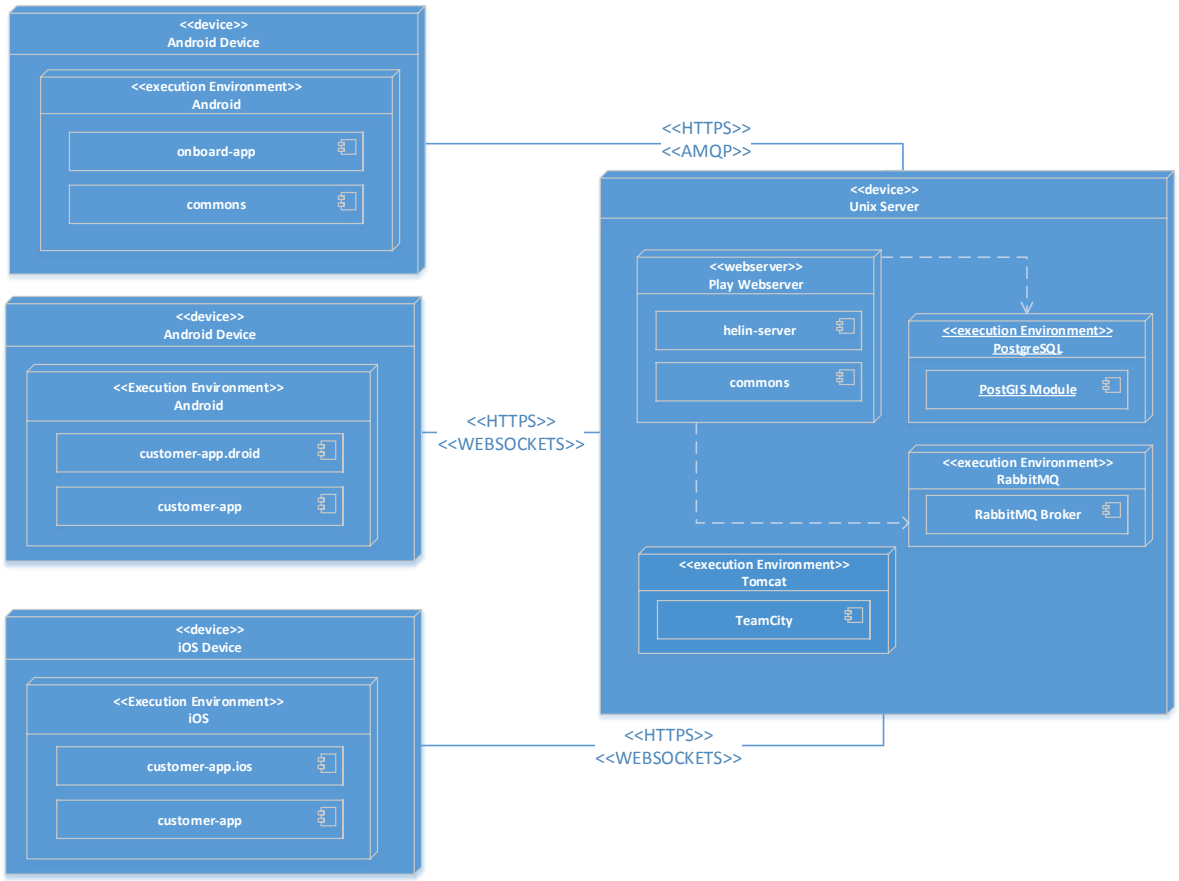


Abbildung C.1.: Deployment Diagram

Für die Helin Applikation wird ein physischer Server verwendet. Wie in der Abbildung 3.3 dargestellt, wird dieser als Build-, Test- und Applikations-Server verwendet. Auf dem Server läuft eine PostgreSQL

Datenbank mit einer PostGIS Extension, welche für die geografischen Daten benötigt wird. Gleichzeitig sind ein RabbitMQ-Message-Broker und eine Instanz der Helin Server Applikation installiert. Aus Ressourcengründen wird alles auf einem Server ausgeführt, könnte aber bei Leistungsproblemen ohne weiteres auf mehrere Server verteilt werden. Als Build- und Test-Server für alle Applikationen wird TeamCity verwendet.
Neben dem Server werden auch Apps für Android und iOS Geräte veröffentlicht. Die Onboard-App kommuniziert primär über AMQP mit dem Server, während die Customer-App HTTPS und WebSockets nutzen.

