

# Documentation Technique

---

Par Axel Bacrie 1A [axel.bacrie@etu.univ-tlse2.fr](mailto:axel.bacrie@etu.univ-tlse2.fr) et Théo Chaves 1A  
[theo.chaves@etu.univ-tlse2.fr](mailto:theo.chaves@etu.univ-tlse2.fr) (correcteur)

---

# Table des matières

1. Introduction . . . . .	5
2. Structure et fonctionnement . . . . .	5
2.1. Structure de l'application . . . . .	5
2.1.1. Point sur les technologies. . . . .	7
2.1.1.1. Capteur Electrique (NKE TRIPHAS'O 60A) . . . . .	7
2.1.1.2. LoRaWAN . . . . .	7
2.1.1.3. MQTT . . . . .	8
2.1.1.4. Node-Red . . . . .	9
2.1.1.5. InfluxDB . . . . .	9
2.1.1.6. Grafana . . . . .	9
2.1.1.7. PHP. . . . .	9
2.1.1.8. Apache . . . . .	9
2.2. Structure du site. . . . .	10
2.3. Paradigme MVC2. . . . .	13
2.4. Fonctionnalités du site . . . . .	13
2.4.1. L'authentification . . . . .	14
2.4.1.1. L'obtention de privilèges . . . . .	15
2.4.1.2. La vérification des privilèges . . . . .	16
2.4.2. Les Menus . . . . .	16
2.4.3. Les paramètres. . . . .	17
3. Configuration . . . . .	18
3.1. Le fichier config.php. . . . .	18
3.1.1. Informations sur InfluxDB . . . . .	19
3.1.2. Stockage des d'URL. . . . .	20
3.1.3. Mise en place des couleurs de thème . . . . .	20
3.1.4. Informations de navigation/routes . . . . .	25
3.1.5. Gestion des cookies . . . . .	27
3.1.6. Construction des menus. . . . .	27
3.2. Configurer l'authentification . . . . .	32
4. Mise en place du serveur . . . . .	33
4.1. Introduction . . . . .	33
4.2. Installation de NodeRed. . . . .	33
4.2.1. Installation . . . . .	33
4.2.1.1. Node . . . . .	33

4.2.1.2. Node-RED . . . . .	34
4.2.1.3. Node-RED Admin et Configuration user . . . . .	35
4.2.1.4. Packet Node-RED pour InfluxDB . . . . .	37
4.2.1.5. Lancer Node-RED au démarrage . . . . .	37
4.2.2. Sources . . . . .	38
4.3. Installation de InfluxDB . . . . .	39
4.3.1. Installation . . . . .	39
4.3.2. Configuration . . . . .	39
4.3.2.1. Réseau . . . . .	40
4.3.2.2. User et base de données . . . . .	40
4.3.3. Sources . . . . .	41
4.4. Installation Grafana . . . . .	41
4.4.1. Installation . . . . .	41
4.4.2. Configuration . . . . .	43
4.4.2.1. Boot . . . . .	43
4.4.2.2. Réseau . . . . .	43
4.4.2.3. Première connexion . . . . .	43
4.4.2.4. Source de données . . . . .	45
4.4.3. Sources . . . . .	49
4.5. Installation Apache et PHP . . . . .	49
4.5.1. Instalation . . . . .	50
4.5.1.1. Apache . . . . .	50
4.5.1.2. PHP . . . . .	50
4.5.2. Configuration Apache . . . . .	50
4.5.3. Source . . . . .	51
4.6. Configuration réseau . . . . .	51
5. Simulation de données avec Node-Red . . . . .	51
5.1. Introduction . . . . .	52
5.2. Valeurs de simulation . . . . .	52
5.3. Code . . . . .	53
5.4. Fonctionnement de la simulation . . . . .	54
5.5. Sources . . . . .	56
6. Créer de nouveaux SVG . . . . .	56
6.1. Étape 1 : Choisir un modèle . . . . .	58
6.2. Étape 2 : Décalquer (optionnel) . . . . .	59
6.3. Étape 3 : Passage en SVG . . . . .	60
6.4. Étape 4 : Nettoyage du fichier . . . . .	62

6.5. Étape 5 : Ajouter le SVG au site : ..... 67

# 1. Introduction

Cette documentation a plusieurs objectifs :

- Permettre le déploiement de l'application

Voir : [Mise en place du serveur, Structure du site](#)

- Permettre de comprendre la structure de l'application

Voir : [Structure de l'application, Structure du site](#)

- Permettre la modification de l'existant

- Modifier les processus du site

Voir : [Les Menus, L'authentification, Les paramètres](#)

- Étendre les processus existants sans ajout de fonctionnalité

Voir : [Configuration, Créeer de nouveaux SVG](#)

- Permettre l'extension future de l'application et du site (ajout de fonctionnalités).

**NOTE**

Ce que nous appellerons "*Le Site*" est la partie web du projet (plus de détails [ici](#)).

# 2. Structure et fonctionnement

## 2.1. Structure de l'application

Afin de correctement décrire ce projet, nous pouvons le découper en trois grandes parties. Nous retrouvons ces trois parties dans le schéma des technologies ci-dessous (trois lignes). Nous regrouperons donc les technologies et les tâches à réaliser

dans ces trois catégories.

- **Acquisition** : Capteur Electrique (NKE TRIPHAS'O 60A), LoRaWAN
- **Stockage** : Node-Red, InfluxDB
- **Affichage / Interprétation** : PHP, Apache, Grafana

Ici, nous constatons que le MQTT n'est pas vraiment classable entre Acquisition et Stockage puisque par sa nature de protocole de communication, c'est une technologie de "transition".

Nous pouvons apporter un peu plus de détails quant aux rôles de ces différentes parties.

- **Acquisition** : Récupérer les données (consommation électrique) "sur le terrain" pour les rendre disponibles sur le réseau de l'IUT.
- **Stockage** : Lire les données envoyées sur le réseau et les insérer dans une base de données.
- **Affichage / Interprétation** : Collecter les données dans la base pour les présenter sous différentes formes et les présenter à l'utilisateur.

**NOTE**

"Interprétation" fait référence à des traitements faits sur les données et qui déclenchaient des actions en fonction.

