20/12/2023

# Rapport de projet IoT

*skatepark intelligent*

Christoph Samuel, Mouche Valentin

# Rapport de projet IoT

*skatepark intelligent*

# I) Introduction

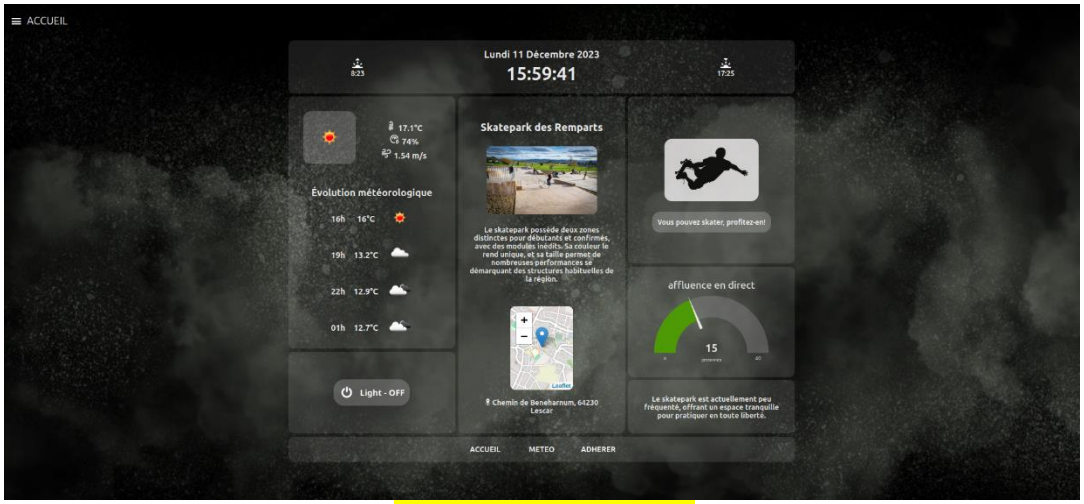
Ce projet a été réalisé dans le cadre du **cours d'IoT** dans le but de développer un projet via **Node-RED,** une plateforme de développement visuel pour les applications IoT. En l'absence de directives spécifiques, nous avons choisi de créer un **tableau de bord (dashboard) interactif** axé sur la **surveillance météorologique**, la **gestion de l'éclairage, la prise en compte de l'affluence…** afin **d'améliorer l'expérience des skateurs.**

Les défis liés à l'affluence, aux conditions météorologiques imprévisibles, aux risques climatiques et à l'organisation d'événements ont stimulé **la création d'un système** pour répondre aux besoins et aux **contraintes de cet environnement,** afin d'en informer les **personnes concernés.**

# II) Présentation du projet

## 1) page ACCUEIL

La page d'accueil du dashboard offre **une interface complète pour les utilisateurs**. Elle est composée de plusieurs **groupes fonctionnels**, c'est la première interface à laquelle les utilisateurs ont accès lorsqu'ils interagissent avec notre projet. Elle offre une expérience complète et pratique pour **faciliter la décision** des skateurs **quant à leur venue au skatepark**.



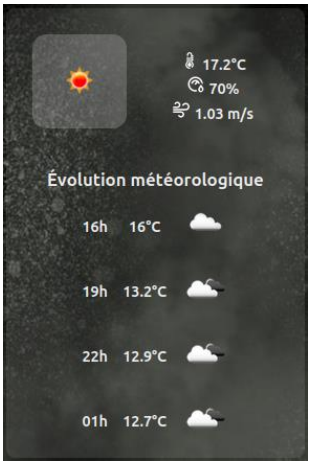

page d'accueil du dashboard

### *a) group « top »*

Ce groupe est **la première section visible** sur la page d'accueil du dashboard, il affiche **en temps réel** des données telles que **l'heure, la date,** ainsi que **l'heure de lever** et de **coucher du soleil** du jour, aidant les utilisateurs à planifier leur visite au skatepark de manière efficace.

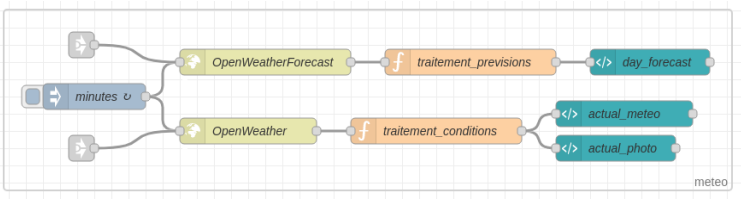
## *b) group « meteo »*

Ce groupe utilise **l'API OpenWeather** pour présenter les données **météorologique actuelles** (tendance, humidité, température et vent**) par tranche de 3 heures** pour la **journée**, offrant aux utilisateurs une perspective avant de se rendre au skatepark.



