Projet Météo

IOT – Gilles Ménez

Université de Nice
Sophia Antipolis - MIAGE

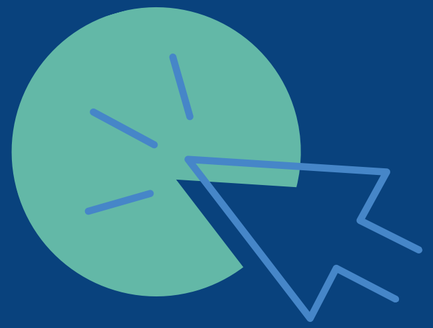
Groupe 13

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

PROJET METEO

SOMMAIRE

PROJET METEO



Introduction

PROJET METEO

Présentation du Projet

PROJET METEO

Projet Meteo

- Création de station météo connecté

- Récupération des données de la station

- Création d'une application web

- Connexion des ESP au site web

PROJET METEO

Présentation de l'équipe

PROJET METEO

Démo

PROJET METEO

Architecture

PROJET METEO

Schéma architecture projet

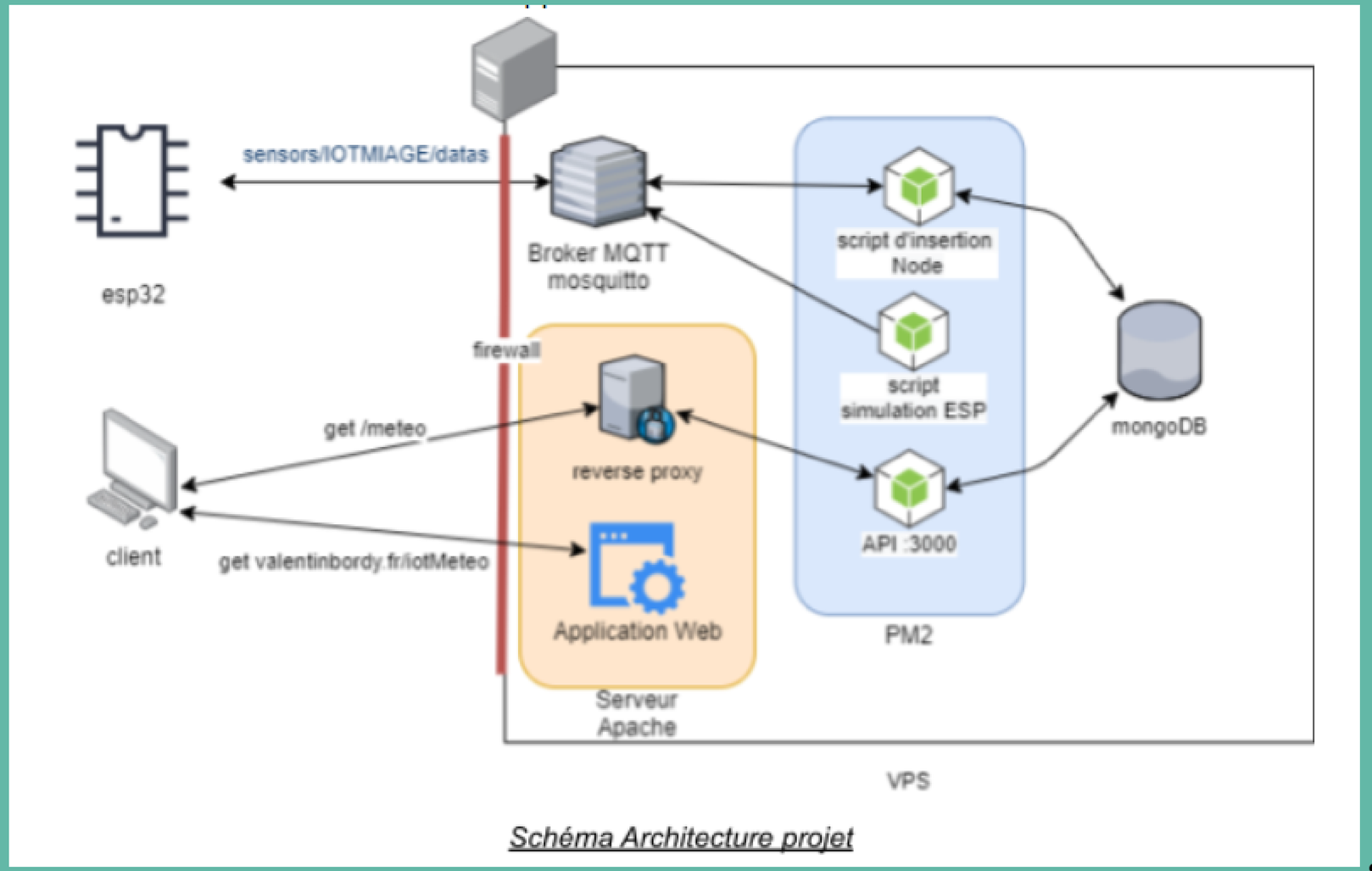


Schéma Architecture projet

PROJET METEO

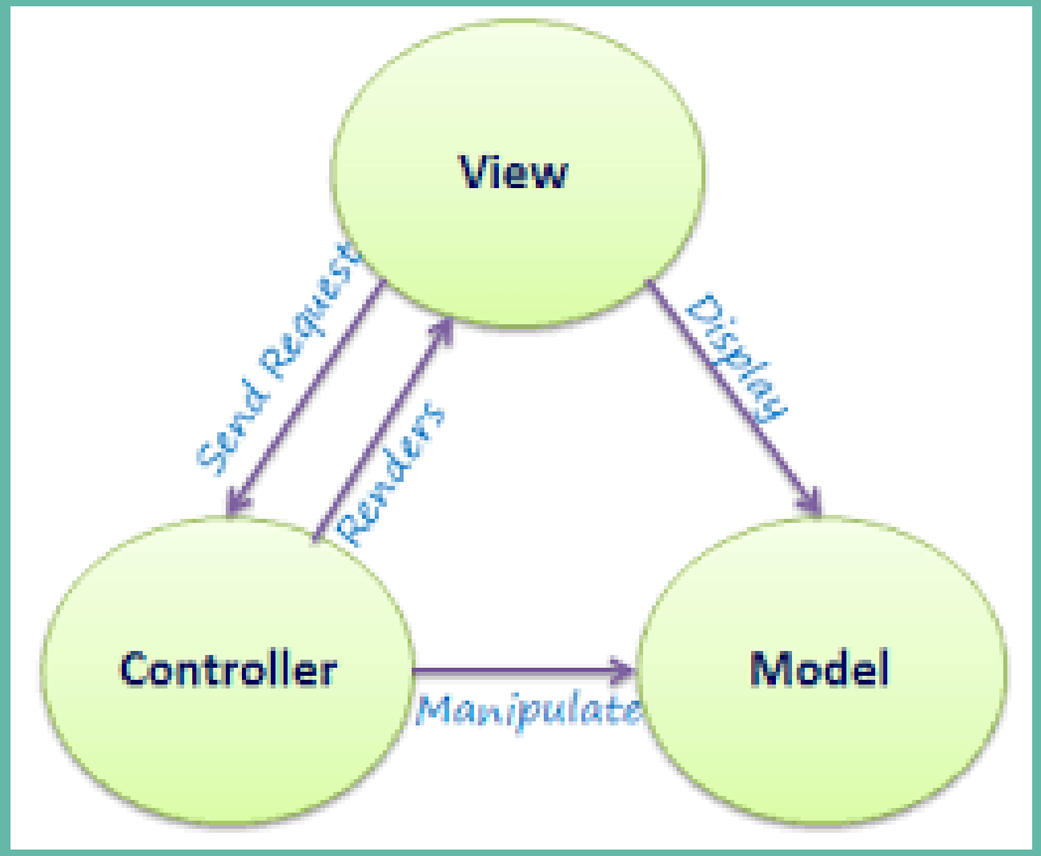
Notre projet est séparée en 3 parties

- Partie Back-end avec MQTT MONGO

- Partie Front-end

- Partie ESP32

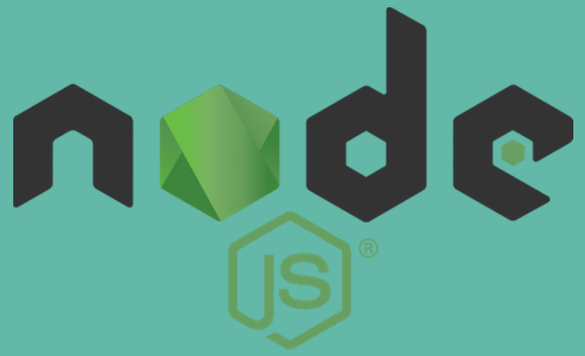
Présence d'un MVC



PROJET METEO

Back-end

Les différentes technologies utilisés :



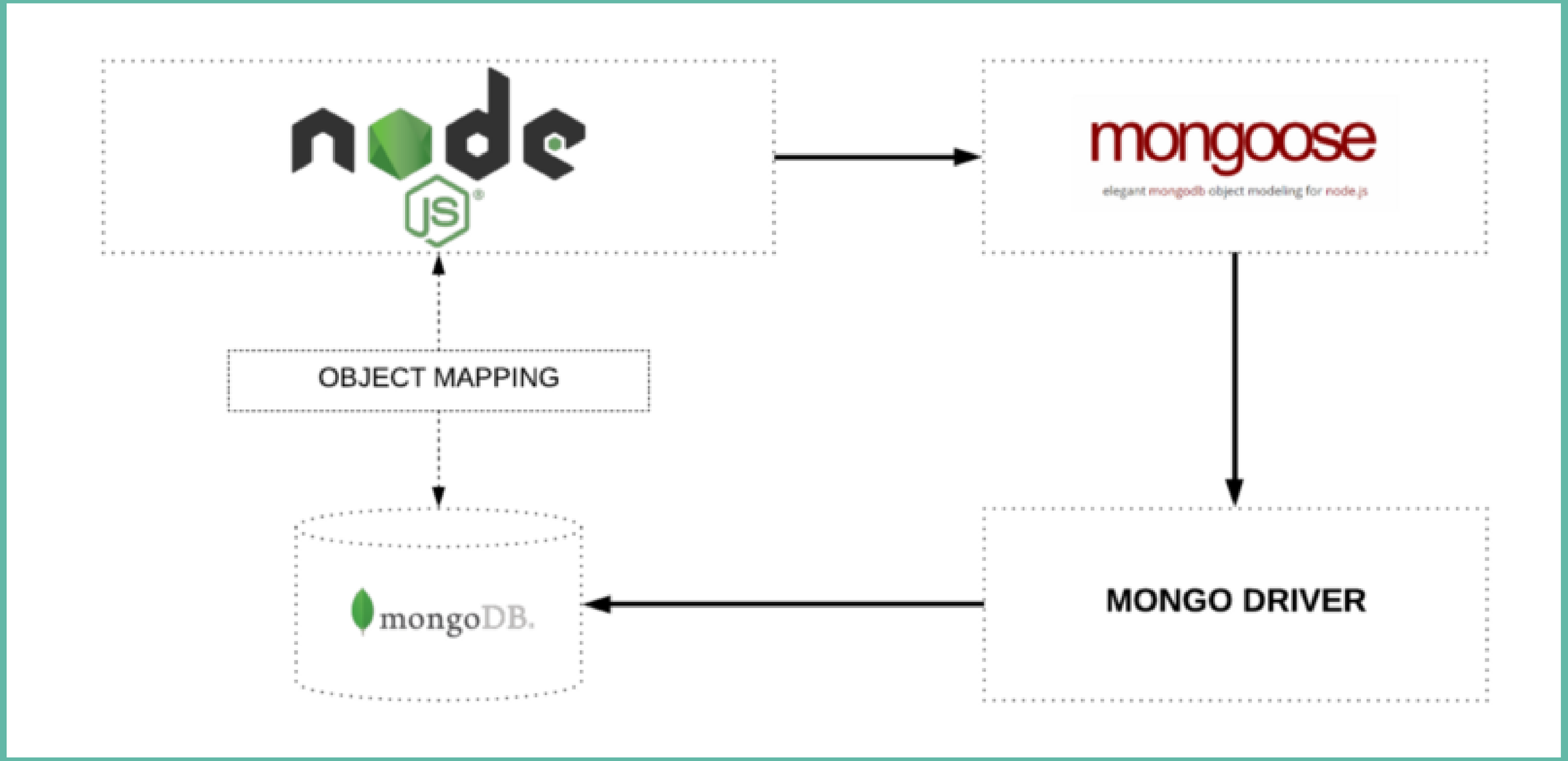
Express JS

Mongoose

mongoDB

PROJET METEO

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);
```

PROJET METEO

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 }
});
```

PROJET METEO

```
//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,
});
```

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

//structure de donnee d'un user
const UserSchema = mongoose.Schema({
    userFirstName: {
        type: String,
        required: true
    },
    userLastName: {
        type: String,
        required: true
    },
    userEmail: {
        type: String,
        required: true
    },
    userPassword: {
        type: String,
        required: true
    },
    userPhoneNumber: {
        type: String,
        required: false
    },
    userAddress: {
        type: String,
        required: true
    },
    picUrl: {
        type: String,
        required: false,
        default: "https://media.discorda
    },
});
```

PROJET METEO

Lien/clé vers Base de données/Api externe