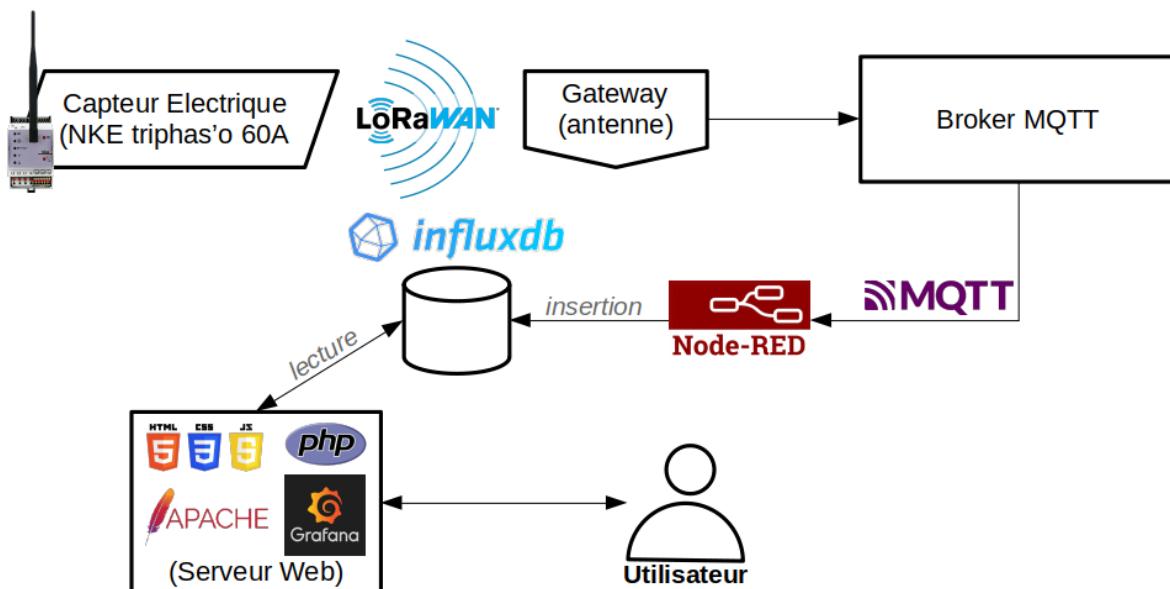


Figure 1. Schéma de répartition des technologies dans l'application

**NOTE**

Ce que nous appelerons "Le Site" dans cette documentation fait référence à la partie "Serveur Web" sur l'image ci-dessus.

## 2.1.1. Point sur les technologies

### 2.1.1.1. Capteur Electrique (NKE TRIPHAS'O 60A)

Le capteur Triphas'O permet la télé relève à distance via le réseau LoRaWAN®, des consommations d'énergies électriques d'une installation. Il est spécialement conçu pour répondre aux besoins de gestion d'énergie des bâtiments industriels, tertiaires, fonctionnant avec des équipements de moyenne et forte puissance et de forte consommation d'énergie. C'est pour ces raisons que ce capteur convient tout à fait à l'utilisation contrôle à distance d'une salle informatique.

### 2.1.1.2. LoRaWAN

La spécification LoRaWAN est un protocole de télécommunication LPWA (Low Power, Wide Area) conçu pour connecter sans fil des "objets" fonctionnant sur batterie et permettant de se connecter et de se connecter à Internet via des passerelles, participant ainsi à l'internet des objets.

Ce protocole se veut simple, peu coûteux à implémenter et économe en énergie. Le protocole LoRaWAN a pour but les communications longues portées à bas coût et basse consommation plutôt que les communications à débit élevé consommatrices en ressource CPU et en énergie. En effet, les défis concernant l'interconnexion des objets résident dans leur coût, leur autonomie ainsi que leur nombre d'un point de vue réseau.

En terme d'architecture, le réseau LoRaWAN est constitué de plusieurs passerelles permettant la communication avec les différents serveurs (par exemple ChirpStack) .

En ce qui concerne la portée, une seule passerelle LoRa peut recevoir et transmettre des signaux sur une distance de plus de 15 kilomètres dans les zones rurales. Même dans les environnements urbains denses, les messages peuvent parcourir jusqu'à cinq kilomètres.

En terme de capacité, un réseau LoRaWAN peut prendre en charge des millions de messages. Toutefois, le nombre de messages pris en charge dans un déploiement

donné dépend du nombre de passerelles installées.

### 2.1.1.3. MQTT

MQTT est un système de messagerie pour objets connectés, leur permettant d'envoyer des informations sur un sujet donné à un serveur qui fonctionne comme un broker de messages. Le broker publie ces informations sur des "topics" que les utilisateurs peuvent suivre en s'y abonnant. Ainsi, les utilisateurs abonnés à ces topics recevront les informations qu'il publie en temps réel.

*Exemple de trame récupéré sur le bus MQTT de l'IUT*

```
application/11/device/8553042fc3905153/rx {
    "applicationID": "11",
    "applicationName": "Chaput-Test",
    "deviceName": "ttgo-1",
    "devEUI": "8553042fc3905153",
    "rxInfo": [
        {
            "gatewayID": "77aaaa5500000001",
            "uplinkID": "ed98d035-c452-41ab-a494-6f73fe757767",
            "name": "",
            "rssi": -120,
            "loraSNR": -4,
            "location": null
        }
    ],
    "txInfo": {
        "frequency": 868300000,
        "dr": 0
    },
    "adr": true,
    "fCnt": 6450,
    "fPort": 1,
    "data": "RW5yZWdpC3RyZW1lbnQgZGUgbGEgZMOpbW9uc3RyYXRpb24="
}
```

Les trames qui circulent sur le bus MQTT prennent la forme de chaînes JSON comme l'exemple ci-dessus. Dans cette trame, la partie utile du dictionnaire JSON est indexée par la clé "*data*" et la valeur est codée en base 64. Ici, la valeur codée est : "RW5yZWdpC3RyZW1lbnQgZGUgbGEgZMOpbW9uc3RyYXRpb24=" et veut dire "Enregistrement de la démonstration". Nous pouvons également remarquer que la trame contient de nombreuses métadonnées comme la *gatewayID* : id de la "gateway" qui a réceptionnée le message ou le *deviceName* : nom de l'émetteur du message.

#### **2.1.1.4. Node-Red**

NodeRED est un environnement de programmation low-code pour les applications événementielles. Il utilise une méthode de programmation graphique basée sur les flux. Ainsi, il est possible via des blocs de code prédéfinis appelés "node" de constituer son programme en reliant les différents nodes . Node-RED a été développé en Javascript et est basé sur NodeJS

#### **2.1.1.5. InfluxDB**

InfluxDB est une time series database (TSDB). Elle est taillée pour stocker un large volume times series venant de différentes sources. Cette base de données vise à collecter le volume croissant de données issues de l'internet des objets et permet de gérer en temps réel les événements de tous ces systèmes.

#### **2.1.1.6. Grafana**

Grafana est un logiciel libre qui permet de générer des graphiques et des tableaux de bord à partir de bases de données de séries temporelles (time series database) tel que Influxdb.

#### **2.1.1.7. PHP**

PHP (Hypertext Preprocessor) est un langage de programmation libre, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP tel que Apache. Ce langage est principalement conçu pour servir de langage de script côté serveur, il est donc capable de faire tout ce que n'importe quel script CGI peut faire, comme collecter des données de formulaire, générer du contenu dynamique ou gérer des cookies et des sessions.

#### **2.1.1.8. Apache**

Le serveur HTTP Apache est un logiciel libre et gratuit qui permet aux utilisateurs de déployer leurs sites web sur Internet. Pour atteindre cet objectif, il agit comme un intermédiaire entre le serveur et les machines clientes. Il extrait le contenu du serveur à chaque demande de l'utilisateur et le diffuse sur le Web.