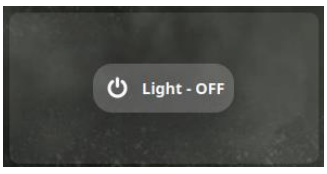

```
const forecastData = msg.payload.list;
const displayData = [];

forecastData.slice(0, 4).forEach(entry => {
    if (entry.main && entry.main.temp_min) {
        const temperatureMin = (entry.main.temp_min - 273.15).toFixed(1);

        let iconUrl = '';

        if (entry.weather && entry.weather[0] && entry.weather[0].icon) {
            iconUrl = `https://openweathermap.org/img/w/${entry.weather[0].icon}.png`;
        }

        const formattedTime = new Date(entry.dt * 1000)
            .toLocaleTimeString('fr-FR', { hour: 'numeric', hour12: false })
            .replace(':', 'h')
            .replace(/\s/g, '');

        displayData.push({time:formattedTime, temperatureMin:parseFloat(temperatureMin), iconUrl:iconUrl});
    }
});

msg.payload = displayData;
return msg;
```

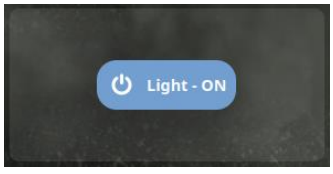
**Toutes les minutes** un signal est envoyé puis récupéré afin **de mettre à jours les données affichées** sur notre dashboard, dans notre cas nous utilisons **deux API différentes** car une se base sur les **données actuelles** et l'autre sur les **prévisions de la journée**.




## *c) group « lumiere »*

Ce groupe assure un **contrôle intelligent de l'éclairage** du skatepark en fonction de **l'heure de la journée** et de **la fermeture du skatepark**. Il dispose d'un **bouton d'état** indiquant si les lumières du skatepark sont **allumées ou éteintes.**



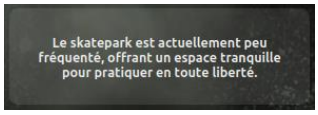

Les lumières sont programmées pour **s'allumer automatiquement au coucher du soleil** et **s'éteindre à minuit**, réduisant ainsi la consommation et évitant les inconvénients liés à un éclairage nocturne excessif et sans perturber les secours en cas d'incident.

```
var currentTimeStamp = Math.floor(Date.now() / 1000);

var midnightTimeStamp = new Date();
midnightTimeStamp.setHours(0, 0, 0, 0);

var sunsetTime = msg.payload.sys.sunset;
var sunsetDate = new Date(sunsetTime * 1000);

if (currentTimeStamp > sunsetTime) {
  msg.payload = { isNight: true };

} else {
  msg.payload = { isNight: false, currentTimeStamp };
}

return msg;
```

## *d) group « affluence»*

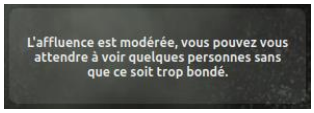
Ce groupe est conçu pour **informer les utilisateurs** sur le **niveau d'affluence actuel** du skatepark, basé sur **une simulation de capteur virtuel**. Il comprend une **jauge** affichant le niveau d'affluence en temps réel ainsi qu'un **message** qui indique si l'affluence est **faible, modérée ou élevée**. Ces niveaux sont déterminés en fonction des **seuils préétablis adaptés à la taille et à la capacité du skatepark.**


nbPersonne < 15





15 < nbPersonne < 25



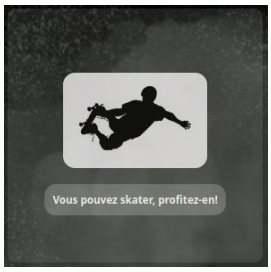


nbPersonne > 25


La **couleur de la jauge varie** en fonction du nombre de personnes présentes. Cette variation des couleurs offre aux utilisateurs **une indication visuelle immédiate** du niveau d'affluence, les aidant à décider s'ils **veulent se rendre au skatepark ou non**.

## *e) group « conseil »*

Ce groupe offre des **recommandations claires**, utilisant un **code couleur (blanc, gris ou noir)** pour indiquer **les conditions actuelles du skatepark** (conseillé, pas recommandé ou défavorable). Les conseils sont **accompagnés de messages** expliquant **les raisons derrière chaque recommandation**.

