

# Projet Météo

IOT - Gilles Ménez

Université de Nice  
Sophia Antipolis - MIAGE

**Groupe 13**

Présenté par : Alaedine Karouia - Valentin Bordy -  
Yohann Laurendeau - Guilhem Fabre--Sauterey -  
Mathieu Birger

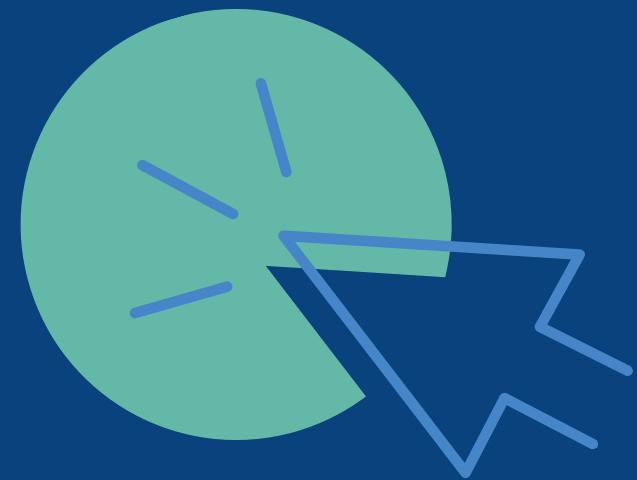


**MIAGE** RÉSEAU  
DES MIAGE  
DE FRANCE  
Nice

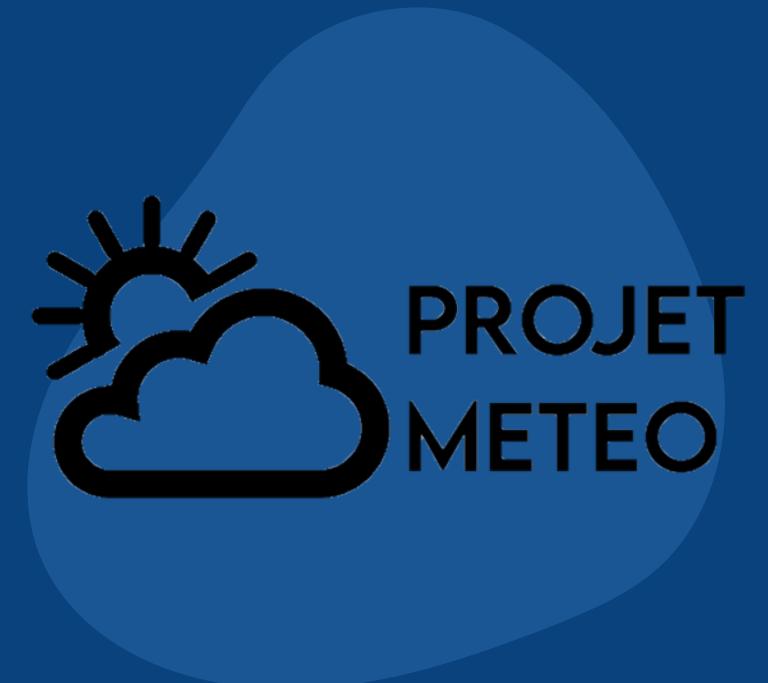
# SOMMAIRE



- Introduction
- Présentation du projet
- Présentation de l'équipe
- Démo
- Architecture
- Usability de l'application
- Intégration de données
- ESP32
- Sécurité
- Déploiement
- Améliorations possibles
- Conclusion

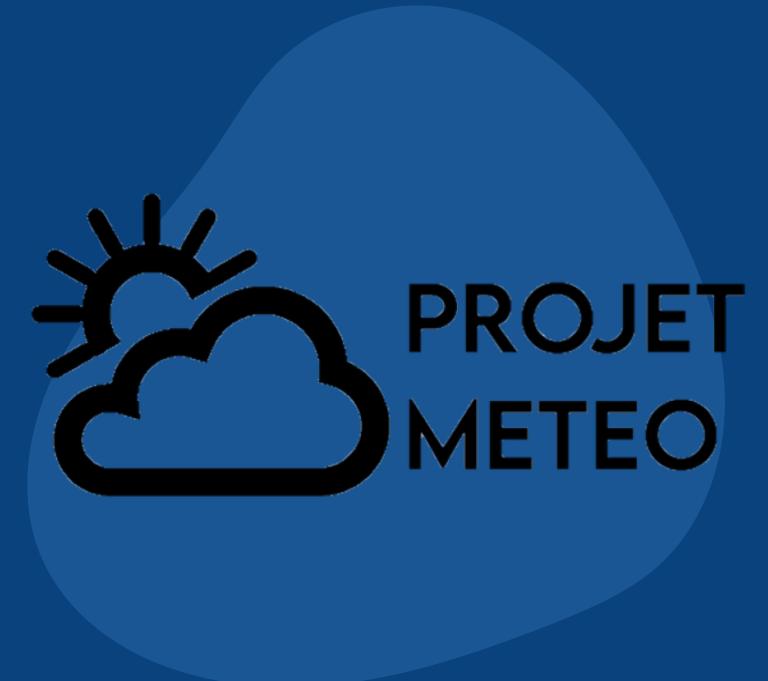


# Introduction





# Présentation du Projet



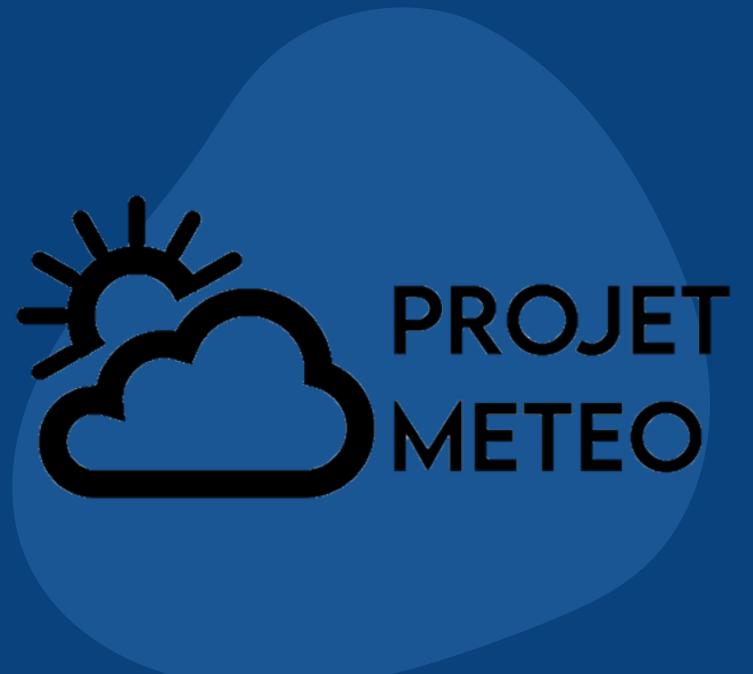
# Projet Meteo

- Crédation de station météo connecté
- Récupération des données de la station
- Crédation d'une application web
- Liaison des ESP et des utilisateurs



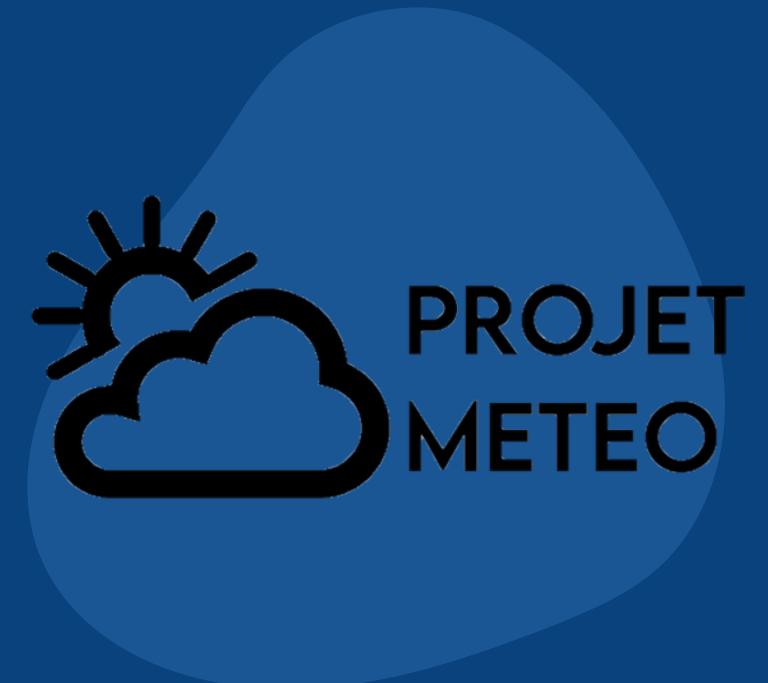


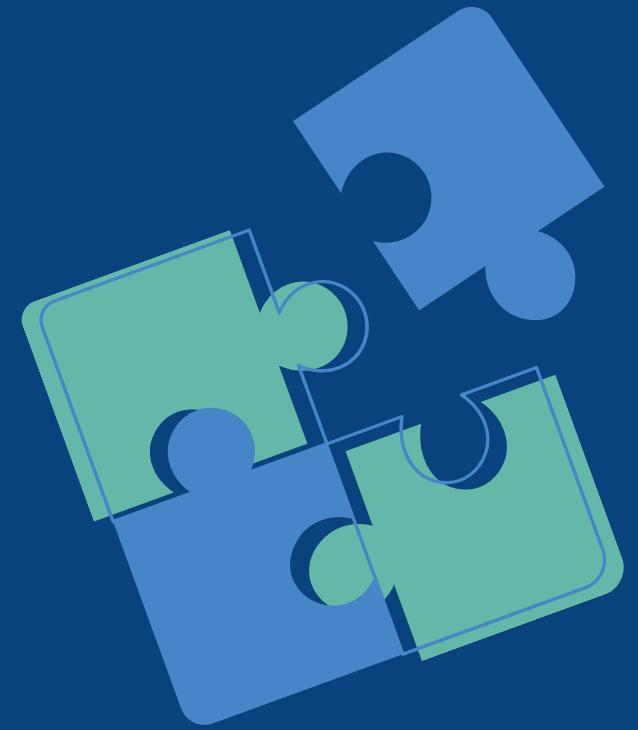
# Présentation de l'équipe



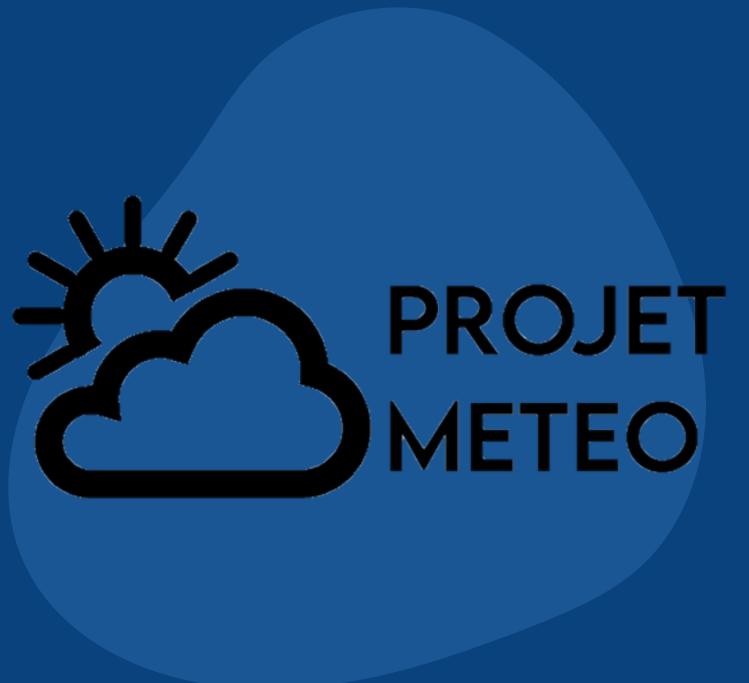


# Démo

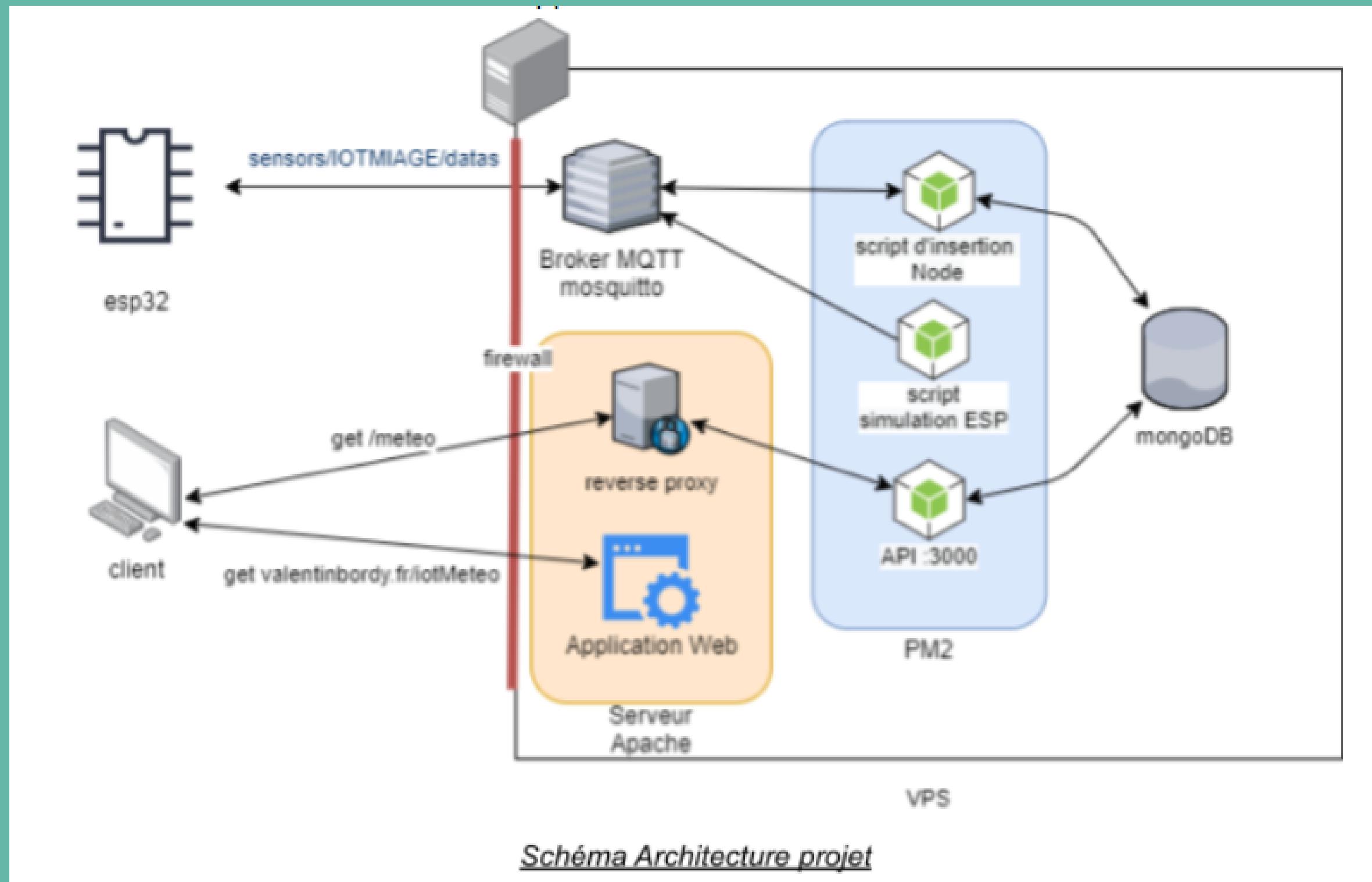




# Architecture



# Schéma architecture projet

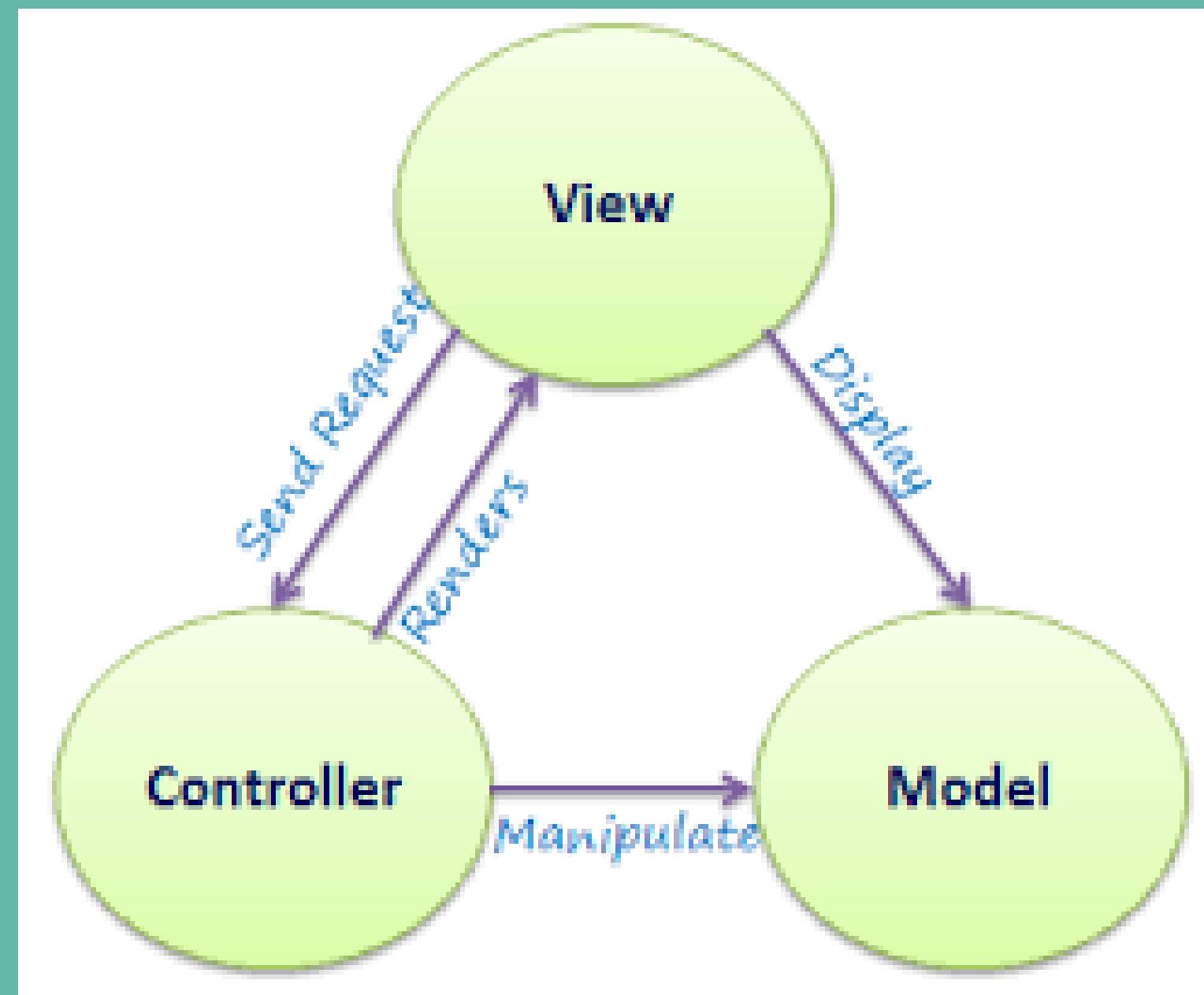


# Notre projet est séparé en 3 parties

- Partie Back-end avec MQTT MONGO
- Partie Front-end
- Partie ESP32



# Utilisation du MVC



# Back-end

Les différentes technologies utilisés :



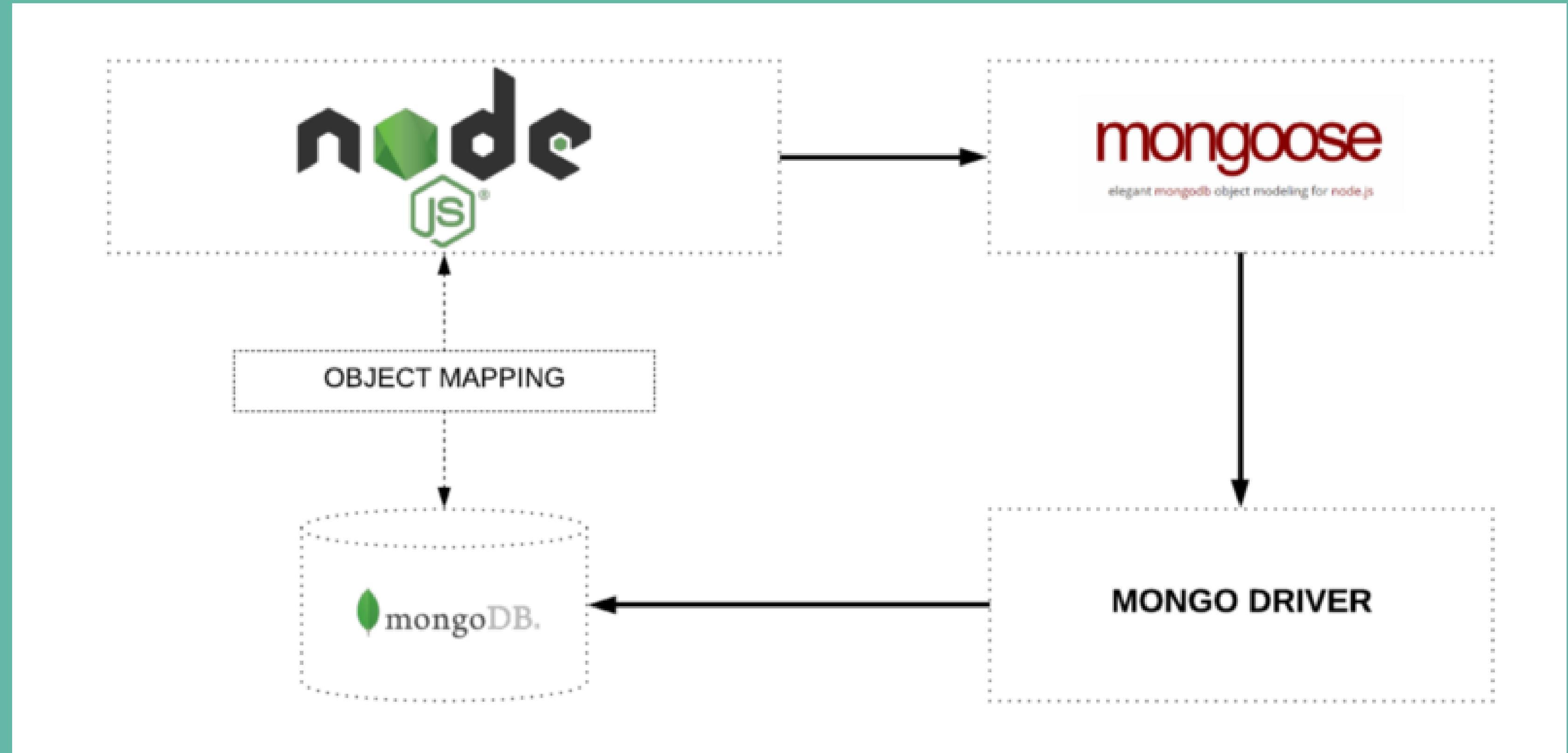
Mongoose



mongoDB



# Schéma des relations de la BDD



# Lancement du serveur

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const port = process.env.PORT || 3000;
const https = require('https');
const mongoose = require('mongoose');
const DB_URI = require('./config/dbconfig');
const meteoRoutes = require('./routes/meteo');
const userRoutes = require('./routes/user');
const espRoutes = require('./routes/esp');
```