```
//environnement de dev
//const uri = 'mongodb+srv://[redacted]&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 = '[redacted]';

module.exports = {
  openWeather_url,
  openWeatherKey
 };
```

PROJET METEO

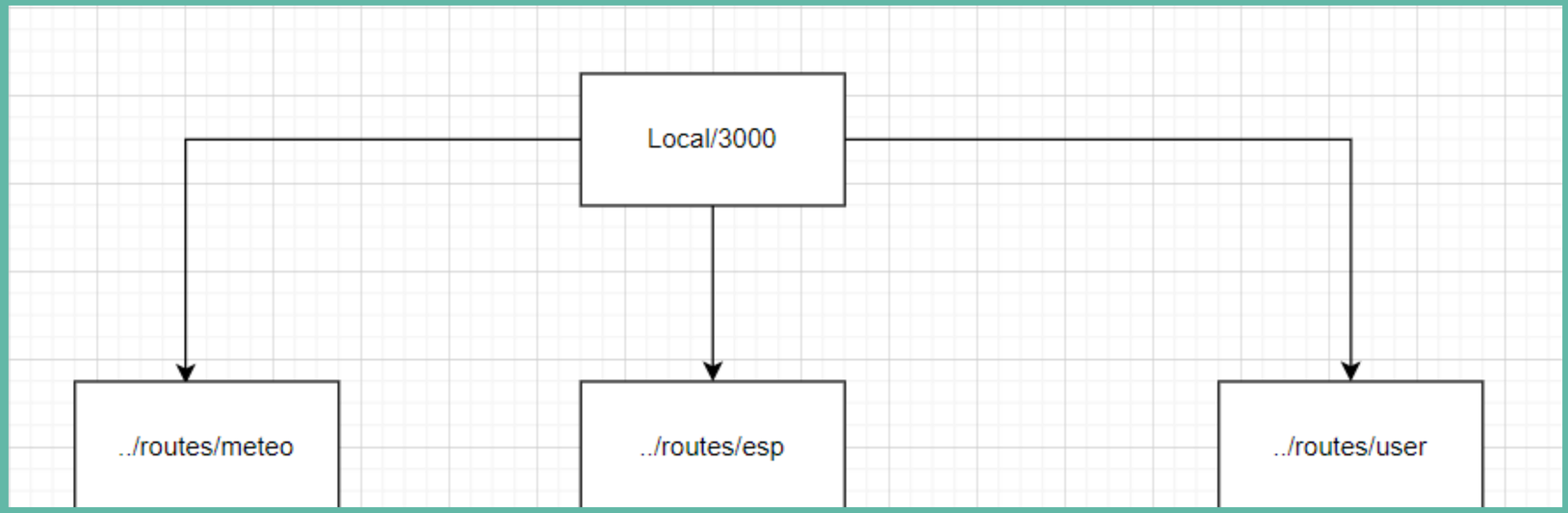
Contrôleurs

```
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});
        })
}
```

PROJET METEO

Routes

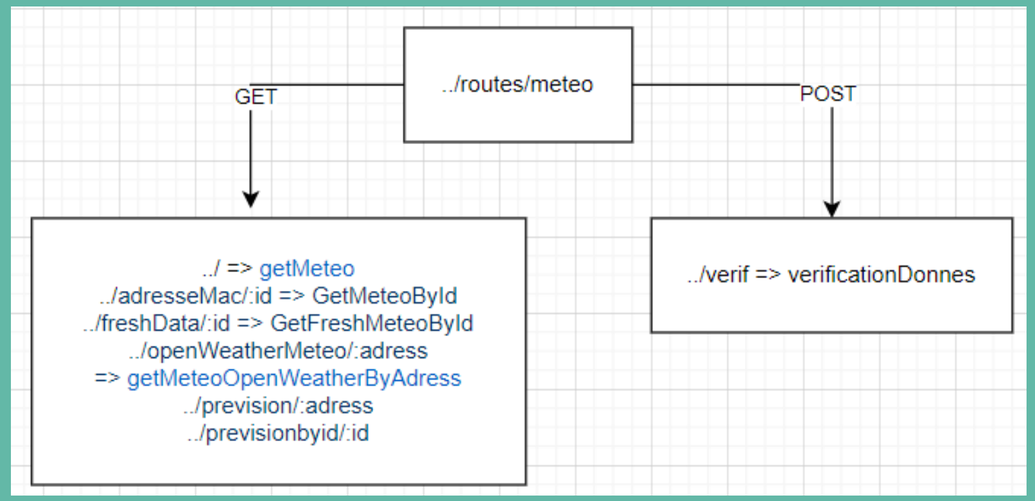


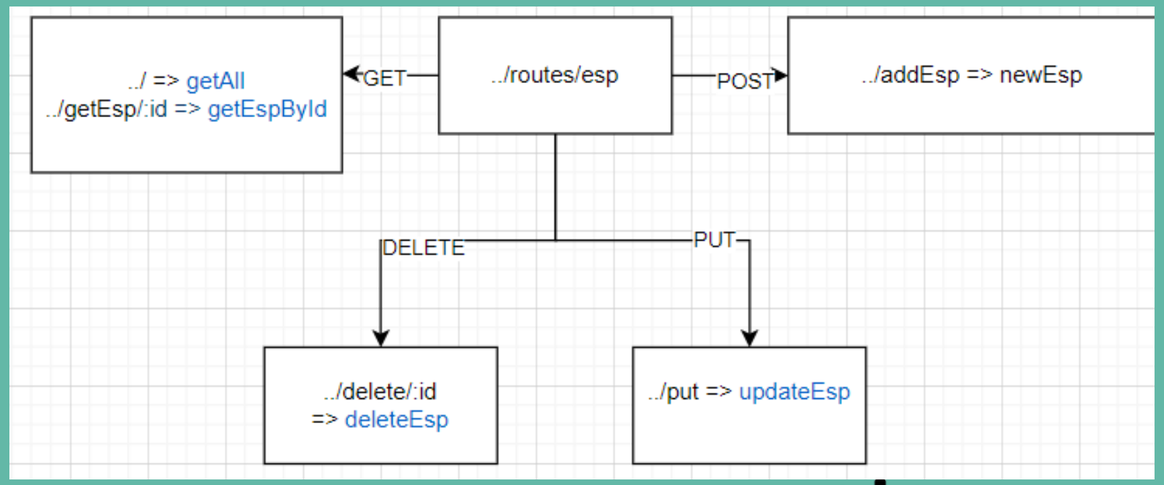

```
app.use('/meteo', meteoRoutes);
app.use('/user', userRoutes);
app.use('/esp', espRoutes);
```

PROJET METEO

Routes

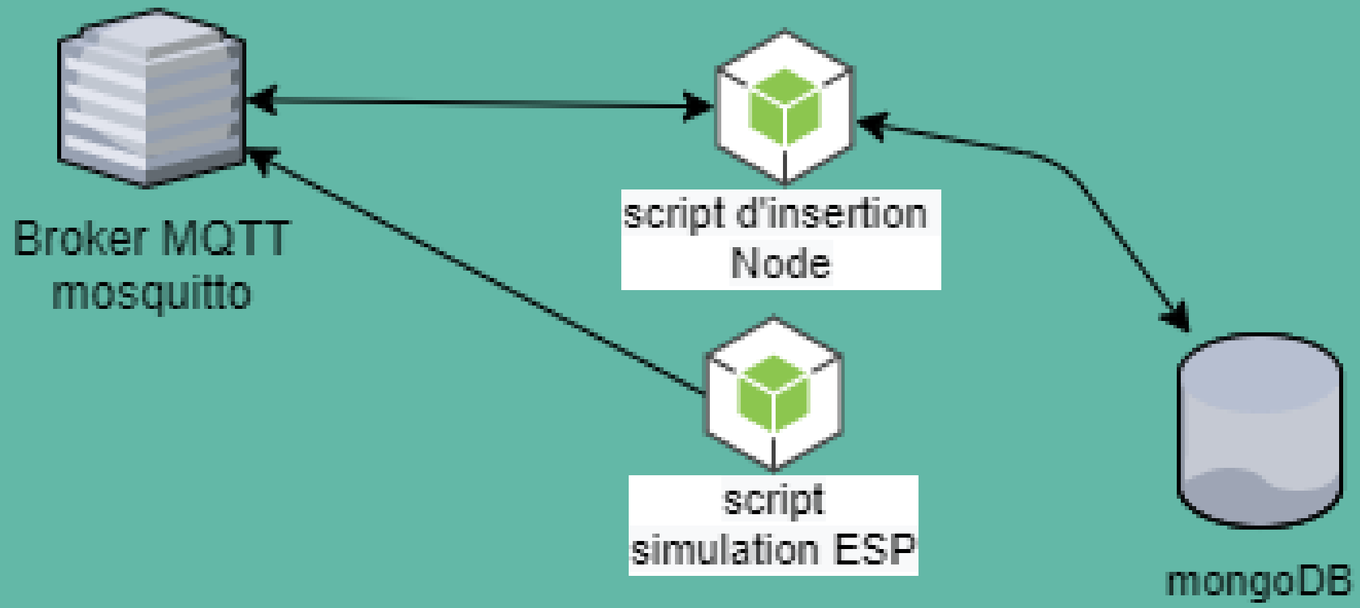
```
//GET
router.get('/', espController.getAll);
router.get('/getEsp/:id', token, espController.getEspById);

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





PROJET METEO

Jointure mqtt mongo



PROJET
METEO

Front-end

PROJET METEO

Librairies utilisés :

Leaflet

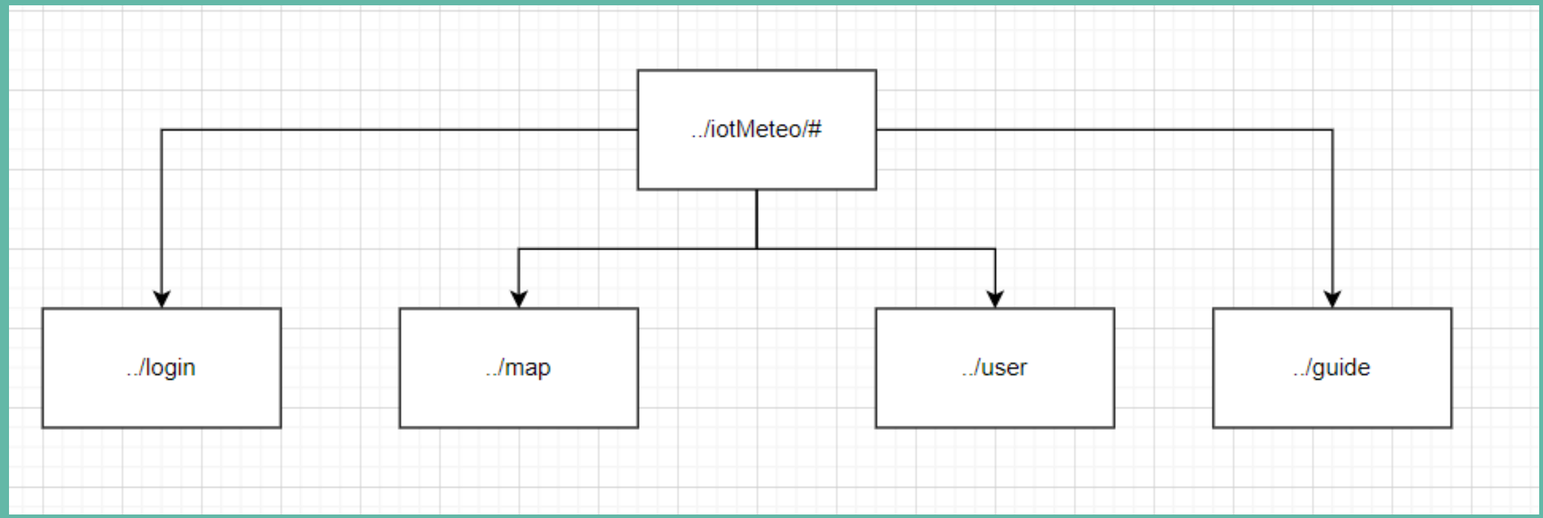
PROJET METEO

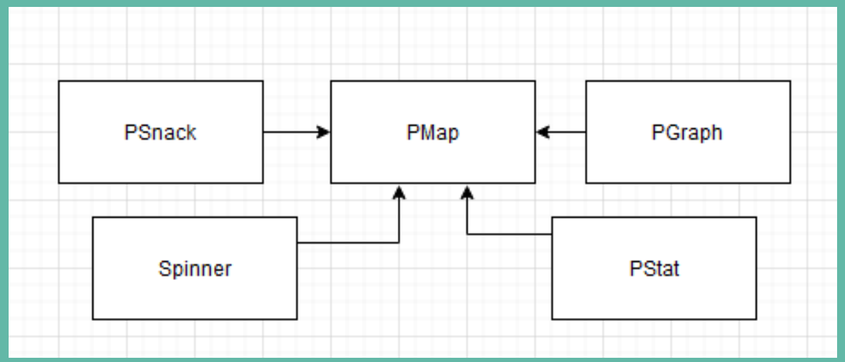
Structure d'un component

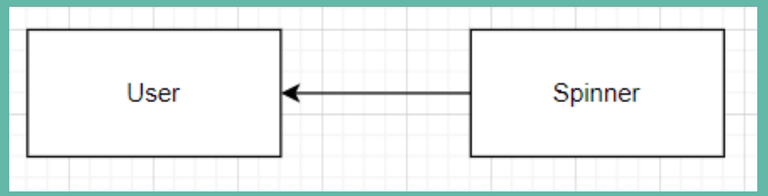
```
<template lang="jade">
div
  p {{ greeting }} World!
  other-component
</template>

<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>
```