## 2.2. Structure du site

Avant tout, nous pouvons commencer par voir comment est structuré le site. Vous pouvez voir ci-dessous une représentation de la structure du site.

**NOTE** Ne sont représentés ici que les répertoires ainsi que les fichiers racines.

*Représentation de l'arborescence des fichiers du site*

```
/  
  -- auth.php  
  -- composer.json  
  -- composer.lock  
  -- composer.phar  
  -- config.php  
  -- index.php  
  --  
    -- grafana -> redirect/grafana/  
    -- nodered -> redirect/nodered/  
  --  
  -- controllers  
  --  
  -- model  
  --  
  -- redirect  
    -- grafana  
    -- nodered  
  --  
  -- sensible  
  --  
  -- static  
    -- img  
    -- include  
    -- js  
    -- style  
    -- svg  
  --  
  -- vendor  
    -- composer  
    -- guzzlehttp  
    -- influxdb  
    -- psr  
    -- ralouphie  
    -- symfony  
  --  
  -- vues
```

Table 1. Détails de la structure du site

Fichier / Répertoire	Rôle
auth.php	Le contrôleur d'authentification du site. Il n'est pas dans le répertoire contrôleur car nous voulons pouvoir directement y accéder avec une URL type "domain.fr/auth.php"
<ul style="list-style-type: none"> <li>• composer.json</li> <li>• composer.lock</li> <li>• composer.phar</li> </ul>	Des fichiers de configuration de composer installés en local. Composer est un utilitaire de gestion des librairies de php.
config.php	Le fichier de configuration du site : il sera détaillé dans la suite de cette documentation. <a href="#">Configuration</a>
index.php	Le routeur du site. Page sur laquelle nous tombons par défaut et qui selon la route indiquée en URL choisi le bon contrôleur. <a href="#">Paradigme MVC2</a>
<ul style="list-style-type: none"> <li>• grafana</li> <li>• nodered</li> </ul>	Des liens symboliques vers redirect/... Leur utilité est simplement que nous préférerons des URLs type domaine.fr/grafana plutôt que domaine.fr:3000 pour accéder à grafana par exemple.
contrôleurs/	Répertoire regroupant les "contrôleurs" du site (partie du MVC2 qui traite l'information et met en forme, utilise les modèles pour accéder aux données et les vues pour les afficher). <a href="#">Paradigme MVC2</a>

model/	Répertoire regroupant les "modèles" du site (partie du MVC2 qui récupère les données, ici requête vers InfluxDB, transmet aux contrôleurs). <a href="#">Paradigme MVC2</a>
• redirect/grafana • redirect/nodered	Des répertoires qui contiennent simplement des fichiers index.php qui redirigent vers grafana et node-red.
sensible/	Répertoire contenant tout ce qui est relatif à la sécurité du site (mots de passe). Il est protégé par des règles d'accès apache (.htaccess et .htpasswd).
static	<p>Contient toutes les ressources du site. Nous y trouvons différents répertoires :</p> <ul style="list-style-type: none"> <li>• img : les images (logo IUT)</li> <li>• include : les fichiers php inclus dans les autres pages du site (menu, vérification d'authentification)</li> <li>• js : les scripts inclus dans les pages du site (actions du menu)</li> <li>• style : les fichiers CSS</li> <li>• svg : les fichiers SVG utilisés ans les menus</li> </ul>
vendor	Répertoire contenant l'ensemble des librairies php importées avec Composer.

vues	Répertoire regroupant les "vues" du site (partie du MVC2 qui affiche les informations à l'utilisateur, met en forme les données traitées par le contrôleur) <a href="#">Paradigme MVC2.</a>
------	---

## 2.3. Paradigme MVC2

Le site est construit avec une structure type MVC2 (Routeur, Contrôleur, Modèle, Vue). Cela permet de correctement séparer les différentes couches de responsabilités entre les fichiers.

**NOTE** Nous n'avons pas employé de méthode de construction orienté objet. La "communication" entre les strats du modèle se fait donc soit par des **fonctions** soit avec des **include/require**