```
// DB connection
mongoose.connect(DB_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true
}, (err) => {
  if(err) console.log(err)
  else console.log('Connected to the database')
})
```

```
const options = {
  key: fs.readFileSync('./key.pem'),
  cert: fs.readFileSync('./cert.pem'),
  passphrase: [REDACTED]
};

https.createServer(options, app).listen(port);
```



# Exemple de modeles:

```
const mongoose = require('mongoose');

//structure de donnee d'une adresse Esp
const adresseEspSchema = mongoose.Schema({
    lng: { type: Number, required: true },
    lat: { type: Number, required: true },
});

//structure de donnee d'un esp
const EspSchema = mongoose.Schema({
    adresseMac: { type: String, required: true },
    adresse: {
        type: adresseEspSchema, required: true
    },
    userId: { type: String, required: true }
});
```



```
//structure de donnee d'une adresse
const adresseSchema = mongoose.Schema({
    lat: Number,
    lng: Number
})

//structure de donnee d'une donnee meteo
const MeteoSchema = mongoose.Schema({
    temperature: {
        type: String,
        required: true
    },
    id: {
        type: String,
        required: true
    },
    pression: {
        type: String,
        required: true
    },
    altitude: {
        type: String,
        required: true
    },
    lumiere: {
        type: String,
        required: true
    },
    humidite: {type: String, required: true}
    idUser: {
        type: String,
        required: true},
    adresse: adresseSchema,
});
```



# PROJET METEO

# Lien/clé vers Base de données/Api externe

```
//environnement de dev  
//const uri = 'mongodb+srv://XXXXXXXXXX@cluster0.yWritestrue&w=majority';  
//environnement de prod  
const uri = 'mongodb://localhost:27017/METEO';  
module.exports = uri;
```

```
//fichier de config pour l'api openWeatherMap  
const openWeather_url = "https://api.openweathermap.org/data/2.5/";  
const openWeatherKey = XXXXXXXXXX  
  
module.exports = {  
  openWeather_url,  
  openWeatherKey  
};
```

```
const weatherBit_url = "https://api.weatherbit.io/v2.0/current";  
const weatherBitKey = XXXXXXXXXX,  
  
module.exports = {  
  weatherBit_url,  
  weatherBitKey  
};
```

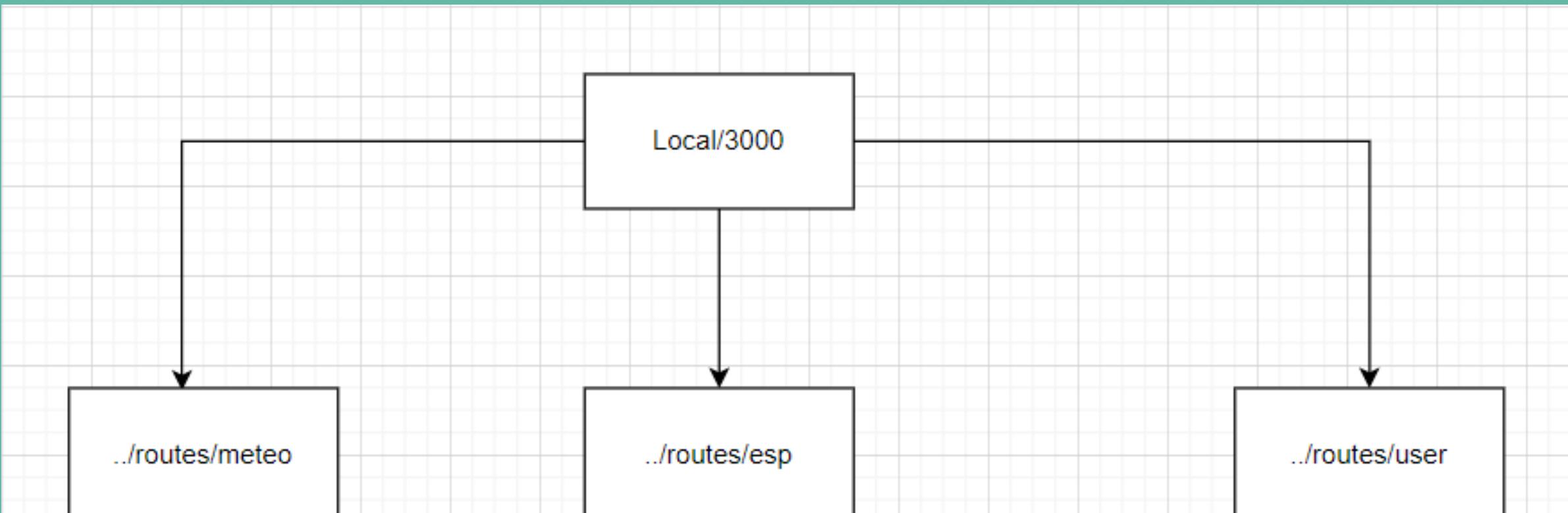


# Contrôleurs

```
const EspModel = require('../models/esp.model');
```

```
const deleteEsp = async (req, res) => { //Requête qui permet de supprimer un esp grace a id
    const id = req.params.id;
    await EspModel.deleteOne({_id: id})
        .then(rslt => {
            res.status(200).send({message: "Great Success ! ESP has been deleted !"})
        })
        .catch(err => {
            res.status(400).send({message: err.message});
        })
}
```

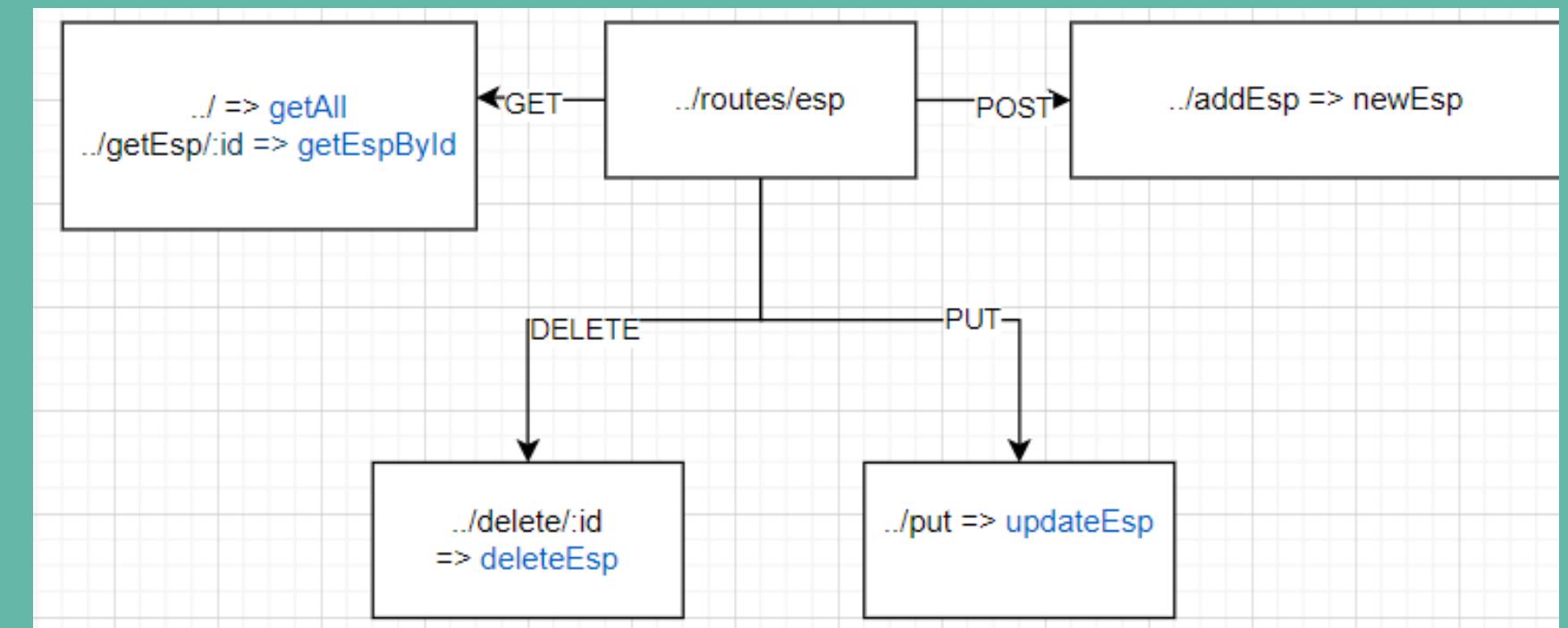
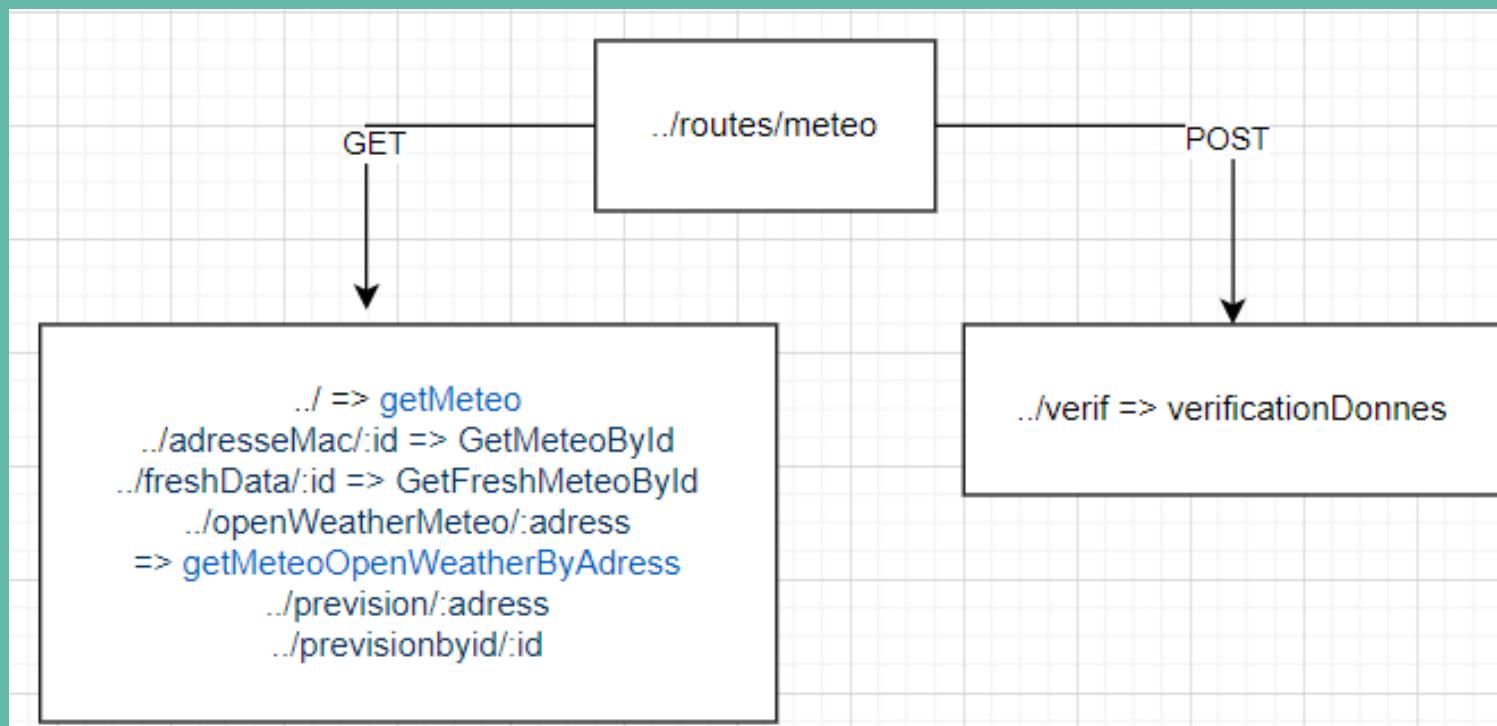
# Routes



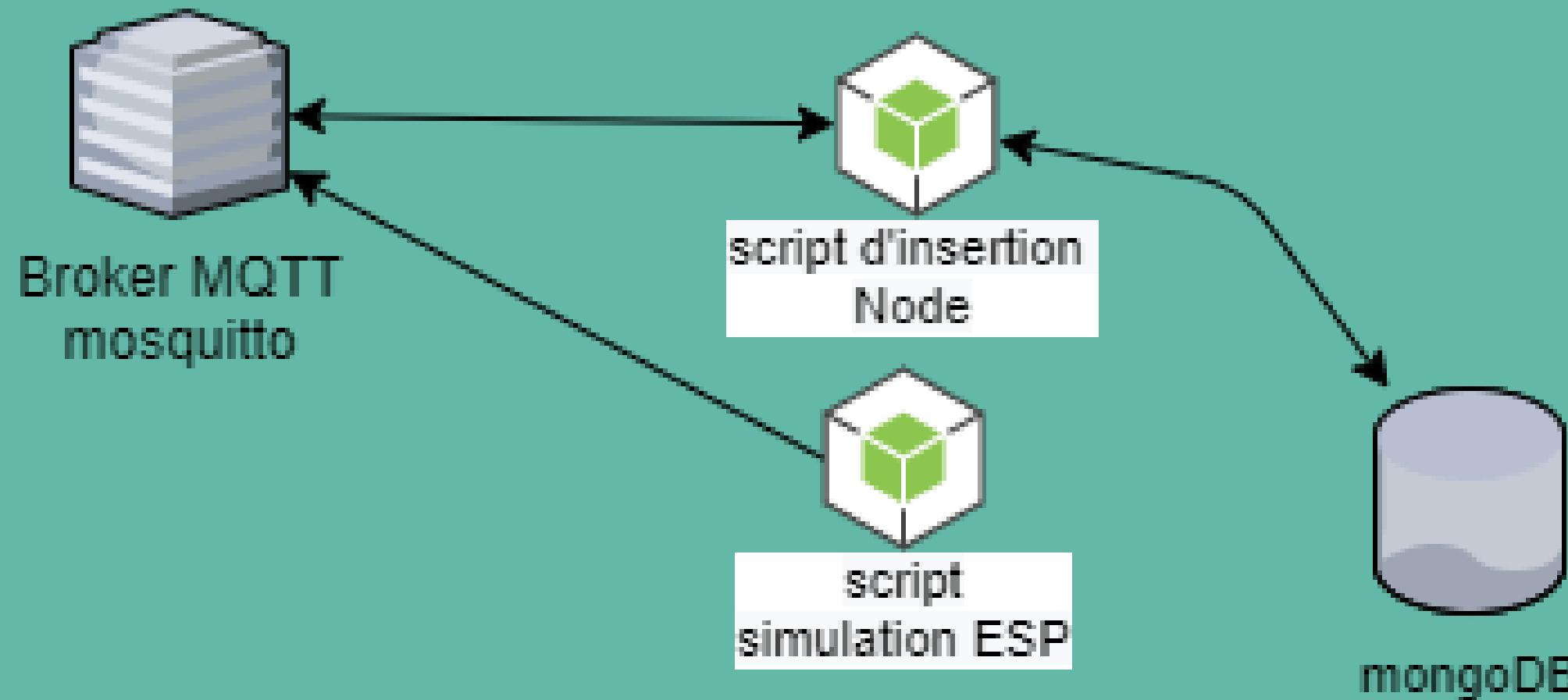
```
app.use('/meteo', meteoRoutes);
app.use('/user', userRoutes);
app.use('/esp', espRoutes);
```

# Routes

```
//GET  
router.get('/', espController.getAll);  
router.get('/getEsp/:id', token, espController.getEspById);  
  
//POST  
router.post('/addEsp', token, espController.newEsp); // créer un nouvel utilisateur (voir le model pour le body)
```



# Jointure mqtt mongo



# Front-end



Librairies utilisées :



# Structure d'un component

```
<template lang="jade">
  div
    p {{ greeting }} World!
    other-component
  </template>

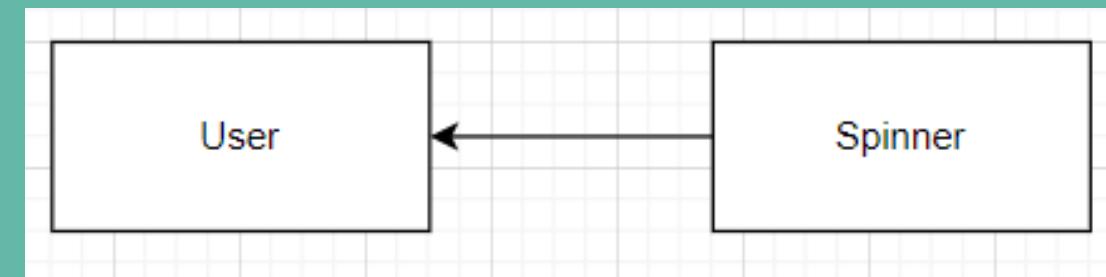
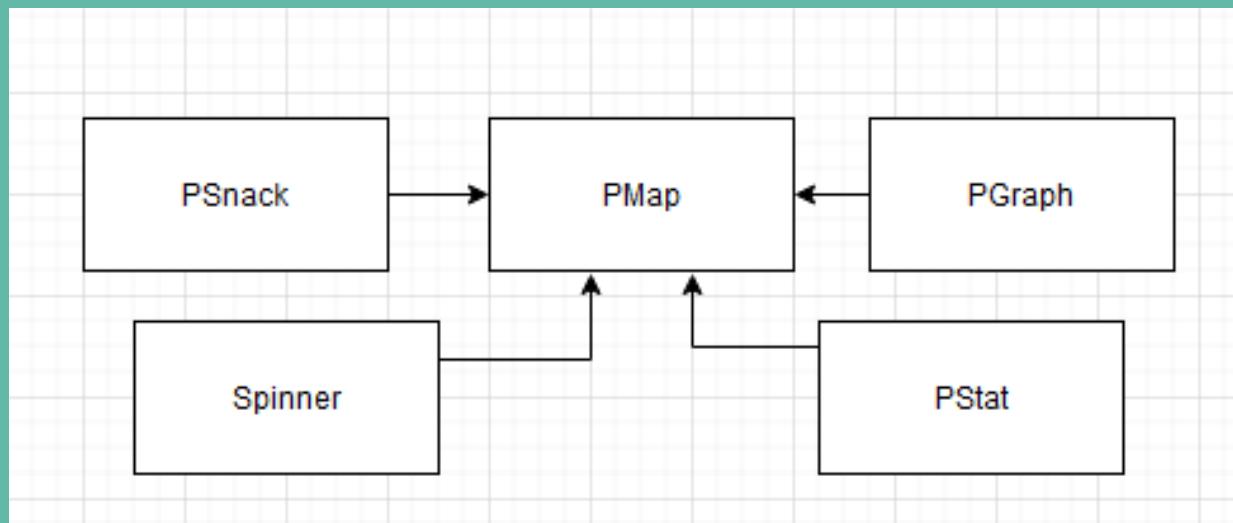
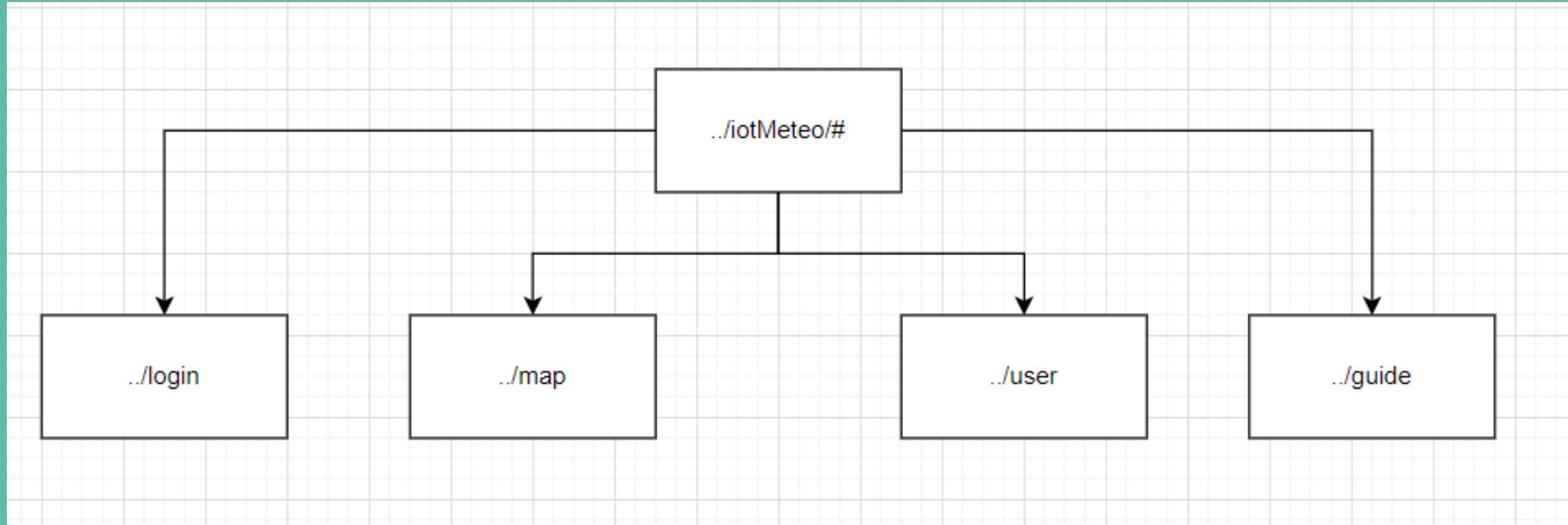
<script>
  import OtherComponent from './OtherComponent.vue'

  export default {
    data () {
      return {
        greeting: 'Hello'
      }
    },
    components: {
      OtherComponent
    }
  }
</script>

