

UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Development of an unmanned aircraft traffic management system

Author:
Anton BORKIVSKYI

Supervisor:
Yurii SHOLOMII

*A thesis submitted in fulfillment of the requirements
for the degree of Bachelor of Science*

in the

Department of Computer Sciences
Faculty of Applied Sciences



APPLIED
SCIENCES
FACULTY ●

Lviv 2020

Declaration of Authorship

I, Anton BORKIVSKYI, declare that this thesis titled, “Development of an unmanned aircraft traffic management system” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

“Do not fly too close to the sun”

Daedalus

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Development of an unmanned aircraft traffic management system

by Anton BORKIVSKYI

Abstract

As drones become more and more popular special rules for their regulation should be created, and special systems for their management should be developed. For that purpose, SESAR has developed a list of services and strategy of their implementation to ensure safe integration of unmanned aviation into airspace. The object of our work is to develop a decomposed system that would match the SESAR list while being flexible and easy to extend. To design the architecture, we were following the IDesign framework. As a result, we provide a system with API documentation to communicate with it.

Acknowledgements

First of all, I would like to thank my supervisor Yurii Sholomii for help during work on this project and for all knowledge-sharing sessions we have had. Also, I am grateful to the whole team of Applied Science Faculty at Ukrainian Catholic University for an excellent academic program and Sigma Software for granting me the scholarship for my study. Finally, I am thankful to my family and friends who were supporting me during all years of my study, especially to Bohdan Borkivskyi for an extraordinary contribution to my growth as a developer and personality.

Contents

Declaration of Authorship	ii
Abstract	iv
Acknowledgements	v
1 Introduction	1
2 Related works	2
2.1 Air traffic management	2
2.2 Unmanned aircraft traffic management	4
2.2.1 U-space	4
2.2.2 Unifly	4
2.2.3 Matternet	5
3 Background information	6
3.1 Interaction of the system with the outside world	6
3.2 Clients	6
3.3 Resources	6
3.4 System	7
3.4.1 Entrypoint	7
3.4.2 Managers	8
3.4.3 Engines	8
3.4.4 Resource accesses	8
3.4.5 Utilities	8
4 Approach	9
4.1 Scope	9
4.2 Resources	10
4.2.1 Database	10
4.2.2 Third-party API	10
4.3 Managers	11
4.3.1 Membership manager	11
4.3.2 Schedule manager	11
4.3.3 Airspace manager	11
4.4 Engines	11
4.4.1 Path Engine	11
4.4.2 Validation Engine	12
4.5 Entrypoint	12
5 Results	26
6 Future works	27

Bibliography

List of Figures

2.1	Modern Air Traffic Management [23]	2
2.2	Example of radar screen [11]	3
2.3	Aircraft tracking [3]	3
2.4	U-Space services [30]	4
2.5	Unifly web application [31]	5
3.1	Interaction of the system	6
3.2	System layers	7
4.1	Architecture of the system with involved agents	9
4.2	Documents example	10
4.3	Making hull to avoid NFZs	12
4.4	C belongs to segment AB	12
4.5	Create new user	14
4.6	User authentication	15
4.7	Get user's drones	16
4.8	Get information about all live drones	17
4.9	Get information about all flights	17
4.10	Create drone	18
4.11	Drone authentication for the flight	19
4.12	Update information about location	20
4.13	Get information about drone	21
4.14	Delete information about drone	22
4.15	Create flight	23
4.16	Get information about flight	24
4.17	Delete information about flight	25
5.1	Path-building algorithm result	26

List of Abbreviations

JSON	J ava S cript O bject N otation
REST	R epresentational S tate T ransfer
API	A pplication P rogramming I nterface
JWT	J SON W eb T oken
SQL	S tructured Q uery L anguage
ORM	O bject- R elational M apping
ANSP	A ir N avigation S ervice P rovider
ATM	A ir T raffic M anagement
NFZ	N on-fly zone

Chapter 1

Introduction

The world is developing faster and faster nowadays. New technologies are replacing old ones each year. Opportunities for such new technologies appear as well.

Drones are incredibly popular: they are cheap, easy to come by, and fun to use. They can be used for all kinds of purposes, creating a whole field of new applications that are rapidly expanding. International reports show [26] [9] [7] that drones will create a multi-billion dollar market with new jobs and opportunities for innovative companies, such as increasing market of delivery and other services (e. g. medicine transportation). As the drone industry continues to grow, the number of drones in the sky is expected to increase exponentially over the coming years. So, special rules of traffic management should be implemented to complete the integration of drone traffic within the existing air traffic management (ATM) eco-system, as left unchecked, these drones present a serious threat to manned aviation. As for now, there are described rules for unmanned aircraft airspace management and several organizations are on the way of implementing them, in particular, FAA (Federal Aviation Administration) with NASA, SESAR Joint Undertaking.

Chapter 2

Related works

2.1 Air traffic management

Air traffic management systems [2] [6] [5] consist of many services that should work together in real-time. Their main purposes are the organization of air traffic (helping planes to take off, transit through the airspace and safely get to the destination), prevention of collisions of aircraft with other aircrafts or some obstacles during the flight

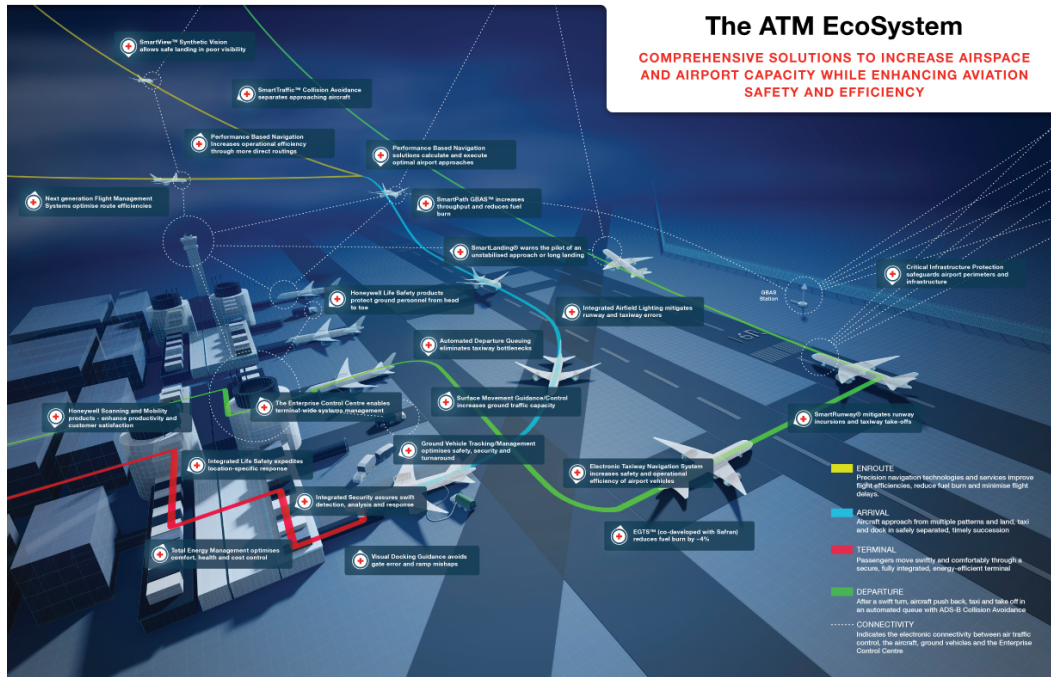


FIGURE 2.1: Modern Air Traffic Management [23]

in the airspace, providing some other information (e. g. weather conditions or notification about some events). To minimize the number of possible collisions between aircrafts aviation authorities designed special separation rules, that is the minimal distance in all directions that should be followed during the whole flight [34].