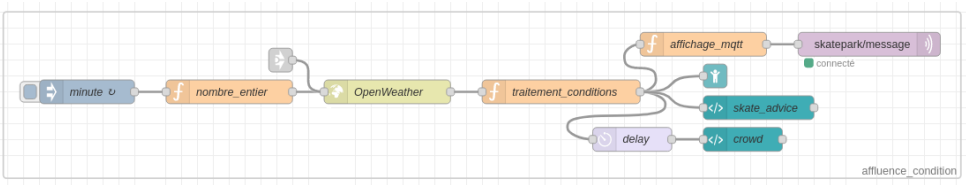





**Par exemple,** en cas de **pluie** ou d'humidité élevée, le message avertit que le **sol** du skatepark pourrait être **mouillé**, suggérant **de ne pas skater dans de telles conditions**. Cette présentation visuelle facilite les utilisateurs à **évaluer les risques potentiels** et à prendre des décisions sur leur sortie.

```
var humidity = msg.payload.main.humidity;
//var humidity = 95;
//var humidity = 50;
var temperature = msg.payload.main.temp - 273.15;
//var temperature = 0;
//var temperature = 20;
//var temperature = 40;
var weatherCondition = msg.payload.weather[0].main;
//const weatherCondition = "Rain";
//const weatherCondition = "Sun";
var windSpeed = msg.payload.wind.speed;
//var windSpeed = 5;
//var windSpeed = 20;
var crowd = msg.crowd;
//var crowd = 5;
//var crowd = 30;

let backgroundColor = 'rgba(242, 241, 241, 0.8)'; // blanc
let recommendation = 'Vous pouvez skater, profitez-en!';
let reasons = [];

if (weatherCondition === 'Rain') {
    crowd = 0;
    humidity = 0;
    recommendation = 'Il est déconseillé de skater maintenant,';
    reasons.push('il pleut actuellement');
    backgroundColor = 'rgba(50, 50, 50, 0.8)'; // noir
}

if (humidity > 90) {
    crowd = 0;
    recommendation = 'Il est déconseillé de skater maintenant,';
    reasons.push('le sol du skatepark risque d\'être mouillé');
    backgroundColor = 'rgba(50, 50, 50, 0.8)'; // noir
}

if (temperature > 35) {
    recommendation = 'Il est conseillé de skater avec prudence,';
    reasons.push('risque d\'insolation');
    backgroundColor = 'rgba(141, 140, 140, 0.8)'; // gris
}

if (temperature < 5) {
    recommendation = 'Il est conseillé de skater avec prudence,';
    reasons.push('couvrez-vous');
    backgroundColor = 'rgba(141, 140, 140, 0.8)'; // gris
}

if (windSpeed > 15) {
    recommendation = 'Il est conseillé de skater avec prudence,';
    reasons.push('le vent est trop élevé');
    backgroundColor = 'rgba(141, 140, 140, 0.8)'; // gris
}

if (crowd > 25) {
    recommendation = 'Il est conseillé de skater avec prudence,';
    reasons.push('affluence élevée');
    backgroundColor = 'rgba(141, 140, 140, 0.8)'; // gris
}

if (weatherCondition === 'Rain' || humidity > 90) {
    recommendation = 'Il est déconseillé de skater maintenant,';
    backgroundColor = 'rgba(50, 50, 50, 0.8)'; // noir
}

if (reasons.length > 0) {
    msg.payload = {
        recommendation: recommendation,
        reasons: reasons.join(', ').replace(/,([^,]*)$/, ' et$1') + '.',
        crowd: crowd,
        backgroundColor: backgroundColor
    };
} else {
    msg.payload = {
        recommendation: recommendation,
        reasons: '',
        crowd: crowd,
        backgroundColor: backgroundColor
    };
}

return msg;
```

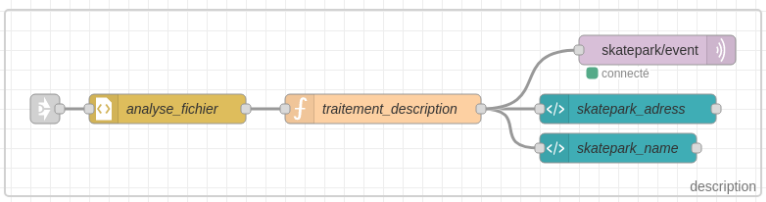
**Vous pouvez modifier** les différents **paramètres** du nœud fonction **« traitement_ conditions »** (certains cas sont déjà en commentaire) afin **de tester tous les éventuels cas possibles** (conditions météorologiques, affluence…) afin **d'observer les modifications sur le dashboard.**




Le principe est simple, on génère **un nombre aléatoire** pour **simuler l'affluence du lieu**, on récupère les **infos météorologiques sur l'API OpenWeather**, ainsi que les **données du groupe lumière** pour adapter les conseils, de plus **ces données sont renvoyées via un broker MQTT** afin d'être récupérée dans de futures pages.