<style lang="stylus" scoped>
  p
    font-size 2em
    text-align center
</style>
```

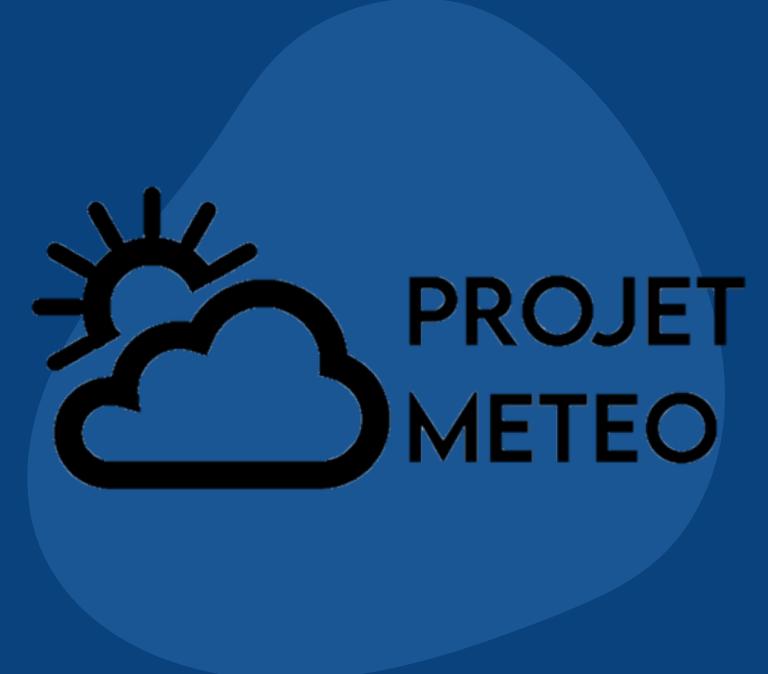


# Router

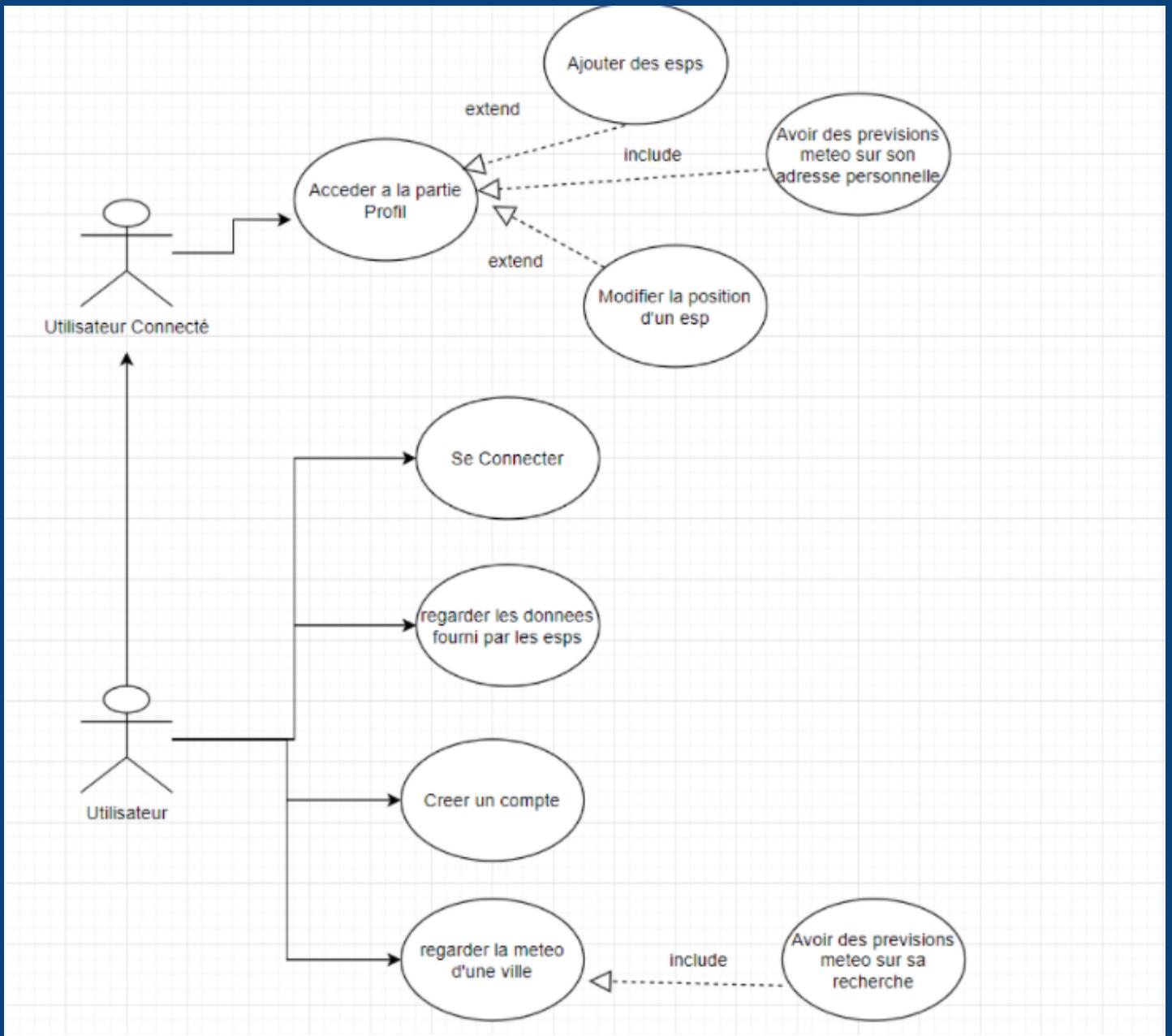




# Usability de l'application



# Use Case



# Accueil

SE CONNECTER



S'ENREGISTRER



INVITÉ



E-mail

---

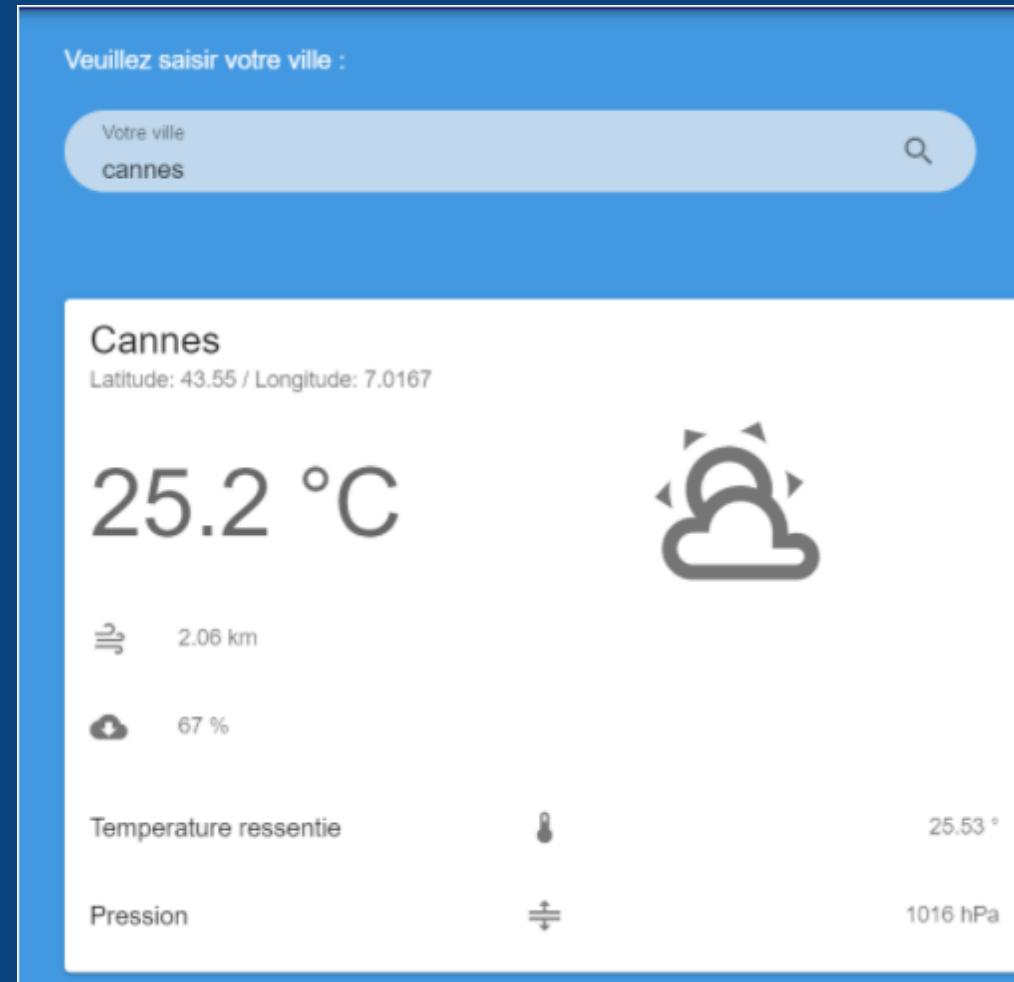
Mot de passe

---

0

CONNEXION

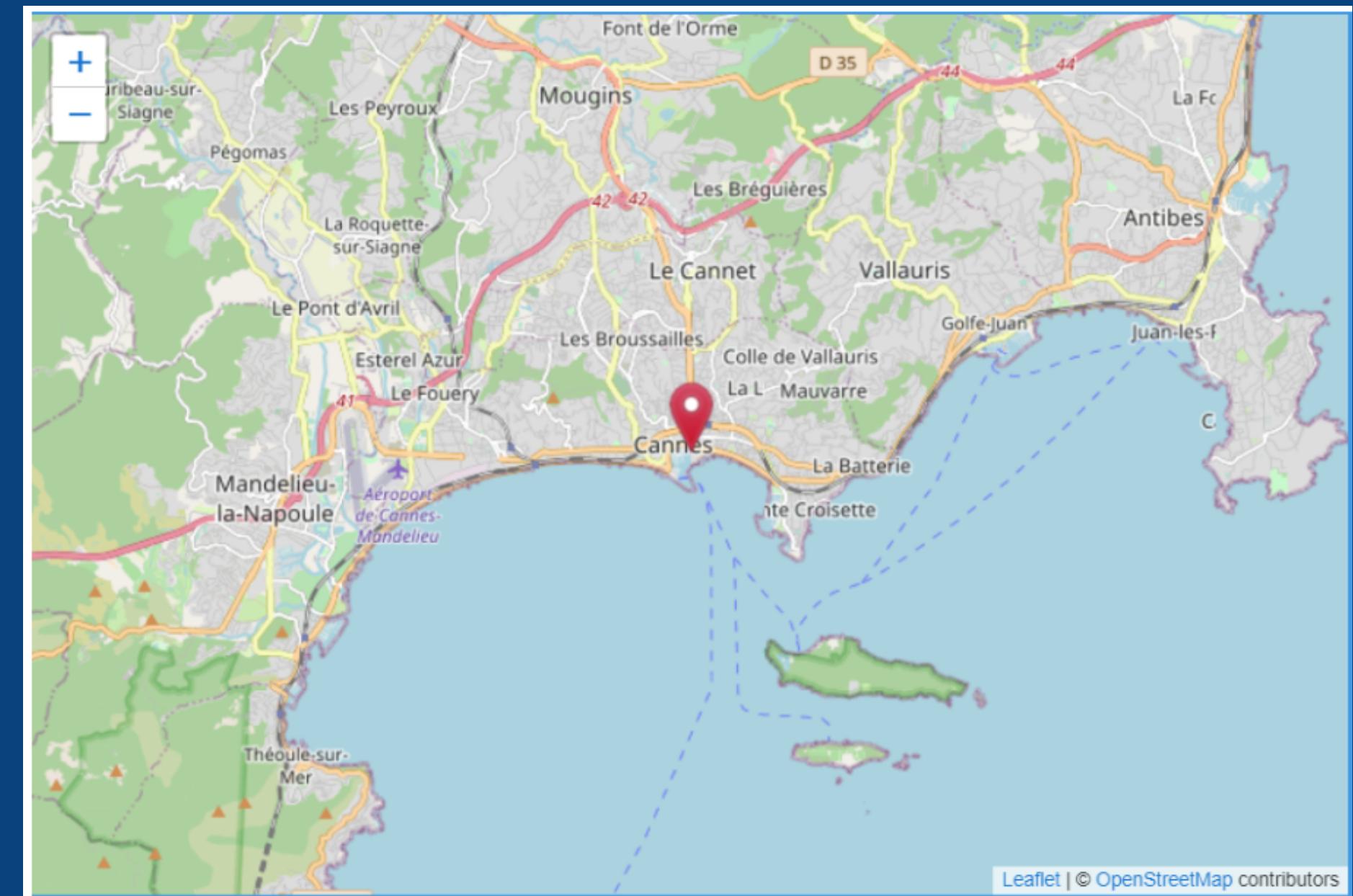
# Cas d'utilisation: Utilisateur invité



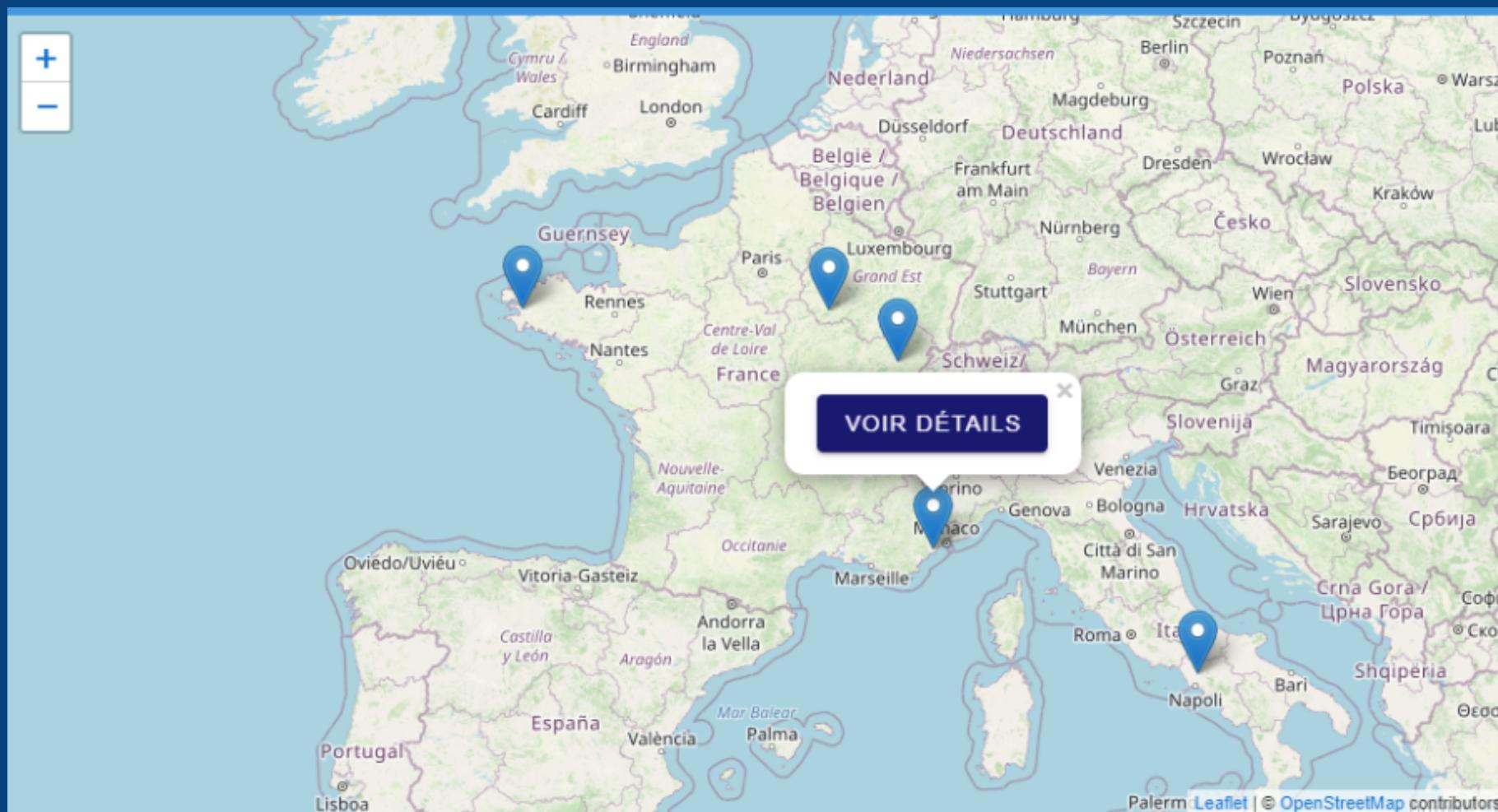
VOIR LES PRÉVISIONS

La météo aujourd'hui sera pluvieux  
La météo dans 1 jour sera nuageux  
La météo dans 2 jours sera pluvieux  
La météo dans 3 jours sera nuageux  
La météo dans 4 jours sera nuageux

CLOSE



# Donnees des ESP

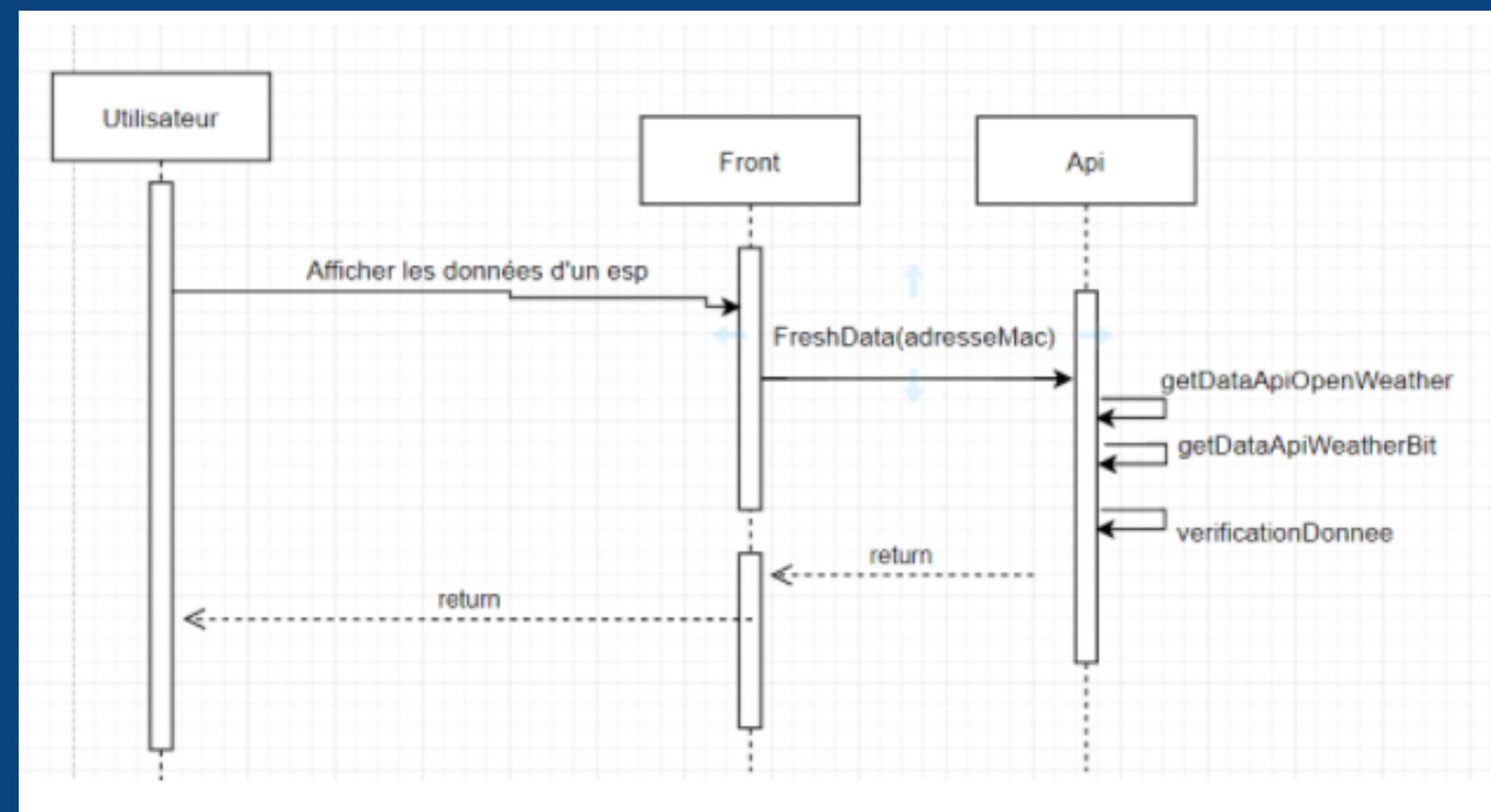


## Historique des 4 derniers jours :

Type/Date	2021-06-08	2021-06-07	2021-06-06	2021-06-05
-----------	------------	------------	------------	------------

humidité	!	!	!	54.07
----------	---	---	---	-------

# Verification des donnees



# Recuperation des données de traitement