Identification of the aircraft in the airspace is made using radar. There are two types of radars airspace controllers usually use: primary radar [14] and secondary radar [25].

Primary air traffic control radars have the fan-shaped beam, narrow in the horizontal direction and wide in the vertical direction to reach high-flying planes. This beam scans around in a circle once every two-three seconds and echoes are displayed on a circular display. The air traffic controller can track the echoes on display to determine where the aircraft is heading.

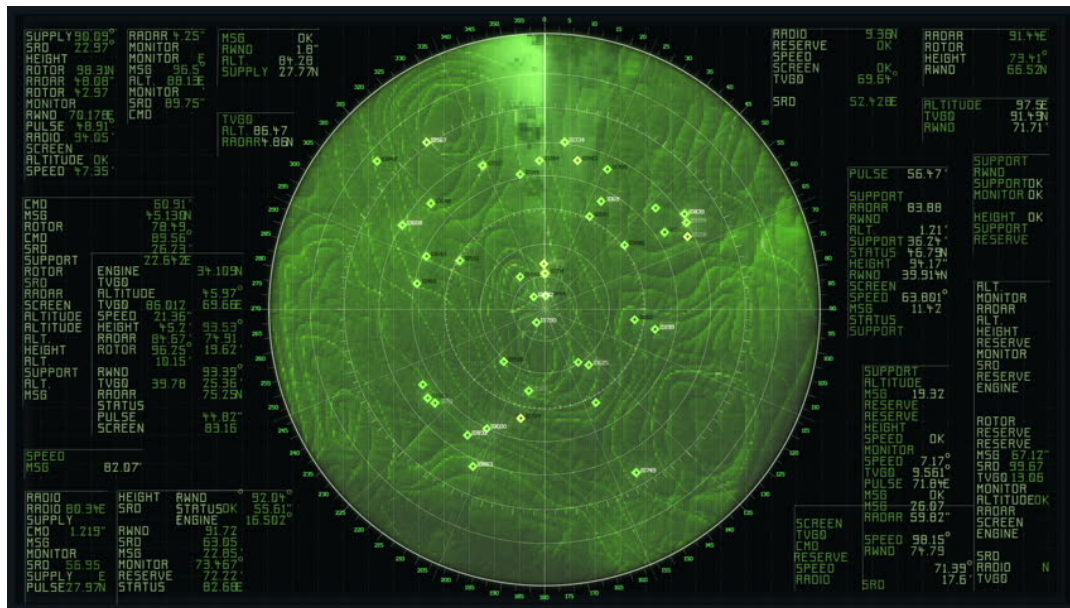


FIGURE 2.2: Example of radar screen [11]

Secondary radar is partly a communication system between aircraft and air traffic controllers on the ground. A limited amount of information (e.g. aircraft height and flight identification number) is requested by a controller on the ground and automatically supplied by a transponder on the aircraft [17] [12]. This used to be called identification friend or foe, or IFF [20]. Secondary radar also acts as a radar system because the position of the aircraft is found by measuring the range (from the time delay between request and reply) and the azimuth, as measured by an antenna on the ground.

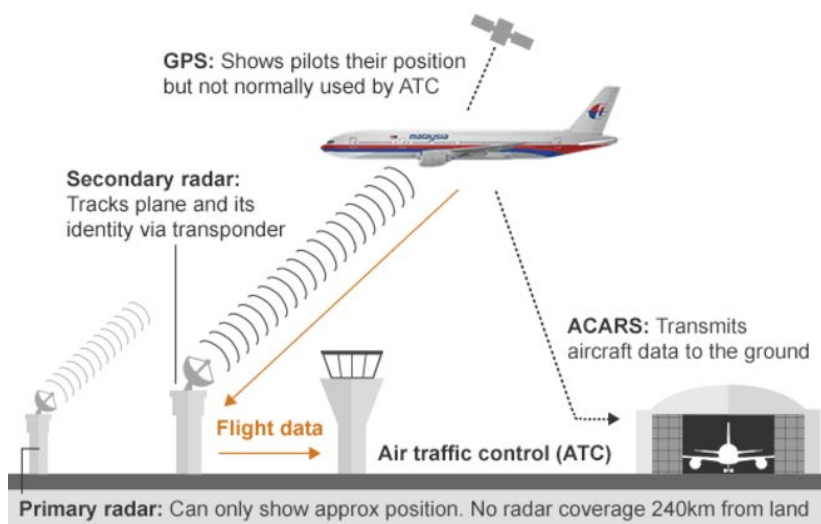


FIGURE 2.3: Aircraft tracking [3]

Nowadays, secondary radar is used more than the primary radar, because there is a huge amount of planes in the airspace, so it is easier to communicate with them by means of secondary radar.

2.2 Unmanned aircraft traffic management

As unmanned aviation (especially for delivering or long flights) is only becoming a common thing, so rules for it are only in the developing stage. However, there are several companies that already develop ATM systems, but they are responsible for very limited areas of applying.

2.2.1 U-space

U-space [29] is a set of new services and specific procedures designed to support safe, efficient and secure access to airspace for large numbers of drones. It was designed by SESAR - the mechanism which coordinate EU research and development (R&D) activities in ATM. They came up with the solution on how to build a good system for unmanned aircraft ATM. It should consist of four layers:

- U-space foundation services that provide e-registration, e-identification and geofencing.
- U-space initial services may include flight planning, flight approval, tracking, airspace dynamic information, and procedural interfaces with air traffic control.
- U-space advanced services support more complex operations in dense areas and may include capacity management and assistance for conflict detection.
- U-space full services offer integrated interfaces with manned aviation, support the full operational capability of U-space and will rely on very high level of automation, connectivity and digitalisation for both the drone and the U-space system.

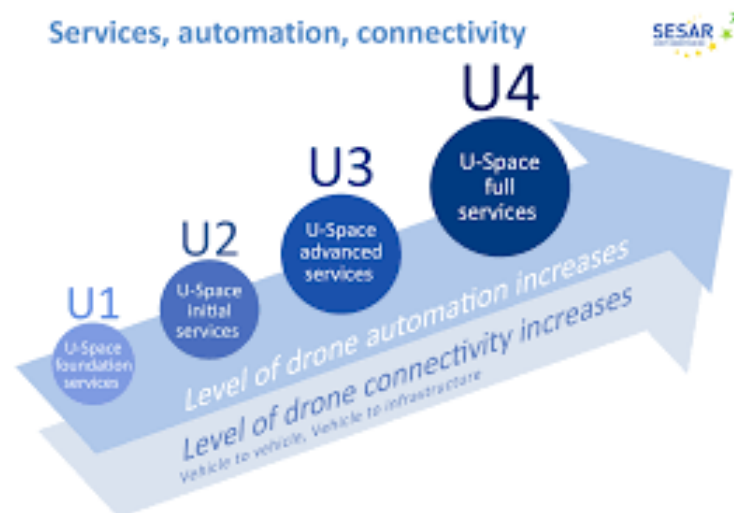


FIGURE 2.4: U-Space services [30]

2.2.2 Unifly

Unifly [31] is a private company founded by air traffic controllers and pilots. Their solution is also based on U-Space blueprint, and they try to implement its layers

step-by-step. For now, they developed an ecosystem that could work in Germany, Austria, Denmark and Belgium due to contracts with local ANSPs. Their solution includes Flight View module that designed to display both drone traffic and manned traffic on a UTM supervisor screen, Map Manager that allows managing geographical data, Administration module allows the management of users or drones registration, The Flight Approval module will confirm if the drone operation is allowed or not, etc. Also, they provide web and mobile applications for drone operators.