## f) group « description »

Ce groupe permet un **affichage dynamique des détails du skatepark** sélectionné, à partir d'un **fichier XML**, incluant le nom, une photo, une description ainsi que la localisation sur une carte du skatepark. Cela permet une **extension aisée pour ajouter de nouveaux skateparks,** offrant ainsi une portée étendue pour une base de données de lieux disponibles.




```
var skatepark = msg.payload.skateparks.skatepark[0];

var formattedData = {
    title: skatepark.title[0],
    image: skatepark.image[0],
    description: skatepark.description[0],
    address: skatepark.address[0],
    latitude: Number(skatepark.latitude[0]),
    longitude: Number(skatepark.longitude[0]),
    eventimage: skatepark.eventimage[0],
    eventdescription: skatepark.eventdescription[0],
    email: skatepark.email[0],
}

msg.payload = formattedData;
return msg;
```

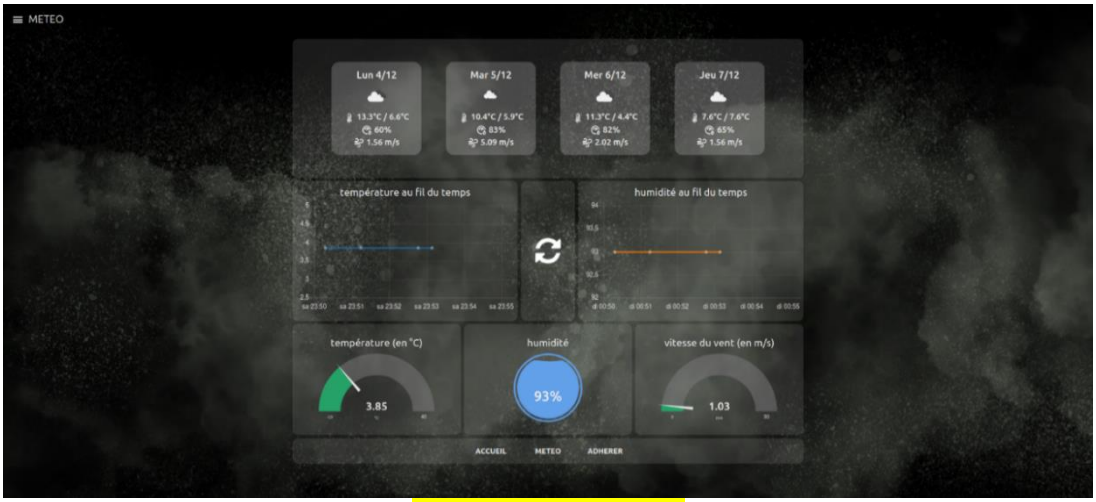

Le système **analyse le fichier XML,** extrait les détails sur le skatepark désigné, et les présente de **manière structurée sur le dashboard**. Les informations **aident les utilisateurs** à sélectionner le skatepark adéquat **en fonction de leurs préférences**, de leur **emplacement géographique** et **des conditions météorologiques.** Les informations peuvent êtres modifié **à tout moments** en fonction, des différents événements ou modifications de modules sur un skatepark par exemple.

```
<skateparks>
  <skatepark>
    <title>Skatepark des Remparts</title>
    <image>https://github.com/tahlisfove/IoT/blob/main/img/sk8park.jpg?raw=true</image>
    <description>
      Le skatepark offre deux zones distinctes adaptées aux niveaux débutants et confirmés, équipées...
    </description>
    <address>Chemin de Beneharnum, 64230 Lescar</address>
      <latitude>43.33268403112534</latitude>
      <longitude>-0.4335628116439437</longitude>
    <eventimage>https://github.com/tahlisfove/IoT/blob/main/img/evenement.png?raw=true</eventimage>
    <eventdescription>
      Inscrivez-vous dès maintenant à "Skoot 2", une compétition mêlant skateboard et trottinette...
    </eventdescription>
    <email>lescarskatefest@gmail.com</email>
  </skatepark>
</skateparks>
```

fichier skateparks.xml

## 2) page METEO

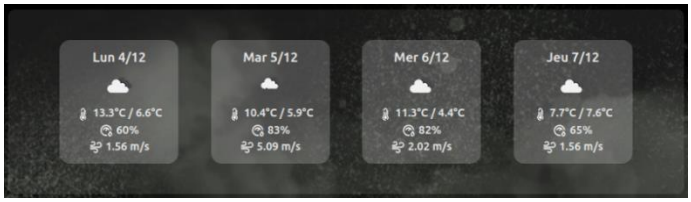
La page météo fournit une **analyse détaillée** des **données météorologiques**, elle offre une **perspective approfondie des conditions**, permettant aux utilisateurs de planifier leurs sessions au vu de la semaine en fonction des prévisions détaillées. Elle présente une série **d'éléments visuels** et informatifs pour une analyse complète des **conditions climatiques.**




page meteo du dashboard

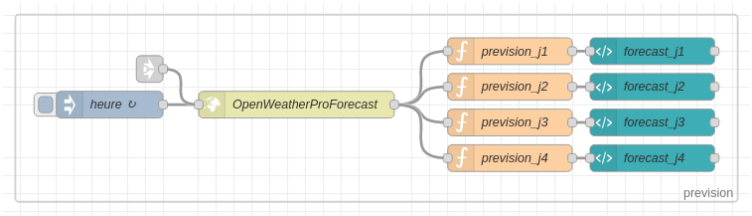
### *a) group « prevision »*

Ce groupe est conçu pour fournir des **informations** météorologiques **détaillées**, aidant les utilisateurs à **planifier leurs sortie** en fonction des conditions future. Il comprend la **date**, la **tendance** météo, la température **minimale** et **maximale**, le **vent moyen** et **l'humidité moyenne de chaque journée et sa pour** les **4 jours suivants en prévision.**