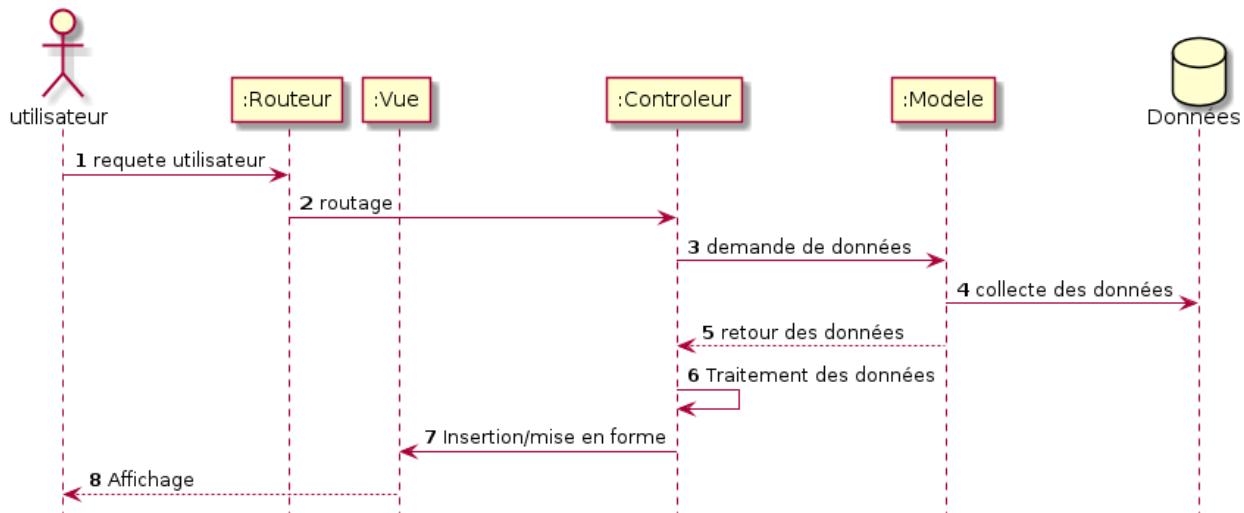


Figure 2. Diagramme de séquence type MVC2

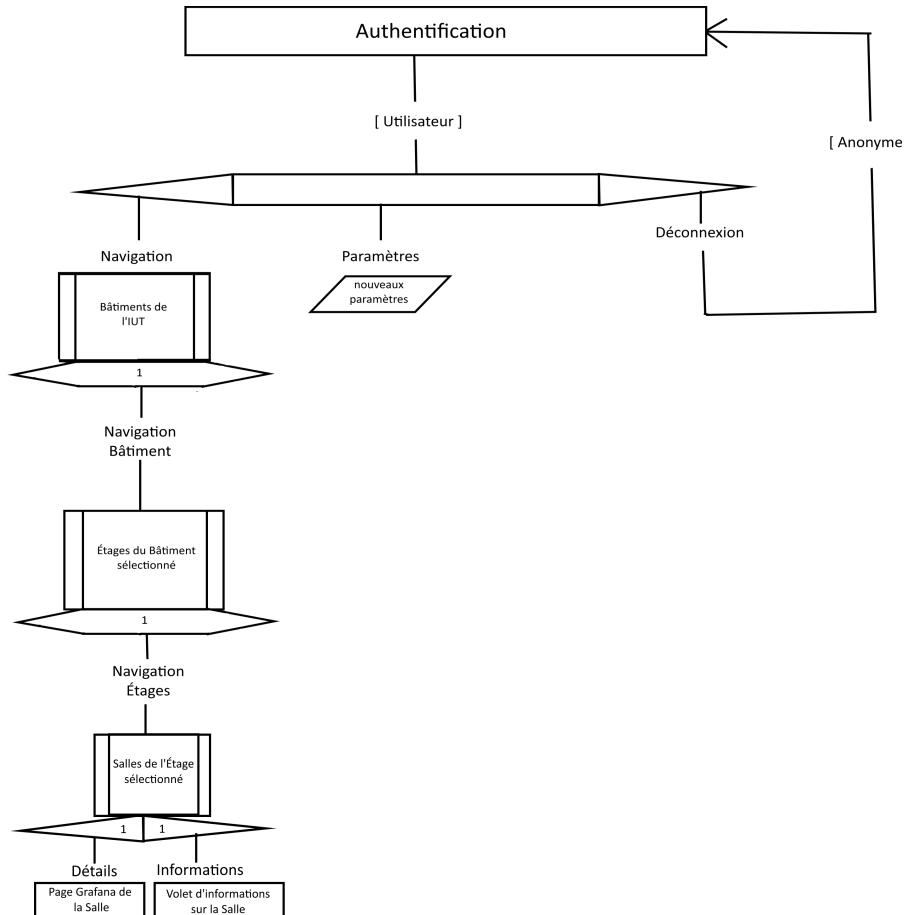
Comme présenté dans la partie [Structure du site](#), nous avons réuni les fichiers .php dans des répertoires selon leurs responsabilités.

## 2.4. Fonctionnalités du site

L'objectif du site est de visualiser les données. La partie PHP sert à naviguer et visualiser un résumé des données. La partie Grafana sert à consulter les données dans le détail.

Le site est plutôt petit, il n'y a pas beaucoup de voies de navigation. Sur le SNI ci-dessous nous pouvons voir se démarquer 3 grandes parties :

- la phase d'authentification : [L'authentification](#)
- la phase de navigation : [Les Menus](#)
- la phase de paramétrage : [Les paramètres](#)



*Figure 3. Schéma Navigationnel d'Interface du site*

## 2.4.1. L'authentification

Ici, le but est de vérifier si un utilisateur peut ou non accéder aux pages du site, dans l'optique de garder confidentielles les informations affichées.

Il y a ici deux processus en jeu :

- l'obtention de priviléges

- la vérification des priviléges

#### 2.4.1.1. L'obtention de priviléges

L'obtention des priviléges se fait via l'authentification. Le site compare le login et le hash du mot de passe saisis avec ceux stockés dans le fichier sensible/.mdp.json. Le hash et la vérification sont faits avec les méthodes php crypt() et hash\_equals(). Le salt actuel utilise l'algorithme blowfish.

Pour modifier les mots de passe, voir [Configurer l'authentification](#).

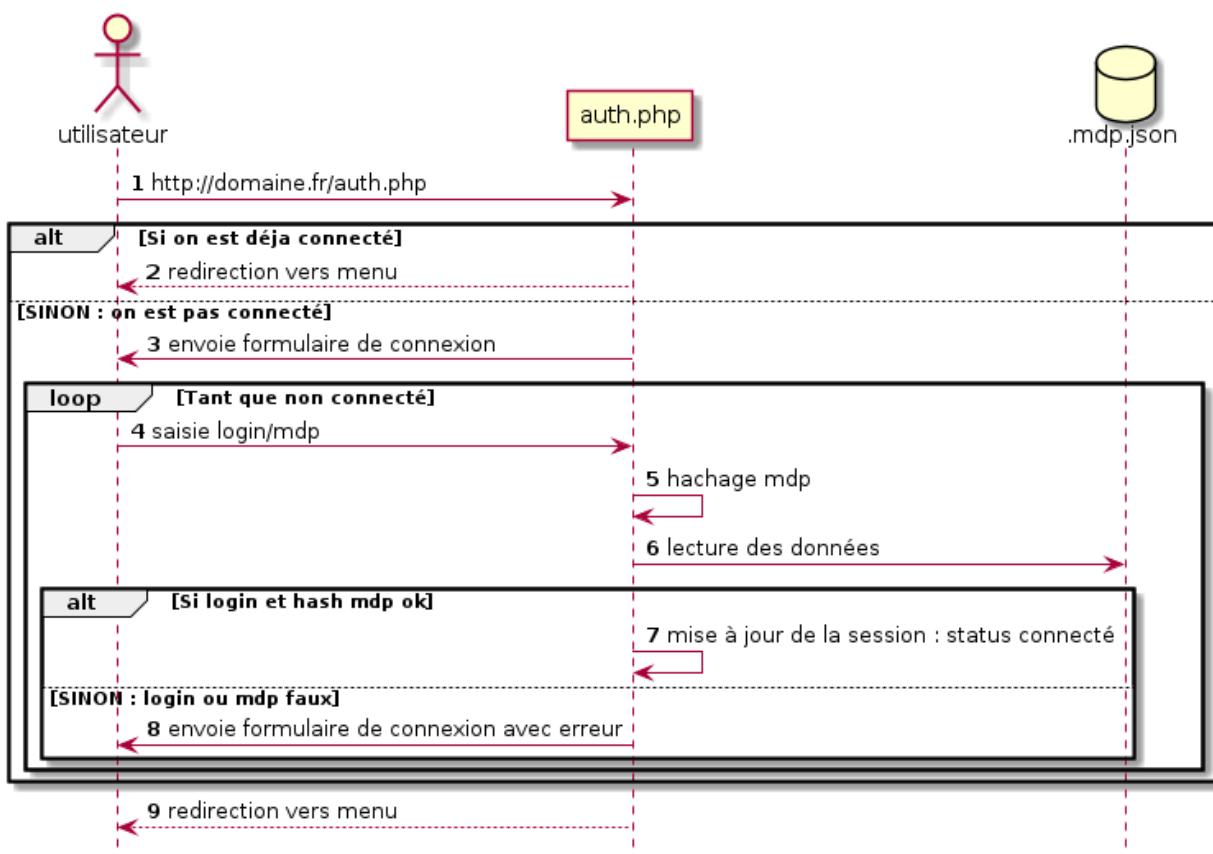


Figure 4. Diagramme de séquence du processus d'authentification

Nous vérifions si l'utilisateur est déjà connecté. Si oui, nous le redirigeons directement vers les menus. Si non, nous lui envoyons un formulaire d'authentification. Il le remplit, nous hashons le mot de passe et vérifions la correspondance avec ceux enregistrés. Tant que l'authentification échoue, le formulaire est renvoyé avec un message d'erreur. Quand l'authentification réussit, le site enregistre le login dans `$_SESSION['log']` ce qui équivaut à donner le statut "connecté". Enfin nous le redirigeons vers la page des menus. [Les Menus](#)

#### 2.4.1.2. La vérification des priviléges

Cette opération doit être faite pour chaque page du site (hormis la page d'identification). Pour cela un petit morceau de code est inclus au début de chaque fichier .php. Ce morceau de code se trouve dans /static/include/authCheck.php et est intégré avec la fonction php requier\_once().