PROJET METEO

Router





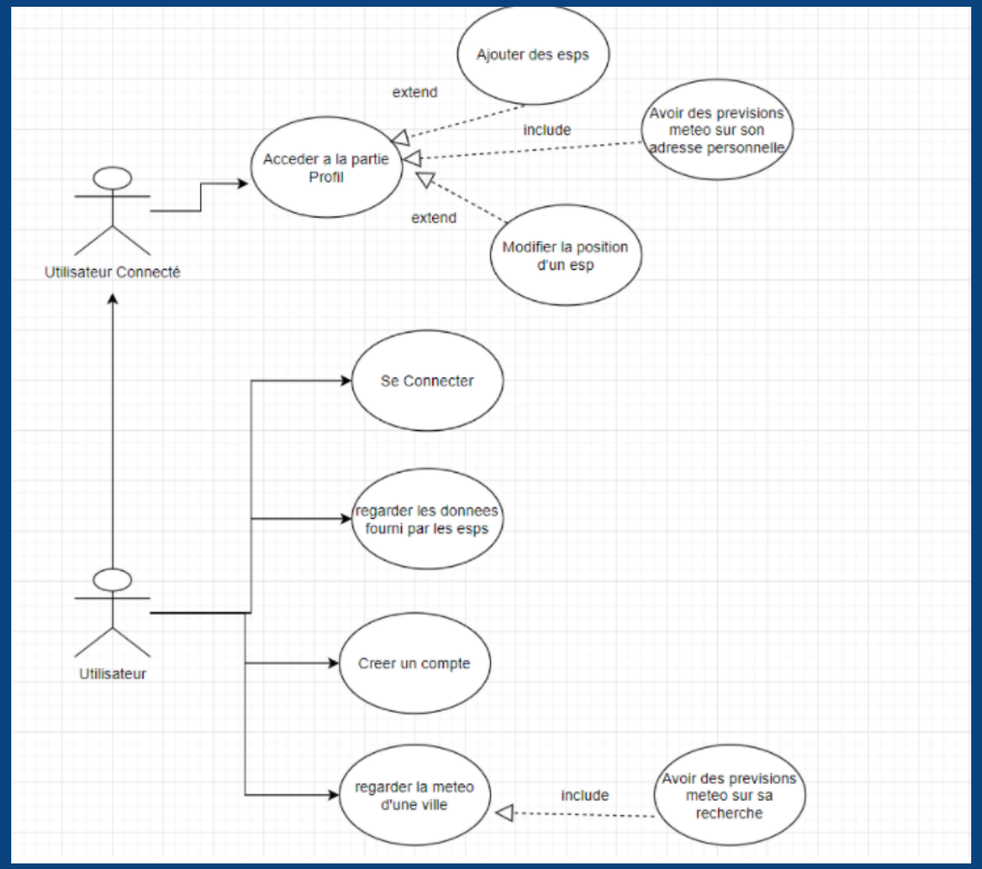


PROJET METEO

Usability de l'application

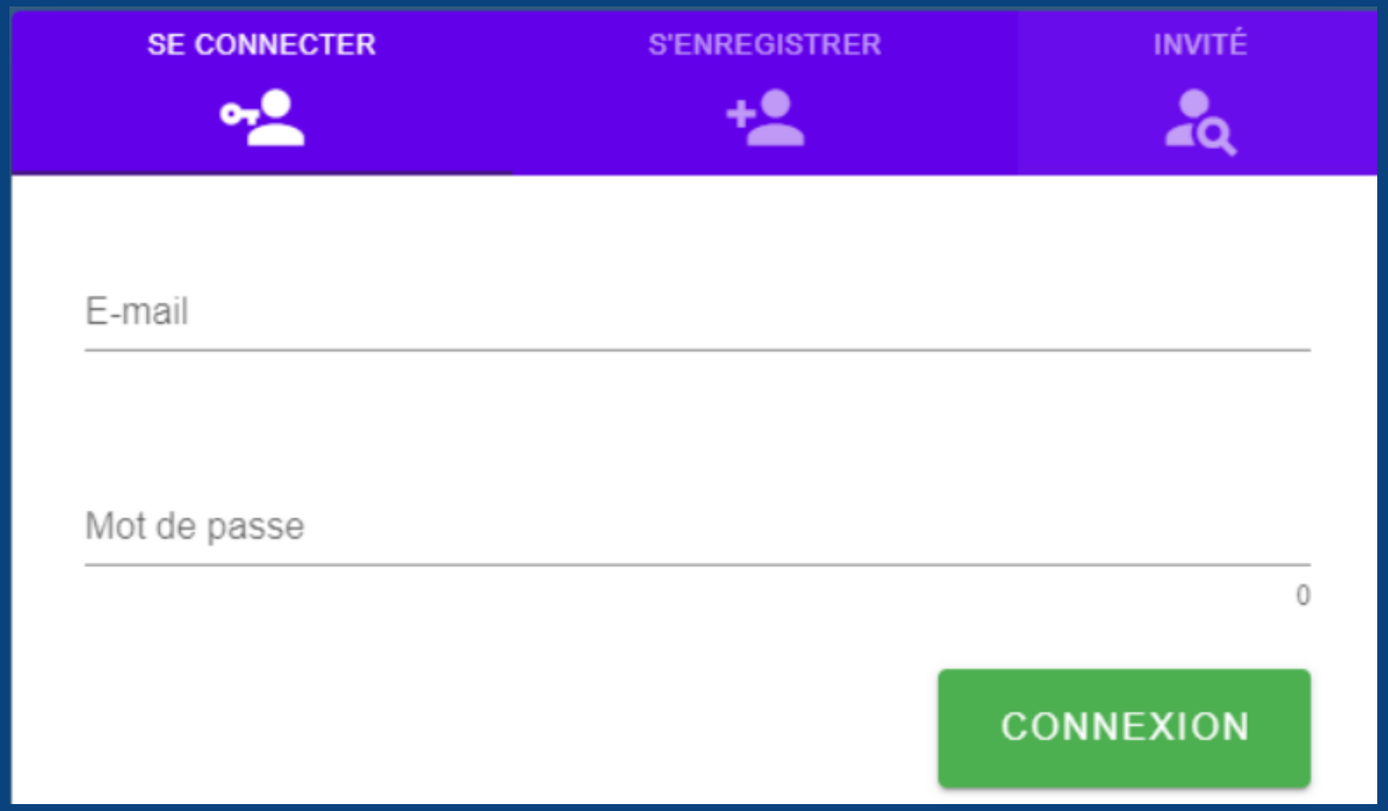
PROJET METEO

Use Case

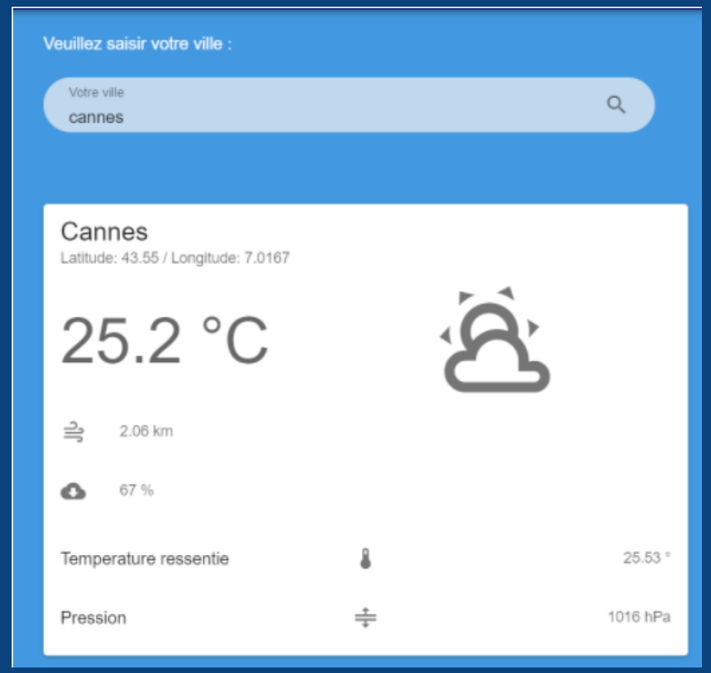


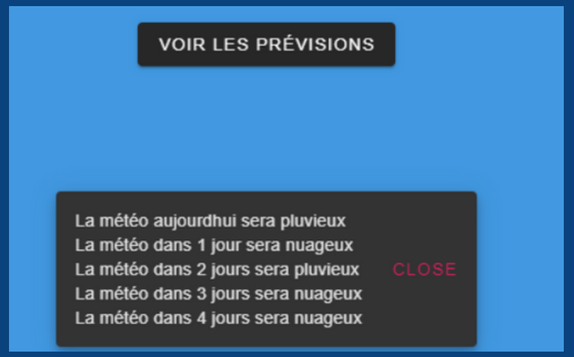
PROJET METEO

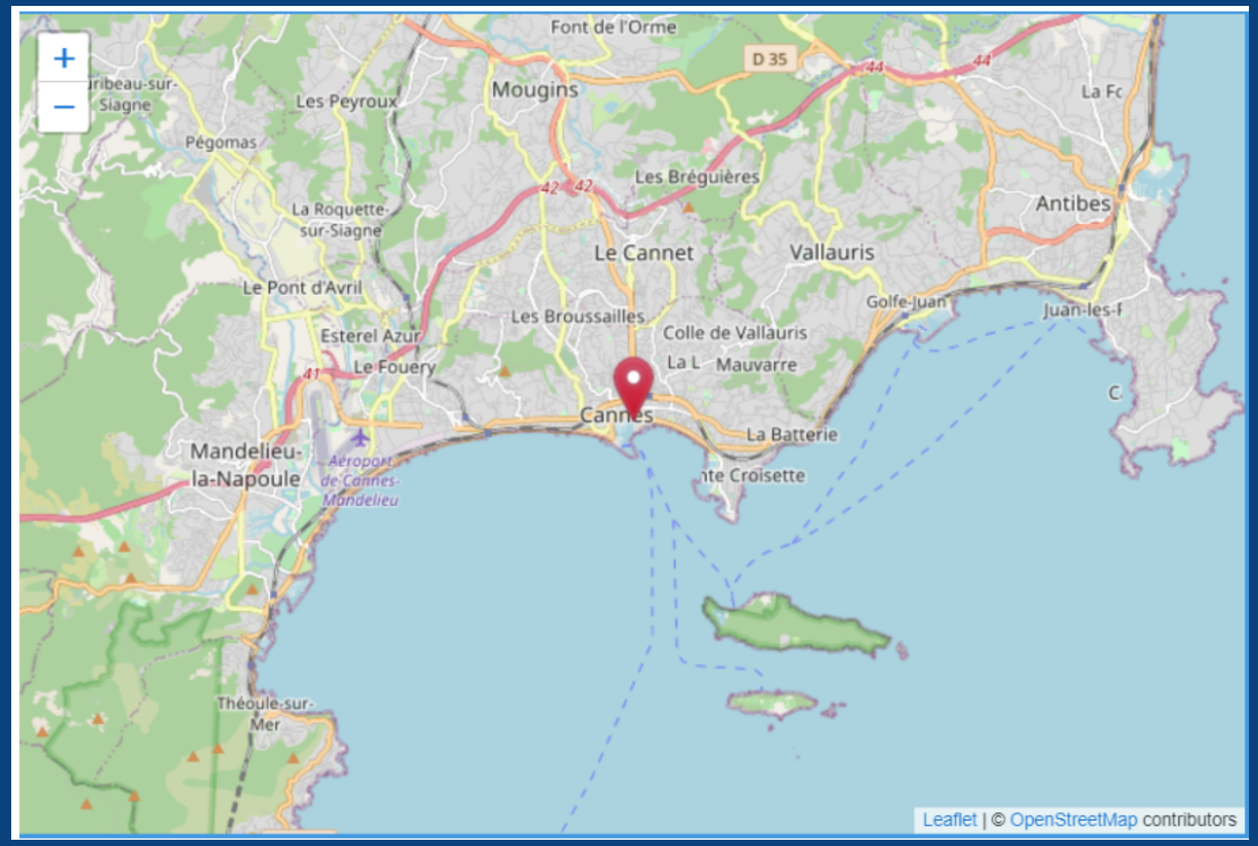
Accueil


SE CONNECTER
S'ENREGISTRER
INVITÉ
E-mail
Mot de passe
0
CONNEXION

Cas d'utilisation: Utilisateur invité

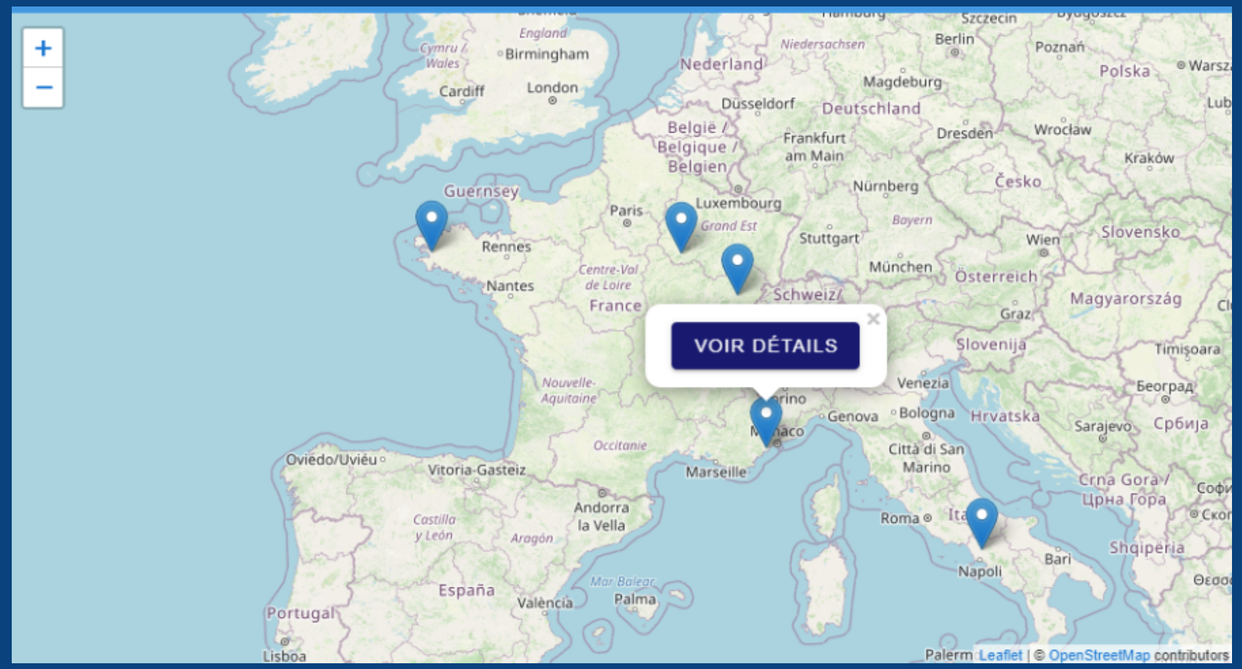






PROJET METEO

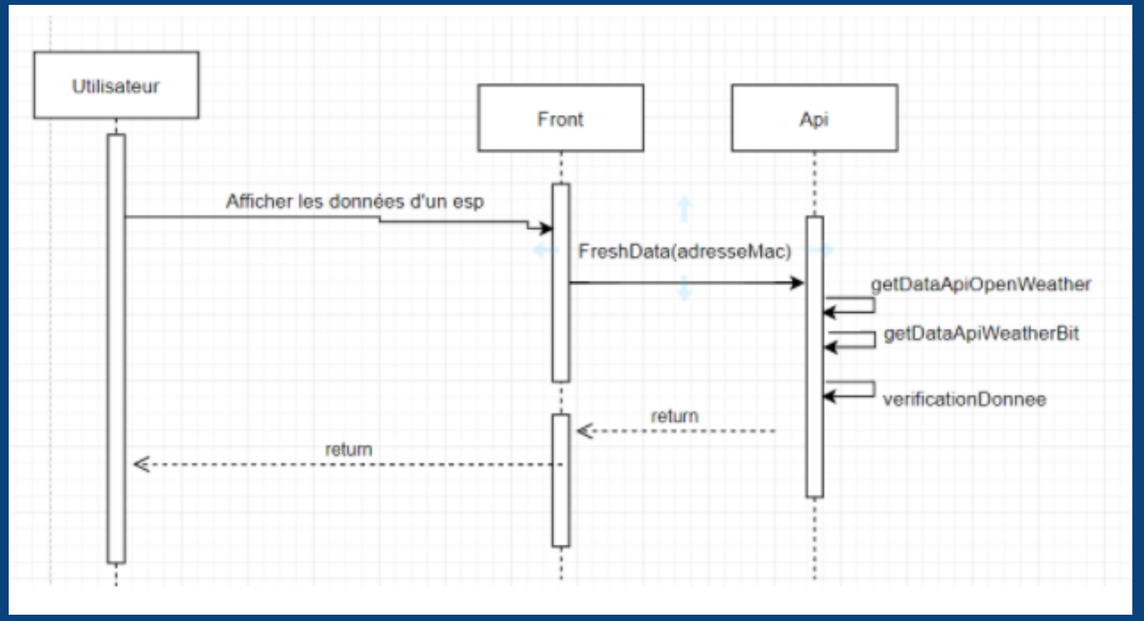
Donnee des Esp



Historique des 4 derniers jours :

Type/Date	2021-06-08	2021-06-07	2021-06-06	2021-06-05
humidite	⚠	⚠	⚠	54.07

Verification des donnees



Recuperation des données de traitement

```
//permet de récuperer les dernières datas de l'esp
const getFreshMeteoById = async (req, res) => {
    //maximum de data par requete