```
//permet de récupérer les dernières datas de l'esp
const getFreshMeteoById = async (req, res) => {
    //maximum de data par requête
    const maxData = 7000;
    const addMac = req.params.id;
    const today = new Date();
    today.setDate(today.getDate() - 14);

    //.find permet de chercher dans le MeteoModel
    //.limit permet de limiter le nombre de données à récup
    //.sort permet de trier par date -1 signifie qu'on récup les données depuis la dernière insérée
    await MeteoModel.find({id: addMac, date: {$gte: today.toLocaleString("sv-SE", {timeZone: "Europe/Paris"})}}).limit(maxData).sort( { date: -1 } )
        .then(rslt => {
            //reverse permet d'inverser le tableau
            rslt.reverse();
            rslt.length ? res.status(200).json(rslt) : res.status(200).json({erreur: "ESP inconnu ...."})
        })
        .catch(err => {
            res.status(400).send({message: err.message});
        })
}
```

# Critere d'acceptance

```
function testHumidity(espH1, h2) {//Permet de comparer les données d'humidité des api et celle de l'esp
    //console.log('humid : ',espH1, h2)
    return (espH1 <= h2 + 10) && (espH1 >= h2 - 10);
}

function testTemp(espT1, t2) {//Permet de comparer les données de la température de l'api et des esp
    //console.log('temp : ',espT1, t2)
    return (espT1 <= t2 + 2.5) && (espT1 >= t2 - 2.5);
}

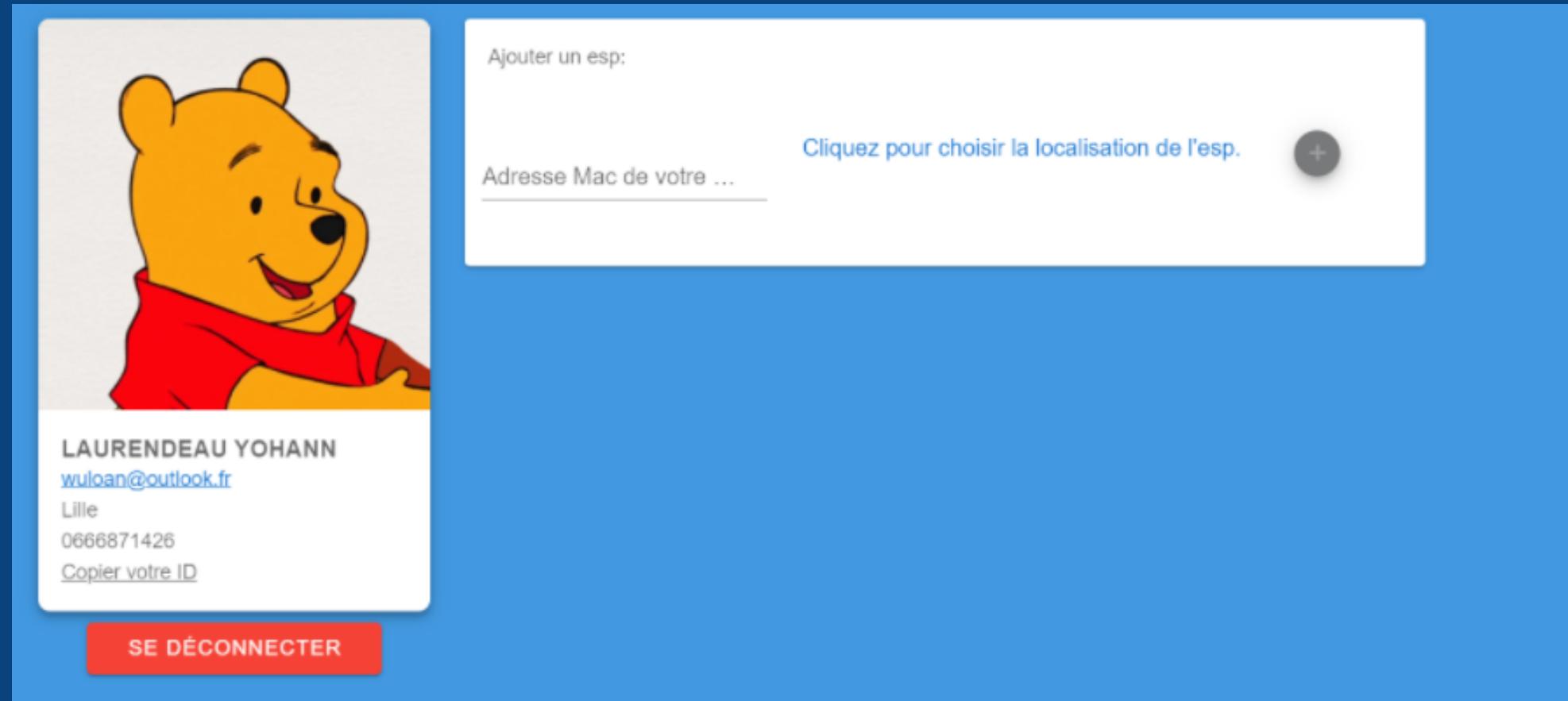
function testPression(espP1, p2) {//Permet de comparer les données de pression de l'api et des esp
    //console.log('pression: ',espP1, p2)
    return (espP1 <= p2 + 5) && (espP1 >= p2 - 5);
}
```

# Validation des données

```
let objetValid;
try {

    objetValid = {
        temperature: testTemp(esp.temperature, dataApi1.data[0].temp) || testTemp(esp.temperature, dataApi2.main.temp),
        pression: testPression(esp.pression, dataApi1.data[0].pres) || testPression(esp.temperature, dataApi2.main.pressure),
        humidite: testHumidity(esp.humidite, dataApi1.data[0].rh) || testHumidity(esp.humidite, dataApi2.main.humidity)
    }
} catch (e) {
    try {
        objetValid = {
            temperature: testTemp(esp.temperature, dataApi2.main.temp),
            pression: testPression(esp.temperature, dataApi2.main.pressure),
            humidite: testHumidity(esp.humidite, dataApi2.main.humidity),
        }
    } catch (e) {
        let message = "Vérification indisponible pour le moment. La limite de vérification a peut-être été atteinte.";
        res.status(400).send(message);
    }
}
```

# Cas d'utilisation: Utilisateur Connecté:



Cliquez sur la carte pour ajouter la localisation de votre esp :



```
adressMacRules: [
  (v) => !!v || "l'adresse mac est requises",
  (v) =>
    /^[0-9A-Fa-f]{2}[:]{5}([0-9A-Fa-f]{2})$/.test(v) ||
    "L'adresse mac n'est pas valide.", //permet de tester avec un regex si l'adresse mac a un bon format
],
```



# Intégration des données

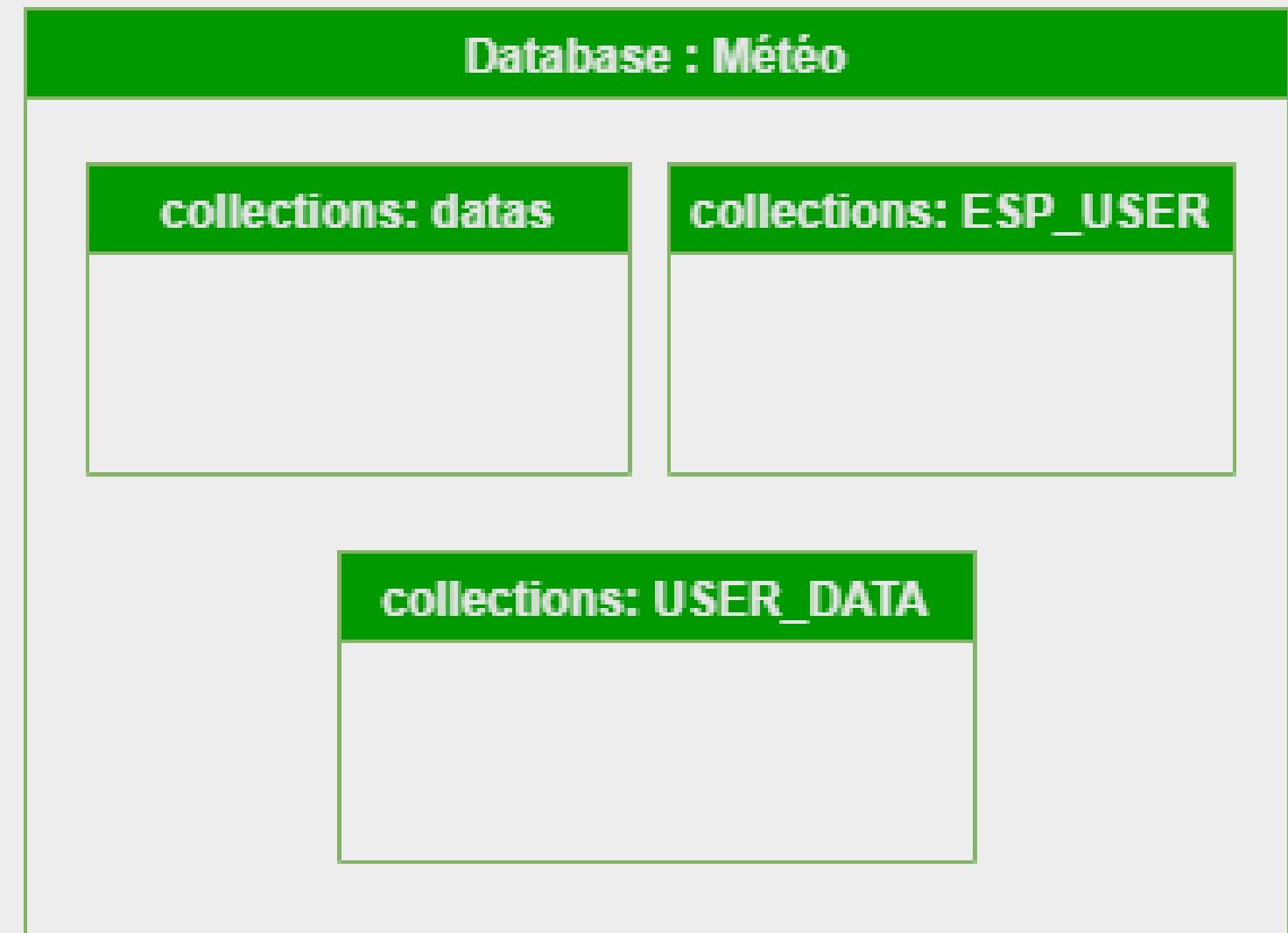


# MongoDB

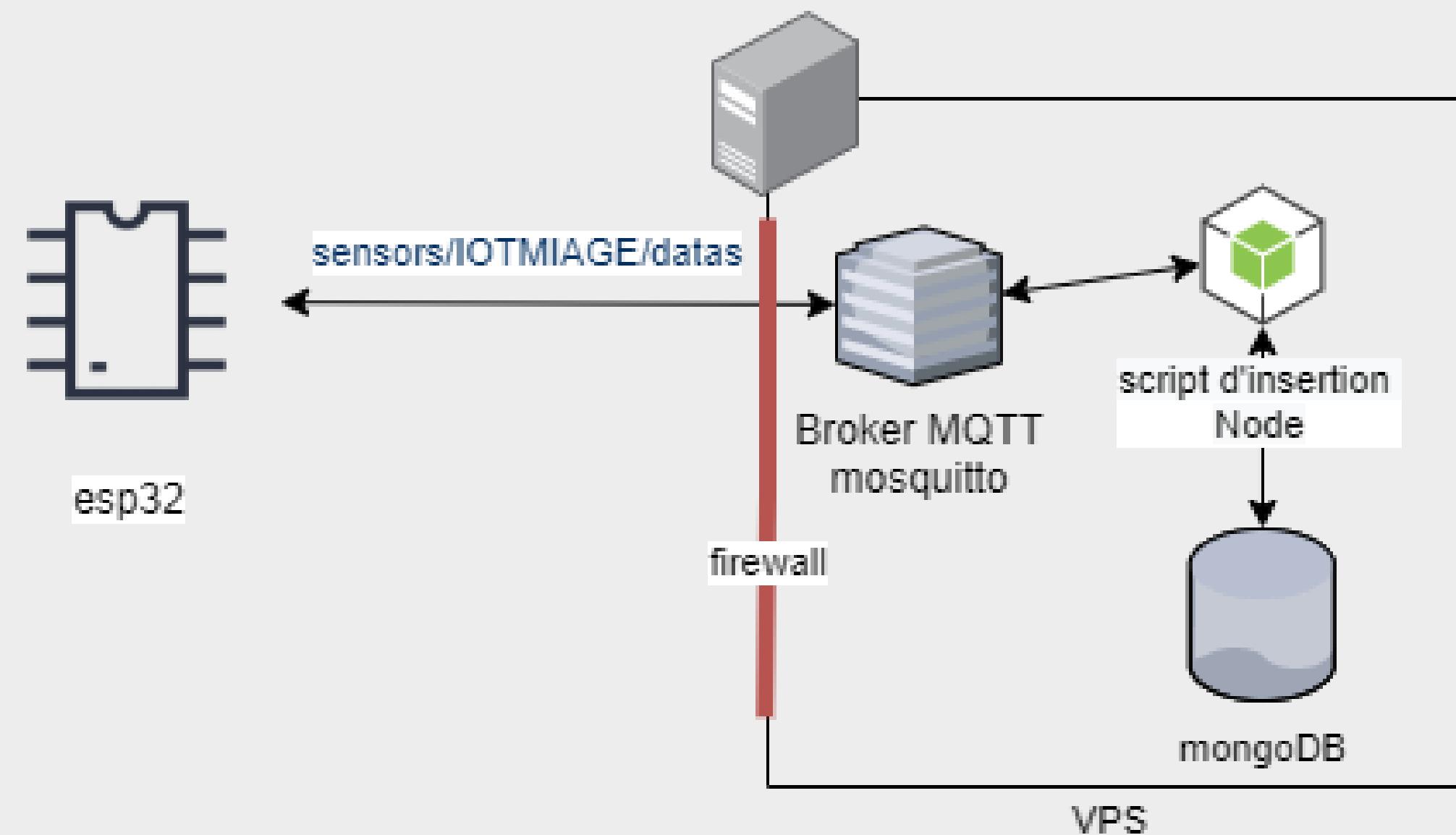
- NoSQL
- orienté Document
- Format BSON (JSON binaire)



mongoDB



# Schéma chemin des données de l'esp



# Validateur MongoDB : collection ESP\_USER

-adresseMac: string  
-userId: string  
-adresse: object  
{-lat: number,-lng: number}

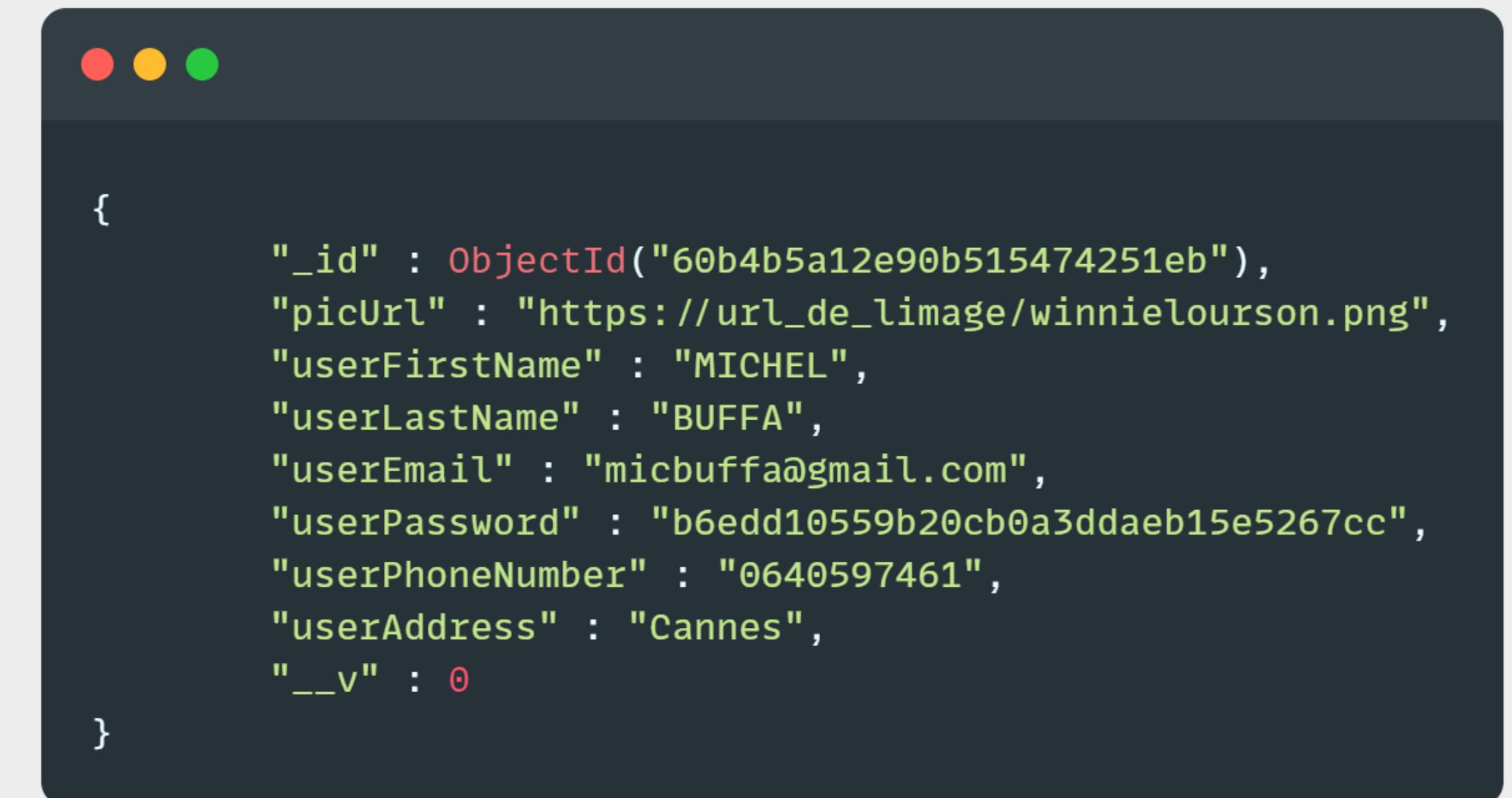


A screenshot of a dark-themed terminal window. At the top, there are three small colored circles (red, yellow, green) representing window control buttons. The main area contains a JSON document:

```
{
  "_id" : ObjectId("60b9e28b8f8f119a3da94178"),
  "adresseMac" : "25:6F:68:69:C9:14",
  "adresse" : {
    "_id" : ObjectId("10b9e28c8f8f119a3da94179"),
    "lat" : 43.65707029201576,
    "lng" : 7.135788121548403
  },
  "userId" : "60b2aa582ea85f6b509d5435",
  "__v" : 0
}
```

# Validateur MongoDB : collection USER\_DATA

- userFirstName: string
- userLastName: string
- userEmail: string
- userPassword: string
- userAddress: string

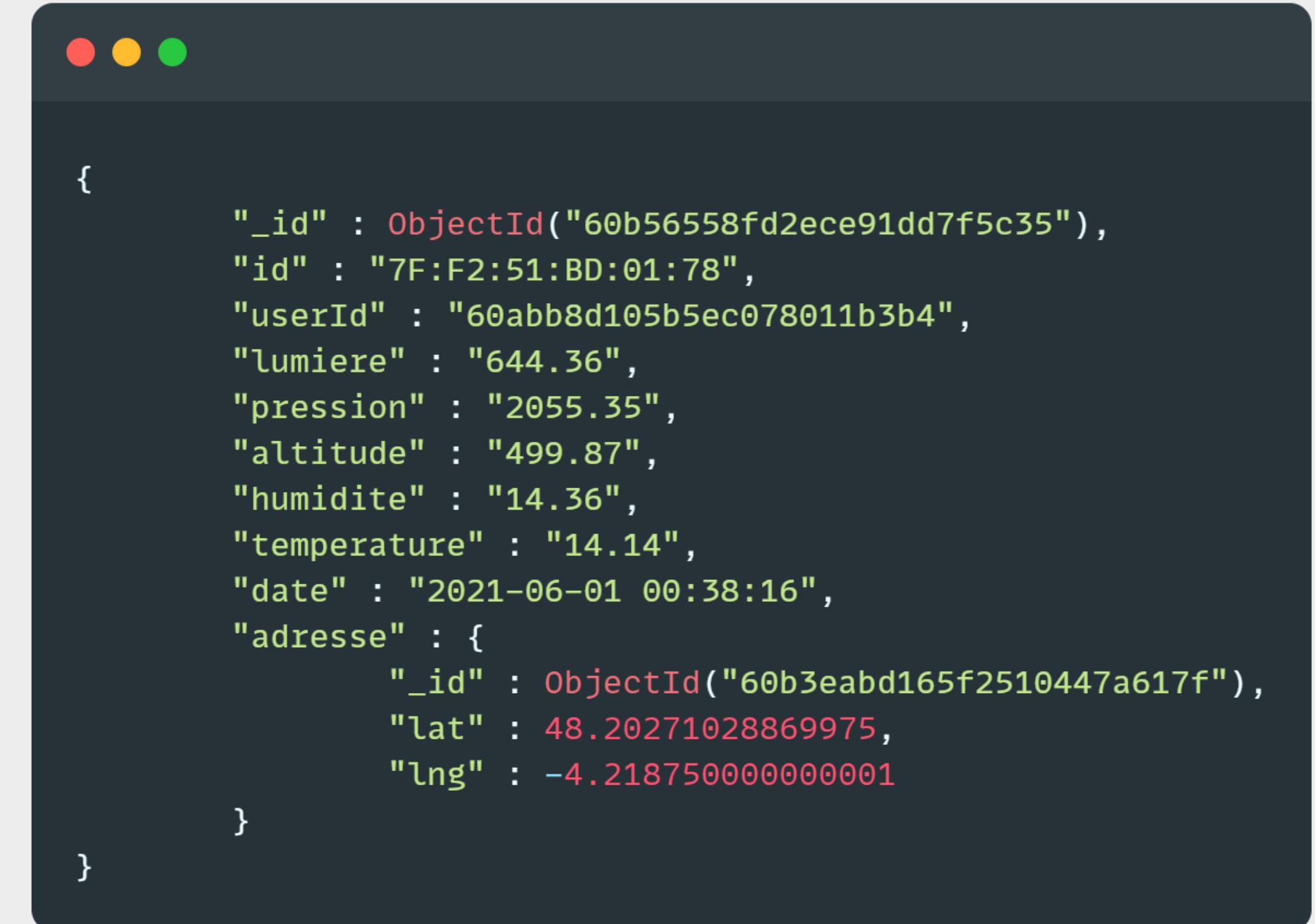


A screenshot of a MongoDB document in a dark-themed interface. The document is represented by a JSON object with curly braces. It contains several fields: '\_id' (an ObjectId), 'picUrl' (a URL), 'userFirstName' (MICHEL), 'userLastName' (BUFFA), 'userEmail' (micbuffa@gmail.com), 'userPassword' (a hashed password), 'userPhoneNumber' (0640597461), 'userAddress' (Cannes), and '\_\_v' (version number 0). The background is dark, and the text is white or light-colored.

