Core-System

HTTP

HTTP

drone-api

:80001

Save route, get info, get battery,
comm with core

MongoDB

RCS

drone-manager

:80002

:27017

Get route and control drone
(private protocol)

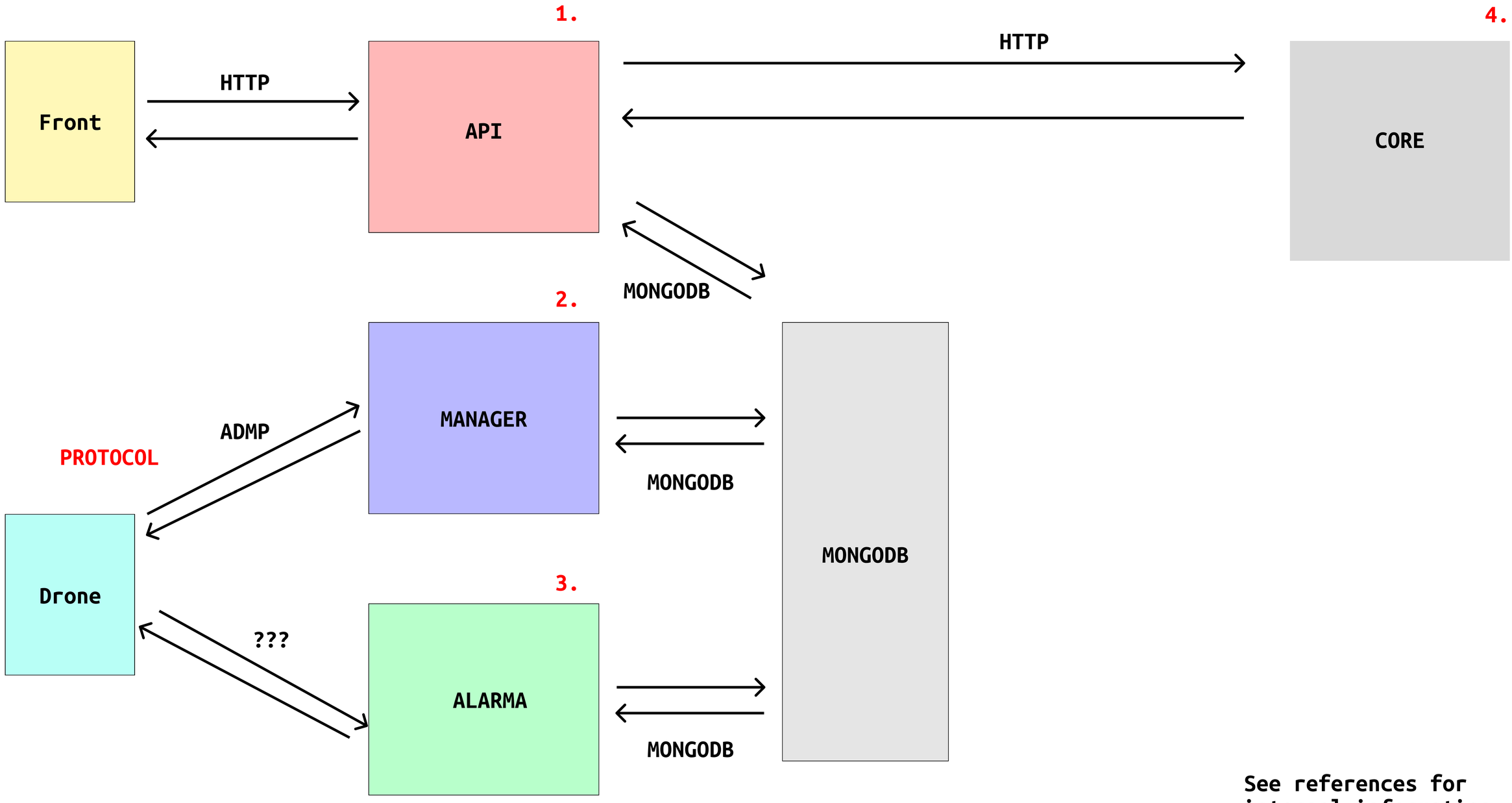
HTTP

drone-ai

:80003

Get images from drone and
check and modify alert in db

This database store temporal
data such **(routes, alarm,
battery, gps)**

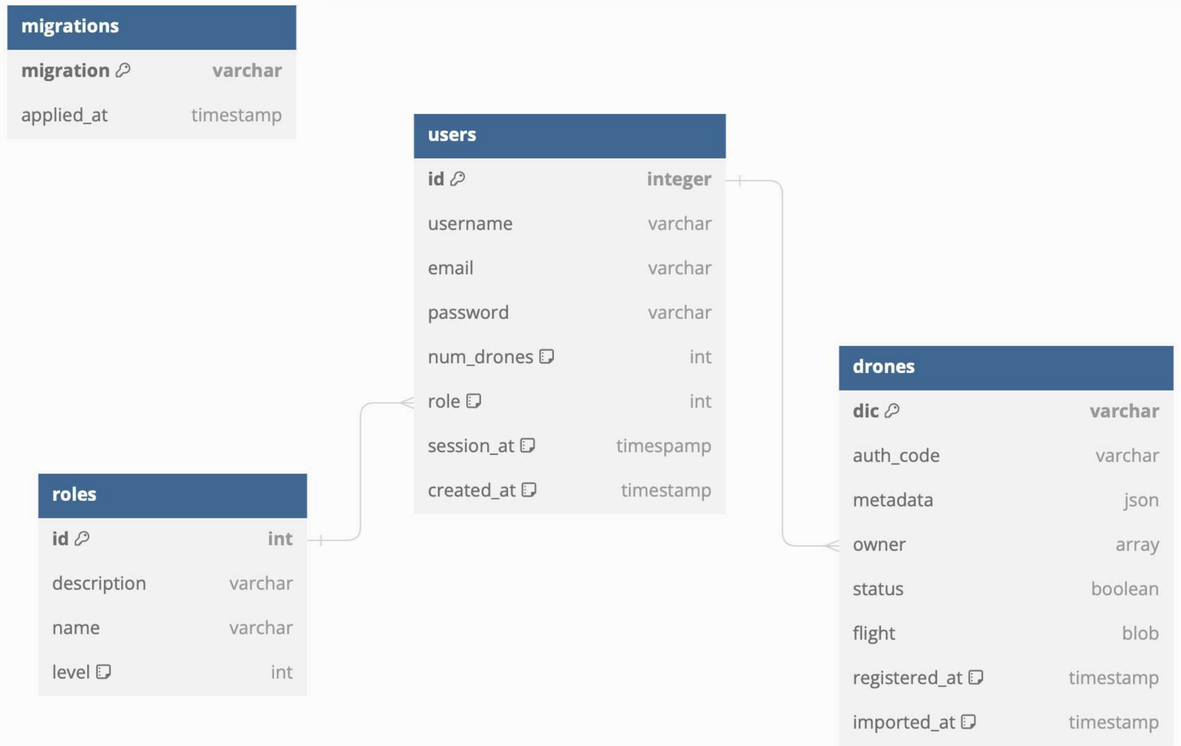


See references for internal information

```
// Use DBML to define your database structure
// Docs: https://dbml.dbdiagram.io/docs



Table users {
  id integer [primary key]
  username varchar
  email varchar
  password varchar
  num_drones int [note: 'Amount of drones']
  role int [note: 'Privilegies']
  session_at timespamp [note: 'Last logged time']
  created_at timestamp [note: 'Account creation time']
}

Table roles {
  id int [primary key]
  description varchar
  name varchar
  level int [note: 'Privilegy level (Low 0 - 9 High)']
}

Table drones {
  dic varchar [primary key]
  auth_code varchar
  metadata json
  owner array
  status boolean
  flight blob
  registered_at timestamp [note: 'First registration date']
  imported_at timestamp [note: 'First time imported to platform']
}

Ref: drones.owner > users.id
Ref: users.role > roles.id
```