    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 recup
    //.sort permet de trier par date -1 signifie qu'on recup 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});
        })
}
```

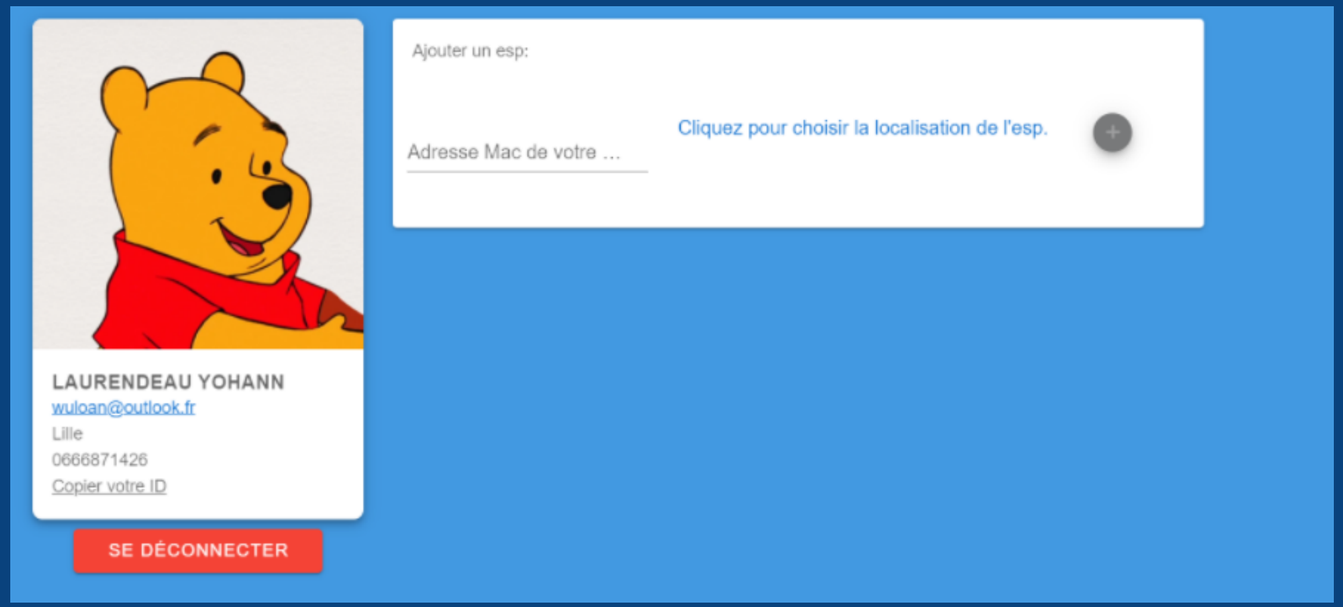
Critere d'acceptance

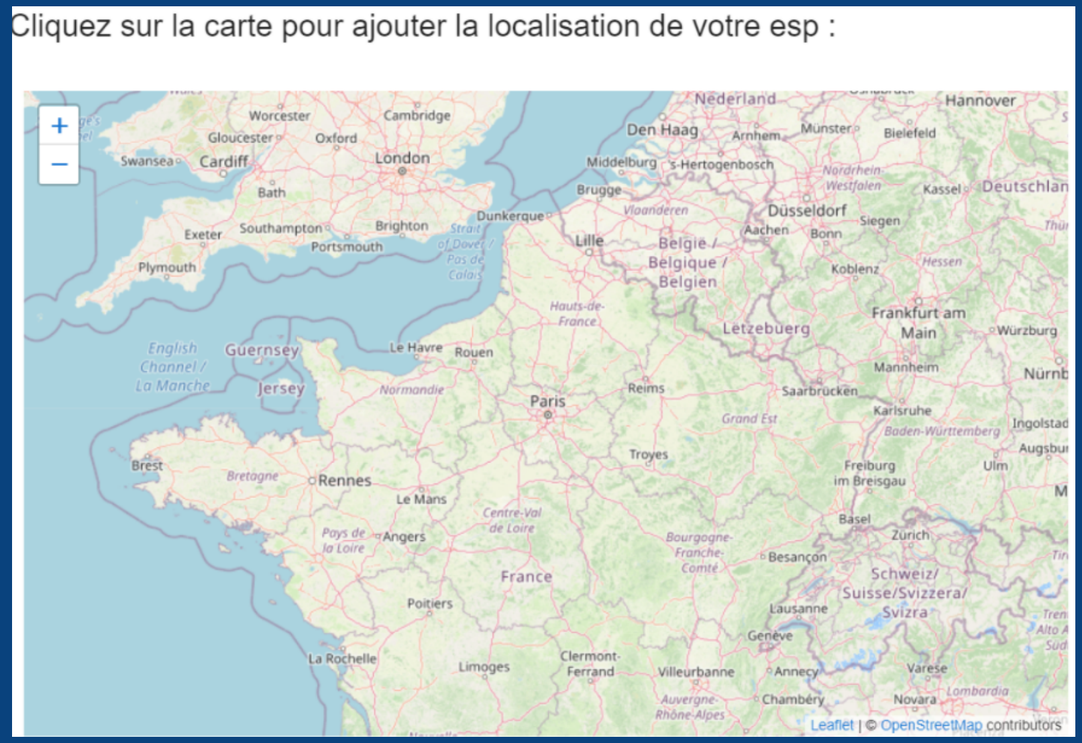
```
function testHumidity(espH1, h2) {//Permet de comparer les données d'humidité des api et celle de l'esp
    //console.log('humid : ',espH1, h2)
    return (espH1 <= h2 + 10) && (espH1 >= h2 - 10);
}

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 donnees

```
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 = "Verification indisponible pour le moment. La limite de vérification a peut-être été atteinte.";
        res.status(400).send(message);
    }
```

Cas d'utilisation: Utilisateur Connecté:





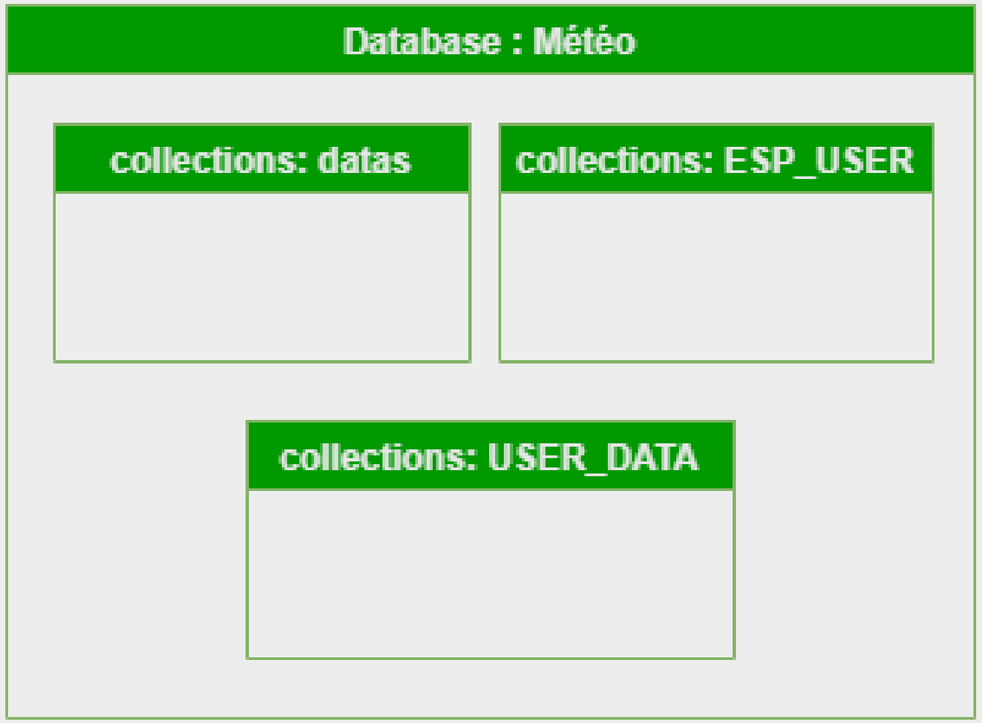