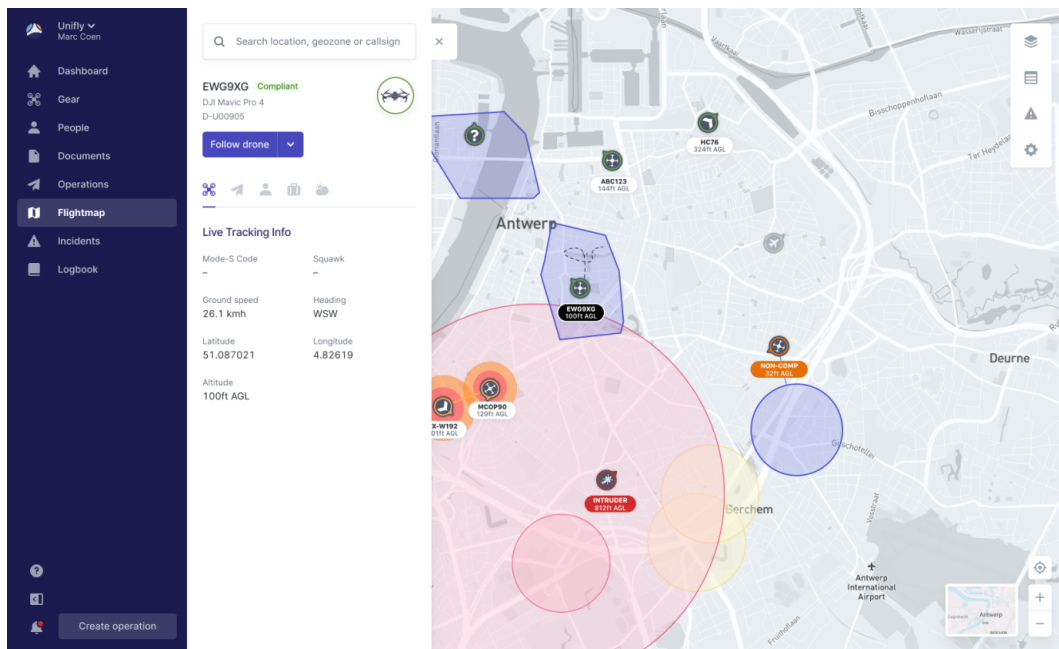


FIGURE 2.5: Unify web application [31]

2.2.3 Matternet

One more example is Matternet [22]. This is a private company that develops system mainly for healthcare, but also e-commerce and logistic organizations. As institutions in the medical domain sometimes urgently needed in some medicine or even organ transportation to hospitals which are not provided with it at the moment - drone transportation could help in such transportation. Their solution includes Matternet M2 Drone, the Matternet Station and the Matternet Cloud Platform. The Stations are the endpoints for drone flights and work only with M2 Drones.

Chapter 3

Background information

3.1 Interaction of the system with the outside world

Our system is a server which aggregates the processing of all requests that were addressed to it, calculations and data transformations, all business logic and flows, accesses to needed data. The general flow of interaction between all needed parts looks like the following:

- some client sends the request to our system; each request corresponds to the specific use case that should be done;
- based on given information there are some calculations or other actions based on flow inside the system;
- if the given information is not enough - then the system will ask some data resource to provide it;
- the client receives the corresponding response.

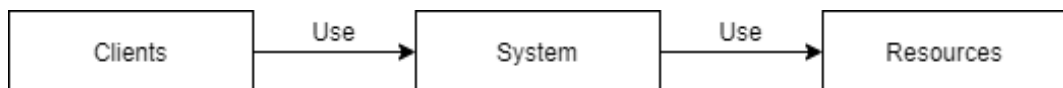


FIGURE 3.1: Interaction of the system

3.2 Clients

Our system is designed and built to be used by clients. Usually, in the web-applications front-end part of the whole application is the main client. There are several ways to connect the front-end and back-end parts of the application. The most common one is REST using HTTP requests. Also, this way of communication with the server could be used not only by front-end client but also by other clients (e.g. another server that needs some data from our server). Clients that could probably use our system are officer portal (to check if the system is consistent) or drones themselves. As usually unmanned aircraft use GSM [10] or WiMAX [33] technologies as a way of the communication, and therefore have access to the Internet, they could easily communicate with our system via it.

3.3 Resources

To function properly, our application should not only implement business logic and do some calculations but also should have the ability to operate with some data.

These data could be obtained by sending requests and getting a response from other services by API or by connecting to some database server and getting data using ORM. Despite the fact databases are not part of the system, sometimes they could be developed and used by the same development team and mistakenly considered as part of the system. In this case, they should also be considered as independent instances.

3.4 System

During the system's development, we should think about the separation of different logic parts, so changes in one part will not impact other parts. So, we have developed the architecture of the system that consists of the following layers [18] [4]: endpoint, managers, engines, resource accesses, utils. This type of decomposition is also called volatility-based decomposition. All parts of the systems encapsulate some part of logic in themselves.

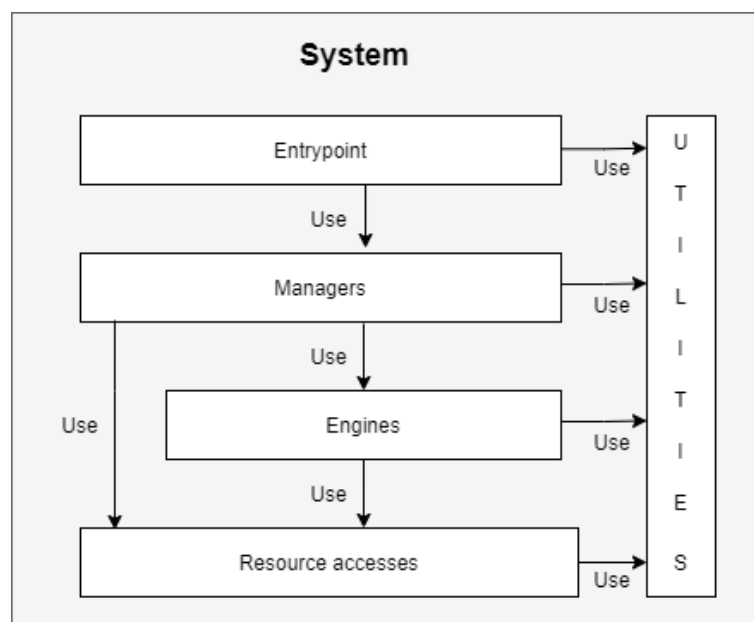


FIGURE 3.2: System layers

3.4.1 Entrypoint

Entrypoint stands for communication with clients. Web-portal for officer could communicate with the system over REST API, while aircrafts could communicate over web-sockets or use REST API as well. Other possible ways of communication are scheduled jobs, GUI. The way clients communicate with systems does not affect its other parts. If we need to change the protocol of communication we simply replace it with another library or framework, as the flows of business logic that will be called from endpoint will remain the same - we should only think of converting the result to the appropriate format for the delivery mechanism.