```
{  
    "_id" : ObjectId("60b4b5a12e90b515474251eb"),  
    "picUrl" : "https://url_de_limage/winnielourson.png",  
    "userFirstName" : "MICHEL",  
    "userLastName" : "BUFFA",  
    "userEmail" : "micbuffa@gmail.com",  
    "userPassword" : "b6edd10559b20cb0a3ddaebe15e5267cc",  
    "userPhoneNumber" : "0640597461",  
    "userAddress" : "Cannes",  
    "__v" : 0  
}
```

# Validateur MongoDB : collection datas

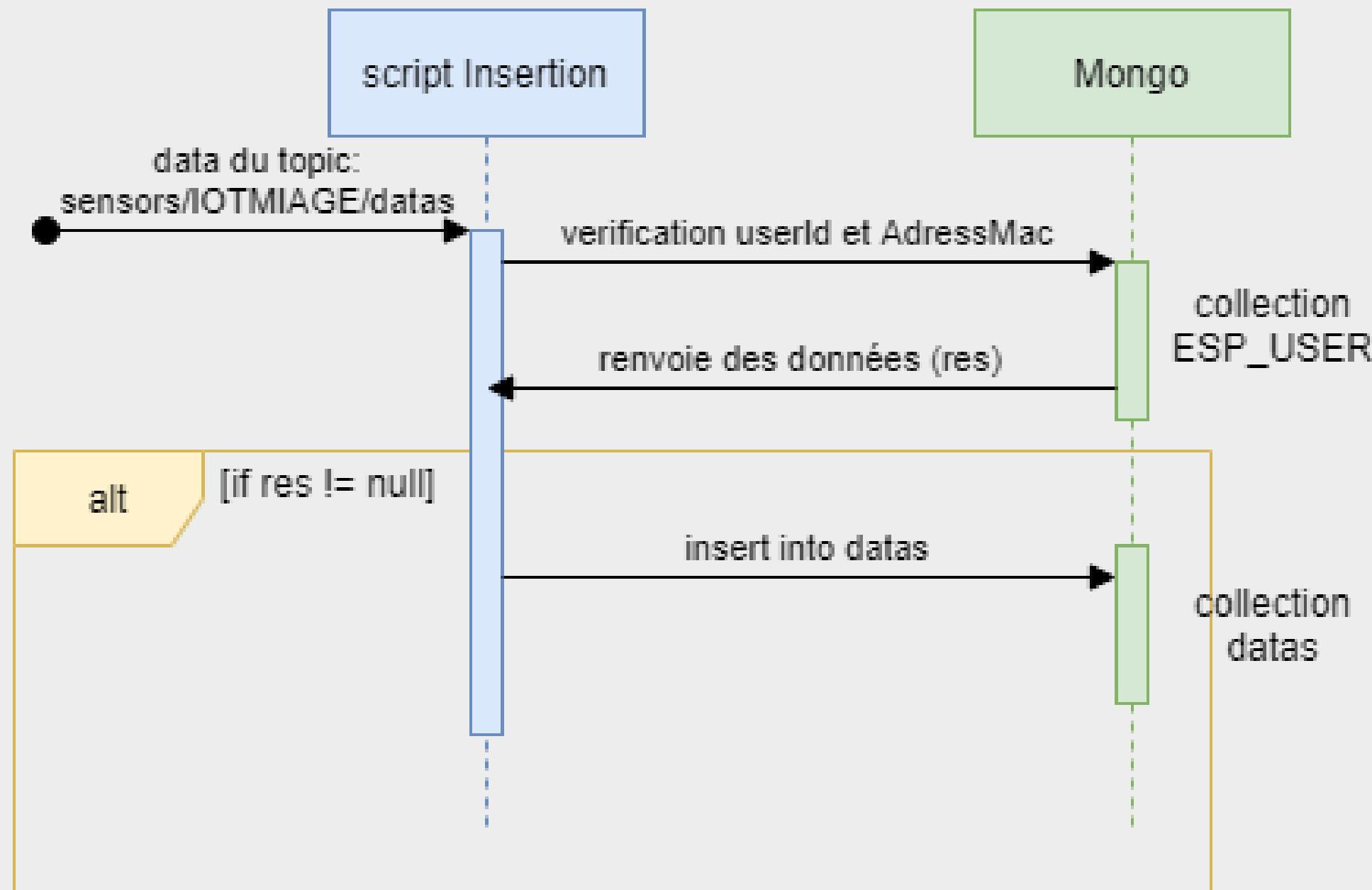
```
-id: string
userId: string
lumiere: string
pression: string
altitude: string
humidite: string
temperature: string
date: string
adresse: object
{-lat: number,-lng: number}
```



A screenshot of a mobile application window showing a JSON document. The document consists of a single object with the following fields and values:

```
{
  "_id" : ObjectId("60b56558fd2ece91dd7f5c35"),
  "id" : "7F:F2:51:BD:01:78",
  "userId" : "60abb8d105b5ec078011b3b4",
  "lumiere" : "644.36",
  "pression" : "2055.35",
  "altitude" : "499.87",
  "humidite" : "14.36",
  "temperature" : "14.14",
  "date" : "2021-06-01 00:38:16",
  "adresse" : {
    "_id" : ObjectId("60b3eabd165f2510447a617f"),
    "lat" : 48.20271028869975,
    "lng" : -4.2187500000000001
  }
}
```

# Script d'insertion des données



# API externes

## OpenWeatherMap

- Current Weather Data
- 5 day weather forecast
- 60 appels/min ou 1M/mois
- compte premium étudiant 6mois



## WeatherBit

- Current Weather Data
- 500 appels/jour



# Script Simulation des données

```
2  var esp = {id:"48:A3:21:5A:BE:17", userid:"60b2aa582ea85f3b509d5435", lumiere:600, pression:1017, alt:200, hum:50, temp:22};
3  var esp2 = {id:"28:30:D1:83:E8:CB", userid:"60b2aa582ea85f3b509d5435", lumiere:750, pression:1117, alt:199, hum:60, temp:20};
4  var esp3 = {id:"04:15:24:81:E2:51", userid:"60b2aa582ea85f3b509d5435", lumiere:200, pression:1057, alt:220, hum:70, temp:21};
5  var esp4 = {id:"7F:F2:21:BD:01:78", userid:"60abb8d105b5ec078018b3b4", lumiere:650, pression:2057, alt:500, hum:20, temp:15};
6  var esps = [esp,esp2,esp3,esp4];
7  var mqtt = require('mqtt');
8  console.log("connection");
9  var client = mqtt.connect('mqtt://152.228.216.110:8883',{clientId:"scriptounet",
10    username:"iot",
11    password:"salutcestleprojetiot"})
12
13  client.on("connect",function(){
14    console.log("connection");
15    onConnect();
16  })
17
18
19  function sleep(ms) {
20    return new Promise(resolve => setTimeout(resolve, ms));
21  }
22  async function onConnect() {
23    console.log("onConnect");
24    while (true)
25    {
26      //temps entre chaque envoie en l'occurrence 10min
27      await sleep(600000);
28      let i = 0;
29      for (var esp in esps) {
30        console.log("envoie");
31        console.log(esps[i]);
32        client.publish("sensors/IOTMIAGE/datas",getDataRandomData(esps[i]),1);
33        i++;
34      }
35    }
36  }
37
```



PROJET  
METEO

```
function getDataRandomData(esp){
    return "{\"id\": \"\" + getID(esp) + "\", \"userId\": \"\" + userID(esp) + "\", \"lumiere\": \"\" + get_light(esp) + "\", \"pression\": \"\" + get_pressure(esp) + "\", \"alt\": \"\" + get_alt(esp) + "\", \"humidity\": \"\" + get_humidity(esp) + "\"}";
}

function getID(esp){
    return (esp.id);
}

function userID(esp){
    return (esp.userid);
}

function get_light(esp){
    let res = Math.random() * (esp.lumiere+6+0.8- esp.lumiere-6) + esp.lumiere-6;

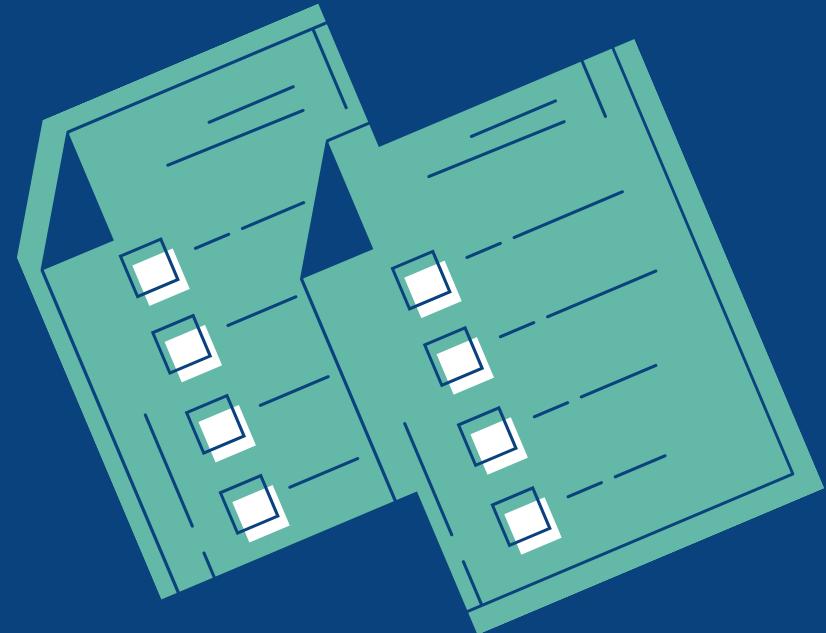
    if(res <0)
        res=0;
    return res.toFixed(2)
}

function get_pressure(esp){
    let res = Math.random() * (esp.pression+2+0.8- esp.pression-2) + esp.pression-2;
    if(res <0)
        res=0;
    return res.toFixed(2)
}

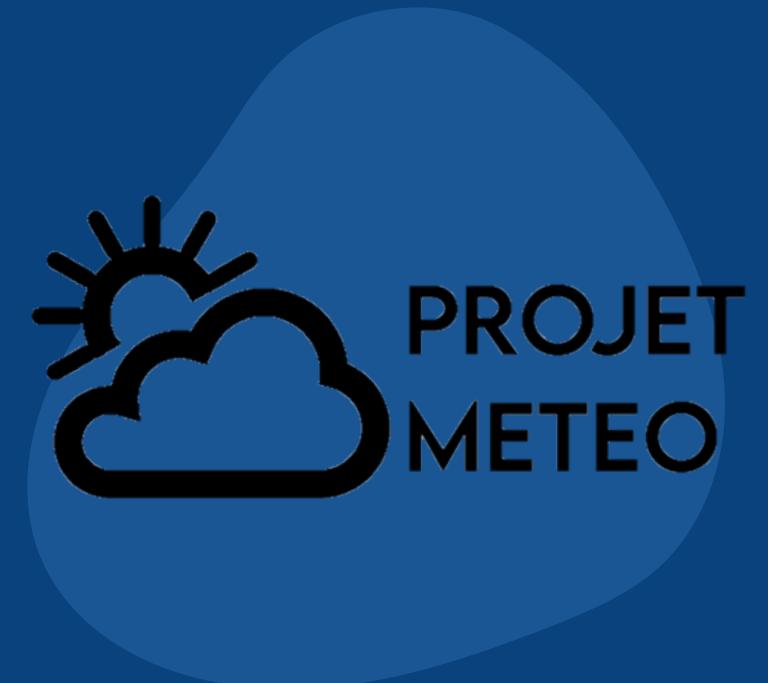
function get_alt(esp){
    let res = Math.random() * (esp.alt+0.2+0.8- esp.alt-0.2) + esp.alt-0.2;

    if(res <0)
        res=0;
    if(res >1000)
        res=1000;
    return res.toFixed(2)
}

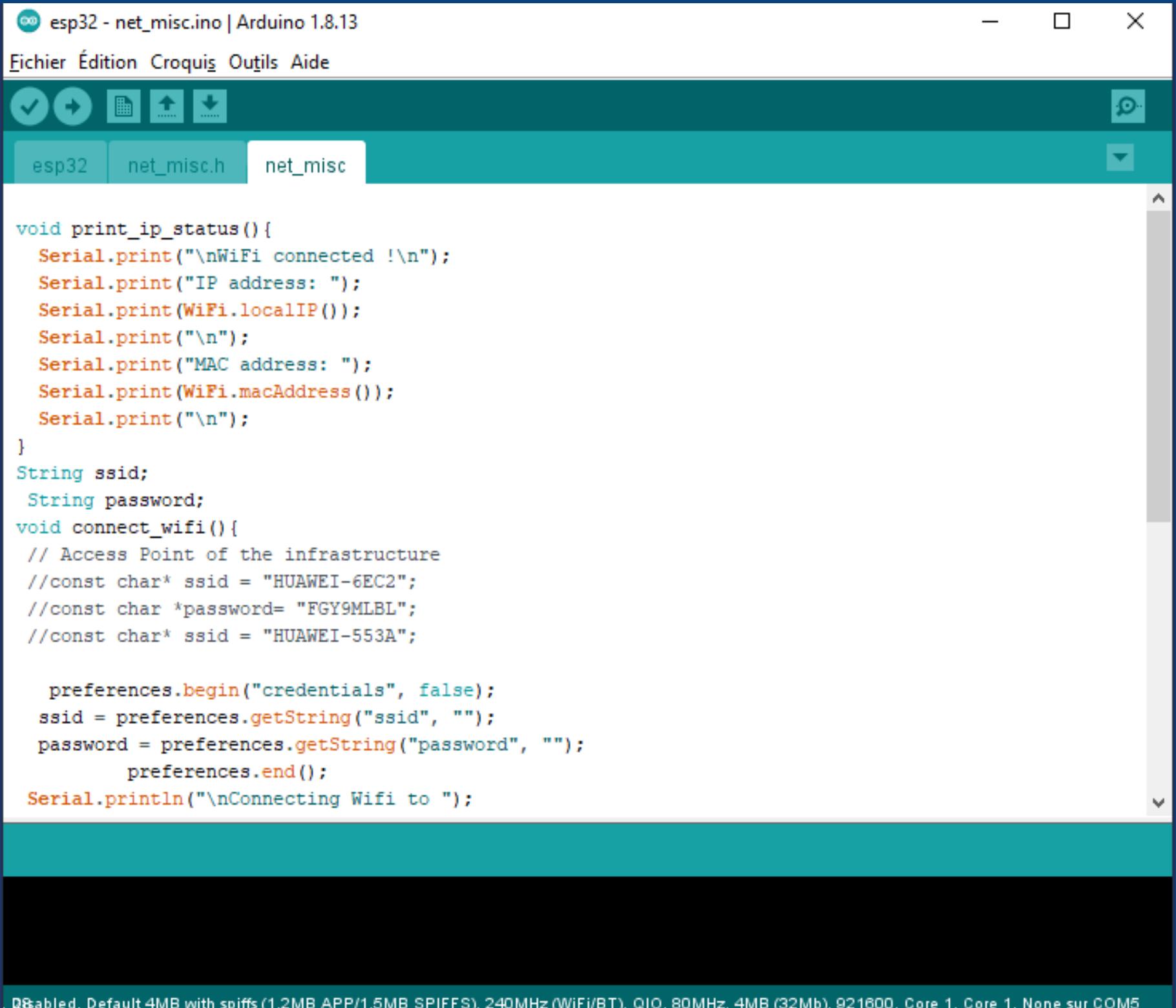
function get_humidity(esp){
    let res = Math.random() * (esp.hum+6+0.8- esp.hum-6) + esp.hum-6;
    if(res <0)
        res=0;
    if(res >100)
        res=100;
    return res.toFixed(2)
}
```



# ESP 32



# IDE :



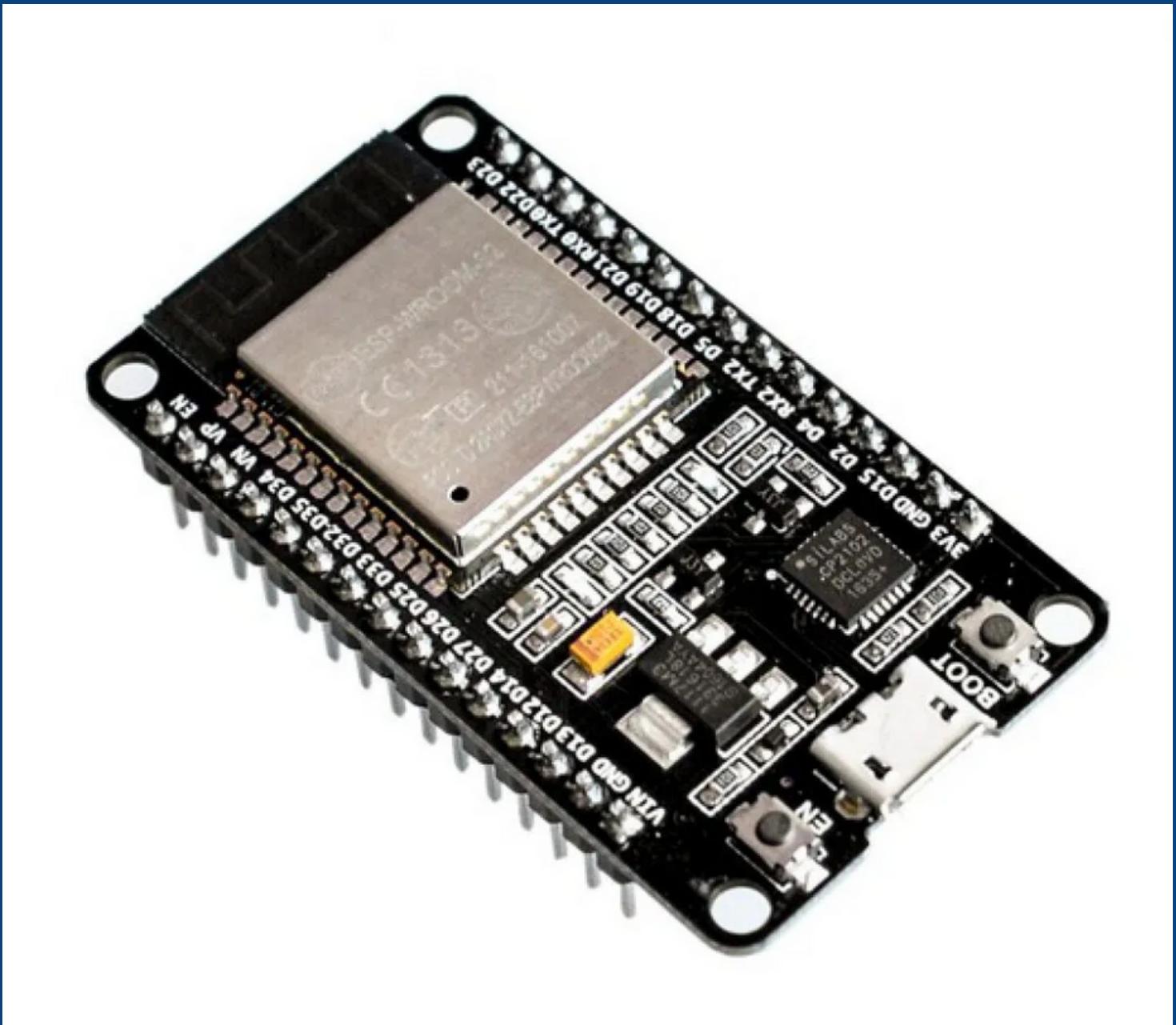
```
esp32 - net_misc.ino | Arduino 1.8.13
Fichier Édition Croquis Outils Aide
esp32 net_misc.h net_misc

void print_ip_status() {
    Serial.print("\nWiFi connected !\n");
    Serial.print("IP address: ");
    Serial.print(WiFi.localIP());
    Serial.print("\n");
    Serial.print("MAC address: ");
    Serial.print(WiFi.macAddress());
    Serial.print("\n");
}
String ssid;
String password;
void connect_wifi(){
// Access Point of the infrastructure
//const char* ssid = "HUAWEI-6EC2";
//const char *password= "FGY9MLBL";
//const char* ssid = "HUAWEI-553A";