```
adressMacRules: [
  (v) => !!v || "l'adresse mac est requises",
  (v) =>
      /^([0-9A-Fa-f]{2}[:]){5}([0-9A-Fa-f]{2})$/.test(v) ||
      "L'adresse mac n'est pas valide.", //permet de tester avec un regex si l'adresse mac a un bon format
],
```

Intégration des données

MongoDB

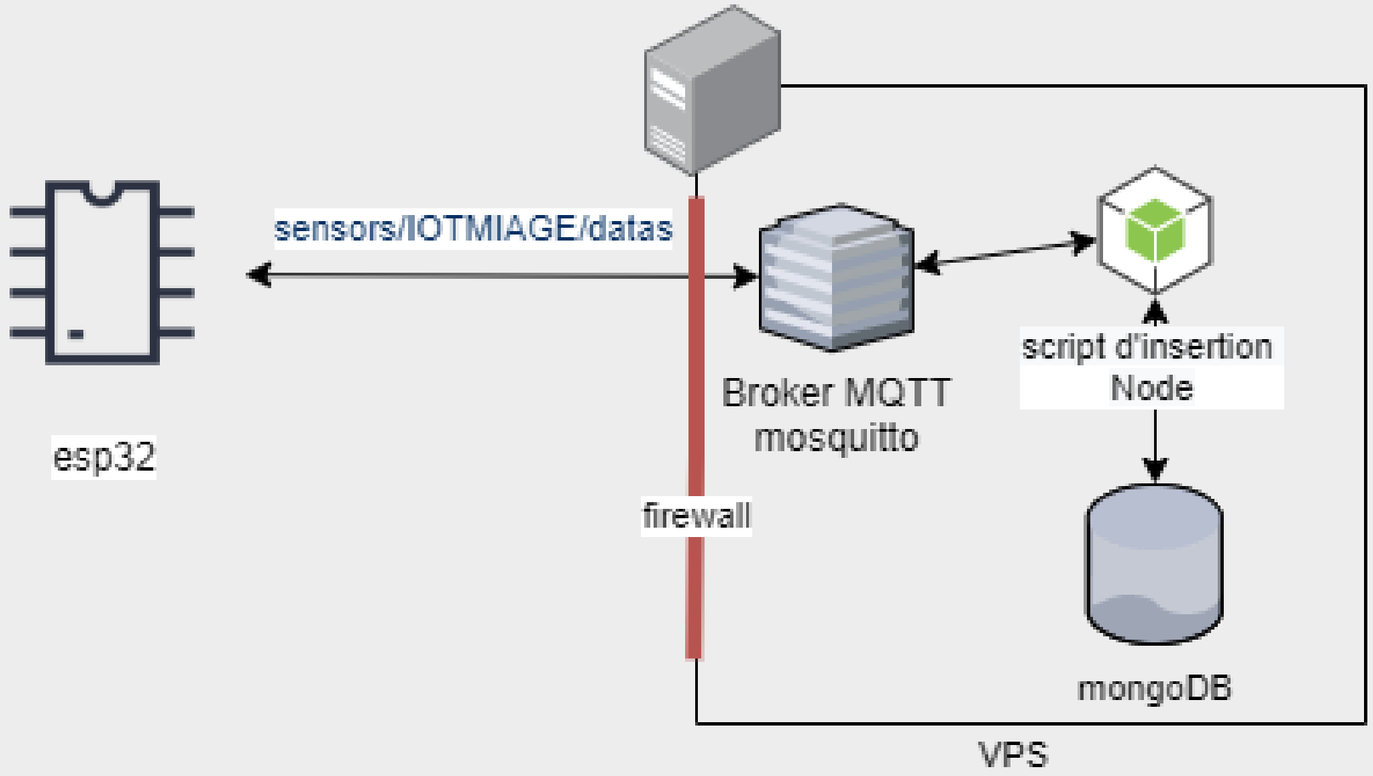
- NoSQL
- orienté Document
- Format BSON (JSON binaire)

mongoDB



PROJET METEO

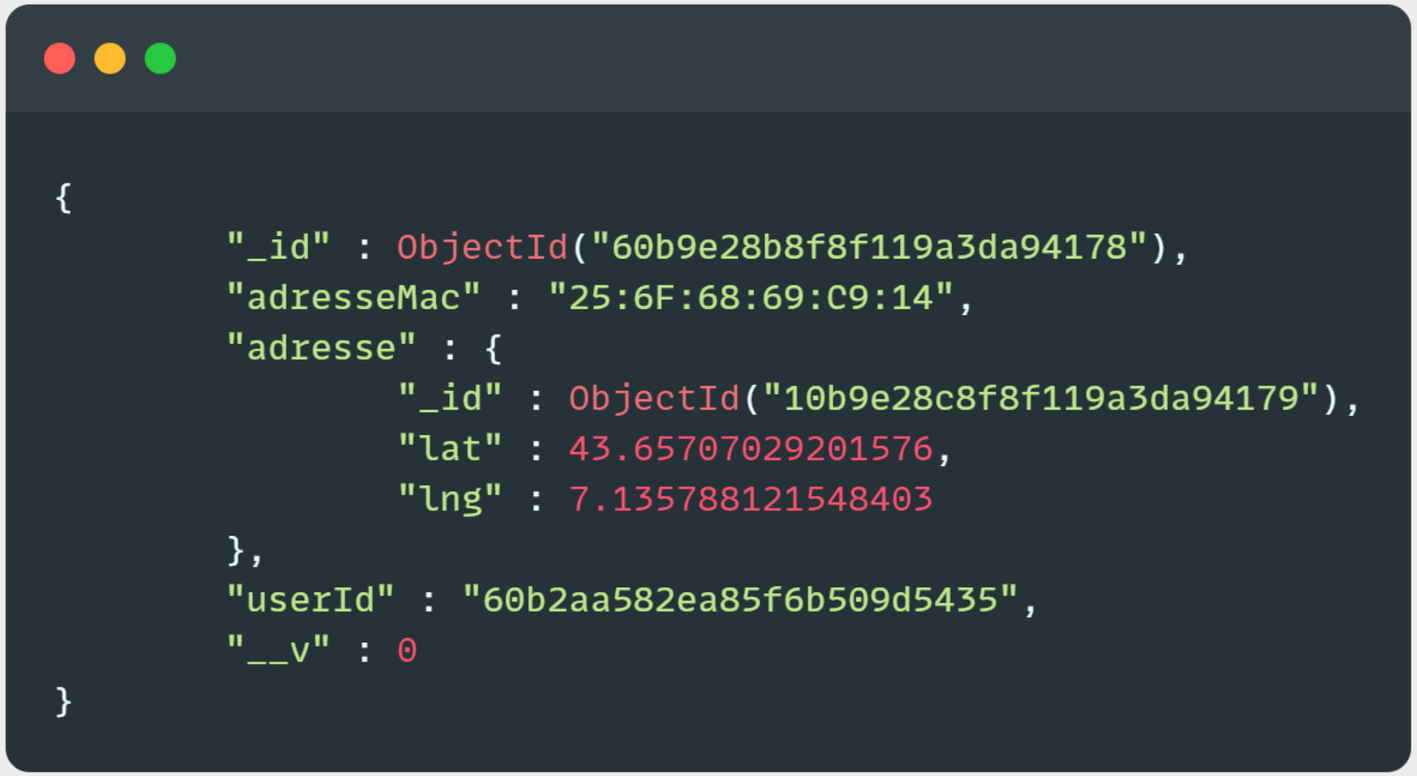
Schéma chemin des données de l'esp



PROJET METEO

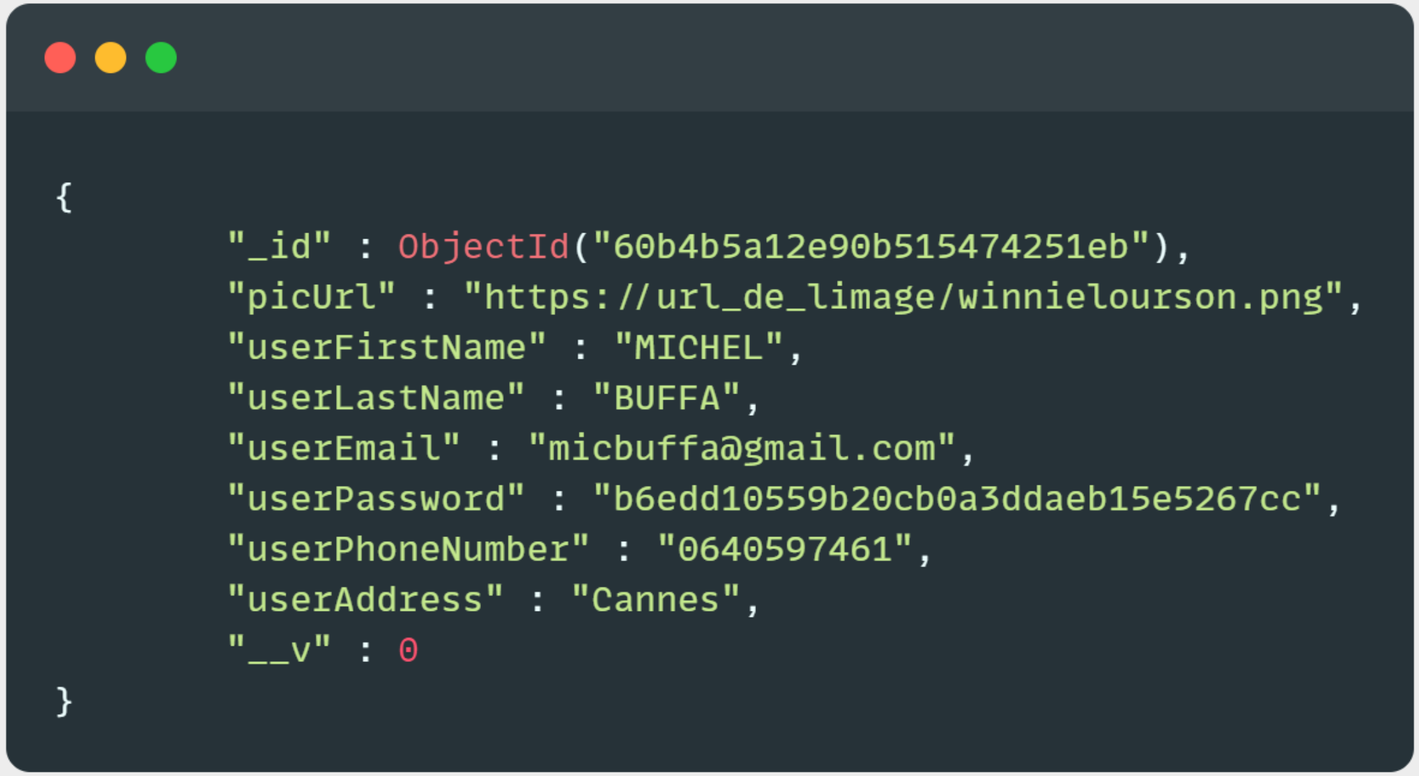
Validateur MongoDB : collection ESP_USER

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



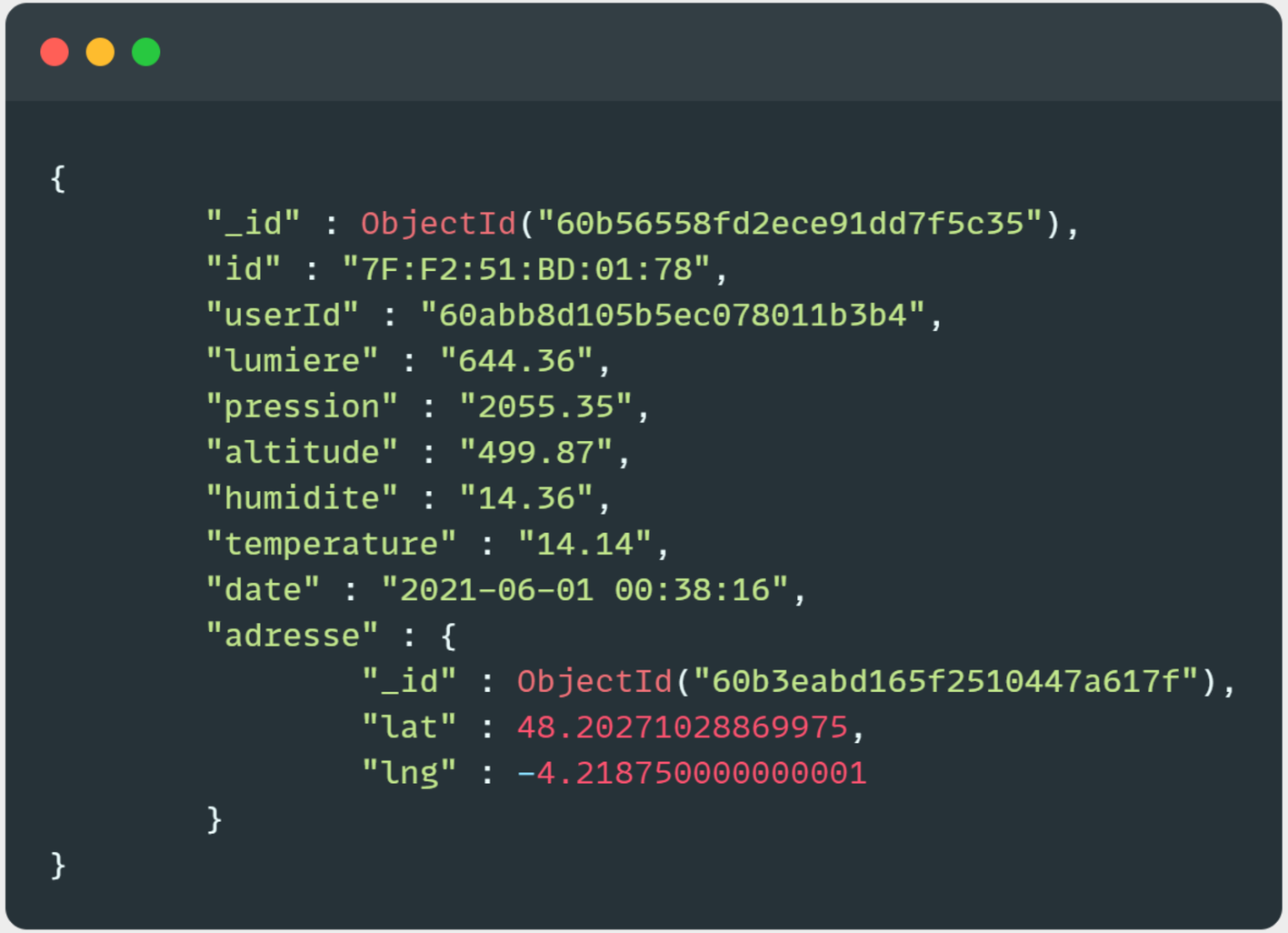
Validateur MongoDB : collection USER_DATA

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

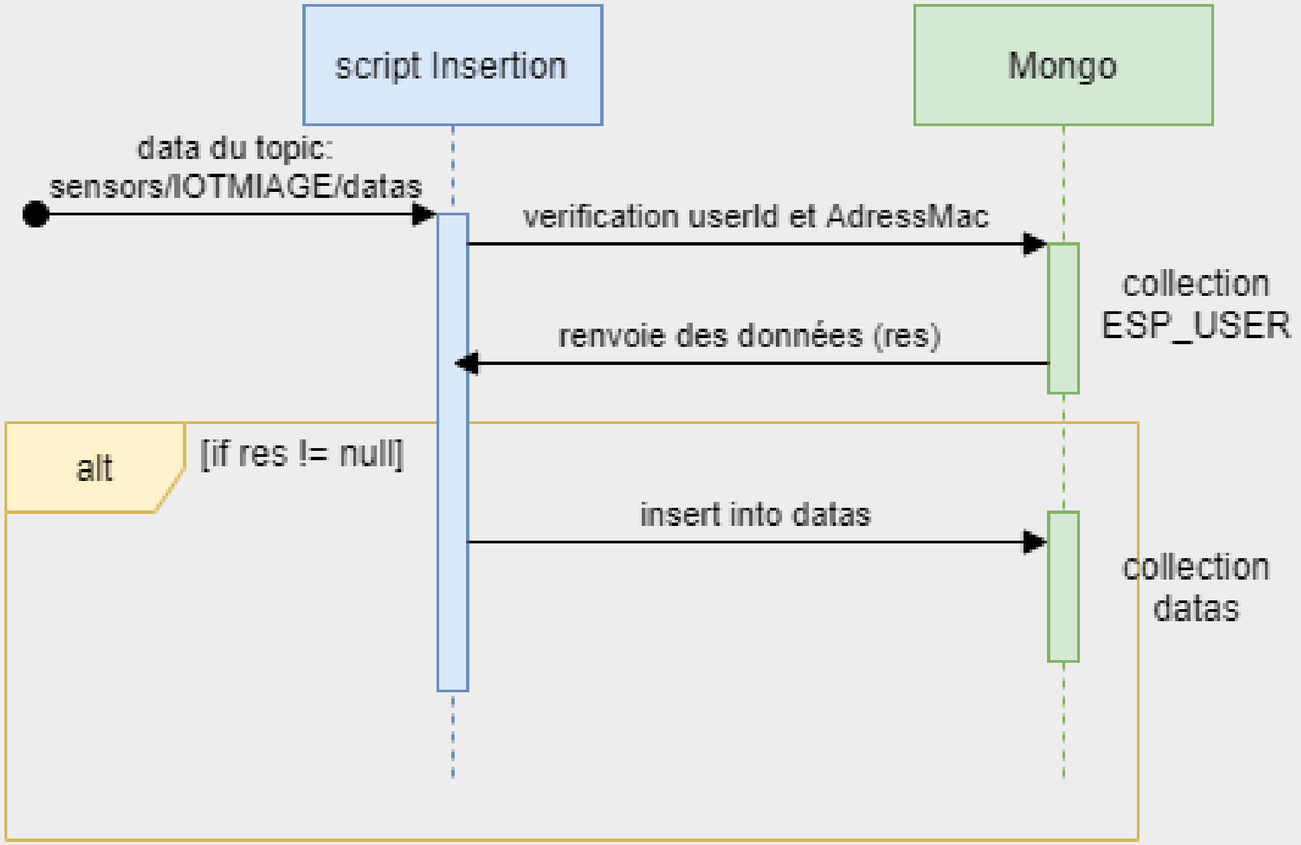


Validateur MongoDB : collection datas

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



Script d'insertion des données



PROJET METEO

API externes

OpenWeatherMap

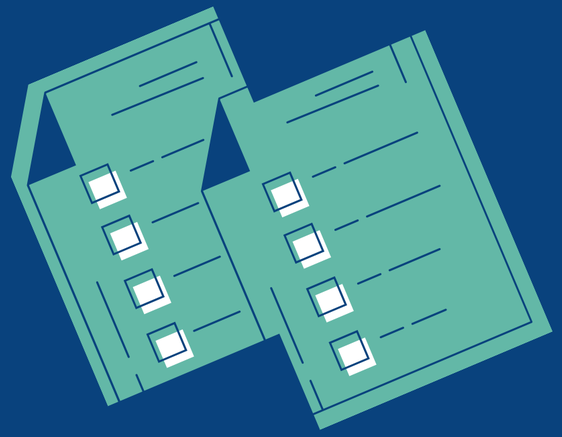
-Current Weather Data
-5 day weather forecast
-60 appels/min ou 1M/mois
-compte premium étudiant 6mois

WeatherBit

-Current Weather Data
-500 appels/jour

Script Simulation des données

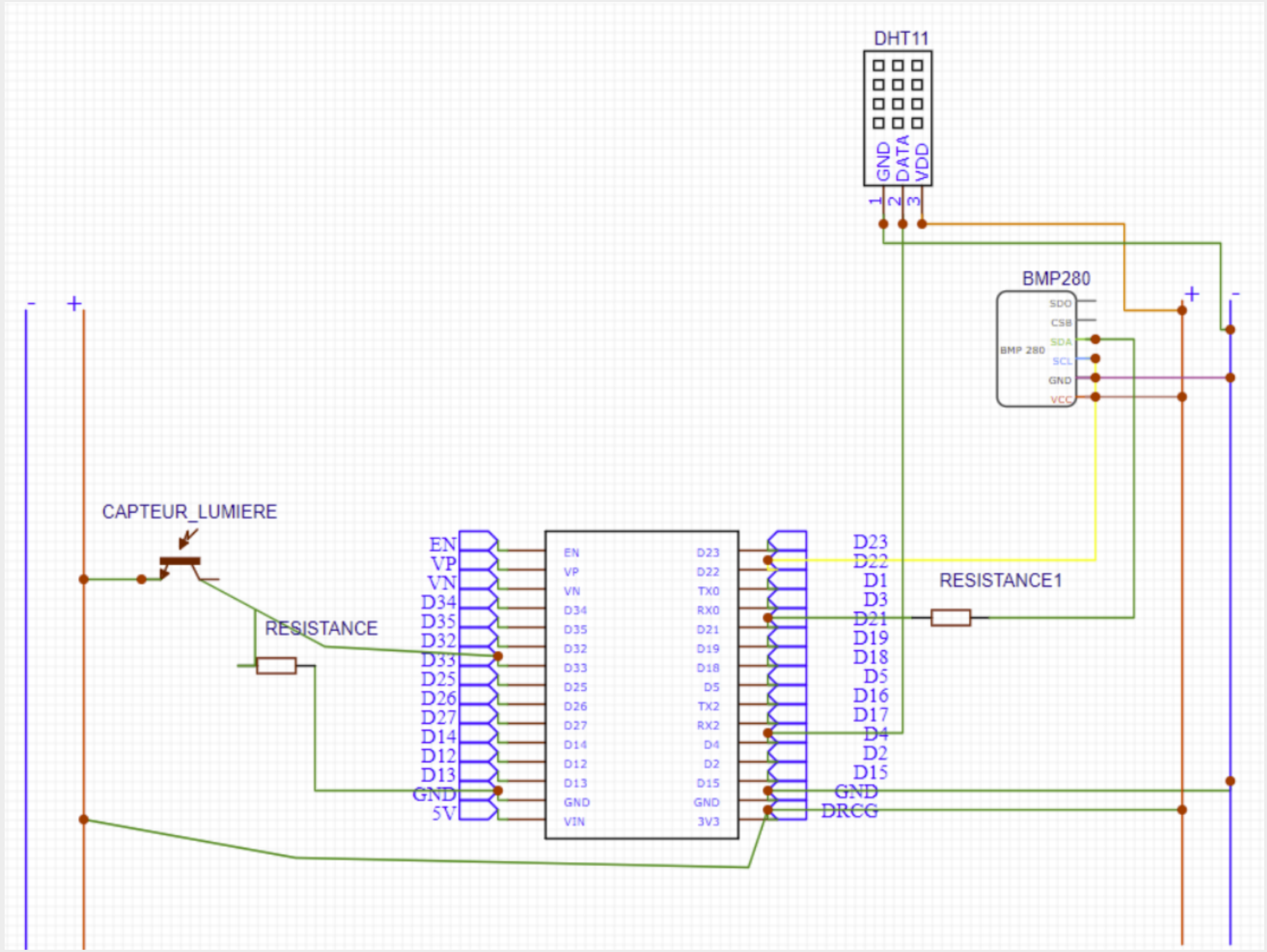
PROJET METEO



ESP 32

PROJET METEO

Schéma de l'ESP32



Initialisation 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);
```