3.4.2 Managers

Managers encapsulate business logic or use cases in themselves. Each use case is consisting of some other smaller activities. These activities, such as data transformation, some calculations or data access, are located in other parts of the system that will be called by Manager.

Usually, there are several Managers in the system, and they stand for different groups of related use cases.

3.4.3 Engines

Engines encapsulate part of business logic in themselves, and they are mainly responsible for calculations and data transformation. All such activities are grouped in different Engines by main activity of each Engine. If some extra data needed during the activity flow, then Engine calls corresponding Resource access.

3.4.4 Resource accesses

Resource accesses are adapters between the system and resources. This part of the system could be changed most frequently as we can change the database to store our data or data provider's API, but our system should not be affected by these changes.

If some data is needed inside specific activity - Resource access could be called directly from Engine. However, if a few consecutive activities will need these data, then Resource access could be called from the Manager.

3.4.5 Utilities

Utilities are the cross-system part that consists of additional packages, that does not relate to business logic but provide additional services (e.g. logging).

Chapter 4

Approach

4.1 Scope

In our work, we focus on the first and partly on the second layers of U-Space services. We provide registration and identification of drones and users, as well as geofencing (possibility to avoid obstacles) from the first set of services. From the second layer, we provide a basic implementation of flight planning, flight approval, tracking, airspace dynamic information and we do not provide support of procedural interfaces with air traffic control as it requires strong interaction with local ANSP. These activities were divided into corresponding Engines which are assigned to corresponding Managers.

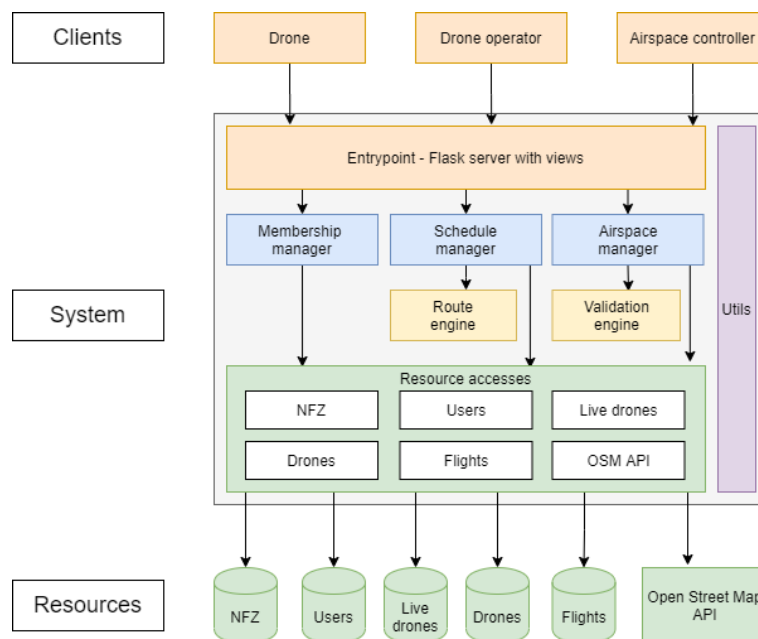


FIGURE 4.1: Architecture of the system with involved agents

To implement such a system, we decided to use Python programming language [32] for a few reasons [19].

Python provides high development quality as it has a very simple syntax. Comparing with compiled or strong typed programming languages the source code volume usually is times smaller. Thus, source code written in Python is easier to read, reuse and support.

Also, Python has a huge amount of support packages which make development easier.

4.2 Resources

4.2.1 Database

Most of our data is stored in MongoDB [15]. It is more convenient for our application for a few reasons.

First of all, we store flights and non-fly zones in the database. Each flight path and all non-fly zones are represented as the lists of coordinates. To represent them well in SQL, we should use two schemas - for path/zone itself and all related coordinate points (connection 1-N). However, MongoDB is NoSQL [1] storage, which works with documents, represented in JSON objects, and it does not require the description of such documents. So, we can simply create the specific key in such a JSON object with a list of coordinates as the corresponding value.

Also, MongoDB is known to be faster than SQL-based databases in some cases [24]. It is an important aspect, as we have collection for unmanned aircrafts that are live in our airspace. We need to update their position and get it in almost real-time. However, it is not as fast as an in-memory database called Redis [16] which is also NoSQL storage, but this is a key-value store while MongoDB is a document store. Usually, it is used as cache as it holds data in RAM with the option to write data to the database.

So, we decided to use MongoDB as a compromise decision between speed and usability aspects.

Collections in which we store documents:

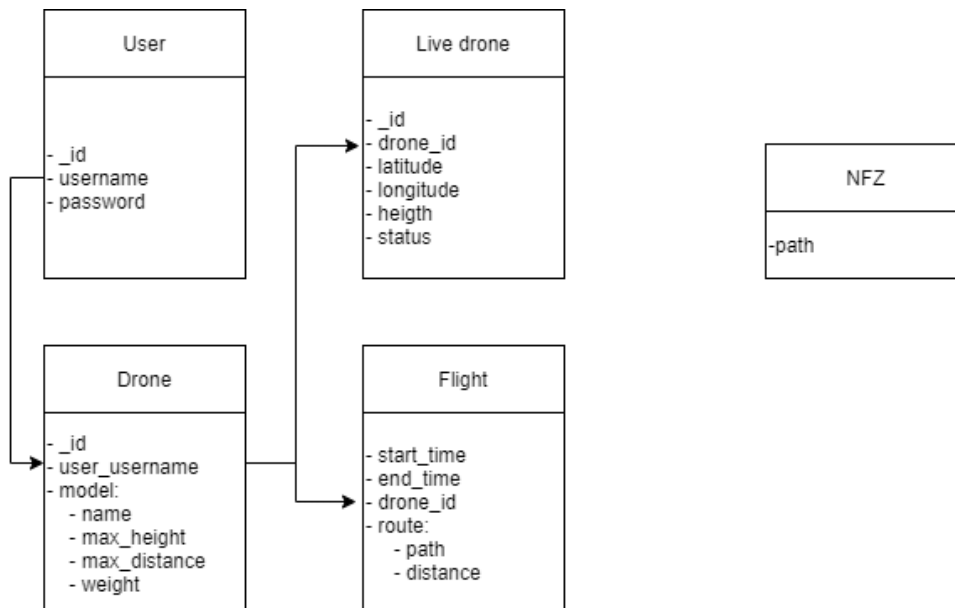


FIGURE 4.2: Documents example

4.2.2 Third-party API

However, not all data needed for our system is located in the databases. Some part of our logic is related to geographical data. Nevertheless, there are cases when these pieces of data are not coordinates of specific place themselves, but just a plain text - the name of some location. To deal with this problem, we use OpenStreetMap API. By giving the name of the location, it returns the exact geographical coordinates of that place.

4.3 Managers

As managers are part of the system that encapsulate business logic in themselves, and there are quite different use cases in the scope of our system - we decided to group them into the following three managers.

4.3.1 Membership manager

Main manager use cases are registration and identification of users and drones. Registration of the user is done by a unique username and password that will be hashed. During authentication, the user will send their username and password and based on these data, JWT will be generated. This token then will be sent back and can be used to make requests related to operations with drones.

