

# **Conception Logicielle, smart phone au wearable computer**

---

**Soutenance intermédiaire**  
**Projet : TrackMap**

---

---

# Présentation sujet

---

## Objectif:

**Créer une application mobile connectés utilisant au minimum 2 dispositifs (téléphones et SmartWatch) en stockant ces données et en les visualisant**

## Notre sujet:

**L'idée serait d'avoir une application mobile permettant de visualiser différentes données des utilisateurs possédant l'application lors d'une activité sportif tel que la course**

# Description de l'application

**Actuellement l'application prend:**

- **capteur de luminosité**
- **capteur podométrique**
- **capteur de géolocalisation**

**Actuellement possible pour l'utilisateur actuel de l'application, a terme ces données apparaitront pour tous les utilisateurs**

- **La valeur de la luminosité est envoyé sur Influx qui est affiché sur l'application dans l'infobox de l'utilisateur est peut être visualisé sur Grafana**
- **A l'aide du capteur podométrique nous pouvons sur l'application lancer un compteur de pas qui envoie la donnée à Influx (0) et envoie la donnée une autre fois au stop du compteur, visualisation possible sur Grafana**
- **On récupère la localisation de l'utilisateur en stockant la donnée sur influx afin de voir notre position sur une map l'application telephone ainsi que sur une map sur Grafana**

---

# Stratégie adopté

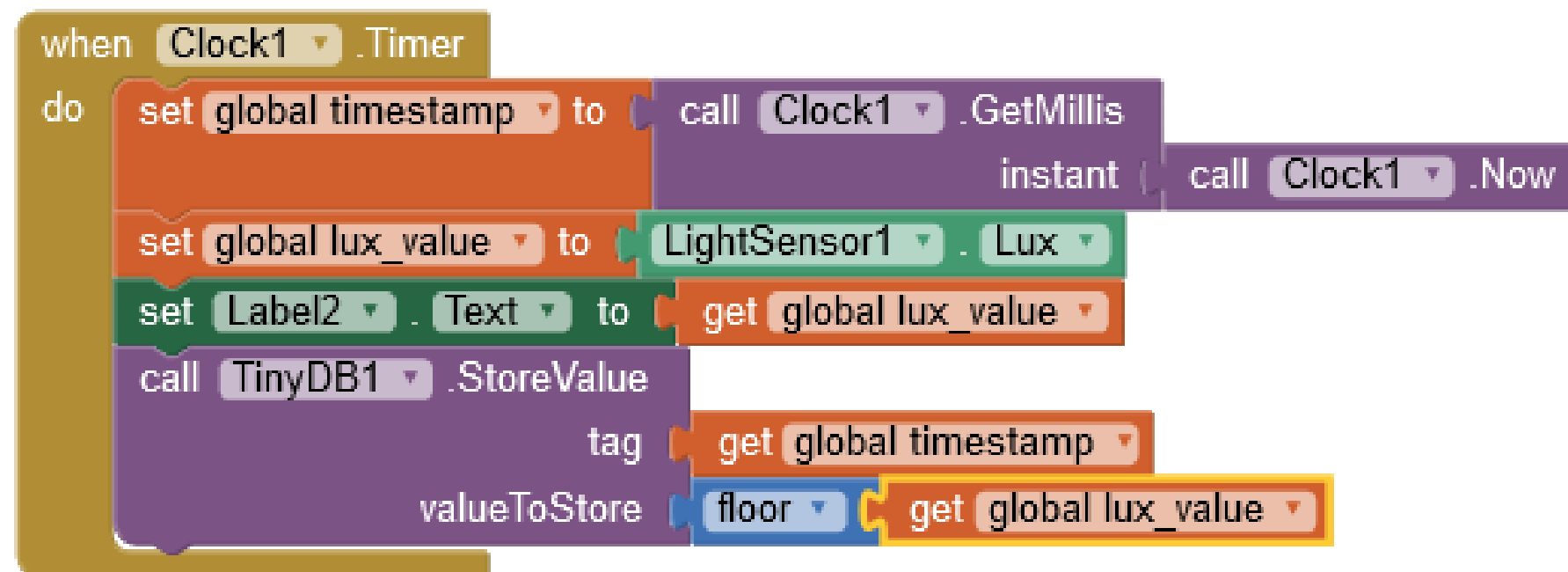
---

- **Optimisation de l'envoi de donnée avec TinyDB par package**
- **Application fonctionnel sur téléphone et donnée qui apparait sur une map afin de voir facilement ou se situe les utilisateurs de l'application**
- **Ajout interactif de consulter ses données via des infobox sur le marqueur de chaque utilisateurs et autres éléments de l'UI**