PROJET METEO

La configuration du client MQTT:

```
WiFiClient espClient; // Wifi
PubSubClient client(espClient) ; // MQTT client
const char* mqtt_server = "ip_vps";
#define TOPIC "sensors/IOTMIAGE/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);
```

PROJET METEO

Configurer l'ESP

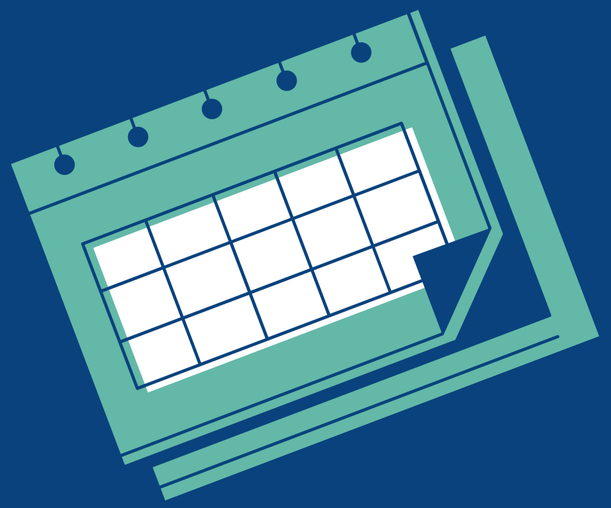
```
Button button1 = {0, 0, false};

//Quand le bouton est pressé on appel 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);
}
```

```
    void loop () {
      if (button1.pressed) {
        isEmit = !isEmit;
        configured = false;
        button1.pressed = false;
      }
      if(!isEmit)
      {
        if (!configured)
        {
          WiFi.mode(WIFI_STA);
          connect_wifi();
          configured = true;
        }
```

```
//configuration de l'esp32
        else {
          if (!configured)
          {
          //on emmet du wifi
            Serial.print("Setting AP (Access Point)…");
            WiFi.softAP("Esp32", NULL);
            IPAddress IP = WiFi.softAPIP();

            Serial.print("AP IP address: ");
            Serial.println(IP);
              //on envoie la page web au client
            server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
              request->send_P(200, "text/html", index_html, processor);
            });
           //Quand le client a sauvegardé ses datas on les recoit ici
            server.on("/get", HTTP_GET, [] (AsyncWebServerRequest *request) {
              String inputMessage;
              int inputMessageFreq;
              String inputParam;
      //pour chaque if on enregistre le paramètre dans la mémoire morte si le paramètre a bien été changé
              if (request->hasParam(SSID)) {
                inputMessage = request->getParam(SSID)->value();
                if (inputMessage[0] != '\0')
                  {

                inputParam = SSID;

                preferences.begin("credentials", false);
                preferences.putString("ssid", inputMessage);
                Serial.println(inputParam);
                preferences.end();
              }
              }
```

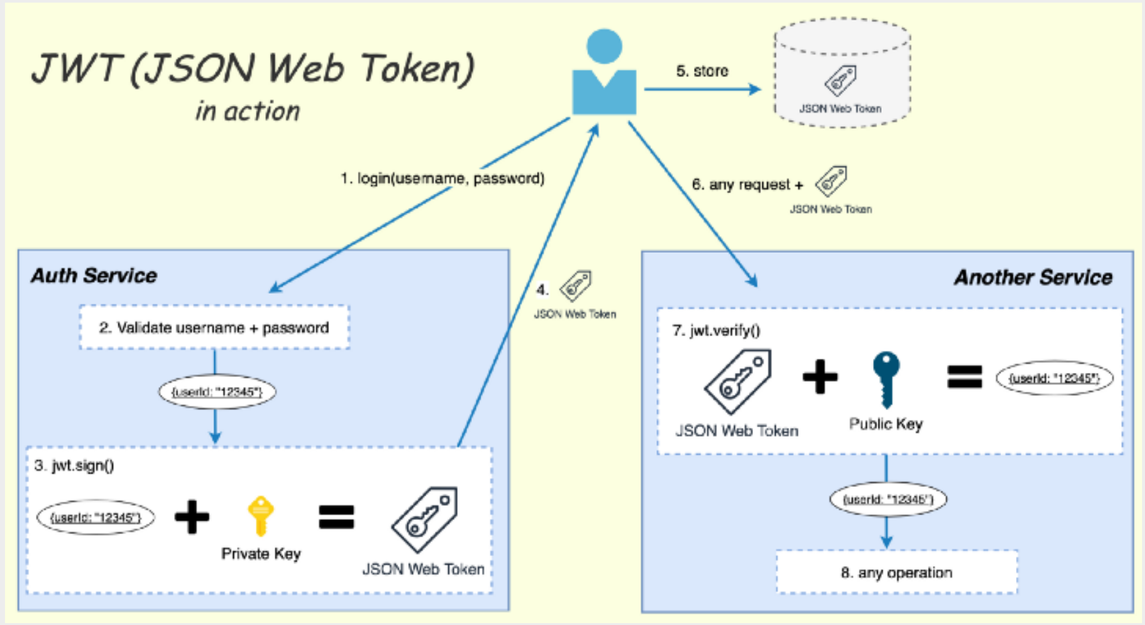
```
#define ONBOARD_LED  2
  digitalWrite(ONBOARD_LED,1);
```