Registration of drone could be done by the user after authentication step, and then it will be linked to this user in the database.

4.3.2 Schedule manager

The main goal of this manager's use cases is everything related to flight planning. When the user is planning to establish the flight they give all needed information, that is information about start and end points of the flight (these points could be in coordinates notation or in place's name notation that will be converted to coordinates one), information about the time when a flight should take place and one of the drones that are registered by this user.

4.3.3 Airspace manager

This manager is responsible for live airspace events. Authentication of the drone before the beginning of the flight should take place with JWT issued to the user before. If the system does not find a connection between the user and drone - flight might not be started. Moreover, during the flight drone should regularly send information about its current position in the airspace, that is latitude, longitude and height.

4.4 Engines

To make the application flexible and decomposed, we identified two Engines in our architecture. They encapsulate parts of calculation logic inside themselves in such a way that changes in one engine will not affect the functionality of other engines. Changes in resource accesses that are used inside engines and changes in managers that call specific engine will not affect the engine, as well.

4.4.1 Path Engine

The main goal of this engine is to build the path by given route, where the route is coordinates of the start and the end, while the path is a sequence of adjacent coordinate points. Using appropriate resource accesses it can operate with such data as a list of NFZs to build path correctly and flights that are compatible with the given flight.

To build a path we applied an algorithm that first checks if there any NFZs (red rectangles on Figure 4.3) intersecting direct segment (grey line on Figure 4.3) from

start point to the end point (black dots on Figure 4.3). To achieve this, we use calculations in 2D space, as we assume that flights will not be too long in the distance and the sphericity of the Earth will not have an effect on calculations. Then we use a function that calculates the convex hull [8] of all point of all NFZs that are between route points to find the convex ways (green lines on Figure 4.3) that will not intersect with any of given NFZ. Then we select the shorter of the convex ways (green line beneath the level of grey line on Figure 4.3) Although, this algorithm will work excellently in most situations, it is not 100% accurate and could show the worse result in edge situations.

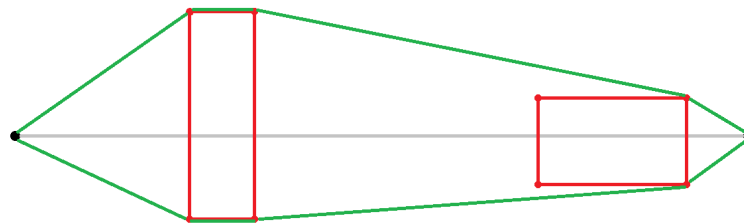


FIGURE 4.3: Making hull to avoid NFZs

Also, in this engine, there are checks if the flight with this path could be allowed to be done. For example, if the length of the flight's path is shorter than maximum distance drone assigned to this flight could cover (the length of the path is calculated by summing length of segments the path consist of; length of each segment is calculated using Vincenty's method [28]); or if this flight's path does not intersect with other flights' paths in a given area at the same point of time.

4.4.2 Validation Engine

The main purpose of this engine is to validate the state of the drone in the live airspace. First of all, there is a regular check of the time when specific unmanned aircraft has sent its data. If the time value extends the threshold - the system will change the status of this drone. Also, there is a validation if the drone is keep going in his way and has not deviated from the right direction. We can calculate it by checking firstly if segment and point are aligned using vector cross product [27] and if so, whether it belongs to the segment or not (Figure 4.4).

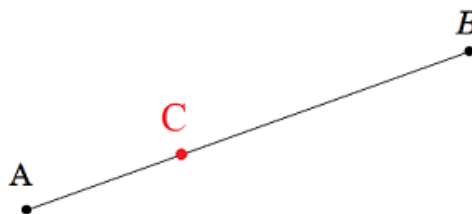


FIGURE 4.4: C belongs to segment AB

4.5 Entrypoint

For communication with clients, we provide API. All information about corresponding requests and responses they provide could be found in the Figures:

- Create user - Figure 4.5
- User authentication - Figure 4.6
- Get all drones of specific user - Figure 4.7
- Get info about all live drones - Figure 4.8. Note, that if during the flight drone will not send data about its position or will deviate from its route - the status of the corresponding drone will be changed, and client that will receive a response will be able to notify responsible people (e. g. air traffic controller)
- Create drone - Figure 4.10
- Drone authentication for the flight - Figure 4.11
- Update information about drone location - Figure 4.12
- Get information about drone - Figure 4.13
- Delete information about drone - Figure 4.14
- Get information about all flights = Figure 4.9
- Create flight - Figure 4.15
- Get information about flight by id - Figure 4.16
- Delete flight - Figure 4.17

POST /users/registration Create user

Request body required

Example Value | Schema

```
{
  "username": "username",
  "password": "password"
}
```

Responses

Code	Description	Links
201	User created	No links
Media type: <input type="text" value="application/json"/>		
<small>Controls Accept header.</small>		
Example Value Schema		
<pre>{ "description": "User created" }</pre>		
400	Invalid data	No links
Media type: <input type="text" value="application/json"/>		
Example Value Schema		
<pre>{ "description": "User with this username already exists" }</pre>		

FIGURE 4.5: Create new user

PATCH /users/auth User authentication

Request body required

Example Value | Schema

```
{
  "username": "username",
  "password": "password"
}
```

Responses

Code	Description	Links
200	User successfully authenticated	No links

Media type

Controls Accept header.

Example Value | Schema

```
{
  "token":
  "sv64w549w.sdf9w4v9wdd8f4w98fw4f.wf98w4fv9e8f4we9f"
}
```

400	Invalid data	No links
-----	--------------	----------

Media type

Example Value | Schema

```
{
  "description": "Password does not match"
}
```

404	User not found	No links
-----	----------------	----------

Media type

Example Value | Schema

```
{
  "description": "User with this username does not
  exist"
}
```

FIGURE 4.6: User authentication

GET `/users/drones` Returns a list of user's drones

Parameters

Name	Description
token * required	Specify user's token
string (query)	<input type="text" value="sv64w549w.sdf9w4v9wdd8f4w98fw4f.wf98w4fw9e8f4we9f"/>

Responses

Code	Description	Links
200	OK	No links
Media type: <input type="text" value="application/json"/>		
Controls Accept header.		
Example Value Schema		
<pre>{ "drones": [{ "id": "99df55fr35sd559r8", "max_height": 500, "max_distance": 20, "weight": 2, "name": "M1" }] }</pre>		
400	Invalid token	No links
Media type: <input type="text" value="application/json"/>		
Example Value Schema		
<pre>{ "description": "Invalid token" }</pre>		

FIGURE 4.7: Get user's drones

The screenshot shows a REST client interface for the endpoint `GET /drones` with the description "Get info about all live drones". The response is a 200 OK status with no links. The media type is set to `application/json`. The example value is a JSON array of drone objects.

```
{
  "drones": [
    {
      "id": "45sdg49sv948av65y19",
      "drone_id": "99df55fr35sd559r8",
      "lat": 46.138465,
      "lon": 46.235465,
      "height": 300,
      "status": "OK | DEVIATION | NO SYGNAL",
      "last_updated": "2020-04-20:12-00-00"
    }
  ]
}
```