# Principe et choix architecturaux

- Conception de TinyDB sur MIT App Inventor:

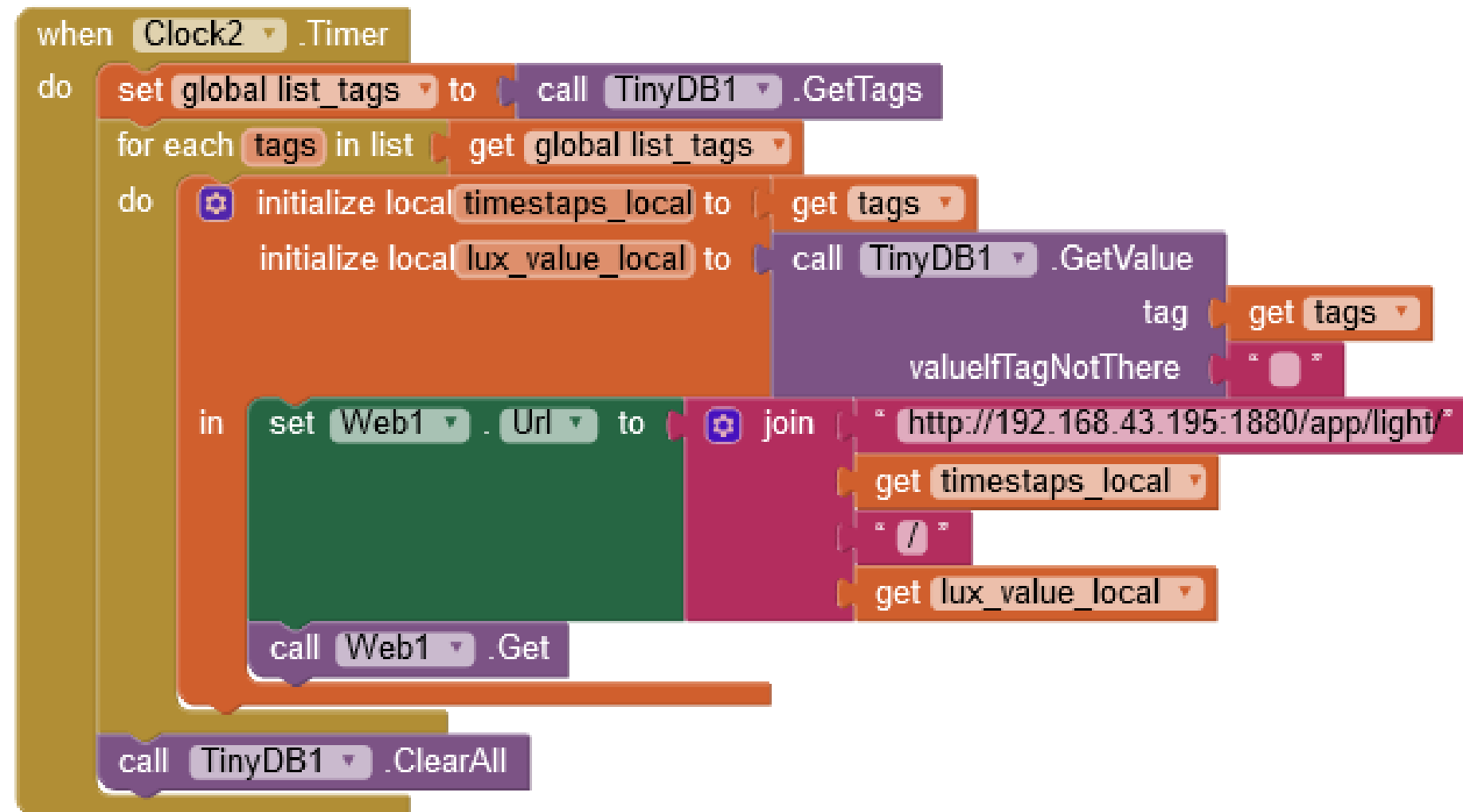
**On lit les informations toutes les secondes et on les enregistre dans une TinyDB**



# Principe et choix architecturaux

- Conception de TinyDB sur MIT App Inventor:

**Toute les information stocker sont envoyer toutes les 10 secondes puis sont supprimer de la TinyDB**



---

# Principe et choix architecturaux

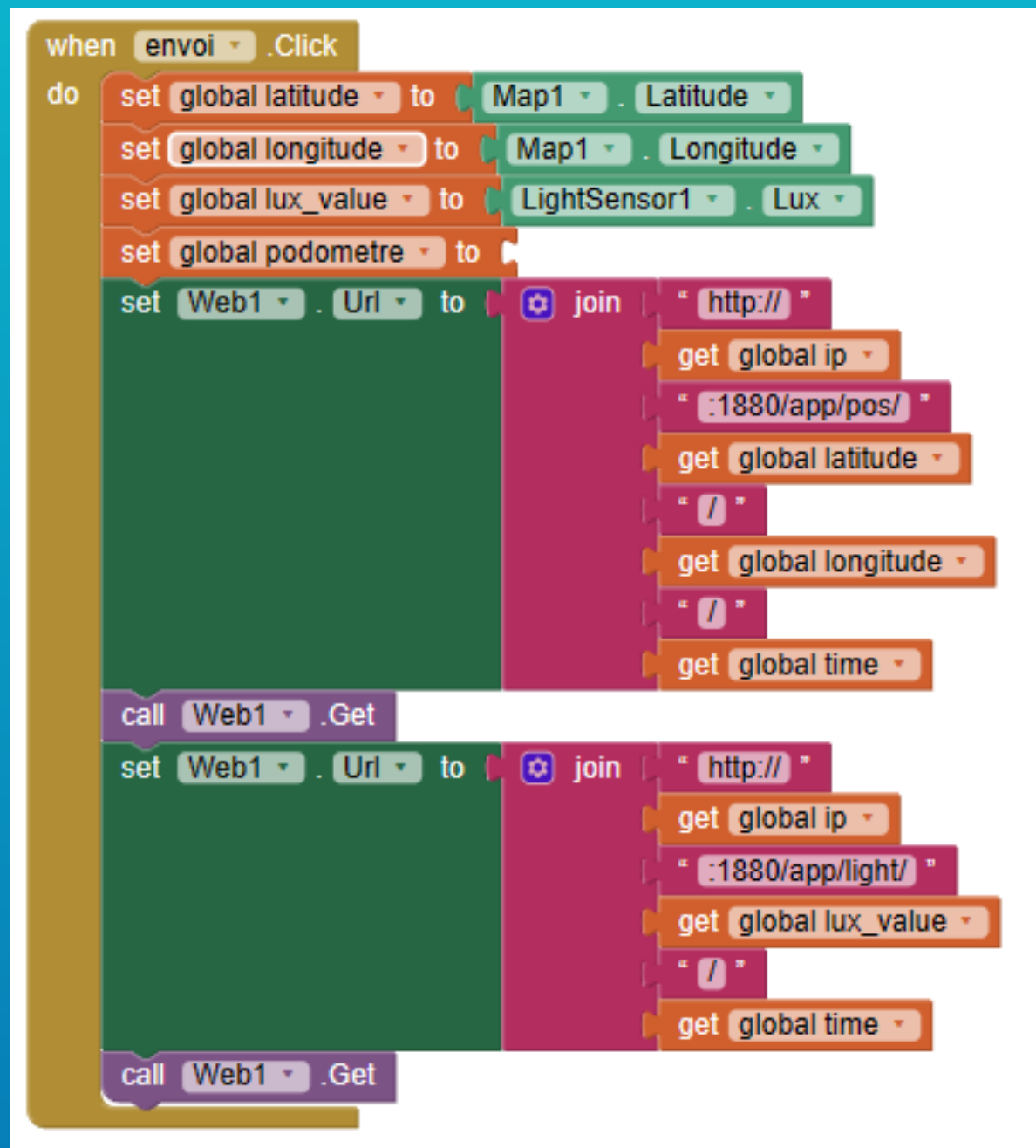
---

- **Utilisation de MIT App Inventor/InfluxDB (pourquoi ?):**
  - permet de stocker un large volume de données chronologiques, ou séries temporelles
  - fonctionne avec de larges données
- **Utilisation de Grafana:**

Grafana est très utile mêlée à InfluxDB afin de visualiser de plusieurs manière possible via différente technique de visualisation nos données (map, graphique bar, graphique donuts etc)