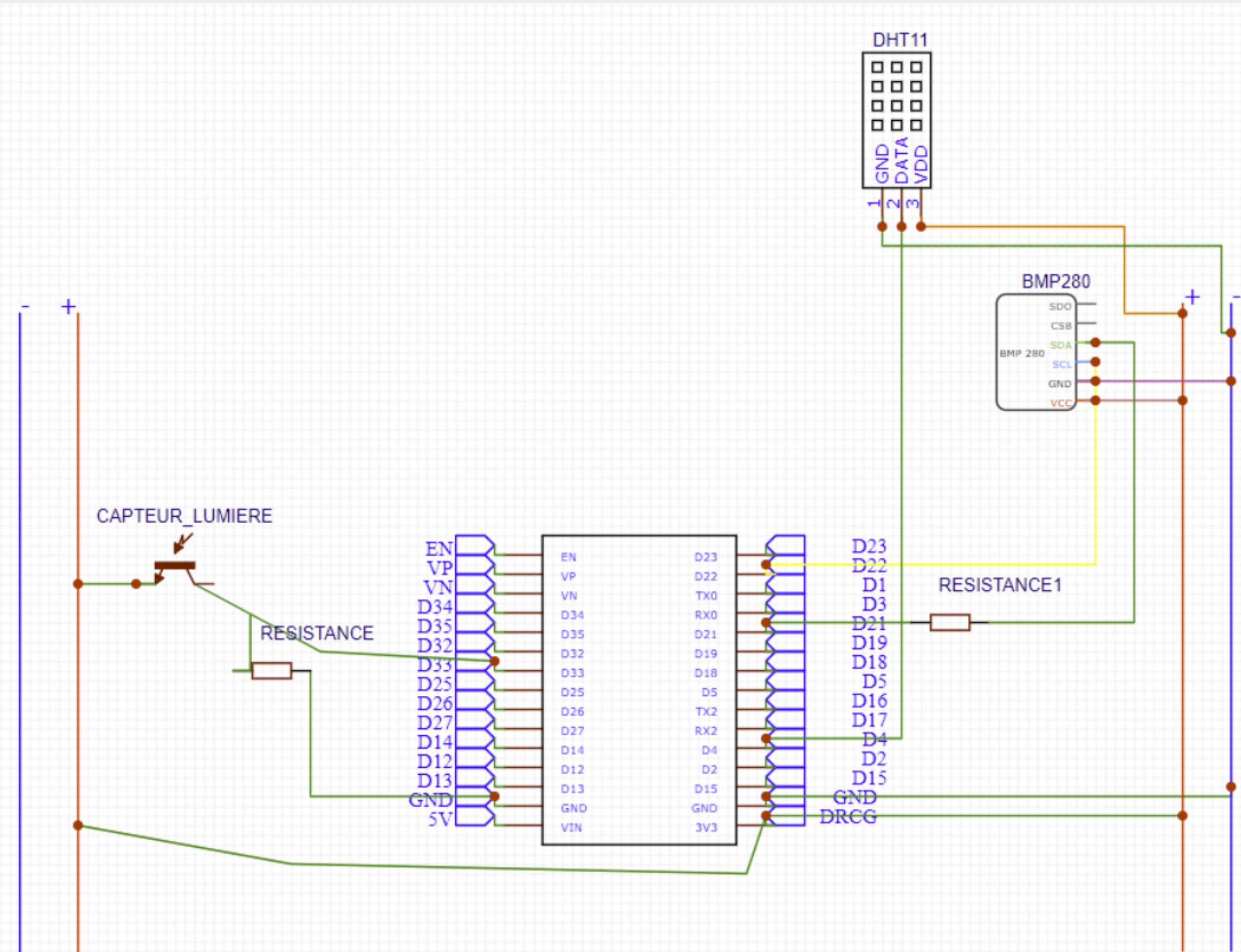
    preferences.begin("credentials", false);
    ssid = preferences.getString("ssid", "");
    password = preferences.getString("password", "");
    preferences.end();
    Serial.println("\nConnecting Wifi to ");
}

, Q8abled, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, Core 1, Core 1, None sur COM5
```

# ESP32:

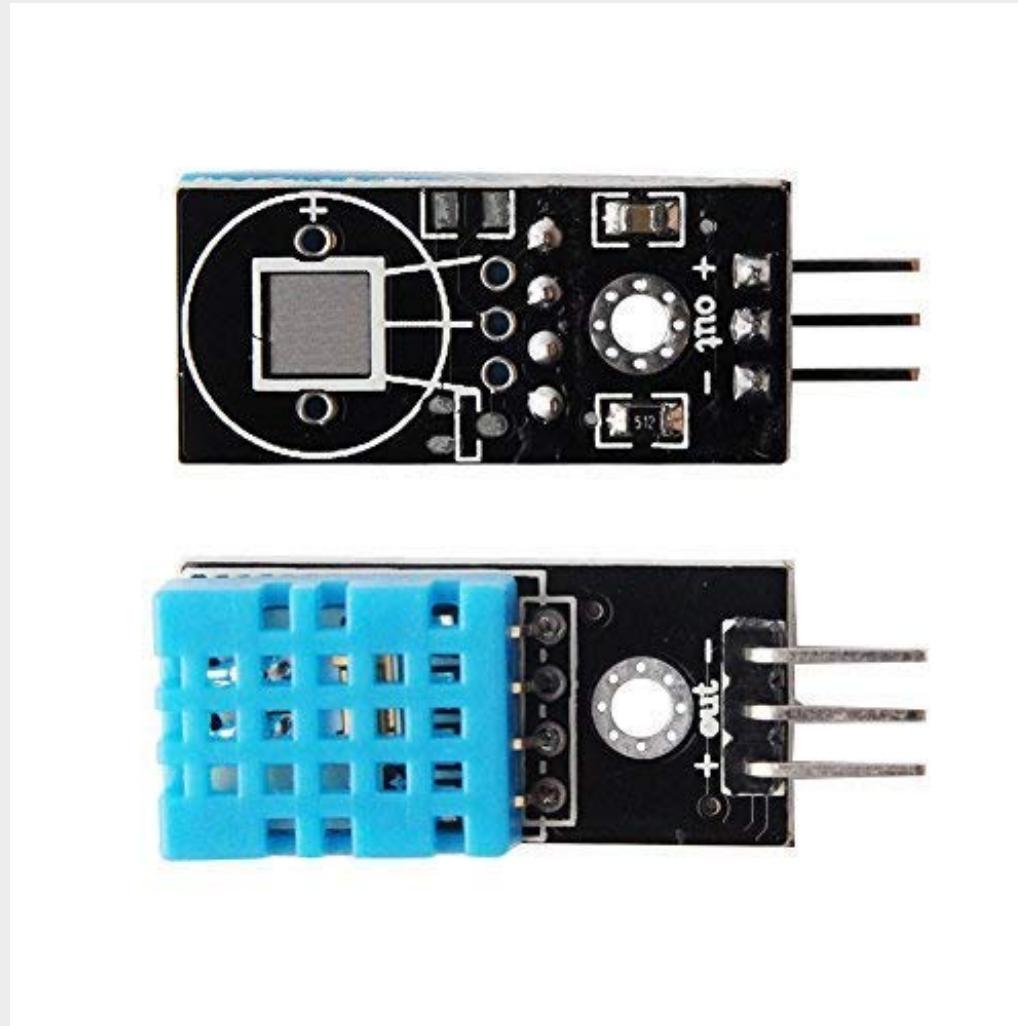


# Schéma de la station météo

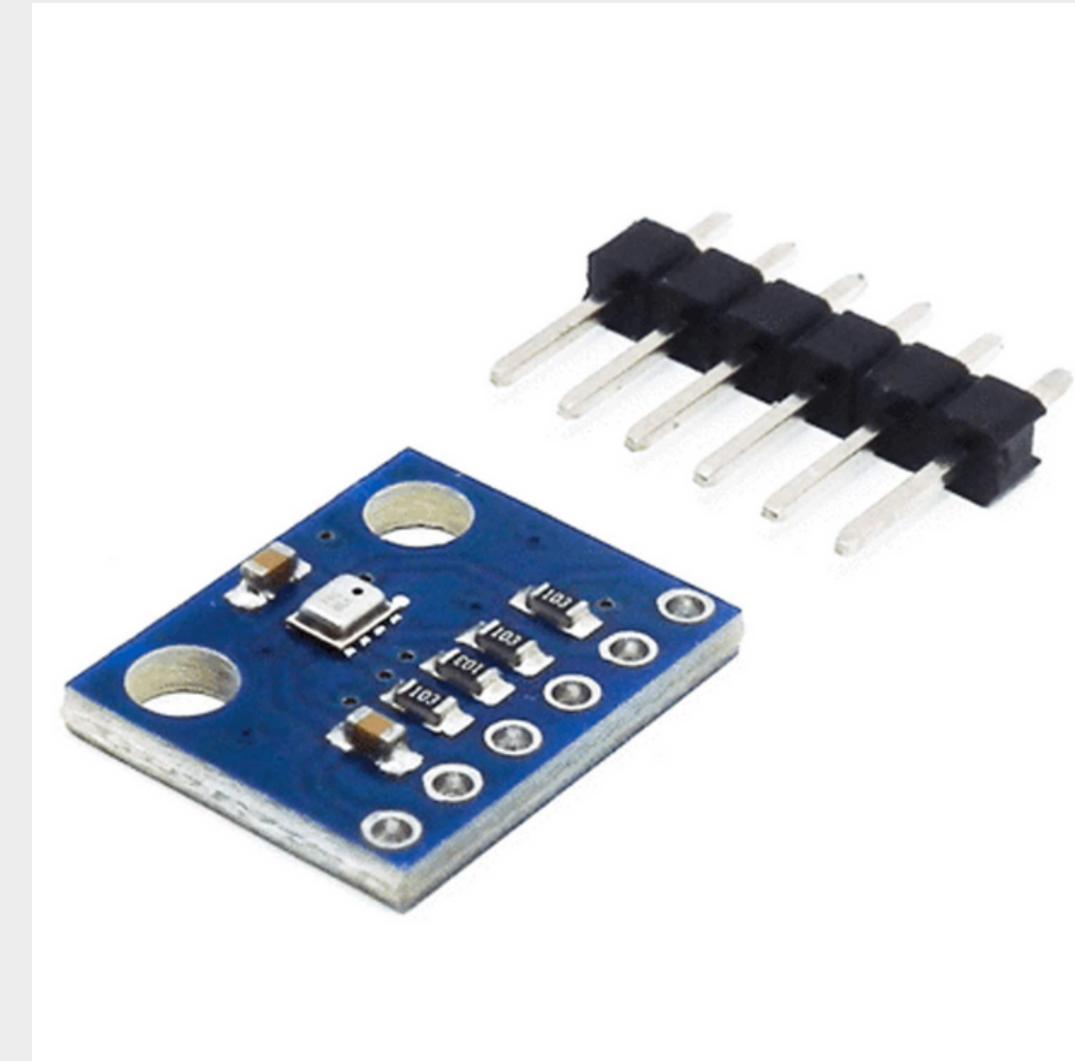


**Capteurs :**

Capteurs : DHT11 x 2 = 10€



Capteurs : BMP280 x 3 = 5.29€



## Initialiser les capteurs:

```
if (!bmp.begin(0x76)) {
    Serial.println(F("Could not find a valid BMP280 sensor, check wiring
or "
    "try a different address!"));
    while (1) delay(10);
}
//conf capteur bmp
  bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,           /* Operating Mode.
*/
  Adafruit_BMP280::SAMPLING_X2,      /* Temp. oversampling */
  Adafruit_BMP280::SAMPLING_X16,     /* Pressure oversampling */
  Adafruit_BMP280::FILTER_X16,       /* Filtering. */
  Adafruit_BMP280::STANDBY_MS_500); /* Standby time. */
```

```
#define DHTTYPE DHT11
uint8_t DHTPin = 4;
DHT dht(DHTPin, DHTTYPE);
```



## Récupérer les données

```
6     float get_temperature() {
7         /* float temperature;
8          tempSensor.requestTemperaturesByIndex(0);
9
10         temperature = tempSensor.getTempCByIndex(0);
11         bmp.readPressure() */
12         return bmp.readTemperature();
13     }
14
15     float get_pressure() {
16         return bmp.readPressure();
17     }
18     float get_alt() {
19         return bmp.readAltitude();
20     }
21
22     float get_light(){
23         return analogRead(photo_resistor_pin);
24     }
25     float get_temp(){
26         return dht.readTemperature();
27     }
28     float get_humidity(){
29         return dht.readHumidity();
30     }
31
32     void set_pin(int pin, int val){
33         digitalWrite(pin, val) ;
34     }
35
36     int get_pin(int pin){
37         return digitalRead(pin);
38     }
39
40
```

## La configuration du client MQTT:

```
WiFiClient espClient; // Wifi
PubSubClient client(espClient) ; // MQTT client
const char* mqtt_server = "ip_vps";
#define TOPIC "sensors/IMAGE/datas"
#define TOPIC_LED "sensors/aaa/led"
void setup () {
    client.setServer(mqtt_server, 8883);
}

void loop()
{
    if(client.connect("esp32", mqttUser, mqttPassword )) { // Attempt to
connect
        Serial.println("connected");
        client.subscribe(topic);
    }
}
```

## L'envoie des données

```
    payload = "{\"id\": \"\" + whoami + "\", \"userId\": \"\" + userID +
\"", \"lumiere\": \"\" + get_light() + "\", \"pression\": \"\" +
get_pressure() + "\", \"altitude\": \"\" + get_alt() + "\", \"humidite\":
\"\" + get_humidity() + "\", \"temperature\": \"\" + get_temperature()
+\"\" + \"}";
    payload.toCharArray(data, (payload.length() + 1));
    Serial.println(data);
    client.publish(TOPIC, data);
```



## inscription MQTT:

```
3
9 void mqtt_mySubscribe(char* topic) {
0     /*
1      * ESP souscrit à ce topic. Il faut qu'il soit connecté.
2      */
3     preferences.begin("credentials", false);
4     userID = preferences.getString("userid", "");
5     freq = preferences.getInt("freq", 0);
6         preferences.end();
7     Serial.print(freq);
8     Serial.print(userID);
9     while(!client.connected() && !isEmit) { // Loop until we are reconnected
0     if (button1.pressed) {
1         isEmit = true;
2         button1.pressed = false;
3     }
4
5     Serial.print("Attempting MQTT connection...");
6     if(client.connect("esp32", mqttUser, mqttPassword )) { // Attempt to connect
7         Serial.println("connected");
8         client.subscribe(topic); // and then Subscribe
9     } else { // Connection failed
0         Serial.print("failed, rc=");
1         Serial.print(client.state());
2         Serial.println("try again in 5 seconds");
3         // Wait 5 seconds before retrying
4         delay(2*1000);
5         digitalWrite(ONBOARD_LED,0.5);
6         delay(2*1000);
7         digitalWrite(ONBOARD_LED,1);
8     }
9 }
0     digitalWrite(ONBOARD_LED,LOW);
1 }
2
```

## Connexion WIFI

```
    String ssid;
    String password;
    void connect_wifi(){
        // Access Point of the infrastructure
        //const char* ssid = "HUAWEI-6EC2";
        //const char *password= "FGY9MLBL";
        //const char* ssid = "HUAWEI-553A";

        preferences.begin("credentials", false);
        ssid = preferences.getString("ssid", "");
        password = preferences.getString("password", "");
        preferences.end();
        Serial.println("\nConnecting Wifi to ");
        Serial.println(ssid.c_str());

        Serial.print("Attempting to connect ");
        WiFi.begin(ssid.c_str(), password.c_str());
        while(WiFi.status() != WL_CONNECTED && !isEmit){
            if (button1.pressed) {
                isEmit = true;
                button1.pressed = false;
            }
            digitalWrite(ONBOARD_LED,HIGH);
            delay(500);
            digitalWrite(ONBOARD_LED,LOW);
            delay(500);
            Serial.print(ssid.c_str());
            Serial.print(password.c_str());
            Serial.print(".");
        }

        print_ip_status();
    }
```

# Configurer l'ESP

```
Button button1 = {0, 0, false};

//Quand le bouton est pressé on appelle cette méthode qui va changer
l'état du bouton, on peut alors savoir si celui-ci a été appelé et
changer le comportement de l'ESP
void IRAM_ATTR isr() {
    button1.numberKeyPresses += 1;
    button1.pressed = true;
}

int buttonState = 0;

void setup () {
    attachInterrupt(button1.PIN, isr, FALLING);
}
```

```
Configurationloop qui va vérifier si le bouton a été
    void loop () {
        if (button1.pressed) {
            isEmit = !isEmit;
            configured = false;
            button1.pressed = false;
        }
        if(!isEmit)
        {
            if (!configured)
            {
                WiFi.mode(WIFI_STA);
                connect_wifi();
                configured = true;
            }
        }
    }
}
```

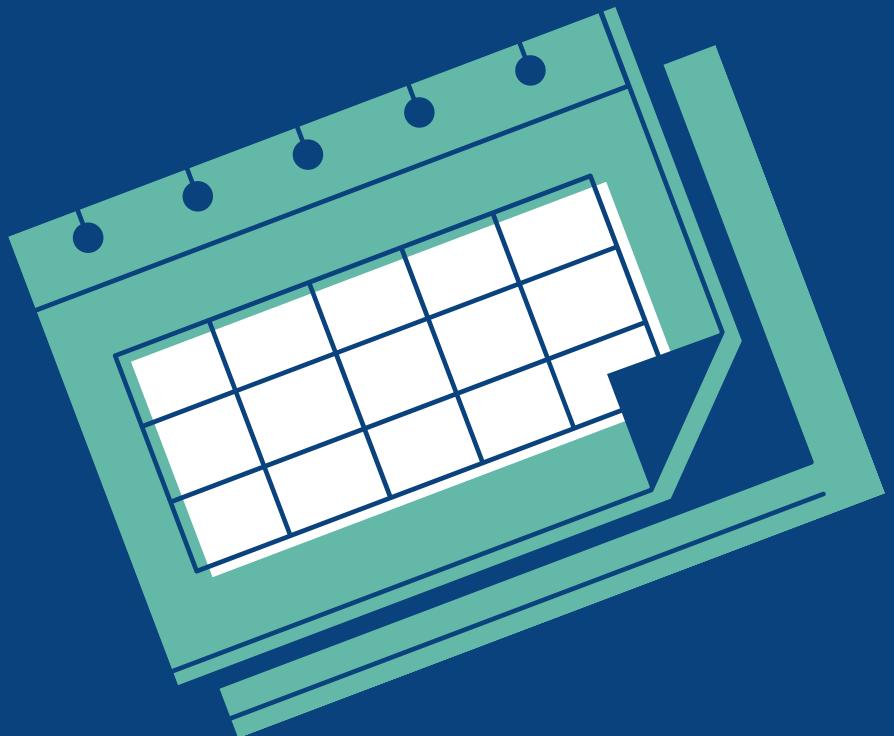
## #include <Preferences.h>