D. Abbildungsverzeichnis

1.1. System Kontext Diagramm mit externen Schnittstellen 4

2.1. Diego[1] 5
2.2. Stefanie [1] 6
2.3. Ricardo [1] 6
2.4. Domain-Model 13

3.1. Übersicht der Project Helin Architektur 15
3.2. Vereinfachte Übersicht der logischen Architektur und deren Komponenten 16
3.3. Übersicht der Kommunikations-Architektur mit den jeweiligen Protokollen. 21
3.4. Ablauf der Bestellung eines Produkts 24
3.5. Zustände der Mission 25

4.1. Übersicht der wichtigsten Schnittstellen und deren Verbindungen 26
4.2. Objekt-Diagramm der Kommunikationsstruktur 28
4.3. APM Planner 2.0 als Bodenstation und Android Emulator mit Onboard App 30
4.4. Übersicht der Customer-App mit allen Verknüpfungen zwischen den Screens 33
4.5. Topologische Ansicht des Campus HSR 39
4.6. Verschiedene Flugzonen am Beispiel der HSR 40
4.7. Erstes Konzept 41
4.8. Visibility Graph in einer Flugzone 42
4.9. Skeleton in Polygon 43
4.10. Beispiel der Höhe am Übergang zweier Polygone 45

5.1. Nutzwertanalyse der verschiedenen Liefervarianten 48
5.2. Skizze des physikalischen Modells eines Fallschirms 49
5.3. DJI F450 Flamewheel Kit 52
5.4. Pixhawk Flight-Controller 53
5.5. GPS-Modul für Pixhawk 53
5.6. Erster Prototyp ohne Landegestell und ohne Smartphone 54
5.7. Drohnen Aufbau mit Smartphone 55
5.8. Drohne mit Handy Halterung 56
5.9. Halterung Modell 56
5.10. Halterung 56
5.11. Halterung montiert 56
5.12. Screenshot der verschiedenen Firmwarevarianten. 58

6.1. Ausschnitt der Webseite `https://my.helin.ch/` 59

6.2. Drohnenliste von MyHelin 60
6.3. Kartenansicht bei der Verfolgung einer Drohne während der Mission 60
6.4. Diverse erfolgreiche Tests mit dem autonomen Abwurf der Ladung 61
6.5. Liefertest mit einem Verbandskasten 62
6.6. Liefertest nach dem Öffnen des Fallschirms 62
6.7. Drohne mit Onboard-App in der finalen Ausbaustufe 63
6.8. Zweite Drohne in der finalen Ausbaustufe 63

B.1. Grobübersicht über den Projektplan 72
B.2. Grobübersicht über den Projektplan 75
B.3. Risiko Matrix 78

C.1. Deployment Diagram 81

E. Literaturverzeichnis

[1] Fernando Cacciola. A CGAL implementation of the straight skeleton of a simple 2d polygon with holes. fcacciola.50webs.com/CGAL_straight_skeleton.pdf, 2004. [Online; 07.06.2016].

[2] Webredactie Communication. Tu delfts ambulance drone drastically increases chances of survival of cardiac arrest patients. http://www.tudelft.nl/en/current/latest-news/article/detail/ambulance-drone-tu-delft-vergroot-overlevingskans-bij-hartstilstand-drastisch/, 2014. [Online; 13.06.2016].

[3] Alan Mackworth David Poole. Artificial Inteligence - Foundation of computational agents. http://artint.info/html/ArtInt_52.html, 2010. [Online; 2016-06-08].

[4] DJI. Turn your Aerial Platform into an Autonomous Flying Robot. https://developer.dji.com/onboard-sdk/. [Online; 13.06.2016].

[5] Mathias Fleck. Usability of Lightweight Defibrillators for UAV Delivery. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, pages 3056–3061, New York, NY, USA, 2016. ACM.

[6] Martin Fowler. Inversion of control containers and the dependency injection pattern. http://martinfowler.com/articles/injection.html, 2004. [Online; 12.6.2016].

[7] Martin Fowler. GUI Architectures - Model View Controller. http://martinfowler.com/eaaDev/uiArchs.html#ModelViewController, 2006. [Online; 16.5.2016].

[8] Bundesamt für Zivilluftfahrt (BAZL). Drohnen und Flugmodelle. https://www.bazl.admin.ch/bazl/de/home/gutzuwissen/drohnen-und-flugmodelle.html. [Online; 10.06.2016].

[9] Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns*. Addison-Wesley, Boston, 2004.

[10] H. Kaluđer, M. Brezak, and I. Petrović. A visibility graph based method for path planning in dynamic environments. In *MIPRO, 2011 Proceedings of the 34th International Convention*, pages 717–721, May 2011.

[11] Xin Liu, Chengping Zhou, Mingyue Ding, and Chao Cai. Skeleton-based fast path planning for UAV, 2009.

[12] Mariella Moon. Yuneec Typhoon H drone uses Intel tech to avoid collisions. https://www.engadget.com/2016/01/05/yuneec-typhoon-h-drone-intel-realsense/, 2016. [Online; 13.06.2016].

[13] Jürg Müller. Apple bleibt Platzhirsch. http://www.nzz.ch/wirtschaft/unternehmen/apple-bleibt-platzhirsch-1.18563593, 16.6.2015. [Online; 07.06.2016].

[14] Narobo. DroneCell V3. http://dronecell.narobo.com/individuals/individuals.html, 2011. [Online; 07.06.2016].

[15] Inc Pivotal Software. Clients & Developer Tools. https://www.rabbitmq.com/devtools.html. [Online; 13.06.2016].

[16] PricewaterhouseCoopers. Clarity from above - pwc global report on the commercial applications of drone technology, 2016.