# Maitrise des éléments technique

## a. Comment les données sont envoyées à Influx DB



```
▼ 0:
result:      "_result"
table:       0
_start:      "2023-10-23T17:38:32.309318506Z"
_stop:       "2023-10-24T17:38:32.309318506Z"
_time:       "2023-10-24T14:15:55.031Z"
_value:      43.61527
_field:      "latitude"
_measurement: "positiongps"

▼ 1:
result:      "_result"
table:       0
_start:      "2023-10-23T17:38:32.309318506Z"
_stop:       "2023-10-24T17:38:32.309318506Z"
_time:       "2023-10-24T14:25:53.941Z"
_value:      43.61527
_field:      "latitude"
_measurement: "positiongps"

▼ 2:
result:      "_result"
table:       0
_start:      "2023-10-23T17:38:32.309318506Z"
_stop:       "2023-10-24T17:38:32.309318506Z"
_time:       "2023-10-24T14:26:18.177Z"
_value:      43.60851
_field:      "latitude"
_measurement: "positiongps"

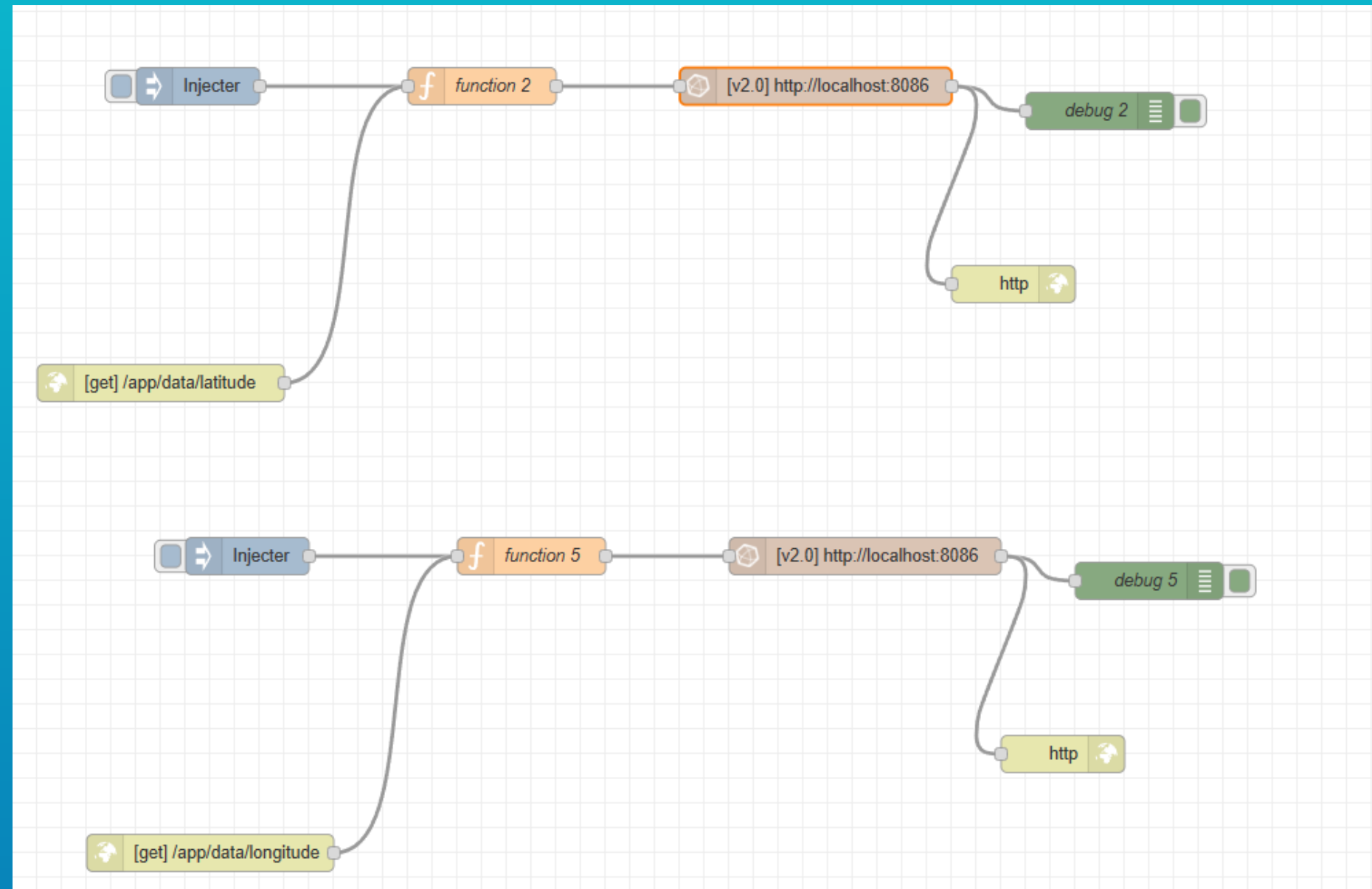
▼ 3:
result:      "_result"
table:       0
_start:      "2023-10-23T17:38:32.309318506Z"
_stop:       "2023-10-24T17:38:32.309318506Z"
_time:       "2023-10-24T14:26:33.132Z"
_value:      43.60851
_field:      "latitude"
_measurement: "positiongps"

▼ 4:
result:      "_result"
table:       0
_start:      "2023-10-23T17:38:32.309318506Z"
_stop:       "2023-10-24T17:38:32.309318506Z"
_time:       "2023-10-24T15:04:25.045Z"
_value:      59.9
_field:      "latitude"
_measurement: "positiongps"
```