drone-front.db

MySQL MariaDB



```
// Use DBML to define your database structure
// Docs: https://dbml.dbdiagram.io/docs

Table migrations {
  migration varchar [primary key]
  applied_at timestamp
}

Table alarms {
  id integer [primary key]
  description text
  level int [note: 'From 0 to 9 (Low to High)']
  action array [note: 'Array of actions to follow']
}

Table actions {
  id varchar [primary key]
  name varchar
}

Table drones {
  dic varchar [primary key]
  auth_code varchar
  metadata json
  status boolean
  flight blob
  alarm_status boolean
  alarm_code int
  default_action varchar [note: 'Default action set by drone in case critic']
  registered_at timestamp
}

Ref: drones.alarm_code > alarms.id
Ref: alarms.action > actions.id
Ref: drones.default_action > actions.id
```

drone-module.mongo

mongoDB.

ai_entities

id	varchar
server_ip	varchar
server_port	int
capacity	int
max_capacity	int

images

id	varchar
data	blob
expire_date	timestamp

alarms

id	integer
description	text
level	int
action	array

drones

dic	varchar
auth_code	varchar
metadata	json
status	boolean
flight	blob
alarm_status	boolean
alarm_code	int
default_action	varchar
registered_at	timestamp
alarm_data	varchar
flight_priority	int
flight_altitude	varchar

migrations

migration	varchar
applied_at	timestamp

actions

id	varchar
name	varchar

```
// Use DBML to define your database structure
// Docs: https://dbml.dbdiagram.io/docs

Table routes {
  dic varchar [primary key]
  auth_code varchar
  flight blob
}

Table registrants {
  id varchar [primary key]
  name varchar
  country varchar
  // More data, but not going to use in v1
}

Table drones {
  dic varchar [primary key]
  auth_code varchar
  registrant_code varchar [note: 'Any registrand platform code']
  owner_name varchar
  owner_identify varchar [note: 'Any identifible document']
  owner_email varchar
  registered_at timestamp
}

Ref: drones.registrant_code > registrants.id
Ref: routes.dic > drones.dic
```

core-system.db

1.

Funcionalidades del frontd

- Listar drones / minimal info de cada drone
- Mapa de todo los drones
- Crear rutas y ver rutas del drone
- Ver las alarmas, con imagen evidencia

Extra Información para todos

Las rutas creadas por el cliente son conjunto de coordenadas en formato **Grado decimal**.

Por ejemplo: [(1,2), (2,3), (3,4)]

Entre coordenada a coordenada no puede existir un gap mayor que 5m. Es decir 1/4 segundos en Europa

API Proporcionadas por el server

Toda peticion deberá llevar su auth_code en header **Autorization**

<dic> : Matricula del drone. Para obtener varias, separar por ;
Ejemplo: ESP12345-001-0001;ESP12340-002-0001

Y auth-code separadas por ; por el mismo orden de <dic>

Los datos en JSON se devolverá con key de la matricula del drone

```
GET  /metadata/[all, battery, gps, smoke, altitude, orientation]/<dic>
```

```
GET  /status/<dic>
```

```
GET  /route/<dic>
```

```
POST /route/<dic>    -----> Body: Conjunto de coordenadas
```

```
GET  /alarm/<dic>
```

```
GET  /alarm/image/<id>
```

```
{
    status: "error",
    message: "401 Unauthorized",
    path: "/metadata"
    errcode: 401
}
```

```
{
    status: "ok",
    path: "/status/ESP00001-001-9999;ESP00001-002-9999"
    data: {
        "ESP00001-001-9999": {
            status: 0
        },
        "ESP00001-002-9999": {
            status: 1
        }
    }
}
```

```
{
    status: "ok",
    path: "/alarm/ESP00001-001-9999"
    data: {
        "ESP00001-001-9999": {
            status: 1,
            image: {
                path: "/img/tmpESP00001-001-9999.png"
            }
        }
    }
}
```

Interacción con DDBB

- Guardar Coordenadas
- Recoger metadatos
- Recoger status
- Recoger alarma y crear archivo con evidencia