[17] UAE Prime Ministers Office. Drones for good award - Project Kwago. http://www.dronesforgood.ae/?q=finals/project-kwago. [Online; 13.06.2016].

[18] Markus Schäfer. Wegsuchealgorithmen in Routenplanungssystemen. http://www.iai.uni-bonn.de/III/lehre/AG/IntelligenteDatenbanken/Projektgruppe/SS03/Seminar/Schaefer-Wegsuchealg_in_Routenplanern.pdf, 2003. [Online; 13.06.2016].

[19] The Verge. Australia uses drones to spot sharks and rescue their victims. http://www.theverge.com/2016/2/29/11132098/australia-drone-shark-attack-little-ripper, 2016. [Online; 13.06.2016].

[20] Arthur H. Watson and Thomas J. McCabe. *Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric*. Computer Systems Laboratory National Institute of Standards and Technolog, Gaithersburg.

F. Glossar

AMQP

Advanced Message Queuing Protocol, wird verwendet für die Übertragung der Messages an den Message Broker. 26

APK

Android Package, ist das Datenformat indem Android Apps gespeichert werden. 9

BEC

Battery Eliminator Circuit ist eine im RC-Modellbau verwendete Terminologie für eine Spannungsstabilisierungsschaltung, um konstante Spannung zu gewährleisten. 46

CGAL

The Computational Geometry Algorithms Library ist eine C++ Bibliothek die einfachen und effizienten Zugang zu Algorithmen auf geometrischen Objekten ermöglicht. Sie bietet unter anderem folgende Algorithmen: ConvexHull, AlphaShape, MeshGeneration, ShapeAnalysis, uvm. 34, 44

CRUD

Abkürzung für C = Create, R = Read, U = Update, D = Delete. Es wird zur Bezeichnung von Operationen auf einem Model/Resource verwendet. Beispielsweise sollen Benutzer erstellt (Create), angesehen (Read), geändert (Update) und gelöscht (Delete) werden können. 8, 10, 17, 27

Data Transfer Objects (DTOs)

Objekte die übertragen werden können und keine Abhängigkeiten zum System haben. 36

DBMS

Ein Database Management System dient zur Verwaltung und zur Abfrage einer Datenbank. Bekannte Systeme sind Oracle, MSSQL und PostgreSQL. 34

E2E-Test

Diese Art von automatisierten Test simuliert einen Benutzer. Typischerweise wird die zu testende Applikation in einem Browser geöffnet und automatisiert die gewünschten Aktionen mit Klicks ausgeführt. 37

Flight-Controller

Der Flight-Controller ist der Kern der Drohne, übernimmt die Stabilisierung und Steuerung der Rotoren. Wird über das MAVLink Protokoll mit Befehlen gesteuert. ii, 2, 4, 9, 14, 19, 29, 31, 46, 51, 52, 65

Integration-Test

Integration-Tests testen normalerweise ein ganzes System ohne Benutzeroberfläche aber mit Datenbank und anderen Abhängigkeiten. 37

JMS

Java Messaging Service. API um aus Java einen Message Broker anzusprechen. 26

JTS

Java Topology Suite ist ein API für 2D Objekte und deren Operationen. Sie erfüllt den Open Geospatial Consortium Standard und implementiert somit die gleichen Objekte wie PostGIS. 29

MAVLink

MAVLink ist ein reines Header-Protokoll, dass es ermöglicht unbemannte Fahr- und Flugzeuge zu kontrollieren. Es kann über verschiedene Protokolle laufen (USB, UDP, TCP). ii, 9, 29–31, 51, 52, 58, 65

Message Producer

Bei einem Messaging-System oder einer MOM (Message Oriented Middleware) gibt es immer Message Producer und Message Consumer. Ein Message Producer erstellt Nachrichten, ein Message Consumer hingegen empfängt und verarbeitet diese. 12

MIT-Lizenz

Eine Software-Lizenz, die es jedem erlaubt den Code in jeder Art weiterzuverwenden. 64

MOM

Message Oriented Middleware, ist Abstraktionsschicht zur Kommunikation zwischen Systemen. ii, 31

OTG

USB On-the-go, ist der Standard um eine direkte Kommunikation zwischen USB-Geräten ohne Host-Controller zu ermöglichen. In unserem Fall um ein USB Gerät an ein Smartphone anzuschliessen. 9

SFCGAL

Simple Feature CGAL ist eine Bibliothek die mit Geometrie Typen aus dem OGC Standard arbeitet. Es kann somit mit den bekannten Datentypen die im Open Source GIS Umfeld weit verbreitet sind, gearbeitet werden. 34

Single-Page-Applications

Eine Webseite, die dynamisch verändert wird, ohne dass der Server ihr neuen HTML-Code schicken muss. 60

UAV

Unpiloted Aerial Vehicle sind Flugzeuge welche Autonom fliegen können, beispielsweise eine Drohne. 43

Unit-Tests

Automatisierte Tests, die nur eine Methode testen, ohne irgendwelche Abhängigkeiten. 37