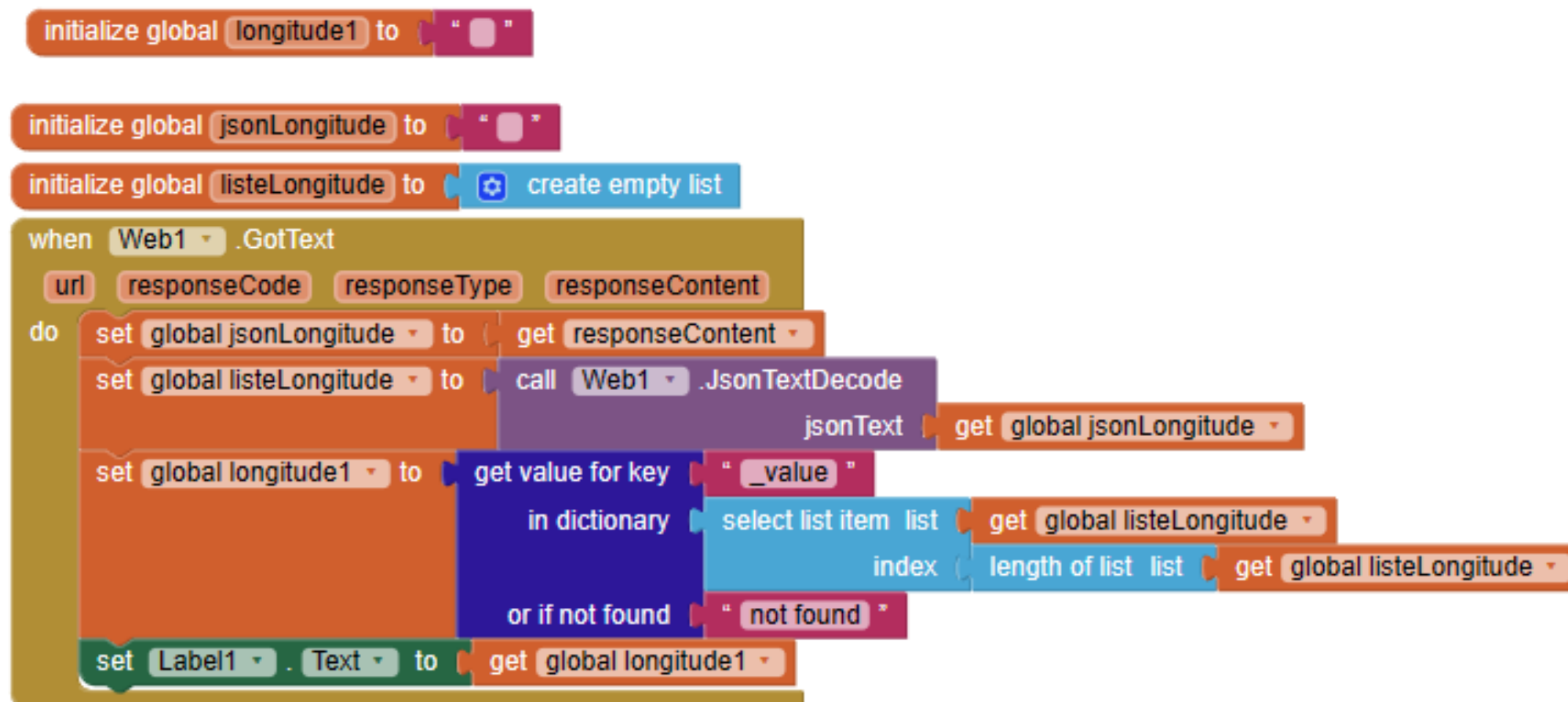
# Maitrise des éléments technique

## b. Comment les données sont récupérées sur l'app depuis Influx (MANUELEMENT)



# Maitrise des éléments technique

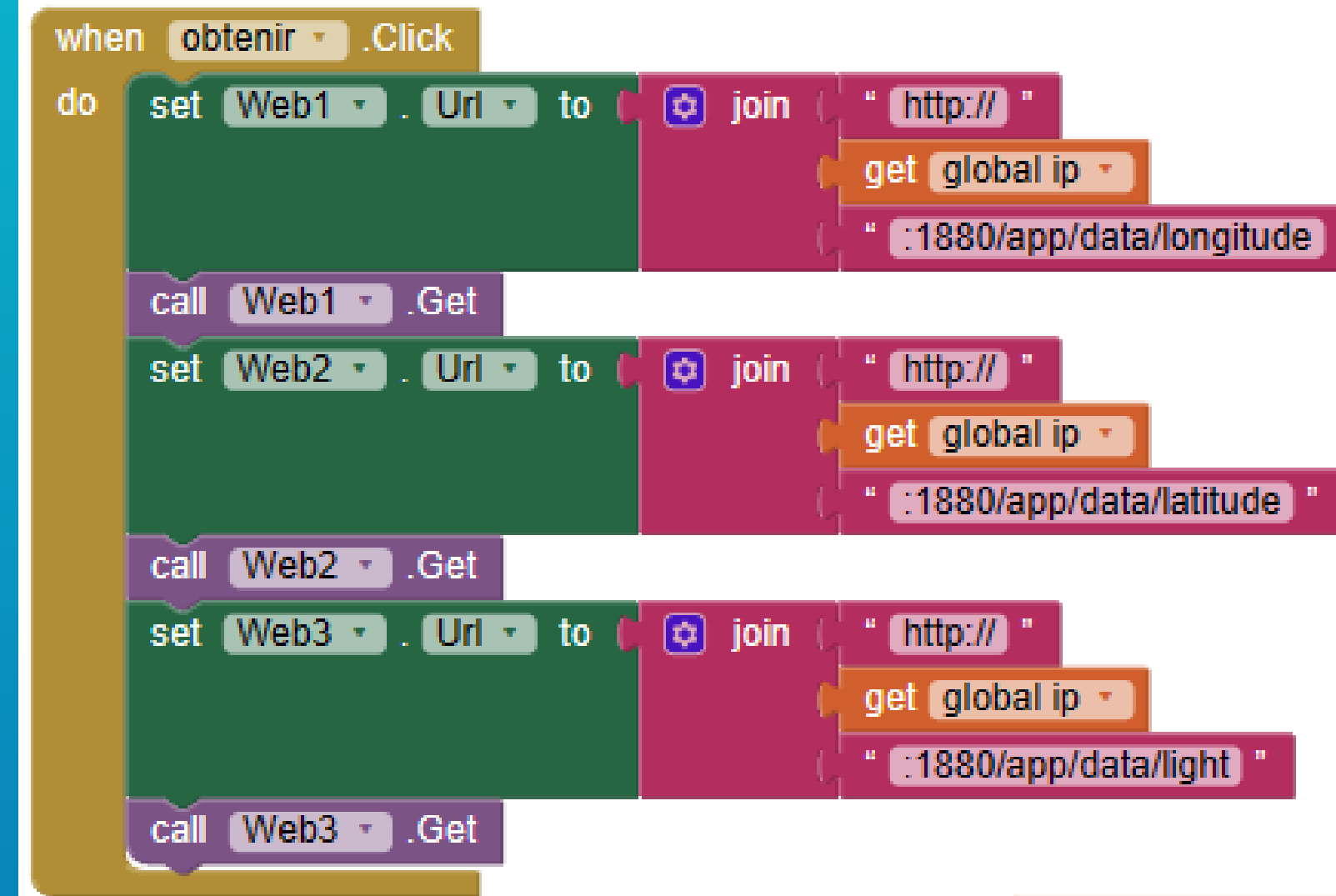
## b. Comment les données sont récupérées sur l'app depuis Influx (MANUELEMENT)



```
initialize global longitude1 to ""
initialize global jsonLongitude to ""
initialize global listeLongitude to create empty list

when Web1 GotText
  url responseCode responseType responseContent
  do
    set global jsonLongitude to get responseContent
    set global listeLongitude to call Web1 JsonTextDecode
    jsonText get global jsonLongitude
    set global longitude1 to get value for key
    in dictionary select list item list get global listeLongitude
    index length of list list get global listeLongitude
    or if not found "not found"
    set Label1 Text to get global longitude1
```