Sécurité des flux de données de notre api:



PROJET METEO

JWT: JSON Web Tokens



PROJET METEO

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;
```

PROJET METEO

Securisation des routes

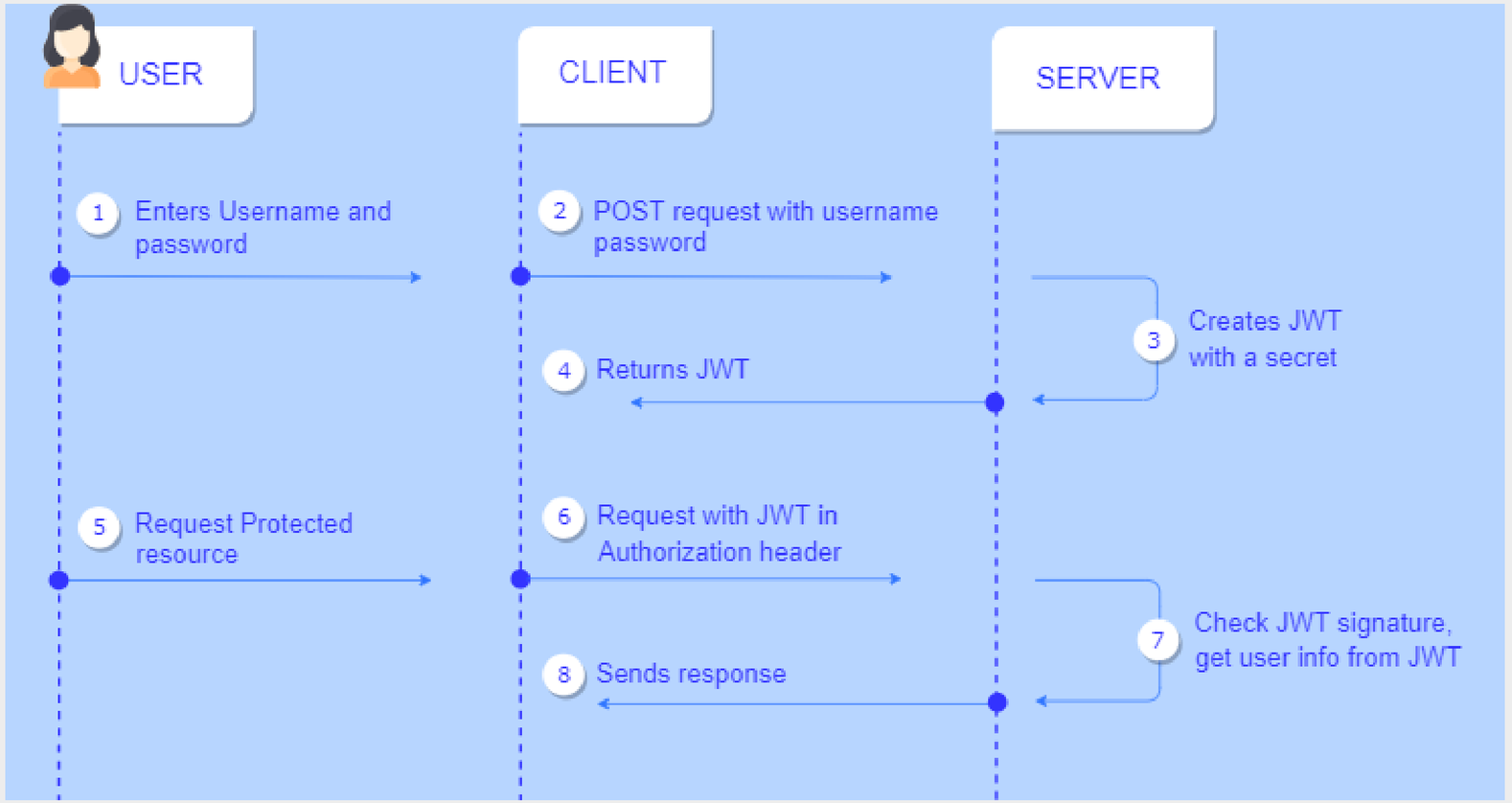
Exemple:

```
//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)
```

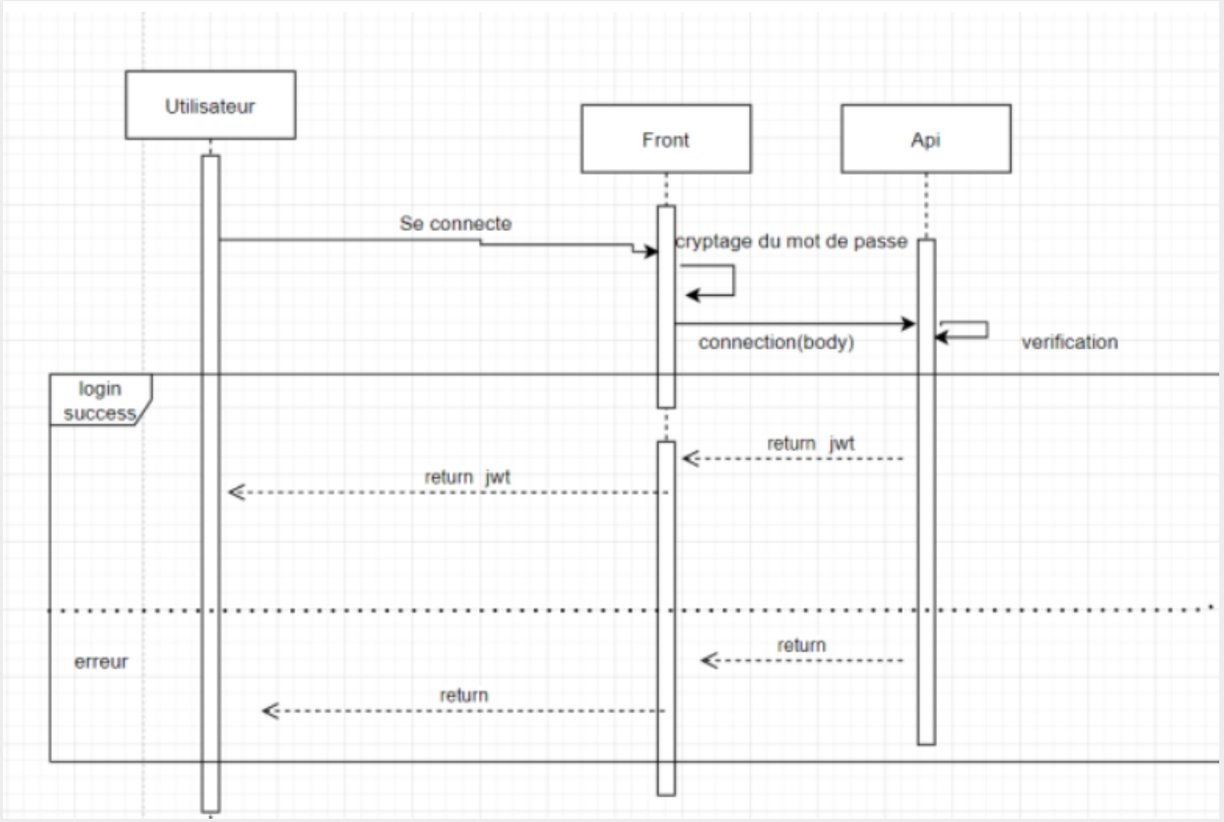
Mise en place du header

```
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 àa récupèrer les données pour ne pas géner l'utilisateur
        this.showSpinner = false;
      });
},
```

Connexion

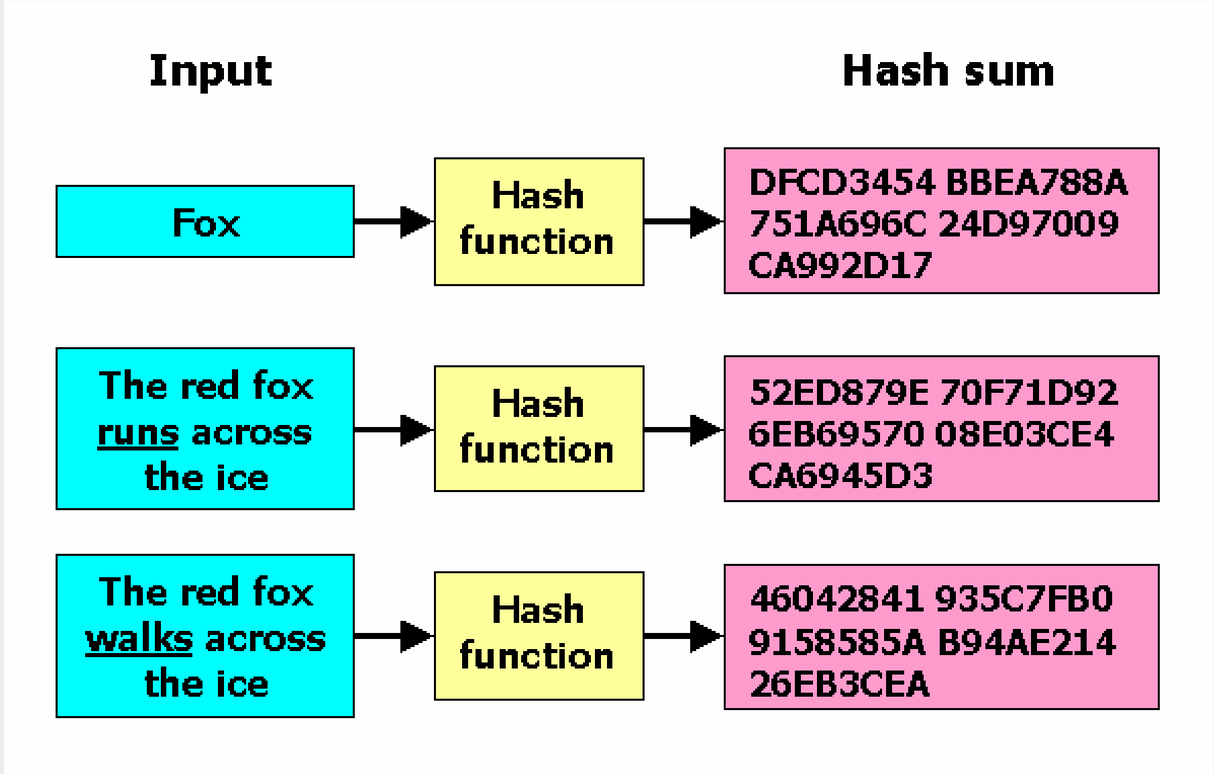

USER
CLIENT
SERVER
1 Enters Username and password
2 POST request with username password
3 Creates JWT with a secret
4 Returns JWT
5 Request Protected resource
6 Request with JWT in Authorization header
7 Check JWT signature, get user info from JWT
8 Sends response

Connexion



PROJET METEO

MD5


Input
Hash sum
Fox
Hash function
DFCD3454 BBEA788A 751A696C 24D97009 CA992D17
The red fox runs across the ice
Hash function
52ED879E 70F71D92 6EB69570 08E03CE4 CA6945D3
The red fox walks across the ice
Hash function
46042841 935C7FB0 9158585A B94AE214 26EB3CEA

Sécurité du VPS : Pare-feu

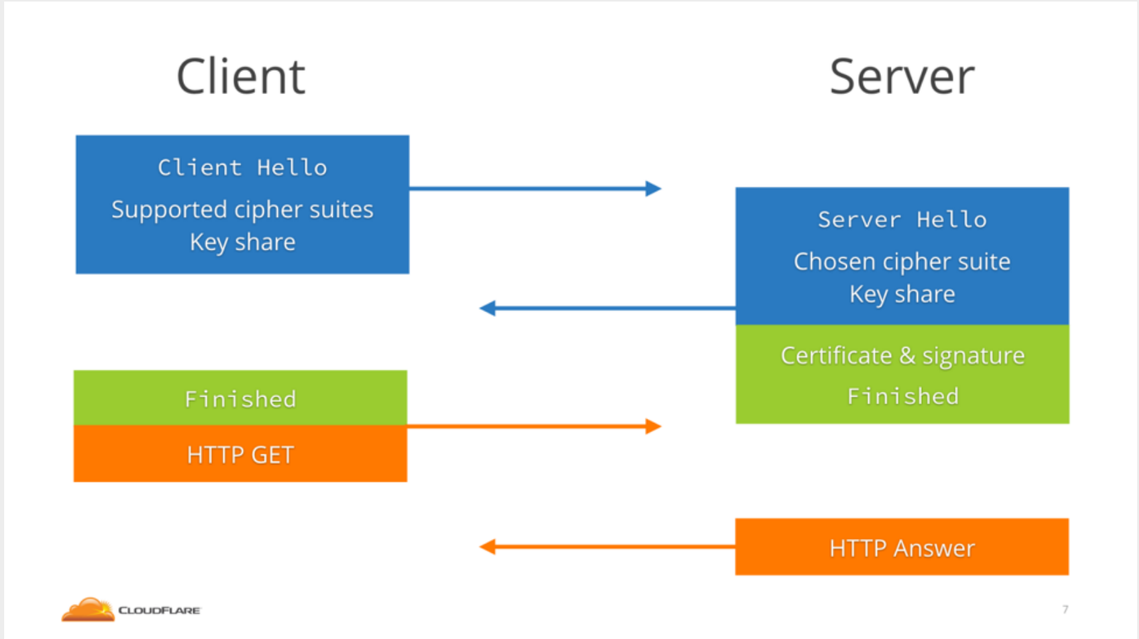
```
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