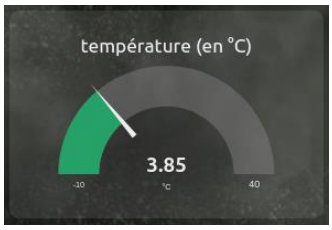

```
const days = ['Dim', 'Lun', 'Mar', 'Mer', 'Jeu', 'Ven', 'Sam'];
const months = ['01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12'];

const kelvinToCelsius = kelvin => kelvin - 273.15;

const weatherData = msg.payload.list.filter(entry => {
    const date = new Date(entry.dt * 1000);
    const tomorrow = new Date();
    const nextDay = new Date(tomorrow);
    nextDay.setDate(tomorrow.getDate() + 1);
    return date.getDate() === nextDay.getDate();
});

if (weatherData.length > 0) {
    let maxTemp = -Infinity;
    let minTemp = Infinity;

    weatherData.forEach(data => {
        maxTemp = Math.max(maxTemp, data.main.temp_max);
        minTemp = Math.min(minTemp, data.main.temp_min);
    });

    const dayDate = new Date(weatherData[0].dt * 1000);
    const formattedDayDate =
        days[dayDate.getDay()] + ' ' + dayDate.getDate() + '/' + months[dayDate.getMonth()];

    msg.payload = {
        dayDate: formattedDayDate,
        weatherIcon: weatherData[0].weather[0].icon,
        tempMax: kelvinToCelsius(maxTemp).toFixed(1),
        tempMin: kelvinToCelsius(minTemp).toFixed(1),
        humidity: weatherData[0].main.humidity,
        windSpeed: weatherData[0].wind.speed
    };
    return msg;
} else {
    return null;
}
```

Comme précédemment le fonctionnement est **similaire** à celui de la **page d'accueil**, on envoie une requête sur une **nouvelle API** (plus puissante, car prévision sur 4j) pour récupérer les informations en temps réel et les afficher sur le dashboard. On répète ce processus pour afficher chaque jour dans l'affichage des prévisions.

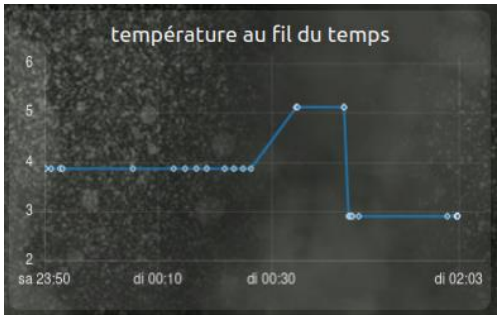


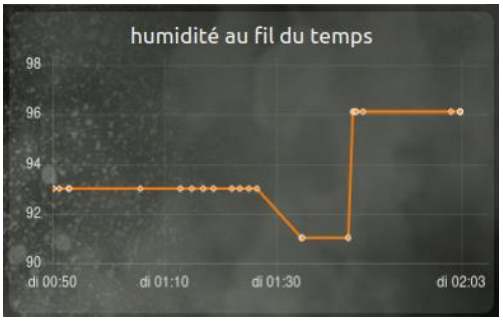

## *b) groups « gauges graphiques »*

Les groupes de **gauges et graphiques** offrent une **vue immédiate** des conditions météos **actuelles** (température, humidité, vents) via des jauges, ainsi qu'un **aperçu graphique** des variations sur les **12 dernières heures** (pour la température et l'humidité), permettant d'observer rapidement les **tendances passées**.