The script on the left is a Scratch script that handles the manual retrieval of data from Influx. It starts with three global variables: `longitude1` (string), `jsonLongitude` (string), and `listeLongitude` (list). The script is triggered when `Web1` receives a text response. It then performs the following steps: 1. Sets `global jsonLongitude` to the `responseContent`. 2. Calls `Web1.JsonTextDecode` with `global jsonLongitude` as the `jsonText` argument. 3. Sets `global longitude1` to the value of the key `"_value"` in the dictionary returned by the previous step, using `select list item list` and `length of list list` to find the correct index. 4. If the key is not found, it sets `global longitude1` to `"not found"`. 5. Finally, it sets the `Text` property of `Label1` to the value of `global longitude1`.



```
when obtenir Click
do
  set Web1 Url to join "http://"
  get global ip
  "1880/app/data/longitude"

  call Web1 Get
  set Web2 Url to join "http://"
  get global ip
  "1880/app/data/latitude"

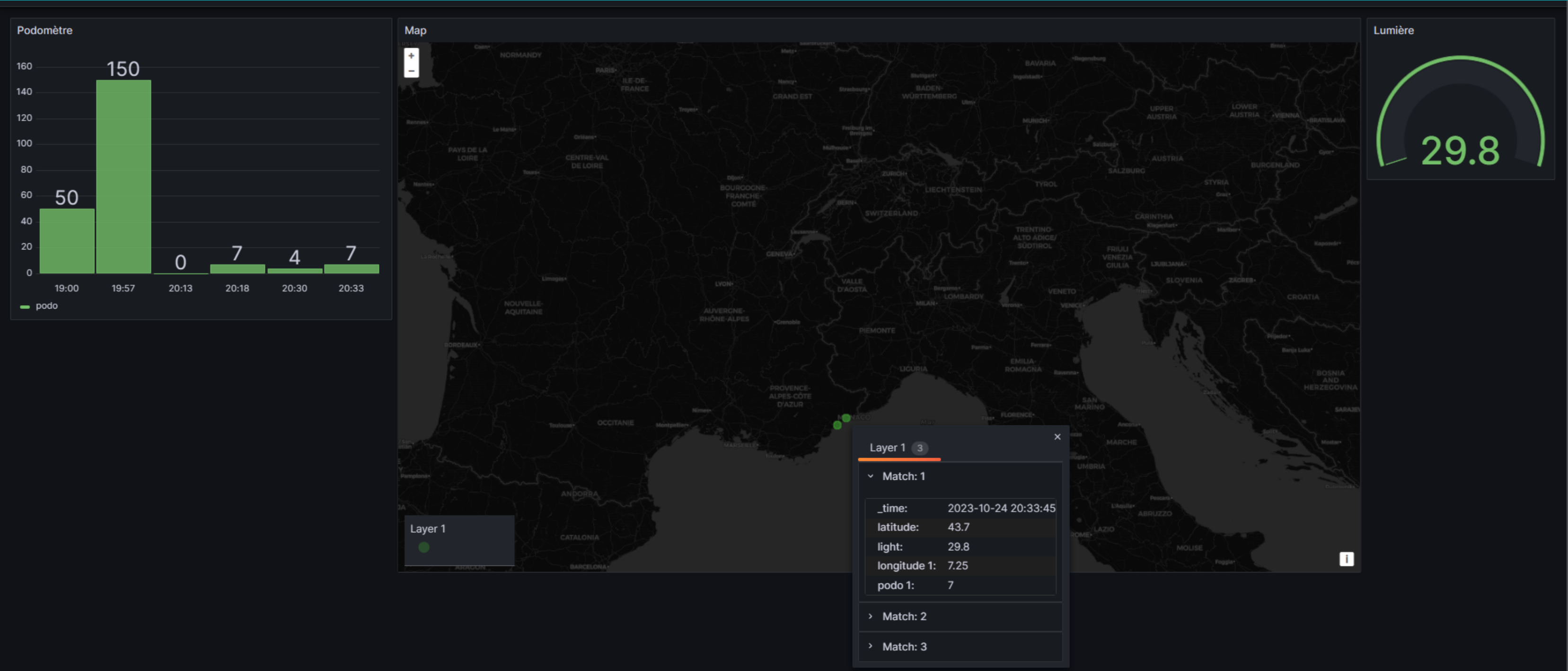
  call Web2 Get
  set Web3 Url to join "http://"
  get global ip
  "1880/app/data/light"

  call Web3 Get
```

The script on the right is a Scratch script that handles the automatic retrieval of data from Influx. It is triggered when the `obtenir` button is clicked. It performs the following steps: 1. Sets the `Url` property of `Web1` to `join "http://"` and `get global ip`. 2. Calls `Web1.Get`. 3. Sets the `Url` property of `Web2` to `join "http://"` and `get global ip`. 4. Calls `Web2.Get`. 5. Sets the `Url` property of `Web3` to `join "http://"` and `get global ip`. 6. Calls `Web3.Get`.