Pour savoir si un utilisateur est connecté nous vérifions sa session. Si \$\_SESSION['log'] contient bien un login, alors il est connecté.

S'il n'est pas connecté, il sera alors redirigé vers la page de connexion.

#### 2.4.2. Les Menus

Il s'agit ici de la partie la plus importante du site. Sa complexité vient du fait que les menus peuvent être affichés sous deux formes différentes : images et tableaux (voir [Les paramètres](#) pour plus de détails). Mais dans le principe, peu importe le mode, nous voulons choisir un bâtiment, puis un étage de ce bâtiment et enfin une salle de cet étage pour finalement être redirigé vers la page grafana de cette salle.

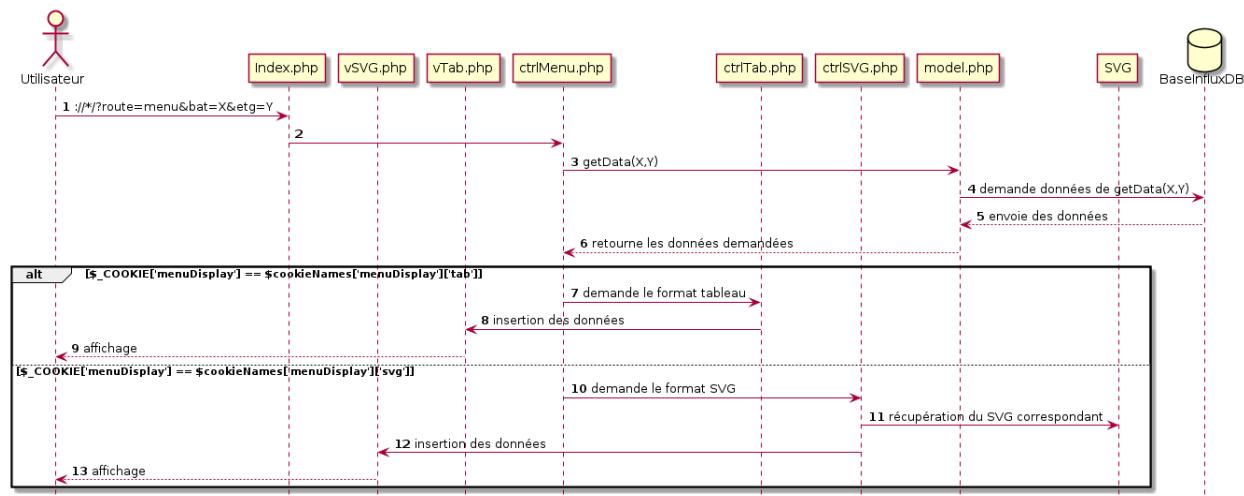


Figure 5. Diagramme de séquence d'affichage d'un menu

Dans le diagramme ci-dessus, l'utilisateur demande une URL qui passe 3 paramètres route=menu, bat=X et etg=Y. Ils signifient respectivement "nous voulons le menu M", "nous nous plaçons dans le bâtiment X" et "nous voulons l'étage Y" de ce bâtiment. Si etg n'est pas précisé alors le bâtiment sera affiché. Et si bat n'est pas précisé alors tout l'IUT sera affiché.

L'information du mode de menu souhaité est passé via cookie (voir [Gestion des cookies](#)).

Le routeur redirige vers le contrôleurMenu qui récupère les données et choisit le bon contrôleur selon le mode : contrôleurTab ou contrôleurSVG. Le contrôleur choisi, génère alors le contenu de la page (va chercher les SVG si nécessaire). Le contrôleur va alors insérer les données dans la vue qui lui correspond : VueTab ou VueSVG.

### 2.4.3. Les paramètres

Afin de rendre l'expérience plus agréable à l'utilisateur, nous lui laissons le choix du "mode de navigation" ainsi que du "thème d'affichage". Ces choix sont faits dans la page de paramètres via un formulaire.

The screenshot shows a dark-themed web application interface. At the top, the title 'CaptElec (v2)' is displayed next to the 'iut BLAGNAC' logo. Below the title is a navigation bar with three items: 'Navigation', 'Paramètres', and 'Déconnexion'. The 'Paramètres' item is currently selected, as indicated by its highlighted background. The main content area contains two sections: 'Navigation:' and 'Theme:'. Under 'Navigation:', there are two radio buttons: 'Mode tableaux' (selected, indicated by a blue dot) and 'Mode images' (not selected, indicated by a red outline). Under 'Theme:', there is a dropdown menu with the option 'Sombre' selected. At the bottom of the form is a large, prominent 'Valider' button.

Figure 6. Formulaire de paramètres

Ici, pas besoin de diagramme de séquence. L'utilisateur reçoit un formulaire pré-rempli avec ses paramètres actuels, il les modifie et soumet le formulaire. Nous parlons ici de paramètres qui s'appliquent à tout le site (navigation et style). De plus, nous voulions que ces paramètres soient conservés entre les visites. Les valeurs sont donc stockées sous forme de cookies (voir [Gestion des cookies](#))