```
400 //configuration de l'esp32
401 else {
402     if (!configured)
403     {
404         //on emmet du wifi
405         Serial.print("Setting AP (Access Point)...");
406         WiFi.softAP("Esp32", NULL);
407         IPAddress IP = WiFi.softAPIP();
408
409         Serial.print("AP IP address: ");
410         Serial.println(IP);
411         //on envoie la page web au client
412         server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
413             request->send_P(200, "text/html", index_html, processor);
414         });
415         //Quand le client a sauvé ses datas on les reçoit ici
416         server.on("/get", HTTP_GET, [] (AsyncWebServerRequest *request) {
417             String inputMessage;
418             int inputMessageFreq;
419             String inputParam;
420             //pour chaque if on enregistre le paramètre dans la mémoire morte si le paramètre a bien été changé
421             if (request->hasParam(SSID)) {
422                 inputMessage = request->getParam(SSID)->value();
423                 if (inputMessage[0] != '\0')
424                 {
425                     inputParam = SSID;
426
427                     preferences.begin("credentials", false);
428                     preferences.putString("ssid", inputMessage);
429                     Serial.println(inputParam);
430                     preferences.end();
431                 }
432             }
433         });
434     }
#define ONBOARD_LED 2
digitalWrite(ONBOARD_LED,1);
```

## Exemple page web

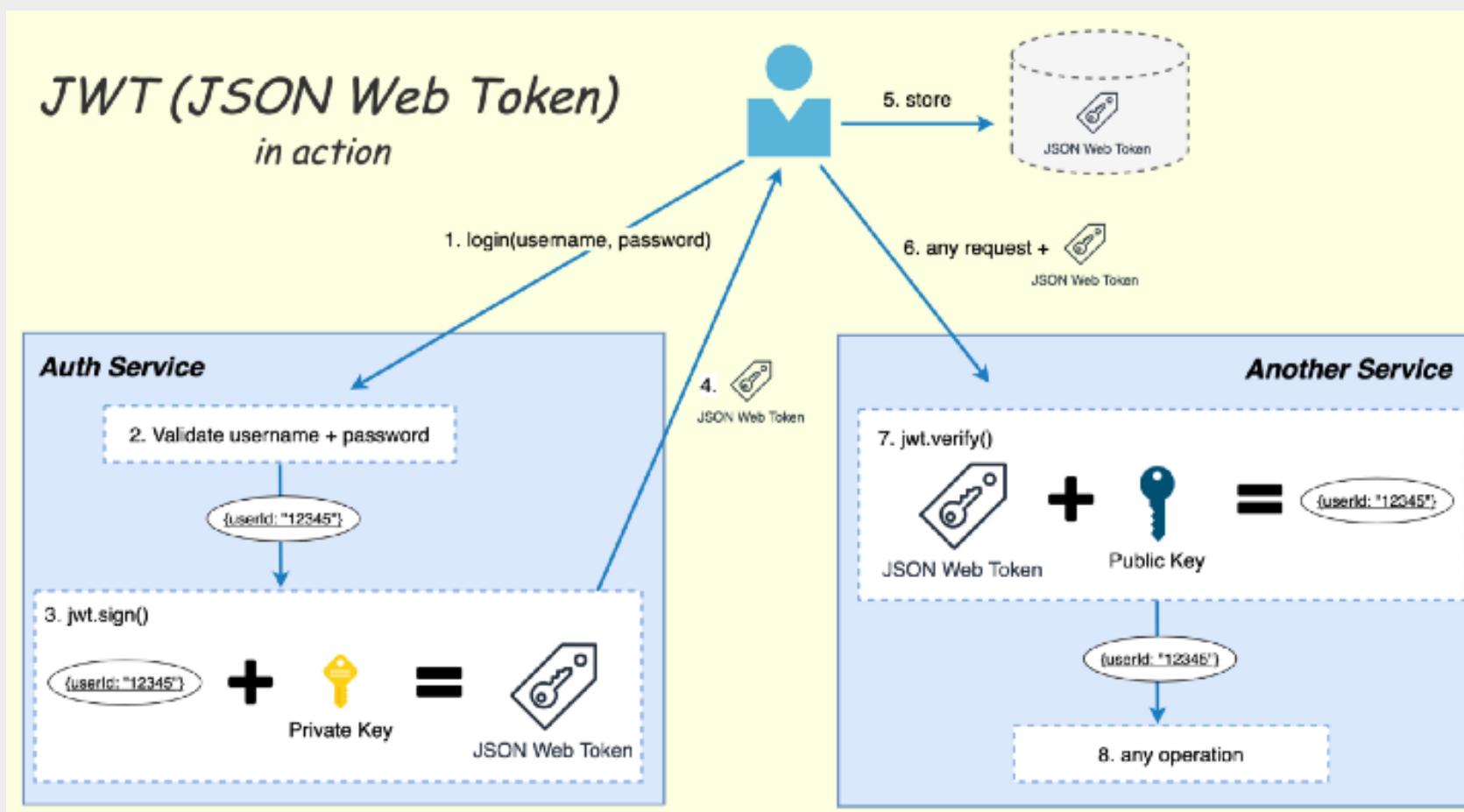
```
2  String processor(const String& var)
3  {
4      if(var == "TEMPLATE")
5          return F(whoamiChar);
6      return String();
7  }
8
9 //La page locale que l'on envoie pour configurer l'ESP
10 char index_html[] = R"rawliteral(
11     <!DOCTYPE HTML><html><head>
12         <title>Configuration ESP </title>
13         <meta name="viewport" content="width=device-width, initial-scale=1" charset="UTF-8">
14     </head><body>
15         <H1>Configurez votre station météo</H1>
16         Adresse MAC: %TEMPLATE%
17         <br>
18         <br>
19         <form action="/get" class="form">
20             <label for="ssid">
21                 SSID:
22             </label>
23             <input type="text" name="ssid" id="ssid">
24             <br>
25             <label for="mdp">
26                 Mot de passe:
27             </label><input type="text" name="mdp" id="mdp">
28             <br>
29             <label for="userid">
30                 ID de l'user sur le site web:
31             </label> <input type="text" name="userid" id="userid">
32             <br>
33             <label for="freq">
34                 Fréquence d'envoie de l'ESP en secondes:
35             </label> <input type="number" name="freq" id="freq">
36             <br>
37             <input type="submit" value="Envoyer" id="submit">
38         </form><br>
39     </body></html>
40     <style>
```

# Sécurité des flux de données de notre api:



PROJET  
METEO

# JWT: JSON Web Tokens



# Authentification

```
const jwt = require('jsonwebtoken')

function auth(req,res,next){
    const token = req.header('x-auth-token');
    try {
        /**
         * 
         * if(!token) {
        res.status(401).json({ message : "Connexion impossible, token manquant", code: res.statusCode });
        }else{
            req.user = jwt.verify(token, process.env.JWT_SECRET);
            next();
        }
        /**
        // next();
        /**
     } catch (e) {
        res.status(400).json({error : 'Token invalide', code: res.statusCode});
    }
}

module.exports = auth;
```



# Securisation des routes

Exemple:

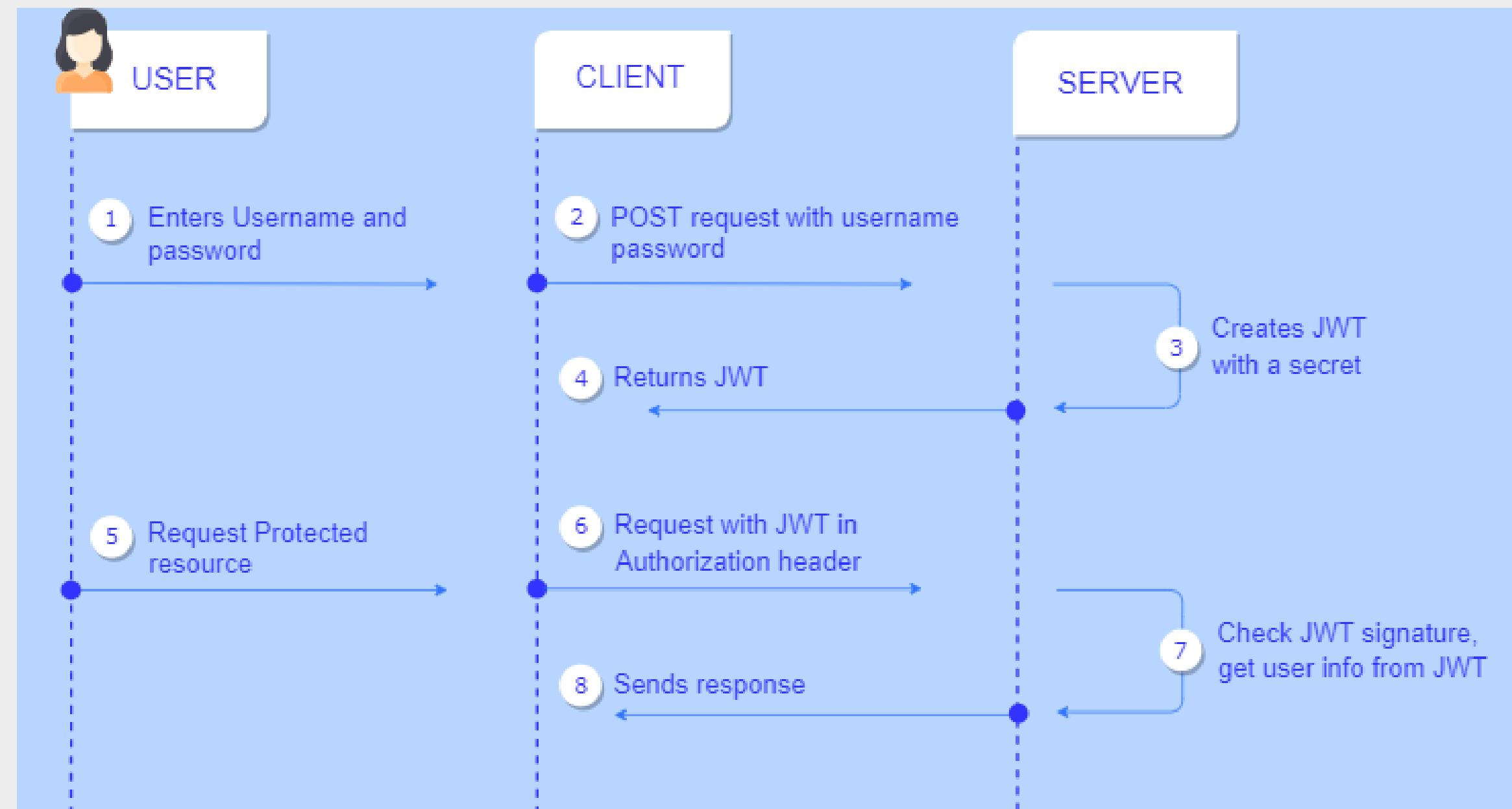
```
//GET
router.get('/', espController.getAll);
router.get('/getEsp/:id', token, espController.getEspById);

//POST
router.post('/addEsp', token, espController.newEsp); // créer un nouvel utilisateur (voir le model pour le body)
```

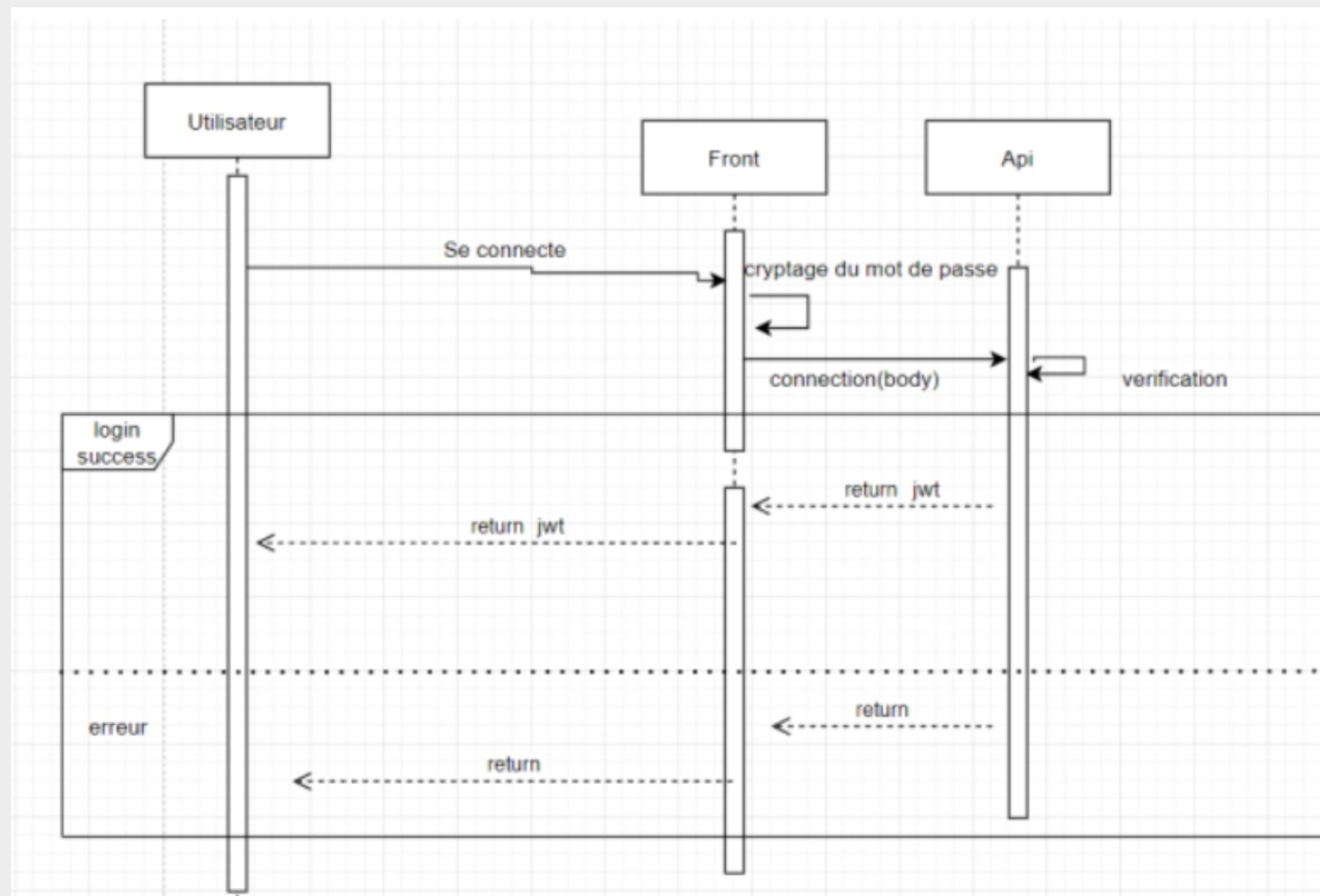
# Mise en place du header