# Maitrise des éléments technique

## c. Visualisation via Grafana



---

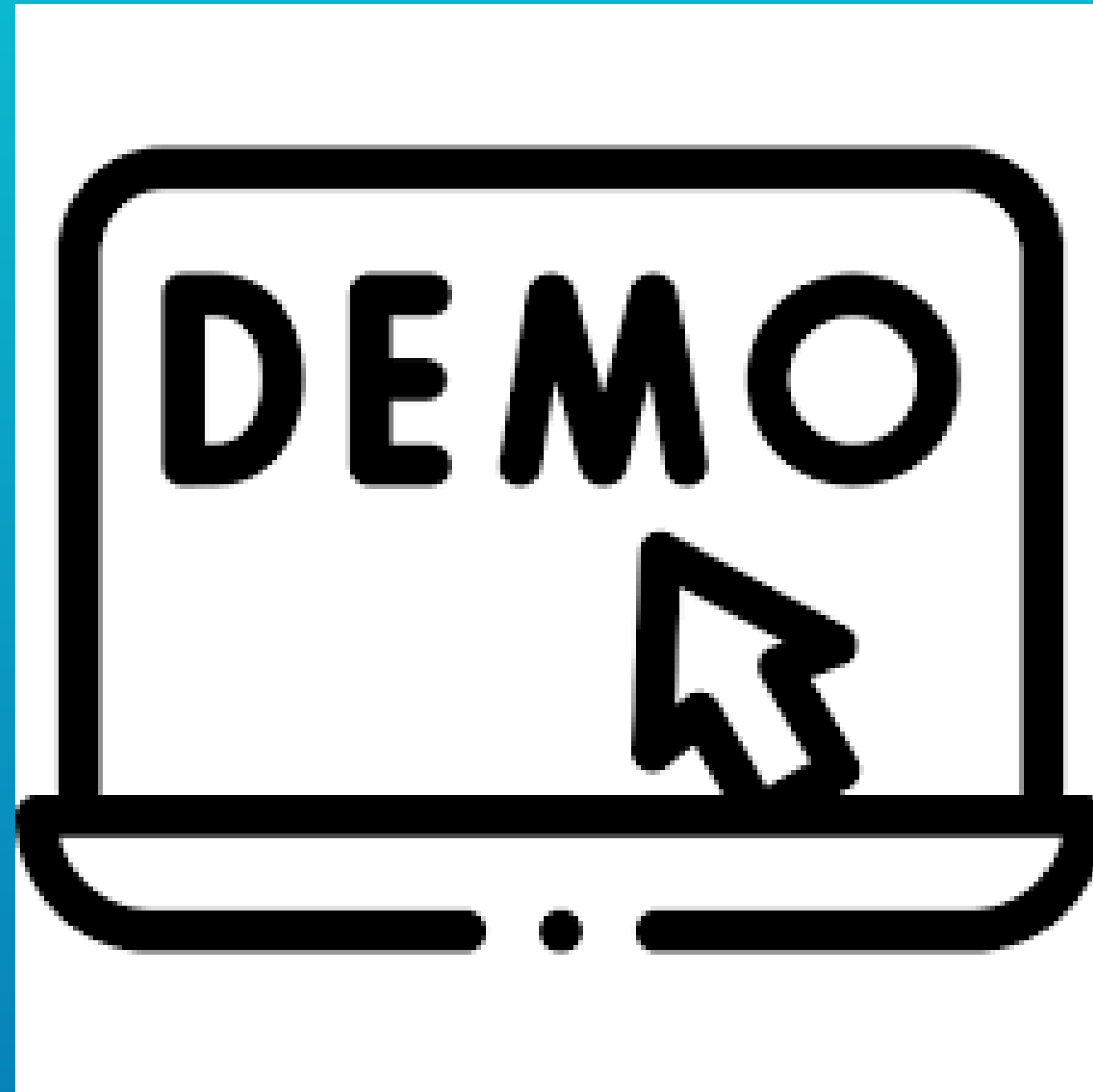
# Maitrise des éléments technique

---

## **d. Les défis techniques rencontrés et solutions**

- When Web1 Got.Text**
- Création de la map sur Grafana**
- Ne pas faire quelque chose de local mais où plusieurs utilisateurs différents apparaitront sur la même app**

# Démonstration



---

**Merci pour  
votre attention**

---