FIGURE 4.8: Get information about all live drones

The screenshot shows a REST client interface for the endpoint `GET /flights` with the description "Get info about flights". The response is a 200 OK status with no links. The media type is set to `application/json`. The example value is a JSON array of flight objects.

```
{
  "flights": [
    {
      "id": "1",
      "route": {
        "path": [
          [
            26.54,
            26.74
          ],
          [
            26.54,
            26.34
          ]
        ],
        "distance": 2.798
      },
      "drone_id": "99df55fr35sd559r8",
      "start_time": "2020-04-20:12-00-00",
      "end_time": "2020-04-20:13-00-00"
    }
  ]
}
```

FIGURE 4.9: Get information about all flights

POST /drones Create drone

Request body required application/json

Example Value | Schema

```
{
  "token": "sd4sdf949.sdv749d48a7dv.a648vwdv7wd",
  "max_height": 500,
  "max_distance": 20,
  "weight": 2,
  "name": "M1"
}
```

Responses

Code	Description	Links
201	Drone created	No links
	Media type application/json <small>Controls Accept header.</small> Example Value Schema <pre>{ "description": "Drone created" }</pre>	
400	Missing required field	No links
	Media type application/json Example Value Schema <pre>{ "description": "Missing required field" }</pre>	
404	Invalid token	No links
	Media type application/json Example Value Schema <pre>{ "description": "Invalid token" }</pre>	

FIGURE 4.10: Create drone

POST /drones/{id} Drone authentication for flight

Parameters Try it out

Name	Description
id * required	Drone id
string (path)	<input type="text" value="id - Drone id"/>

Request body required application/json

Example Value | Schema

```
{
  "token": "sd4sdf949.sdv749d48a7dv.a648vvdv7wd"
}
```

Responses

Code	Description	Links
200	OK	No links
	Media type application/json Controls Accept header:	
	Example Value Schema	
	<pre>{ "description": "Success" }</pre>	
400	Invalid token	No links
	Media type application/json	
	Example Value Schema	
	<pre>{ "description": "Invalid token" }</pre>	
404	No drone with given id	No links
	Media type application/json	
	Example Value Schema	
	<pre>{ "description": "No drone with given id" }</pre>	

FIGURE 4.11: Drone authentication for the flight

PATCH /drones/{id} Update drone location

Parameters

Name	Description
id * required	Drone id
string (path)	<input type="text" value="id - Drone id"/>

Request body required application/json

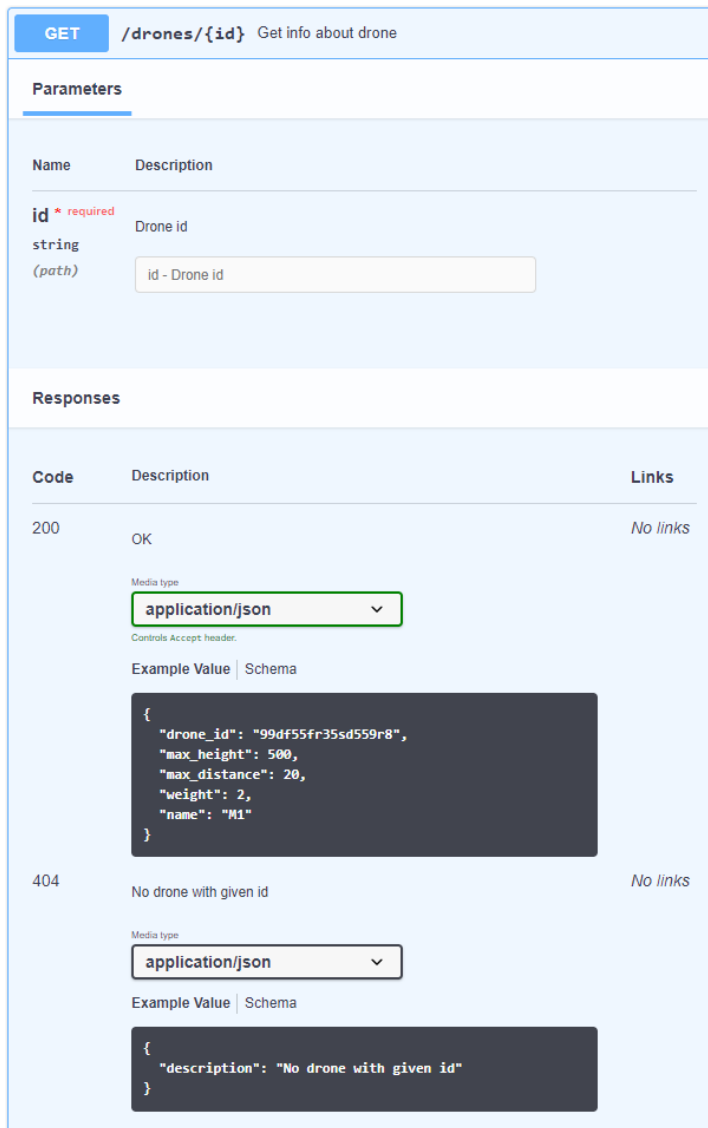
Example Value | Schema

```
{
  "token": "sd4sdf949.sdv749d48a7dv.a648vwdv7wd",
  "lat": 37.254685,
  "lon": 53.154235,
  "height": 350
}
```

Responses

Code	Description	Links
200	OK	No links
	Media type <input type="text" value="application/json"/> <small>Controls Accept header.</small> Example Value Schema <pre>{ "description": "Location updated" }</pre>	
400	Invalid data	No links
	Media type <input type="text" value="application/json"/> Example Value Schema <pre>{ "description": "Invalid data" }</pre>	
404	No drone with given id	No links
	Media type <input type="text" value="application/json"/> Example Value Schema <pre>{ "description": "No drone with given id" }</pre>	

FIGURE 4.12: Update information about location



The image shows a Swagger UI interface for a REST API endpoint. At the top, it indicates a GET method for the path `/drones/{id}` with the description "Get info about drone".

Parameters

Name	Description
id * required string (path)	Drone id

Below the parameter table, there is a text input field containing the value "id - Drone id".

Responses

Code	Description	Links
200	OK	No links

For the 200 response, there is a "Media type" dropdown menu set to "application/json". Below it, there are tabs for "Example Value" and "Schema". The "Example Value" tab is active, showing a JSON object:

```
{
  "drone_id": "99df55fr35sd559r8",
  "max_height": 500,
  "max_distance": 20,
  "weight": 2,
  "name": "M1"
}
```

| 404 | No drone with given id | No links |

For the 404 response, there is also a "Media type" dropdown menu set to "application/json". Below it, there are tabs for "Example Value" and "Schema". The "Example Value" tab is active, showing a JSON object:

```
{
  "description": "No drone with given id"
}
```

FIGURE 4.13: Get information about drone

DELETE /drones/{id} Delete info about drone

Parameters Try it out

Name	Description
id * required	Drone id
string (path)	<input type="text" value="id - Drone id"/>

Responses

Code	Description	Links
200	OK	No links
Media type: <input type="text" value="application/json"/>		
Controls Accept header.		
Example Value Schema		
<pre>{ "description": "Drone deleted" }</pre>		
404	No drone with given id	No links
Media type: <input type="text" value="application/json"/>		
Example Value Schema		
<pre>{ "description": "No drone with given id" }</pre>		

FIGURE 4.14: Delete information about drone

POST /flights Create flight

Request body *required* application/json

Example Value | Schema

```

{
  "start_point": [
    24.25465,
    54.8914
  ],
  "end_point": [
    42.56452,
    45.4198
  ],
  "start_time": "2020-04-20:12-00-00",
  "end_time": "2020-04-20:13-00-00",
  "drone_id": "45sdg49sv948av65y19",
  "token": "bdv5a4dva5.cdv56a1dv55ad5d5v5h.rc54sdv54kv"
}
    
```