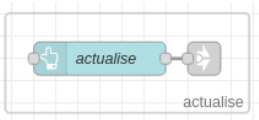



## *c) group « actualise »*

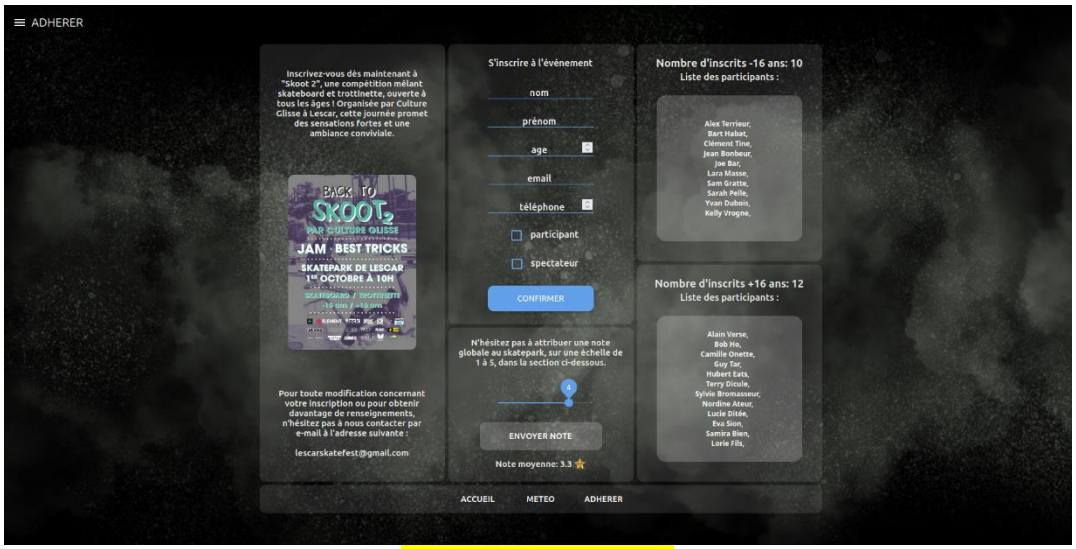
Le bouton d'actualisation permet de **mettre à jour** en temps réel **toutes** les **données** météorologiques **affichées sur la page**. En un clic, il **actualise les informations** pour refléter les conditions météorologiques les plus récentes, il permet entre autre d'ajouter un point sur les deux graphiques vu précédemment.




---

## 3) page ADHERER

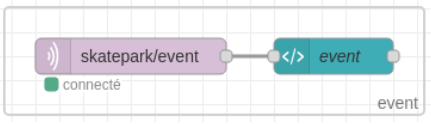
La page Adhérer du dashboard du skatepark intelligent offre une plateforme interactive pour les utilisateurs souhaitant **s'inscrire à des événements** ou **recevoir des notifications concernant le skatepark sélectionné.**



page meteo du dashboard

### *a) group « event »*

Le bloc event de la constitue une **section centrale et informative** pour les utilisateurs souhaitant participer aux **activités organisées au skatepark**. Il contient une mini **description de l'événement** à venir, **l'affiche** de l'événement ainsi que l'adresse **email des organisateurs** de l'événement. L'ensemble de ses **informations sont récupérées** lors de l'analyse du **fichier xml dans la page d'accueil,** puis **envoyé via MQTT** sur cette page.





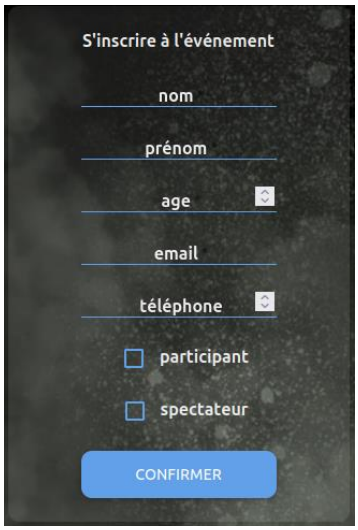
Ce bloc event vise à **présenter** de manière complète les **détails pertinents de l'événement à venir**, encourageant ainsi les utilisateurs à s'y **inscrire** et à y **participer**.

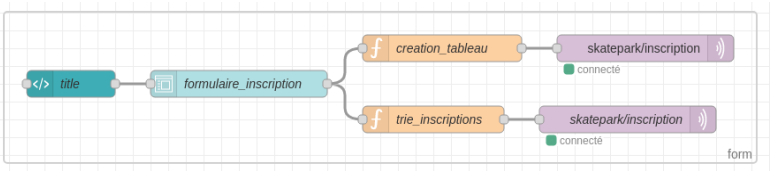
### *b) group « form »*

Le **formulaire d'inscription** est une composante essentielle pour les utilisateurs souhaitant participer aux événements spécifiques du skatepark sélectionné. Il se compose de **champs détaillés pour recueillir des informations** obligatoires pout toutes inscriptions.