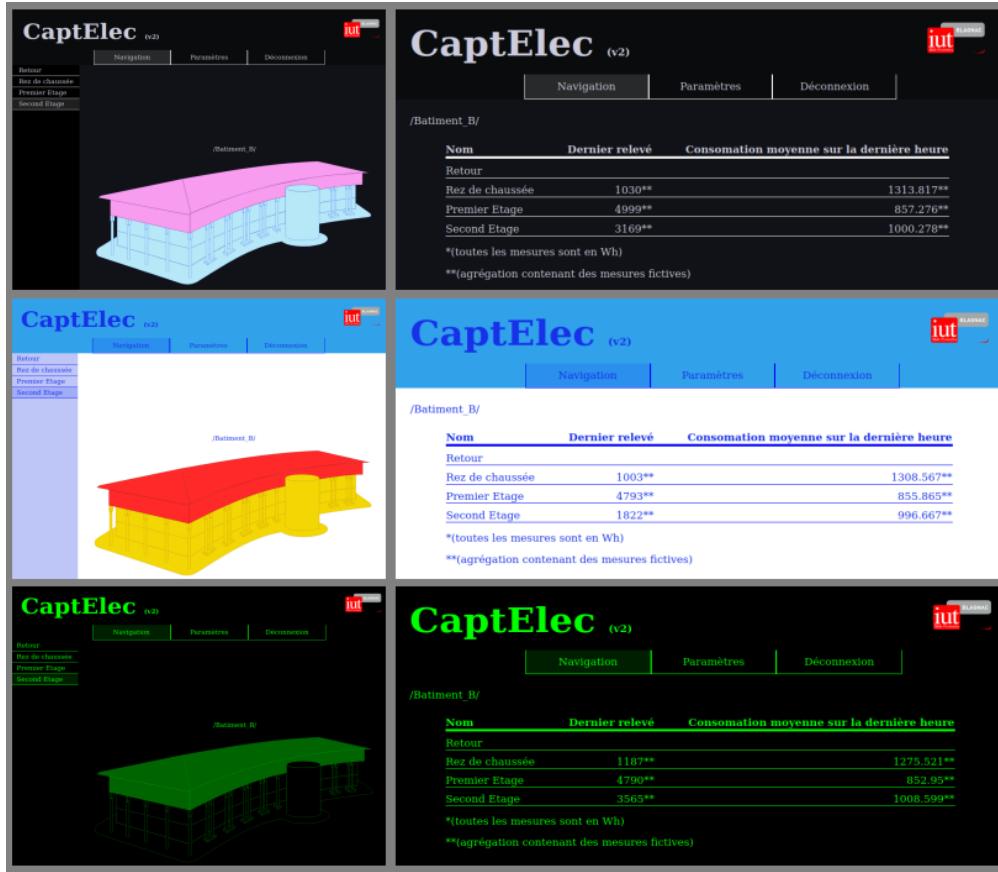


Figure 7. Vue regroupant tous les modes dans tous les thèmes d'affichage

### 3. Configuration

Nous avons essayé de rendre le site facilement configurable et ce, par deux moyens. Un fichier de configuration central `config.php` qui regroupe la plupart des propriétés du site. Et un dossier `/sensible` protégé qui contient le nécessaire pour gérer les connexions au site.

#### 3.1. Le fichier config.php

Le fichier de configuration est un fichier `php` situé à la racine du site. Il est protégé par l'authentification (voir [La vérification des privilèges](#)). Il contient des déclarations de variables utilisées dans tout le reste du site. Il est requis dans les autres fichiers du site : `require_once('config.php')`.

Nous allons détailler toutes ces variables, à quoi elles servent, et comment les modifier. Pour structurer le propos, nous allons les regrouper selon des groupes d'usage.

- ¬Les informations relatives à la base Influx → [Informations sur InfluxDB](#)
- ¬Les informations relatives au domaine URL → [Stockage des d'URL](#)
- ¬Les informations de couleurs pour les thèmes → [Mise en place des couleurs de thème](#)
- ¬Les informations relatives aux routes de navigation sur le site → [Informations de navigation/routes](#)
- ¬Les propriétés gestion des cookies → [Gestion des cookies](#)
- ¬La déclaration de la structure des menus → [Construction des menus](#)

**NOTE**

Les extraits de code donnés ci-après reflètent le fichier de config dans son état au moment de la rédaction de ce document. Si des changements sont appliqués, nous recommandons de modifier une copie de ce document pour y indiquer les modifications apportées.

**WARNING**

Nous recommandons également de faire une copie de sauvegarde de ce fichier avant toute modification.

### 3.1.1. Informations sur InfluxDB

Ici, nous indiquons les informations permettant de nous connecter à InfluxDB :

- l'adresse hôte : dbHost
- le numéro de port : dbPort
- le nom de la base : dbName
- le nom d'utilisateur : dbUsername
- le mot de passe : dbPapassword

#### Déclarations des variables InfluxDB

```
$dbName = 'CaptElec';
$dbPort = 8086;
$dbHost = '127.0.0.1';
$dbUsername = '*****';
$dbPapassword = '*****';
```

### 3.1.2. Stockage des d'URL

Ici, il s'agit principalement de processus automatiques qui stockent des valeurs permettant de construire les liens de navigation. Nous y trouvons également une variable contenant le lien à remplir pour accéder à la page grafana d'une salle.

**WARNING** Nous recommandons de modifier ces variables avec beaucoup de précaution, car elles sont utilisées dans la création de tous les liens de navigation interne.

#### *Déclaration des variables d'URL*

```
$currentURLRoute = (isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] ===
'on') ? 'https://': 'http://';
$currentURLRoute .= $_SERVER['HTTP_HOST'].$_SERVER['REQUEST_URI'];
$rootURL = str_replace(['index', 'auth', '.php/'], '', explode('?', $currentURLRoute)[0]);
$rootURL = str_replace(['.php/', '.php'], '', $rootURL);
$grafanaRoomPage = 'http://captelec.iut-blagnac.fr:3000/d/z1v-
Sk57k/consulter-une-salle?orgId=1&refresh=10s&var-Salle=';
```

### 3.1.3. Mise en place des couleurs de thème

Ici, nous déclarons une structure de données qui permet de définir les thèmes disponibles sur le site. Cette structure est un tableau associatif qui prend comme clé le nom du thème et pour valeur un autre tableau qui associe les propriétés graphiques avec la valeur souhaitée.

### *Exemple de déclaration de thème*

```
$themes = [
    '<nomTheme1>' => [
        '<propriété1>' => '<val1>',
        '<propriété2>' => '<val2>',
        ...
    ],
    '<nomTheme2>' => [
        '<propriété1>' => '<val1>',
        '<propriété2>' => '<val2>',
        ...
    ],
    ...
];
```

Cette structure a plusieurs niveaux de profondeur. Dans le listing qui suit, nous donnerons donc les propriétés comme suit [ 'X' ][ 'Y' ][ 'Z' ].

Propriété	Signification	Valeur possible
[ 'fg' ]	Couleur du texte sur la page	<ul style="list-style-type: none"> <li>• Code couleur hexa (#000 ou #000000)</li> </ul>
[ 'bg' ]	Couleur de fond par défaut de la page	<ul style="list-style-type: none"> <li>• nom de couleur connu par CSS (white, red, gold, teal, ...)</li> </ul>
[ 'border' ]	Couleur de bordure des éléments de la page (lignes de tableau)	<ul style="list-style-type: none"> <li>• fonction de couleur css (rgb(x, y, z) ou rgba(x, y, z, a))</li> </ul>
[ 'headerBg' ]	Couleur de fond de l'en-tête de la page	
[ 'sideMenuBG' ]	Couleur de fond du menu latéral dans le mode "images"	
[ 'fill' ]	Couleur de remplissage des zones des SVGs (par défaut)	
[ 'fillclickable' ]	Couleur de remplissage des zones "cliquables" sur les SVGs	
[ 'fillhighlight' ]	Couleur de remplissage des zones en surbrillance sur les SVGs	
[ 'stroke' ]	Couleur des traits sur les SVGs (par défaut)	
[ 'strokeclickable' ]	Couleur des traits des zones "cliquables" sur les SVGs	
[ 'strokehighlight' ]	Couleur des traits des zones en surbrillance des SVGs	
[ 'highlightTxtBg' ]	Couleur de fond des textes en surbrillance (menu ou tableau)	