Responses

Code	Description	Links
200	OK	No links
	<p>Media type: application/json</p> <p>Controls: Accept header.</p> <p>Example Value Schema</p> <pre> { "description": "Flight created" } </pre>	
400	Invalid token	No links
	<p>Media type: application/json</p> <p>Example Value Schema</p> <pre> { "description": "Invalid token" } </pre>	
404	No drone with given id	No links
	<p>Media type: application/json</p> <p>Example Value Schema</p> <pre> { "description": "No drone with given id" } </pre>	
409	Fail	No links
	<p>Media type: application/json</p> <p>Example Value Schema</p> <pre> { "description": "Cannot create flight with given route at given time" } </pre>	

FIGURE 4.15: Create flight

GET /flights/{id} Get info about flight

Parameters

Name	Description
id * required string (path)	Flight id

id - Flight id

Responses

Code	Description	Links
200	OK	No links

Media type: application/json

Controls Accept header.

Example Value | Schema

```
{
  "route": {
    "path": [
      [
        26.54,
        26.74
      ],
      [
        26.54,
        26.34
      ]
    ],
    "distance": 2.798
  },
  "drone_id": "99df55fr35sd559r8",
  "start_time": "2020-04-20:12-00-00",
  "end_time": "2020-04-20:13-00-00"
}
```

| 404 | No flight with given id | No links |

Media type: application/json

Example Value | Schema

```
{
  "description": "No flight with given id"
}
```

FIGURE 4.16: Get information about flight

DELETE /flights/{id} Delete info about flight

Parameters

Name	Description
id * required string (path)	Flight id <input type="text" value="id - Flight id"/>

Responses

Code	Description	Links
200	OK	No links
	Media type <input type="text" value="application/json"/>	
	Controls Accept header.	
	Example Value Schema	
	<pre>{ "description": "Flight deleted" }</pre>	
404	No flight with given id	No links
	Media type <input type="text" value="application/json"/>	
	Example Value Schema	
	<pre>{ "description": "No flight with given id" }</pre>	

FIGURE 4.17: Delete information about flight

Chapter 5

Results

As a result of our work, we have designed the architecture for the decomposed system and implemented it using Python. This system can control the status of airspace and automatically create flights for it. Access to the system takes place via REST API.

To validate the stability and consistency of our system, we used Python script in which in several threads, we were sending requests to our server. As a result, there were no data collisions.

Example of our path-building algorithm result (red polygons are NFZs and blue polygonal chains are paths):

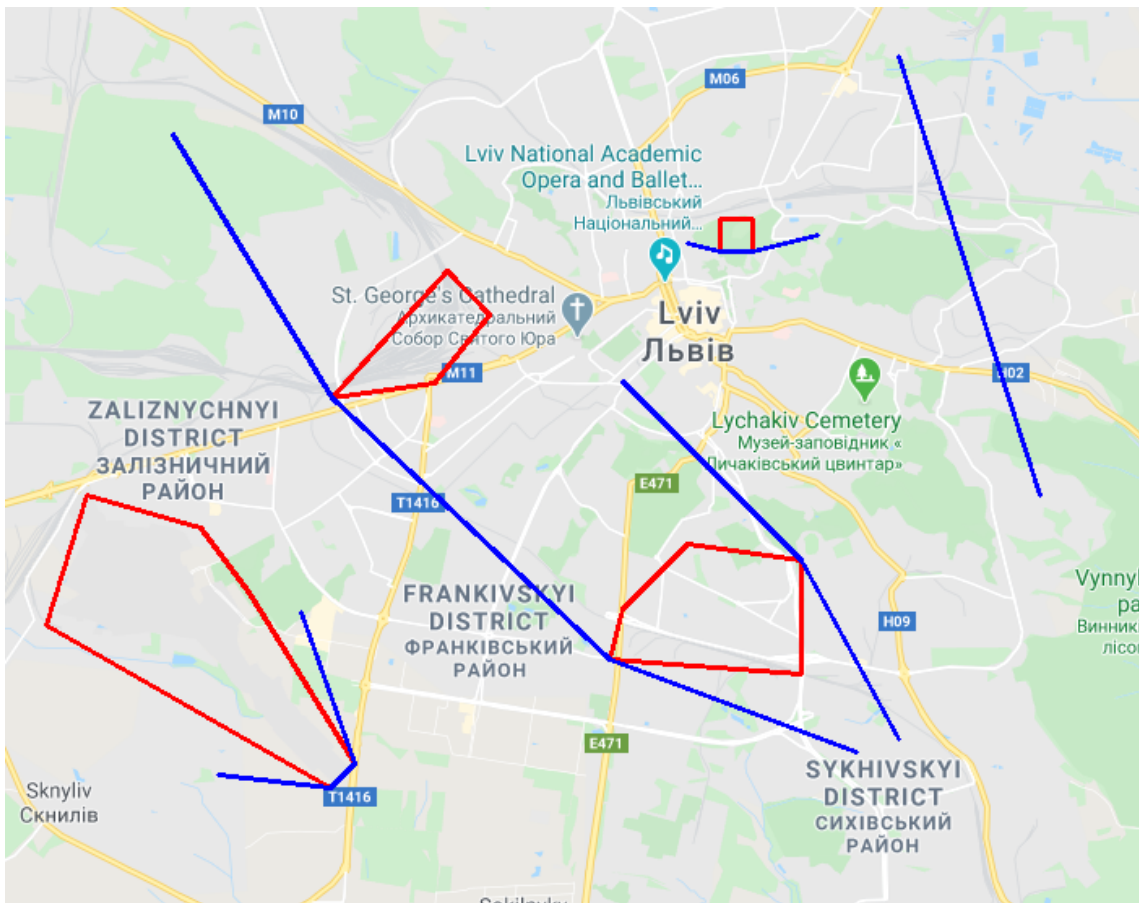


FIGURE 5.1: Path-building algorithm result

Chapter 6

Future works

As next steps, we could try to establish a front-end part for our system with the drone operator and air space controller portals. Drone operator on such portal could register drones and establish flights for them while the controller will have the ability to monitor the state of the live airspace.

One more point to improve is our path-building algorithm as it does not guarantee 100% accuracy. The possible solutions could be Dijkstra's Shortest Path First algorithm [21] or A* search algorithm [13].