Interacción extra con DDBB

- Crear plantilla ddbb
- Aplicar migraciones

Drone

Server

ADMP

automated drone flight management protocol

```
DCMP 1.0\r\n
WAKE
ES46020-098-0001\r\n
SG9sYU11bmRvMTIzRWxlbmFFc0ZlYQ==\r\n
\r\n
set module camera\r\n
set module orientation\r\n
set module humidity\r\n
set module smoke\r\n
set timeval 100\r\n
```

```
DCMP 1.0\r\n
WAKE\r\n
201 OK\r\n
\r\n
set timeval 200\r\n
set camera_server 187.10.92.1\r\n
set camera_port 60003\r\n
```

```
DCMP 1.0\r\n
PUSH\r\n
ES46020-098-0001\r\n
SG9sYU11bmRvMTIzRWxlbmFFc0ZlYQ==\r\n
\r\n
{"gps": [12.01, 12.8], "alt": 60.3, "orientation": 48.3, "humidity": 12, "smoke": 100}\r\n
```

```
DCMP 1.0\r\n
RECEIVE\r\n
\r\n
action 124 10\r\n
action 100 0\r\n
action 124 5\r\n
```

• • •

```
DCMP 1.0\r\n
END\r\n
\r\n
action 1 0
```

```
register code:

    AAANNNNN-RRR-PPPP

    AAA   = COUNTRY CODE in ISO3166-1 alpha-3
    NNNNN = NUMBER
    RRR   = REGISTRANT CODE
    PPPP  = DRONE PROPUSE
        0001 Emergency drone
        0002 Police drone
        0003 Firefighters drone
        ...
        9999 Public drone
```

```
commands:


set <instruction> <value>
action <movement_code> <meters>


movement code:

1xx:
    100 Normal up
    101 Slow up (!)
    102 Fast up (!)
    ...

2xx:
    200 Normal down
    ...

3xx:
    300 Normal left
    ...

4xx:
    400 Normal right
    ...

5xx:
    500 Normal front
    ...

6xx:
    600 Normal back
    ...

7xx:
    700 Stop
```

```
code status:

1xx:
    100 Server Overload
    ...

2xx:
    200 Ok
    201 Ok, but modify
    ...

3xx:
    300 Server moved
    ...

4xx:
    400 Syntax error
    401 Authentication failed
    ...

5xx:
    500 Server collapsed
    501 Server cannot response
```

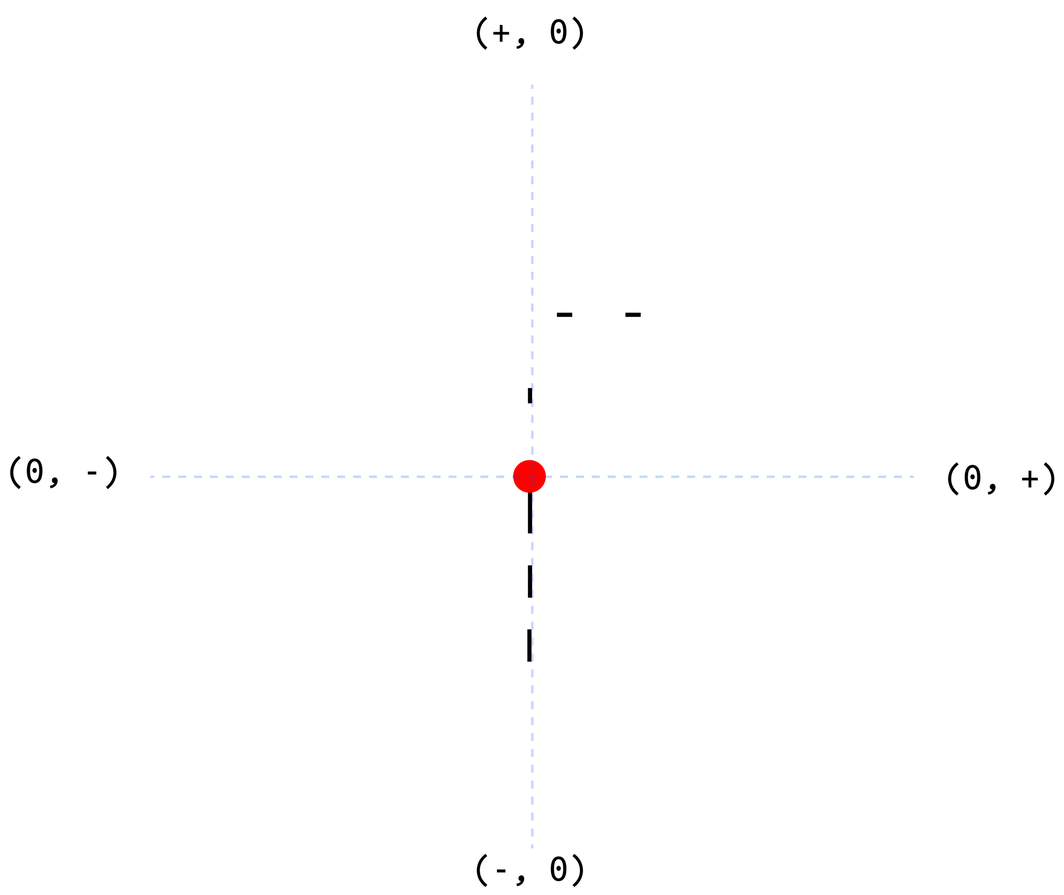
```
message type:

REQUEST -> RESPONSE

WAKE    -> WAKE      (Connection)
PUSH    -> RECEIVE   (Data transmition)
END     -> END       (Route finished)
CLOSE   -> CLOSE     (Force close connection)
```