```
getMyEsps: async function () {//Permet de get les esp de l'utilisateur
    this.showSpinner = true;
    await fetch(` ${urlApi}/esp/getEsp/${this.userId}` , {
        headers: {
            "Content-Type": "application/json",
            "x-auth-token": this.$session.get("token"),
        },
    })
        .then((res) => {
            res.json().then((resJson) => {
                //on cache le spinner
                this.showSpinner = false;
                this.listEsp = resJson;
                //on recupere un code 200 mais on a pas d'esp, juste un objet "erreur" on remet donc la list vide
                if (this.listEsp.erreur) {
                    this.listEsp = [];
                }
            });
            this.showSpinner = false;
        })
        .catch((err) => {
            console.error(err);
            //on cache le spinner si on arrive pas à récupérer les données pour ne pas gêner l'utilisateur
            this.showSpinner = false;
        });
},
```

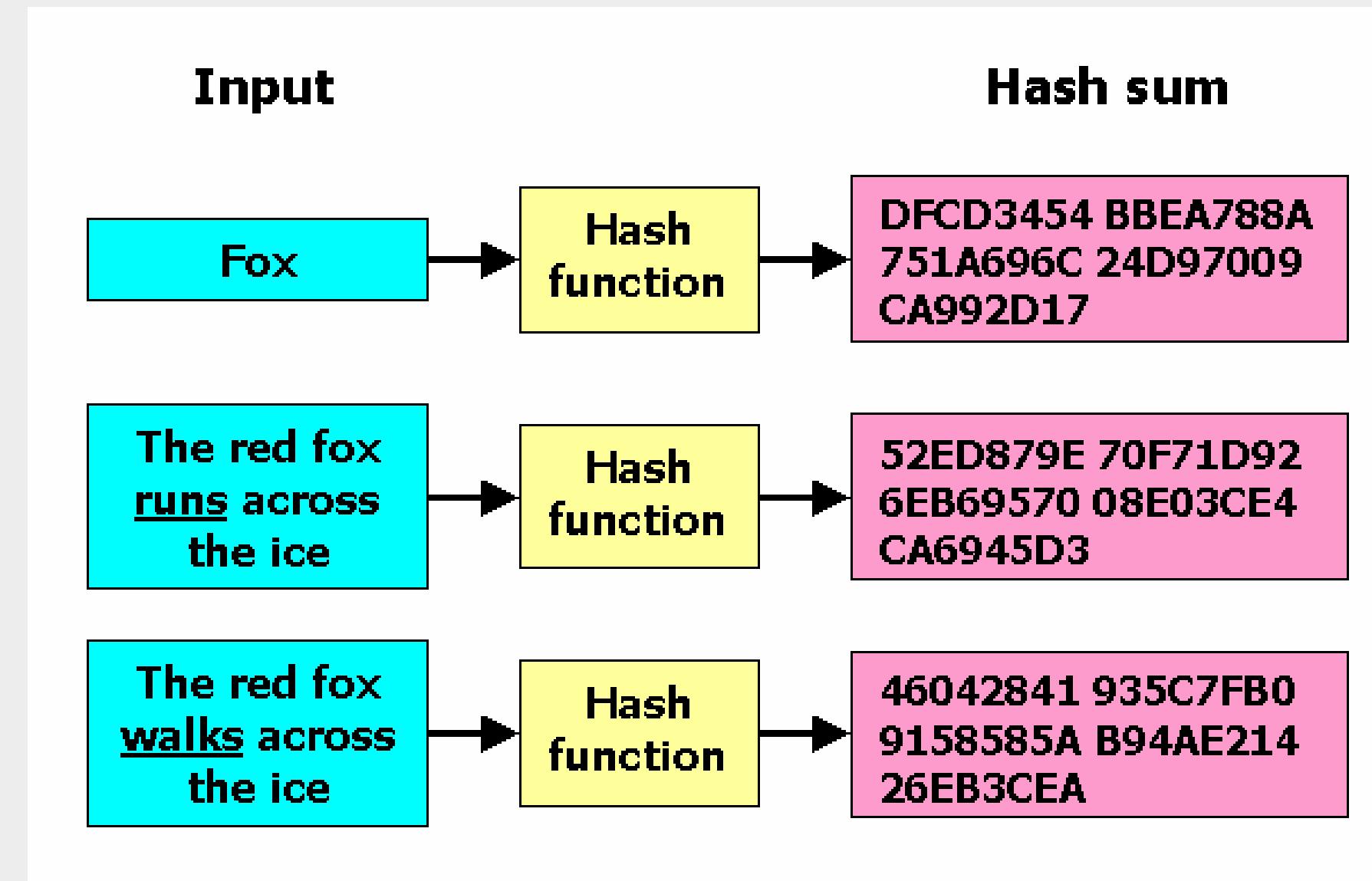
# Connexion



# Connexion



# MD5



# Sécurité du VPS : Pare-feu

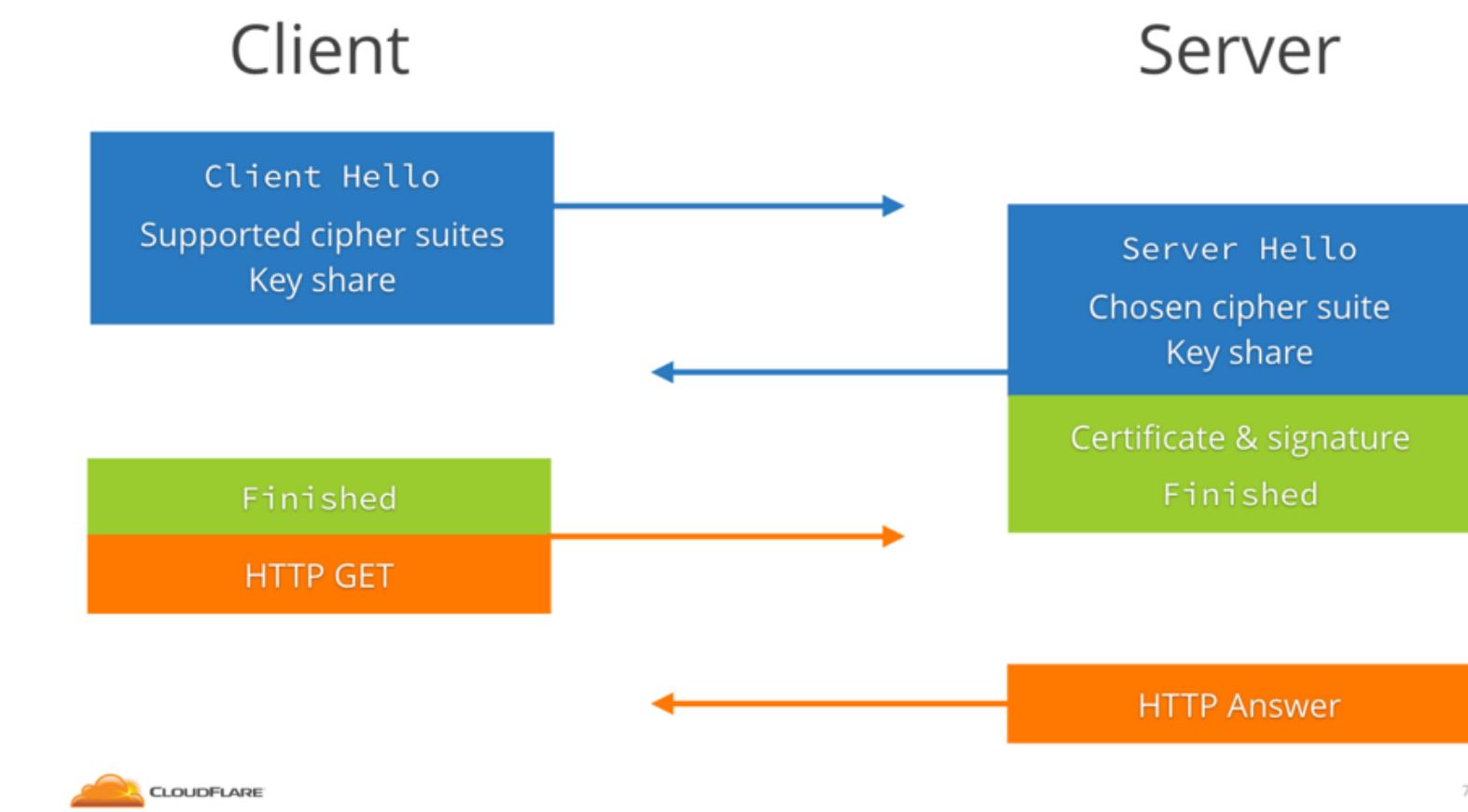
```
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To           Action    From
--          -----
22/tcp        DENY IN  Anywhere
443/tcp       ALLOW IN Anywhere
80/tcp        ALLOW IN Anywhere
50[REDACTED]  ALLOW IN Anywhere
50[REDACTED]/tcp ALLOW IN Anywhere
22            DENY IN  Anywhere
1983          ALLOW IN Anywhere
3000          DENY IN  Anywhere
8883          ALLOW IN Anywhere
1883          ALLOW IN Anywhere
8083          ALLOW IN Anywhere
18083         ALLOW IN Anywhere
3001          DENY IN  Anywhere
22/tcp (v6)   DENY IN  Anywhere (v6)
443/tcp (v6)  ALLOW IN Anywhere (v6)
80/tcp (v6)   ALLOW IN Anywhere (v6)
50[REDACTED] (v6) ALLOW IN Anywhere (v6)
50[REDACTED]/tcp (v6) ALLOW IN Anywhere (v6)
22 (v6)       DENY IN  Anywhere (v6)
1983 (v6)     ALLOW IN Anywhere (v6)
3000 (v6)     DENY IN  Anywhere (v6)
8883 (v6)     ALLOW IN Anywhere (v6)
1883 (v6)     ALLOW IN Anywhere (v6)
8083 (v6)     ALLOW IN Anywhere (v6)
18083 (v6)    ALLOW IN Anywhere (v6)
3001 (v6)     DENY IN  Anywhere (v6)
```

- UFW (Uncomplicated Firewall)
- Fail2Ban

# Sécurité du VPS : certificats SSL/TLS

- Let's encrypt
- TLS 1.3
- support du TLS 1.2

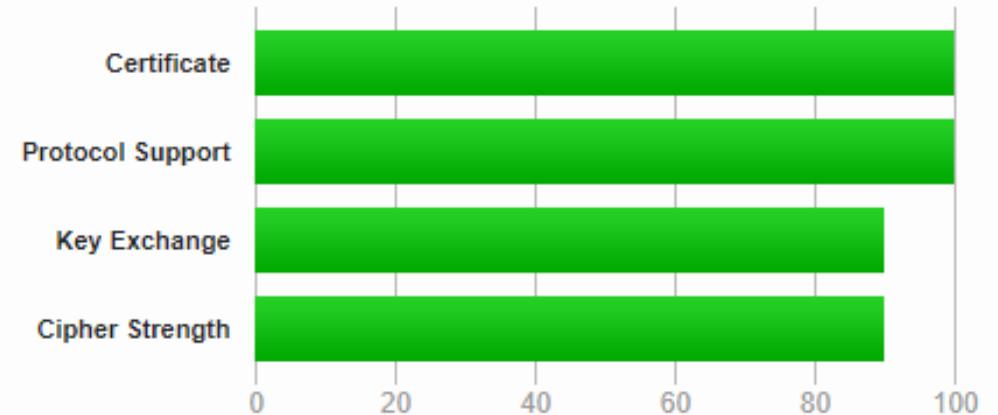


7

## Summary



Overall Rating

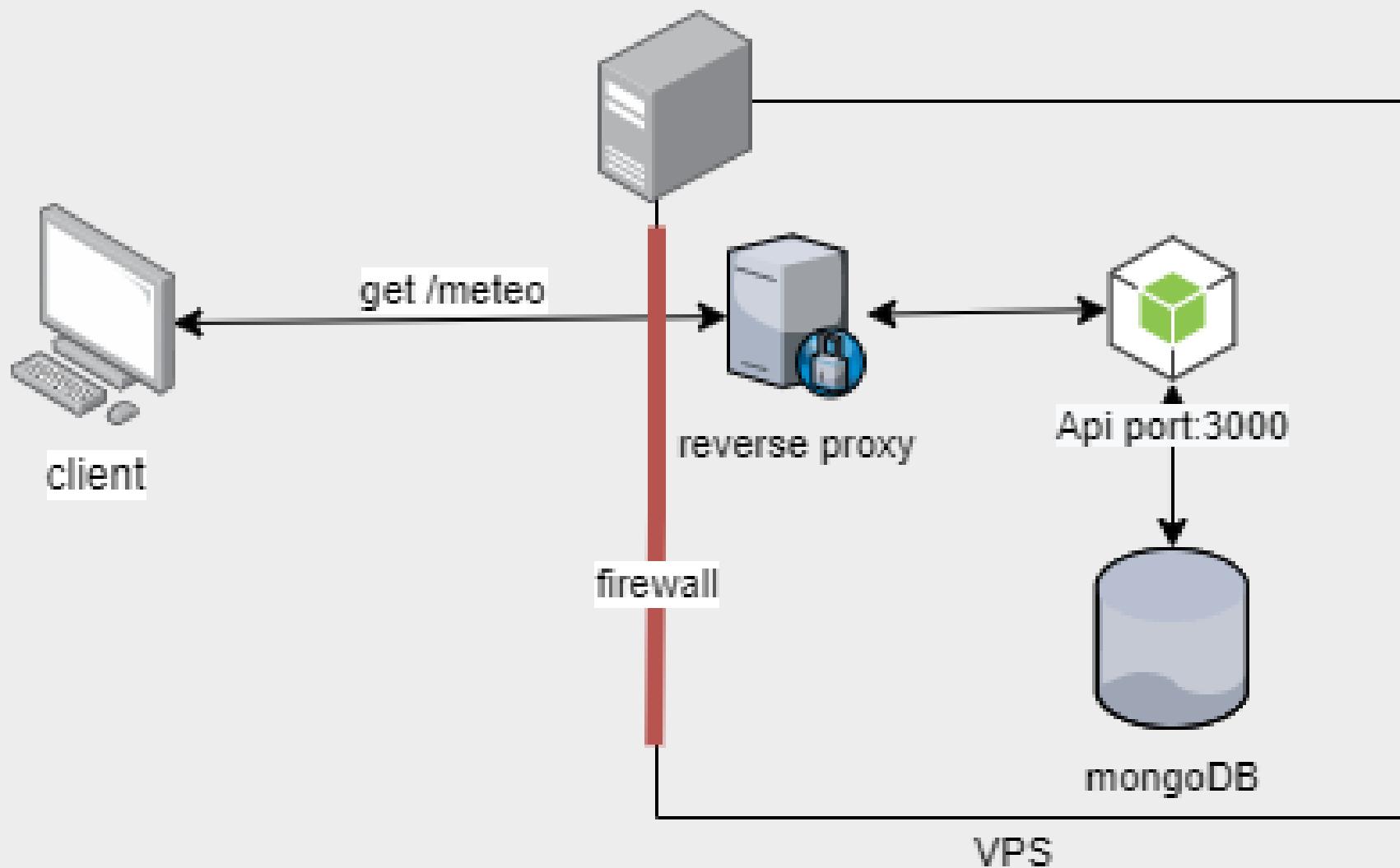


Visit our [documentation page](#) for more information, configuration guides, and books. Known issues are documented [here](#).

This server supports TLS 1.3.

www.ssllabs.com

# Sécurité du VPS : Reverse-proxy



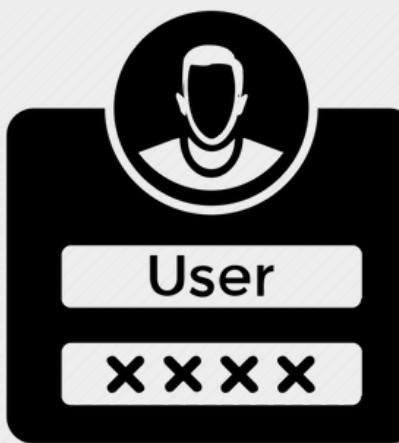
permet de :

- ne pas passer directement pas le port :3000
- forcer les clients à utiliser le port 443.
- redirige les requetes de /meteo au port :3000

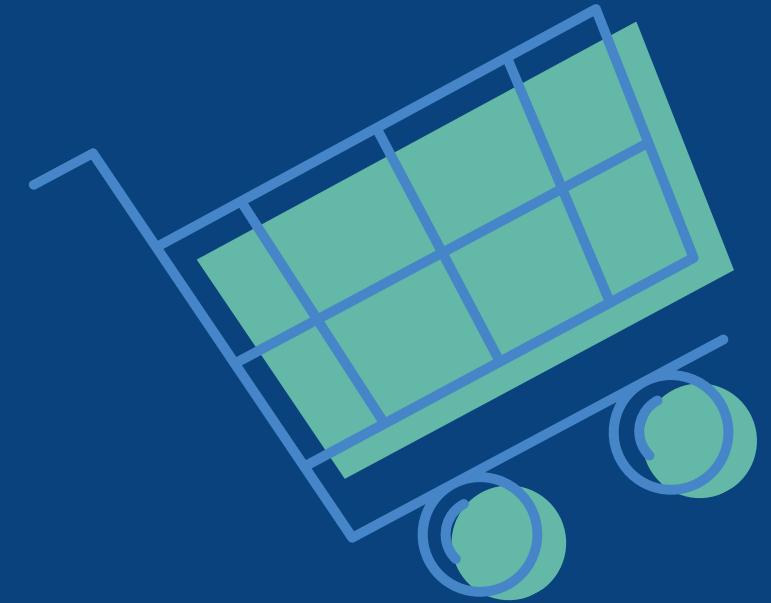
# Sécurité du VPS : Broker



Connections anonymes  
interdites



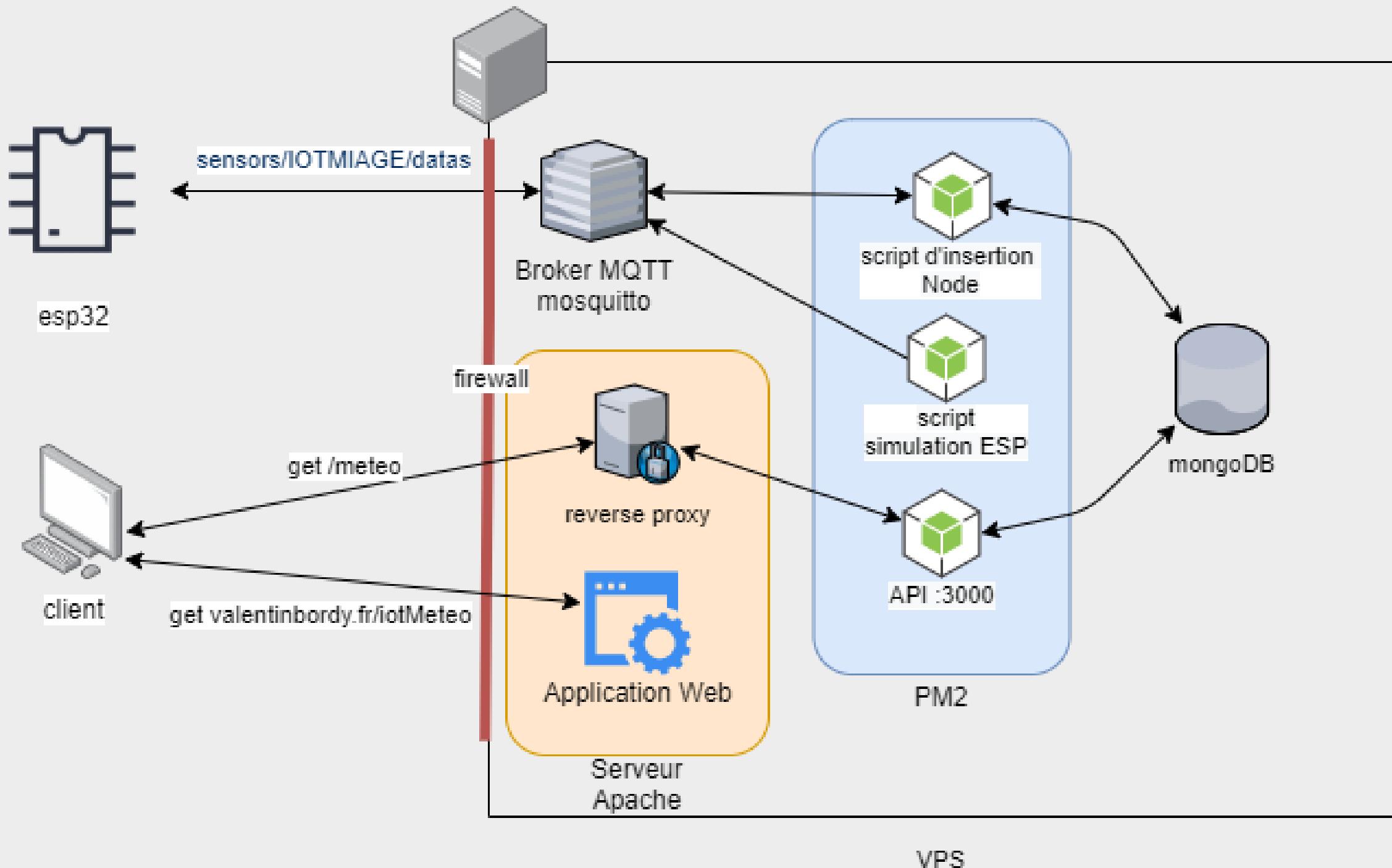
Connections obligatoires.  
Utilisateur/motDePasse  
générique



# Déploiement



# Rappel architecture



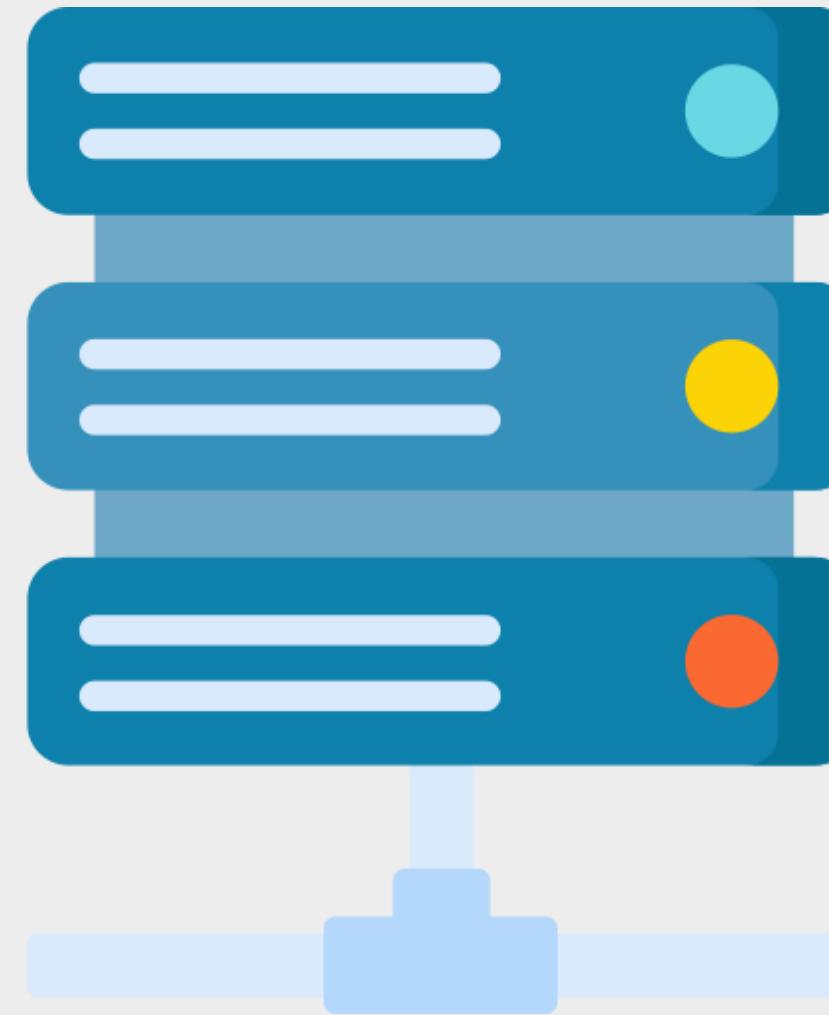
Vps:

- parefeu
- pm2
- mongoDB
- Apache
- Broker

# Pourquoi un vps et pas Heroku ?

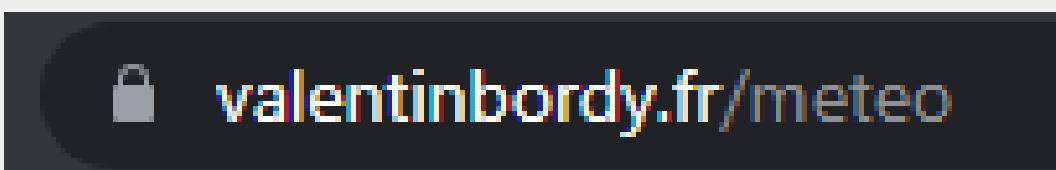


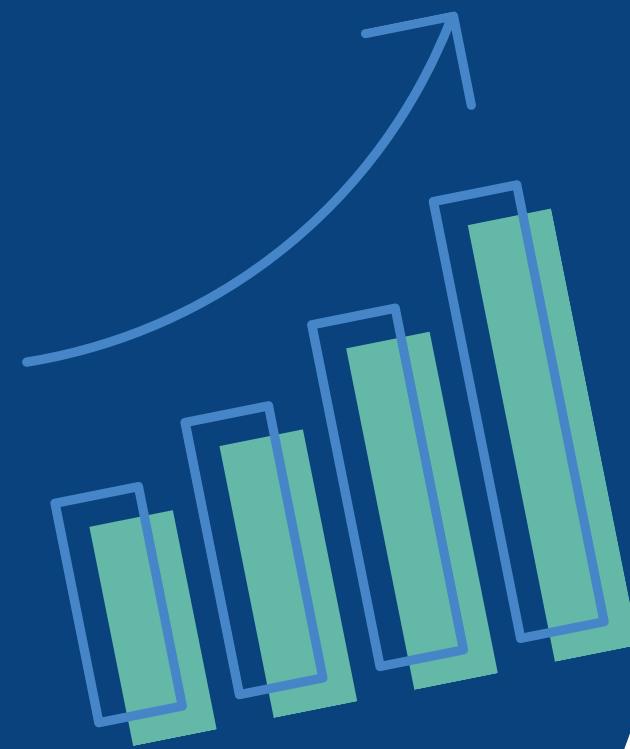
heroku



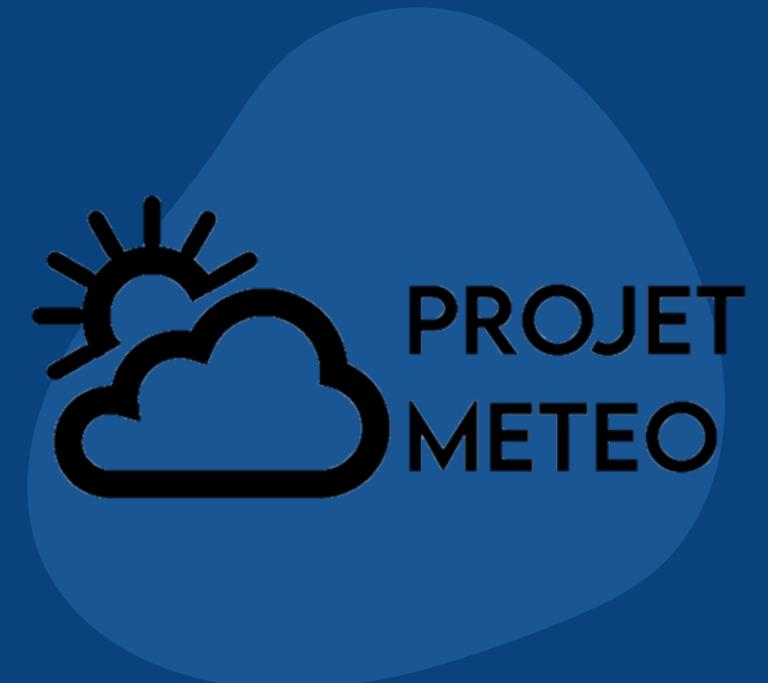
# Amélioration futures pour le déploiement

- Une seule route: /meteo
- Fichier .env





# Améliorations possibles



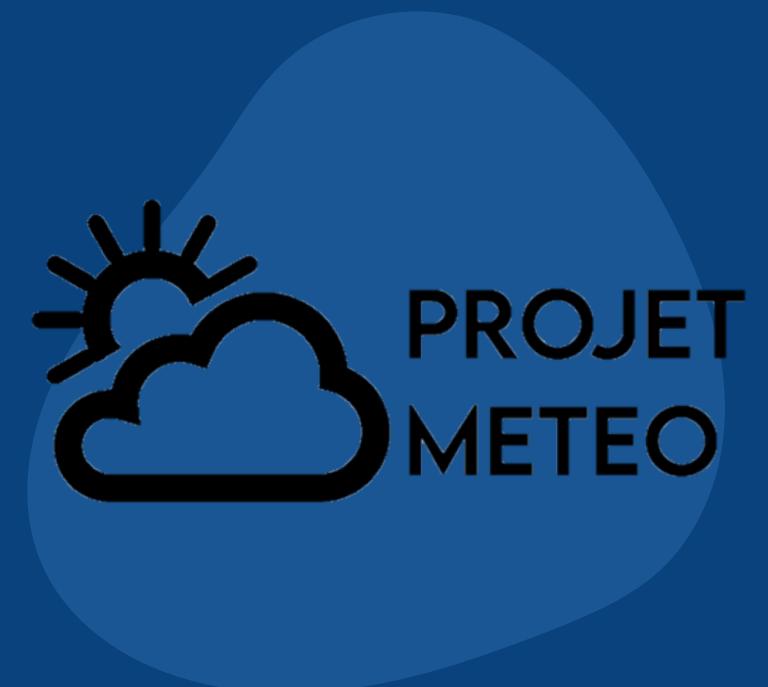
# Les Améliorations

- Lors du changement d'adresse ne plus afficher les données antérieures au changement d'adresse
- Améliorer la précision des prévisions.
- Modification du profil.
- Amélioration de la liaison ESP-USER.
- Amélioration du déploiement.





# Conclusion





Merci pour votre  
attention