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)
```

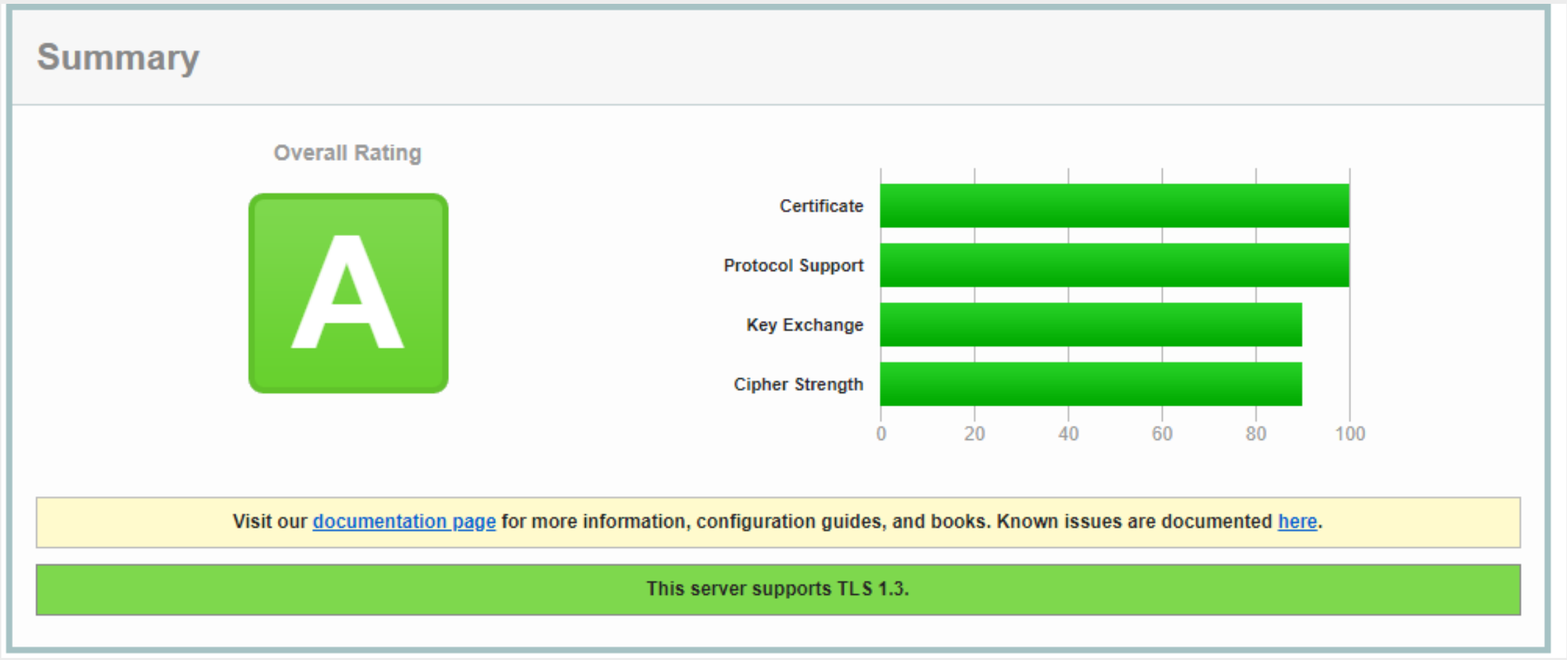
- UFW (Uncomplicated Firewall)
- Fail2Ban

PROJET METEO

Sécurité du VPS : certificats SSL/TLS

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

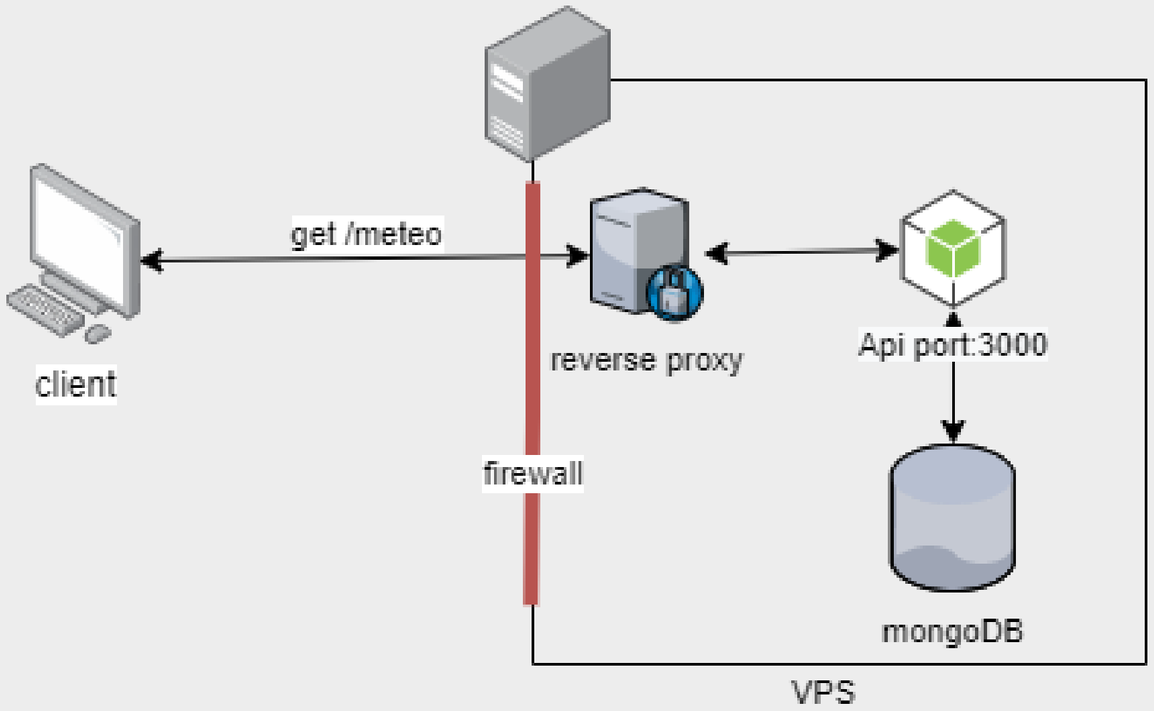




www.ssllabs.com

PROJET METEO

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

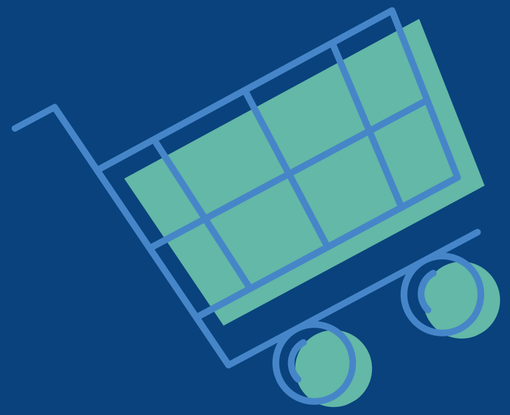
PROJET METEO

Sécurité du VPS : Broker

Connections anonymes interdites

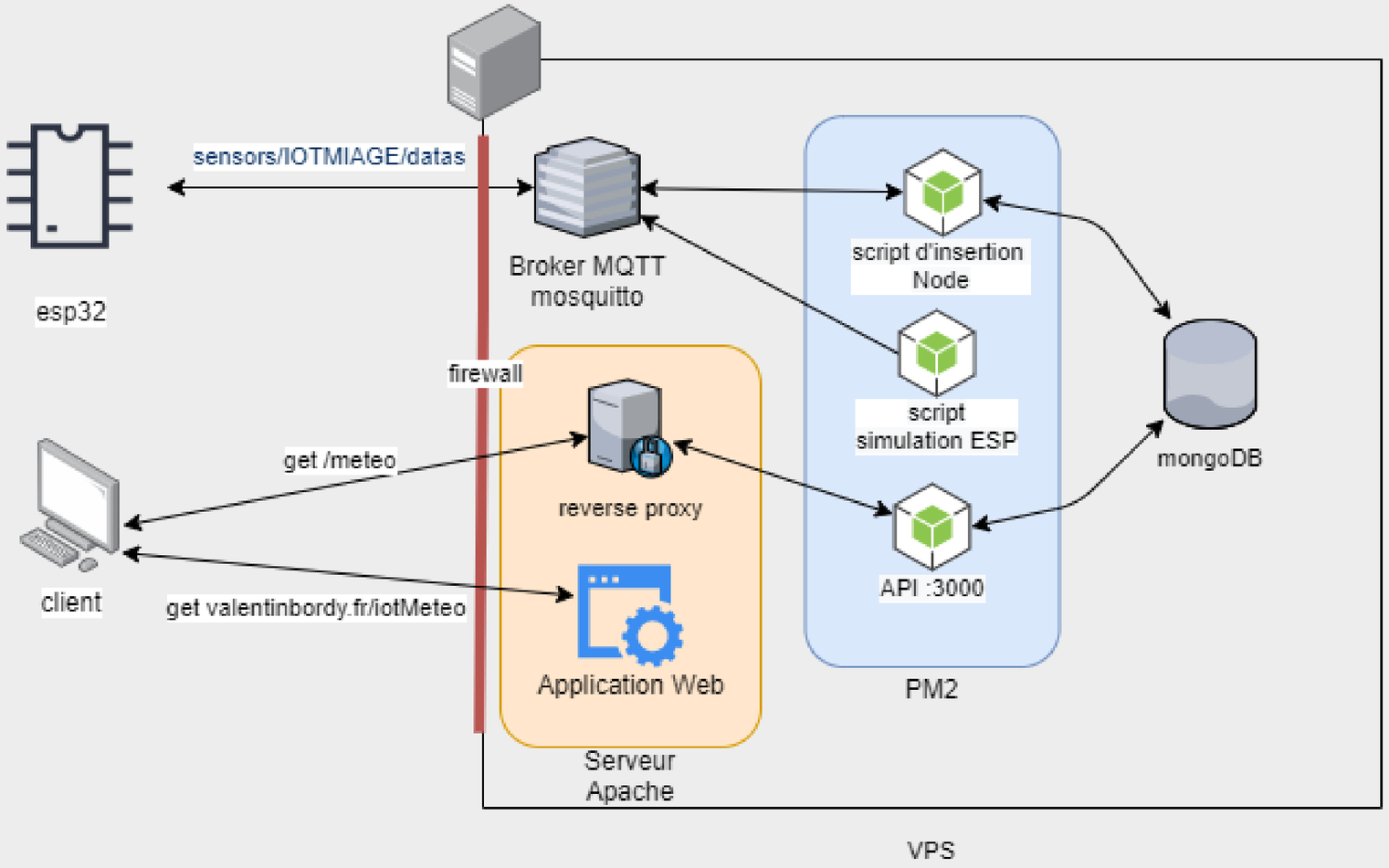
Connections obligatoires. Utilisateur/motDePasse générique

PROJET METEO



Déploiement

Rappel architecture



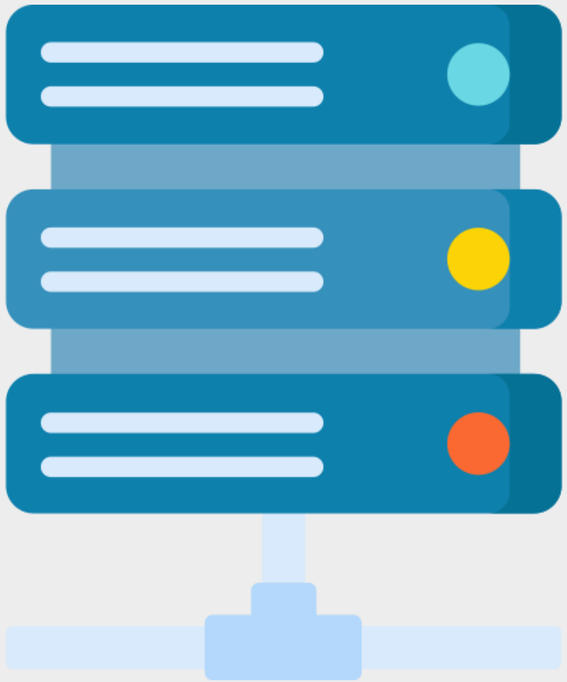
Vps:
-parefeu
-pm2
-mongoDB
-Apache
-Broker

PROJET METEO

Pourquoi un vps et pas Heroku ?

heroku

VS



PROJET METEO

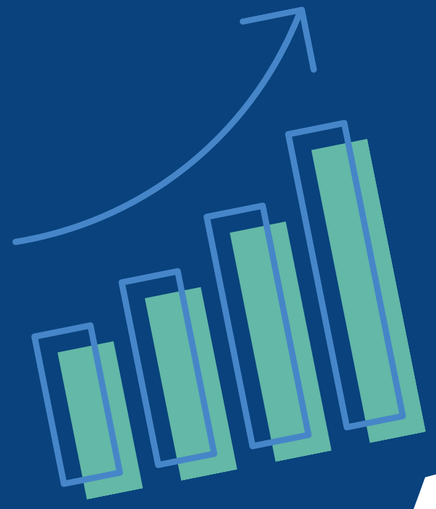
Amélioration futures pour le déploiement

- Une seule route: /meteo
- Fichier .env

.ENV

valentinbordy.fr/meteo

PROJET METEO



Améliorations possibles

PROJET METEO

Les Améliorations

- Lors du changement d'adresse ne plus afficher les données ultérieurs de l'ESP.

- Améliorer la précision des prévisions.

- Modification du profil.

- Amélioration de la liaison ESP-USER.

- Amélioration du déploiement.

PROJET METEO

Conclusion

PROJET METEO

Merci pour votre attention

PROJET METEO

Free Resources

Use these free, recolourable icons and illustrations in your Canva design.