2.

Algoritmo de dirección de drones

Información que tenemos

- GPS Actual del drone
- Ruta del drone (Mapa de coordenadas, Array)
- Altitud del drone
- Orientación del drone



Minimo de 1/4 segundo para dar aviso de giro

Ejemplo:

GPS Drone: (39.482527, -0.346545)
Orientación: NE (45°)
Siguiente Punto: (39.482499, -0.346507)

Diferencia: SiguientePunto - Actual
(-0.000028, 0.000038)

arctan(-0.000028/0.000038) = -36°

Grado - Orientación = Grado giro
Negativo = Derecha
Positivo = Izquierda

Resultado: -81° = 81 Grados a la derecha

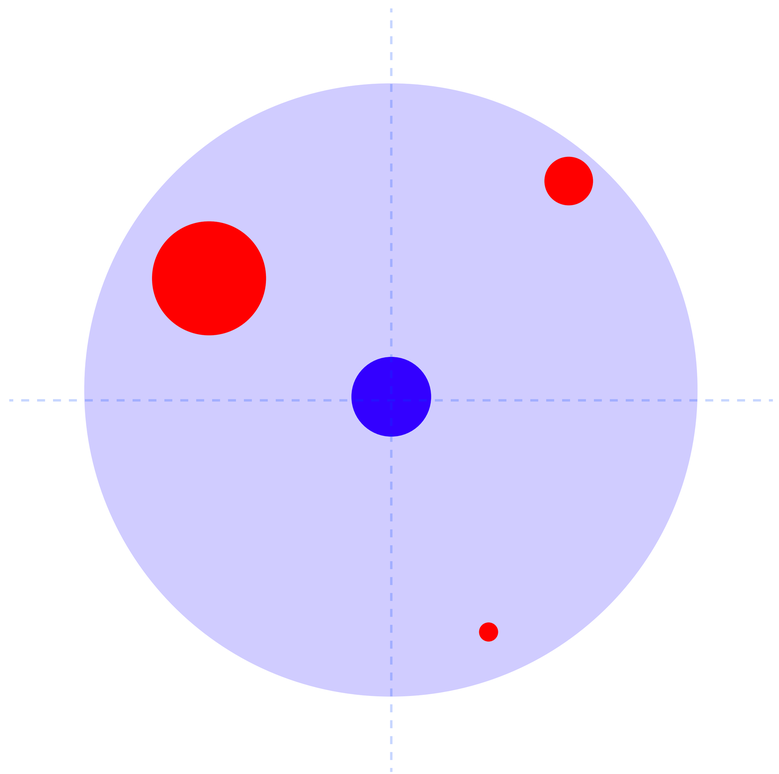
Interacción con DDBB

- Guardar metadatos
- Obtener coordenadas
- Obtener alarma

Extra info

- Cada socket atiende un drone hasta fin de route

Algoritmo de control de choque

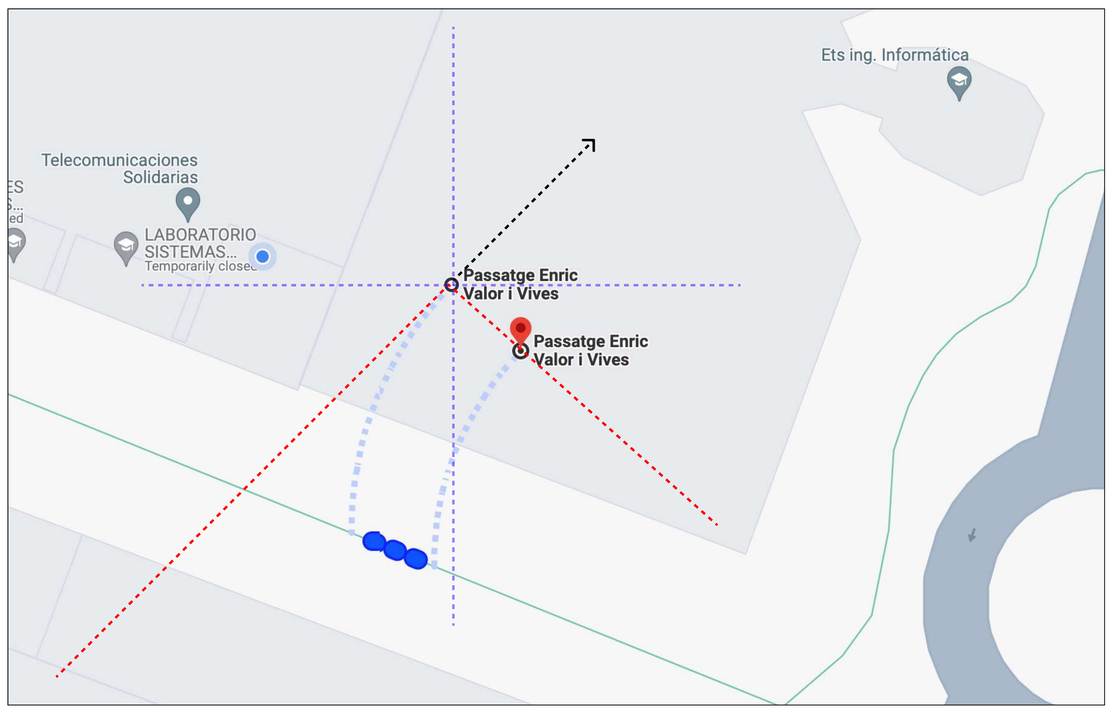


Se debería calcular en un radio de 100 metros los drones que están dentro. Acciones:

* Si el drone es el más prioritario en el area
  Mantener velocidad

* Si el drone es el menos prioritario en el area
  Bajar la velocidad hasta nuevo calculo

* Si el drone es entre varias prioridades
  Reducir la velocidad respecto a la escala de prioridad



Paradoja de las alturas y prioridades

Urgencias y predeterminado

En caso de error en vez de decidir por voluntad propia. Se debe enviar un mensaje de respuesta de tipo ERROR con action de STOP TEMPORAL

Si se obtiene alarma, se pide al drone hacer escaneo de perifericos para confirmar

$$cos(\alpha) = \frac{b}{h}$$

$$sin(\alpha) = \frac{a}{h}$$

3.

Algoritmo de aviso

El servidor recibe una imagen cada x tiempo (live-stream), el servidor a mandar la define y lo comunica el manager.

Modelo pre-entrenada

Código de ejemplo a seguir

Interacción con DDBB

- Guardar la alarm (true, false), alarm_code y alarm_data

```
{
    evidence_path: "https://holamundo.es/data/10jajan18ahu1.jpg",
    expire_datetime: "2024-03-08 21:00:00"
}
```

window: 0 - lang: js

```
//!JS

// Core system message response example

{
    status: 0,
    message: "Route is very dangerous to be used"
}


{
    status: 1,
    dangerous_level: 3,
    message: "Route registered"
}
```