On remarque, **le nom et le prénom, l'âge** (pour catégoriser les participants **en deux groupes**, ceux âgés de **plus de 16 ans** et ceux de **moins de 16 ans**), une adresse mail un numéro de téléphone (facultatif) ainsi qu'un **choix entre participant ou spectateur** permettant aux organisateurs de **prévoir et d'adapter l'événement en conséquence.**

```
var nom = msg.payload.Nom;
var prenom = msg.payload.Prenom;
var email = msg.payload.Email;
var estParticipant = msg.payload['Participant '];
var age = parseFloat(msg.payload.Age);

var userData = {
    Nom: nom,
    Prenom: prenom,
    Email: email,
    Age: age
};

if (estParticipant === true) {
    if (age > 16) {
        msg.topic = "skatepark/inscription/plus16";
    } else {
        msg.topic = "skatepark/inscription/moins16";
    }
} else {
    msg.topic = "skatepark/spectateur";
}

msg.payload = userData;
return msg;

```

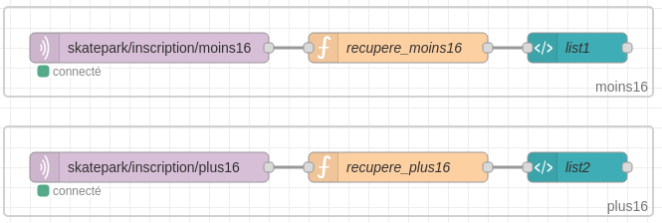


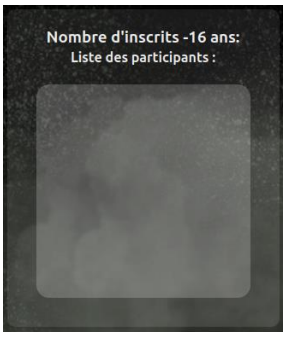

Ce formulaire vise à **séparer les différents participants** pour **créer une future liste** pour l'événement, fournissant aux organisateurs des données utiles **pour planifier et exécuter l'événement**, ainsi que pour adapter **différents groupes d'âge** et/ou **pratique sportive.**

## *c) group « participants »*

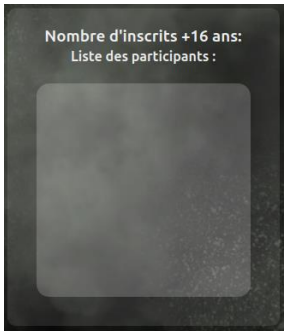
C'est ici que sont récupérés les **noms et prénoms des participants** en fonction de leur âge Il se divise en **deux parties distinctes:** le nombre total de participants âgés de plus de 16 ans ainsi que le nombre total de participants âgés de moins de 16 ans qui **se sont inscrits à l'événement.**

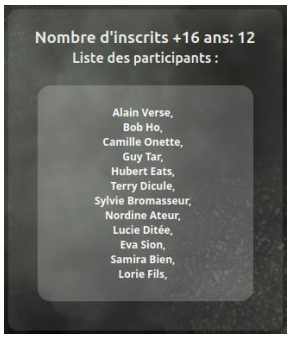



Par **défaut les listes sont vides** et se remplissent au fur et **à mesure des inscriptions** et dans les listes jusqu'à **atteindre une limite** (fixée **selon les événements,** les âges et le skatepark en question).



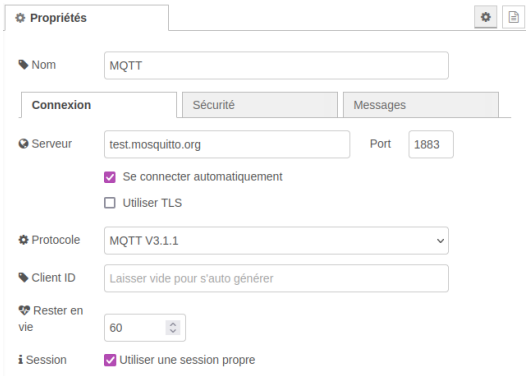


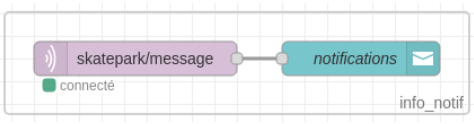


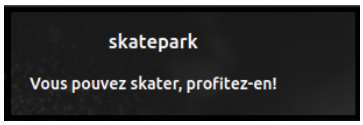




## *d) group « notifications »*

Le bloc notification **affiche des notifications** générales **récupérées depuis la page d'accueil** en **haut à droite de chaque page,** il rappelle la tendance actuelle **toute les 10 minutes**. Les informations récupérées par les blocs de cette page utilisent **MQTT Mosquito** cela garantit que les informations relatives à la faisabilité du skatepark soient **diffusées de manière synchronisée sur toutes les interfaces.**



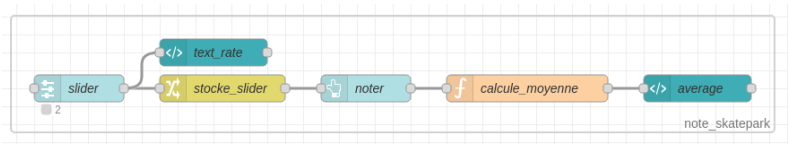




Nous avions pour idée d'ajouter **un système** permettant à l'utilisateur de recevoir ces **notifications par e-mail** en s'inscrivant dans une zone dédiée, lui permettant ainsi d'être informé automatiquement **à une heure définit chaque jour,** cependant nous avons abandonné l'idée pensant que c'était **trop superficiel.**