Propriété	Signification	Valeur possible
[ 'gradient' ]	Règles autour des gradients de couleurs pour le mode DATA	Pas de valeur fixée, il s'agit ici de branches.
[ 'gradient' ][ 'max' ]	Valeurs des composantes RGB pour la valeur maximale du gradient indiquée pour une salle	
[ 'gradient' ][ 'min' ]	Valeurs des composantes RGB pour la valeur minimale du gradient indiquée pour une salle	
[ 'gradient' ][ 'default' ]	Couleur d'une salle si elle se trouve en dessous du seuil minimale de consommation	<ul style="list-style-type: none"> <li>• code hexa</li> <li>• nom couleur</li> <li>• fonction couleur</li> </ul>
<ul style="list-style-type: none"> <li>• [ 'gradient' ][ 'max' ][ 'R' ]</li> <li>• [ 'gradient' ][ 'max' ][ 'V' ]</li> <li>• [ 'gradient' ][ 'max' ][ 'B' ]</li> </ul>	Les composantes "rouge", "vert" et "bleu" pour la valeur maximale du gradient indiquée pour une salle	chaine de caractères représentant un entier entre 0 et 255 compris
<ul style="list-style-type: none"> <li>• [ 'gradient' ][ 'min' ][ 'R' ]</li> <li>• [ 'gradient' ][ 'min' ][ 'V' ]</li> <li>• [ 'gradient' ][ 'min' ][ 'B' ]</li> </ul>	Les composantes "rouge", "vert" et "bleu" pour la valeur maximale du gradient indiquée pour une salle	

Code actuel (3 themes)

```
$themes = [
    "Clair-bleu" => [
```

```

'bg' => '#c0d3ff',
'fg' => '#1731e9',
'border' => '#0000f7',
'headerBg' => '#31a2ea',
'sideMenuBG' => '#bec6f7',
'fill' => 'rgb(242, 215, 5)',
'stroke' => 'rgb(180, 82, 0)',
'strokehighlight' => 'rgb(50, 0, 0)',
'fillclickable' => 'rgb(242, 215, 5)',
'fillhighlight' => 'rgb(255, 41, 41)',
'strokeclickable' => 'rgb(180, 82, 0)',
'highlightTxtBg' => 'rgba(0, 33, 255, 0.15)',
'gradient' => [
  "min" => [
    'R' => "0",
    'V' => "255",
    'B' => "0",
  ],
  "max" => [
    'R' => "255",
    'V' => "0",
    'B' => "0",
  ],
  "default" => "rgb(50, 50, 50)"
]
],
"Sombre" => [
  'bg' => '#111217',
  'fg' => '#ccccdc',
  'headerBg' => '#0a0b0d',
  'sideMenuBG' => 'black',
  'border' => 'whitesmoke',
  'stroke' => 'rgb(0, 26, 255)',
  'fill' => 'rgb(183, 232, 247)',
  'strokehighlight' => 'rgb(56, 0, 102)',
  'strokeclickable' => 'rgb(0, 26, 255)',
  'fillclickable' => 'rgb(183, 232, 247)',
  'fillhighlight' => 'rgb(247, 156, 239)',
  'highlightTxtBg' => 'rgba(255, 255, 255, 0.12)',
  'gradient' => [
    "min" => [
      'R' => "0",
      'V' => "255",
      'B' => "0",
    ],
    "max" => [
      'R' => "255",
      'V' => "0",
      'B' => "0",
    ],
  ],
]

```

```

        "default" => "rgb(250, 250, 250)"
    ]
],
"Matrixx-flat" => [
    'fg' => 'lime',
    'bg' => 'black',
    'stroke' => 'lime',
    'border' => 'lime',
    'headerBg' => 'black',
    'sideMenuBG' => 'black',
    'fill' => 'rgb(0, 0, 0)',
    'strokeclickable' => 'lime',
    'strokehighlight' => 'lime',
    'fillhighlight' => 'darkgreen',
    'fillclickable' => 'rgb(0, 0, 0)',
    'highlightTxtBg' => 'rgba(0, 255, 0, 0.12)',
    'gradient' => [
        "min" => [
            'R' => "0",
            'V' => "255",
            'B' => "0",
        ],
        "max" => [
            'R' => "255",
            'V' => "0",
            'B' => "0",
        ],
        "default" => "rgba(0, 0, 0, 0"
    ]
]
];

```

### 3.1.4. Informations de navigation/routes

Il est ici question de gérer les routes existantes, les liens de navigation et les noms des champs d'URL.

## Déclaration des variables de navigation

```
$urlAttr = [];
$urlAttr['atRoute'] = 'route'; //contient le nom de la page demandée
$urlAttr['atBatiment'] = 'bat'; //contient le batiment demandé
$urlAttr['atEtage'] = 'etg'; //contient l'étage demandé

$routes = [];
$routes['rtSett'] = 'sett'; //contient la chaine identifiant la route
vers la page de settings
$routes['rtMenu'] = 'menu'; //contient la chaine identifiant la route
vers la page des menus
$routes['rtMode'] = 'mode'; //contient la chaine identifiant la route
vers les modeles
$routes['rtDeco'] = 'deco'; //contient la chaine identifiant la route
vers la page de déconnexion

$navLinks = '"btNavig" :
'.'. $rootURL.'index.php/?route='.$routes['rtMenu'].',';
$navLinks .= '"btParam" :
'.'. $rootURL.'index.php/?route='.$routes['rtSett'].',';
$navLinks .= '"btDeco" :
'.'. $rootURL.'index.php/?route='.$routes['rtDeco'].',';
```

Nous avons trois variables :

- `$urlAttr` la liste associative des champs d'URL que le site peut atteindre. Nous déclarons ainsi :

exemple : domain.fr/index.php?truc=x → `$urlAttr['champTruc'] = truc`

Nous donnons en clé le nom que nous utiliserons dans le code, et en valeur la chaîne qui représentera le champs dans les URLs.

- `$routes` la liste associative des valeurs possibles pour le champs d'URL `$urlAttr['atRoute']`. Comme pour `$urlAttr` nous donnons comme clé le nom de constante utilisé dans le code et comme valeur, la chaine qui sera passé à l'attribut.
- `$navLinks` une chaîne de caractères format JSON qui donne les liens pour les boutons du menu.

### 3.1.5. Gestion des cookies

Le site utilise des cookies pour conserver des paramètres au fil des connexions.

Nous définissons :

- `$cookieNames` la liste associative qui prend comme clé le nom du cookie et comme valeur un autre tableau associatif qui associe des noms à des valeurs avec `default` clé ayant pour valeur une autre clé de ce même tableau.
- `$cookieLifeTime` la durée de vie de nos cookies en seconde.