Bibliography

- [1] Khaleel Ahmad and Mohd Javed. “Hands-On Redis”. In: *NoSQL: Database for Storage and Retrieval of Data in Cloud*. Chapman and Hall/CRC, 2017, pp. 355–364. DOI: 10.1201/9781315155579-21. URL: <https://doi.org/10.1201/9781315155579-21>.
- [2] “Air traffic management”. In: *Aircraft Engineering and Aerospace Technology* 74.5 (2002). DOI: 10.1108/aeat.2002.12774eab.044. URL: <https://doi.org/10.1108/2Faeat.2002.12774eab.044>.
- [3] *Aircraft tracking*. URL: https://ichef.bbci.co.uk/news/660/media/images/73552000/gif/_73552365_boeing777_malaysianflight_624gr_v3.gif.
- [4] Mattia Battiston. *Clean Architecture*. Github. 2016. URL: <https://github.com/mattia-battiston/clean-architecture-example>.
- [5] Lucio Bianco and Maurizio Bielli. “Air traffic management: Optimization models and algorithms”. In: *Journal of Advanced Transportation* 26.2 (1992), pp. 131–167. DOI: 10.1002/atr.5670260205. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/atr.5670260205>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/atr.5670260205>.
- [6] Robert Caves and Cheng-Lung Wu. “Research review of air traffic management”. In: *Transport Reviews - TRANSP REV* 22 (Jan. 2002), pp. 115–132. DOI: 10.1080/01441640110074773.
- [7] *Commercial Unmanned Aerial Vehicle (UAV) Market Analysis – Industry trends, forecasts and companies*. 2020. URL: <https://www.businessinsider.com/commercial-uav-market-analysis>.
- [8] *Convex Hull*. 2007. DOI: 10.3840/000404. URL: <https://doi.org/10.3840/2F000404>.
- [9] *Drone market outlook: industry growth trends, market stats and forecast*. 2020. URL: <https://www.businessinsider.com/drone-industry-analysis-market-trends-growth-forecasts>.
- [10] Philippe Dupuis. “Consolidating GSM Phase 1 and Evolving the Services and Features to GSM Phase 2 in ETSI SMG (1992–1995)”. In: *GSM and UMTS*. John Wiley & Sons, Ltd, pp. 61–71. DOI: 10.1002/0470845546.ch4. URL: <https://doi.org/10.1002/2F0470845546.ch4>.
- [11] *Example of radar screen*. URL: <https://ak.picdn.net/shutterstock/videos/6657959/thumb/1.jpg>.
- [12] “General aviation transponder”. In: *Aircraft Engineering and Aerospace Technology* 78.5 (2006). DOI: 10.1108/aeat.2006.12778eab.014. URL: <https://doi.org/10.1108/2Faeat.2006.12778eab.014>.
- [13] P. E. Hart, N. J. Nilsson, and B. Raphael. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. ISSN: 2168-2887. DOI: 10.1109/TSSC.1968.300136.

- [14] Thomas Huber-Obst. "Target filtering for military ATC primary radar". In: *2018 19th International Radar Symposium (IRS)*. IEEE, 2018. DOI: [10.23919/irs.2018.8448061](https://doi.org/10.23919/irs.2018.8448061). URL: <https://doi.org/10.23919/irs.2018.8448061>.
- [15] "Introduction to MongoDB". In: *The Definitive Guide to MongoDB*. Apress, 2010, pp. 3–17. DOI: [10.1007/978-1-4302-3052-6_1](https://doi.org/10.1007/978-1-4302-3052-6_1). URL: https://doi.org/10.1007/978-1-4302-3052-6_1.
- [16] Gurpreet Kaur and Jatinder Kaur. "In-Memory Data processing using Redis Database". In: *International Journal of Computer Applications* 180.25 (2018), pp. 26–31. DOI: [10.5120/ijca2018916589](https://doi.org/10.5120/ijca2018916589). URL: <https://doi.org/10.5120/ijca2018916589>.
- [17] H. Lans. "A GPS/GNSS transponder for use in aircraft". In: *Proceedings of VNIS 93 - Vehicle Navigation and Information Systems Conference*. IEEE. DOI: [10.1109/vnis.1993.585731](https://doi.org/10.1109/vnis.1993.585731). URL: <https://doi.org/10.1109/vnis.1993.585731>.
- [18] J. Lowy. *Righting Software*. Addison Wesley, 2019. ISBN: 9780136524038. URL: <https://books.google.com.ua/books?id=boT4xgEACAAJ>.
- [19] M. Lutz and an O'Reilly Media Company Safari. *Learning Python, 5th Edition*. O'Reilly Media, Incorporated, 2013. URL: <https://books.google.com.ua/books?id=LWM6zQEACAAJ>.
- [20] Daniel L. Magruder. *Counterinsurgency, Security Forces, and the Identification Problem*. Routledge, 2017. DOI: [10.4324/9781315202341](https://doi.org/10.4324/9781315202341). URL: <https://doi.org/10.4324/9781315202341>.
- [21] Neha Makariye. "Towards shortest path computation using Dijkstra algorithm". In: *2017 International Conference on IoT and Application (ICIOT)*. IEEE, 2017. DOI: [10.1109/iciota.2017.8073641](https://doi.org/10.1109/iciota.2017.8073641). URL: <https://doi.org/10.1109/iciota.2017.8073641>.
- [22] *Matternet*. 2020. URL: <https://mtr.net/>.
- [23] *Modern air traffic management*. URL: https://upload.wikimedia.org/wikipedia/commons/c/c5/Air_Traffic_Management.jpg.
- [24] *MongoDB vs. MySQL*. URL: <https://www.mongodb.com/compare/mongodb-mysql>.
- [25] "Secondary surveillance radar". In: *Understanding Radar Systems*. Institution of Engineering and Technology, pp. 159–170. DOI: [10.1049/sbra034e_ch7](https://doi.org/10.1049/sbra034e_ch7). URL: https://doi.org/10.1049/sbra034e_ch7.
- [26] *The Drone Market Report 2019: Commercial Drone Market Size and Forecast (2019-2024)*. 2019. URL: <https://www.researchandmarkets.com/reports/4764173/the-drone-market-report-2019-commercial-drone>.
- [27] "The Vector Cross Product". In: *Rotating Flow*. Elsevier, 2011, pp. 373–375. DOI: [10.1016/b978-0-12-382098-3.00015-9](https://doi.org/10.1016/b978-0-12-382098-3.00015-9). URL: <https://doi.org/10.1016/b978-0-12-382098-3.00015-9>.
- [28] C. M. Thomas and W. E. Featherstone. "Validation of Vincenty's Formulas for the Geodesic Using a New Fourth-Order Extension of Kivioja's Formula". In: *Journal of Surveying Engineering* 131.1 (2005), pp. 20–26. DOI: [10.1061/\(asce\)0733-9453\(2005\)131:1\(20\)](https://doi.org/10.1061/(asce)0733-9453(2005)131:1(20)). URL: [https://doi.org/10.1061/\(asce\)0733-9453\(2005\)131:1\(20\)](https://doi.org/10.1061/(asce)0733-9453(2005)131:1(20)).
- [29] *U-space : blueprint*. Luxembourg: Publications Office of the European Union, 2017. ISBN: 978-92-9216-087-6.

-
- [30] *U-Space services*. URL: <https://www.aerosociety.com/media/7267/u-space-blueprint-to-reality.pdf>.
- [31] *Unifly*. 2020. URL: <https://www.unifly.aero/solutions/unmanned-traffic-management>.
- [32] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.
- [33] “WiMAX Architecture”. In: *WiMAX*. John Wiley & Sons, Ltd, pp. 207–217. DOI: [10.1002/9780470319055.ch13](https://doi.org/10.1002/9780470319055.ch13). URL: <https://doi.org/10.1002/9780470319055.ch13>.
- [34] Zhao Ning Zhang, Zhen Liu, and Dong Man Zhang. “Research on the Latest Maneuver Moment of Aircraft and the Minimum Safety Distance between Aircrafts in Free Flight”. In: *Applied Mechanics and Materials* 538 (2014), pp. 134–145. DOI: [10.4028/www.scientific.net/amm.538.134](https://doi.org/10.4028/www.scientific.net/amm.538.134). URL: <https://doi.org/10.4028/www.scientific.net/amm.538.134>.