## *e) group « note »*

Le bloc note permet aux utilisateurs de **noter le skatepark où ils se trouvent,** offrant donc une évaluation en temps réel de l'expérience utilisateur. Ce bloc comporte un **slider interactif** avec des **valeurs de 1 à 5.** L'utilisateur peut déplacer le curseur pour sélectionner la **note désirée** et l'envoyer en **cliquant sur le boutons dessous « ENVOYER NOTE ».**


A chaque fois qu'une **nouvelle note est soumise,** le système **recalcule la moyenne** des notes attribuées au skatepark en **prenant en compte la note nouvellement ajoutée et les notes précédentes**. La valeur actuelle du slider est **stockée dans un nœud spécifique** (nœud change) **en attente du clic** sur le bouton par l'utilisateur

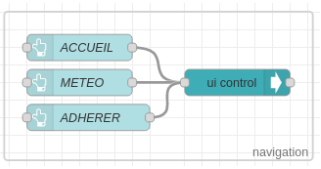



```
var storedValue = flow.get('storedValue');

if (storedValue !== undefined) {
    var previousValues = flow.get('previousValues') || [];
    previousValues.push(storedValue);

    var sum = previousValues.reduce(function(acc, val) {
        return acc + val;
    }, 0);

    var average = sum / previousValues.length;

    flow.set('previousValues', previousValues);
    msg.payload = average;
    return msg;
} else {
    return null;
}
```

---

## 4) navigation et visuel

Les groupes **navigation** permettent aux utilisateurs de **passer d'une section à une** autre via des liens cliquables. Les liens sont des **boutons libellés**, offrant une représentation visuelle de la structure du dashboard. La **transition** entre les pages est **fluide**, assurant une **continuité dans l'expérience utilisateur**.


Une barre d'outils est également disponible sur la partie **gauche de l'écran**, offrant une **alternative pratique** pour la **navigation entre les pages**. Elle reste fixe sur toutes les pages, permettant à l'utilisateur de se déplacer facilement d'une section à une autre **sans retourner à la page d'accueil.**

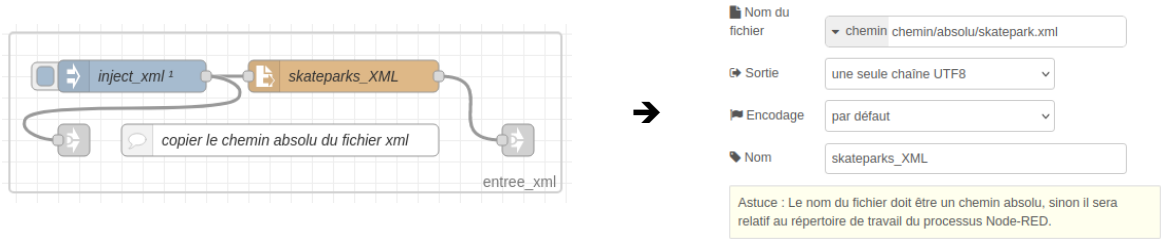
Le **design** du dashboard du skatepark intelligent est pensé pour **l'esthétique et la convivialité des utilisateurs.** Les styles et mises en page sont gérés via **un fichier CSS** dédié placé dans la **page accueil**, ce qui permet une **gestion simplifiée des styles** et offre la flexibilité nécessaire pour des **ajustements futurs**. Ainsi, le dashboard du skatepark intelligent est **fonctionnel**, **esthétique** et **adaptable** pour répondre aux **besoins des utilisateurs.**

# III) Lancer le Projet

1- **Télécharger** les fichiers *skate.json* et *skateparks.xml*,

2- Ouvrir Node-RED (node-red) et **importer** le fichier skate.json,

3- Configurer le nœud XML en **indiquant le chemin absolu** du fichier skateparks.xml,




4- **Déployer** le contenu sur Node-RED et **ouvrir le dashboard**.

Si certains éléments ne se **chargent pas** correctement, essayez de **changer de page,** ou de **réinjecter le flux** principal depuis Node-RED. Cela permet de **mettre à jour** toutes les informations du dashboard.

# IV) Conclusion

En combinant les **données météorologiques en temps réel**, une interface conviviale, des dispositifs de sécurité tels que le **contrôle d'éclairage** et des **événements sportifs**, ce projet vise à améliorer **l'expérience** des utilisateurs. Sa flexibilité, notamment la possibilité d'ajouter facilement **de nouveaux skateparks**, démontre sa capacité **à s'adapter** aux diverses **exigences** des communautés. En résumé, ce projet met en évidence le potentiel **de l'IoT** pour créer des espaces de **loisirs sécurisés et interactifs**, offrant ainsi une **expérience utilisateur optimisée** dans le monde des skateparks.