*définition des variables de gestion des cookies*

```
$cookieNames = [ ];
$cookieNames['menuDisplay'] = [
    'default' => 'tab',
    'tab' => 'displayTab',
    'svg' => 'displaySVG'
];
$cookieNames['themUsed'] = ['default' => array_key_first($themes)];
foreach ($themes as $key => $value) {
    $cookieNames['themUsed'][$key] = $key;
}

$cookieLifeTime = 3600*24*30;
```

### 3.1.6. Construction des menus

Enfin nous devons indiquer au site comment construire ses menus.

D'abord nous définissons `$svgIUT` qui contient le nom du svg représentant l'IUT soit la racine de nos menus.

Puis nous définissons `$menu` dont nous donnons le modèle ci-dessous :

### *légende du modèle*

Les éléments entre chevrons (<element>) sont des champs auxquels nous donnerons une valeur et les éléments entre crochets ([element]) sont des valeurs constantes.

Nous pouvons donc traduire une structure php telle `$a = [$x-[ $y-1 , $z-2 ]]`; en une représentation arborecente comme ci-dessous.

### **TIP**

#### *Exemple de traduction arborecente*

```
$a
  >>> <cleX>
    >>> <cleY>
      >   >>> [1]
    >>> <cleZ>
      >>> [2]
```

## Modèle de la variable menu

```
$menu
  <labelBatiement>
    ["id"]
    <idBatSurSVG>
    ["image"]
    <nomFichierSVG>
    ["requete"]
    <regexQualifiantTouesLesSallesDuBatiment>
    ["etages"]
      <labelEtage>
        ["id"]
        <idEtageSurSVG>
        ["image"]
        <nomFichierSVG>
        ["requete"]
        <regexQualifiantTouesLesSallesDeLetage>
        ["salles"]
          <idSalleSurSVG>
          ["nom"]
          <labelSalle>
          ["range"]
          ["min"]
          <valeurMinimaleDeGradient>
          ["max"]
          <valeurMaximaleDeGradient>
          ...
        ...
      ...
    ...
  ...
...
```

## code de déclaration de \$menu

```
$menu = [
  'Batiment_A-admin' => 'WIP',
  'Batiment_A-bibli' => 'WIP',
  'Batiment_C-locTech' => 'WIP',
  'Batiment_E' => 'WIP',
  'Maison_Intelligente' => 'WIP',

  'Batiment_B' => [
    'id' => 'bat_B',
    'image' => 'bat_B.svg',
    'requete' => '/B_[0-9abc]*/' ,

    'etages' => [
      'Rez de chaussée' => [
        'id' => 'rdc',
        'image' => 'rdc_B.svg' ,
```

```

'requete' => '/B_0[0-9abc]*/' , 

'salles' => [
    'b001' => [ 'nom' => 'B_001', 'WIP' => true],
    'b002' => [ 'nom' => 'B_002', 'WIP' => true],
    'b003' => [ 'nom' => 'B_003', 'WIP' => true],
    'b004' => [ 'nom' => 'B_004', 'WIP' => true],
    'b005' => [ 'nom' => 'B_005', 'WIP' => true],
    'b006' => [ 'nom' => 'B_006', 'WIP' => true],
    'b007' => [ 'nom' => 'B_007', 'range' => [ 'min' =>
20, 'max' => 4000]], 
    'b008' => [ 'nom' => 'B_008', 'WIP' => true],
    'b009' => [ 'nom' => 'B_009', 'WIP' => true],
    'b010' => [ 'nom' => 'B_010', 'WIP' => true],
    'magasin' => [ 'nom' => 'Magasin', 'WIP' => true],
    'cafet' => [ 'nom' => 'Caféteria', 'WIP' => true],
    'san' => [ 'nom' => 'Sanitaires', 'WIP' => true],
    'rgt' => [ 'nom' => 'Rangement', 'WIP' => true]
]
], 
'Premier Etage' => [
    'id' => 'et1',
    'image' => 'et1_B.svg',
    'requete' => '/B_1[0-9abc]*/' , 

    'salles' => [
        'b101' => [ 'nom' => 'B_101', 'range' => [ 'min' =>
20, 'max' => 3000]], 
        'b102' => [ 'nom' => 'B_102', 'range' => [ 'min' =>
20, 'max' => 3000]], 
        'b103' => [ 'nom' => 'B_103', 'range' => [ 'min' =>
20, 'max' => 3000]], 
        'b104' => [ 'nom' => 'B_104', 'range' => [ 'min' =>
20, 'max' => 3000]], 
        'b105' => [ 'nom' => 'B_105', 'range' => [ 'min' =>
20, 'max' => 3000]], 
        'b106' => [ 'nom' => 'B_106', 'range' => [ 'min' =>
20, 'max' => 3000]], 
        'b107' => [ 'nom' => 'B_107', 'WIP' => true],
        'b108' => [ 'nom' => 'B_108', 'WIP' => true],
        'b109' => [ 'nom' => 'B_109', 'WIP' => true],
        'b109b' => [ 'nom' => 'B_109b', 'WIP' => true],
        'b110' => [ 'nom' => 'B_110', 'WIP' => true],
        'b111' => [ 'nom' => 'B_111', 'WIP' => true],
        'b112' => [ 'nom' => 'B_112', 'WIP' => true],
        'b112b' => [ 'nom' => 'B_112b', 'WIP' => true],
        'b113' => [ 'nom' => 'B_113', 'WIP' => true],
        'b115' => [ 'nom' => 'B_115', 'WIP' => true],
        'b116a' => [ 'nom' => 'B_116a', 'WIP' => true],
        'b116b' => [ 'nom' => 'B_116b', 'WIP' => true],
    ]
]
]
```

```

'sousStation' => [ 'nom' => 'Sous-Station', 'WIP'
=> true],
      'san' => [ 'nom' => 'Sanitaires', 'WIP' => true],
      'rgt' => [ 'nom' => 'Rangement', 'WIP' => true]
    ]
  ],
  'Second Etage' => [
    'id' => 'et2',
    'image' => 'et2_B.svg',
    'requete' => '/B_2[0-9abc]*/',
    'salles' => [
      'b201' => [ 'nom' => 'B_201', 'WIP' => true],
      'b202' => [ 'nom' => 'B_202', 'WIP' => true],
      'b203' => [ 'nom' => 'B_203', 'WIP' => true],
      'b204' => [ 'nom' => 'B_204', 'WIP' => true],
      'b205' => [ 'nom' => 'B_205', 'WIP' => true],
      'b206' => [ 'nom' => 'B_206', 'WIP' => true],
      'b207' => [ 'nom' => 'B_207', 'WIP' => true],
      'b208' => [ 'nom' => 'B_208', 'WIP' => true],
      'b209' => [ 'nom' => 'B_209', 'WIP' => true],
      'b210' => [ 'nom' => 'B_210', 'WIP' => true],
      'b211' => [ 'nom' => 'B_211', 'WIP' => true],
      'b212a' => [ 'nom' => 'B_212a', 'WIP' => true],
      'b212b' => [ 'nom' => 'B_212b', 'WIP' => true],
      'b213' => [ 'nom' => 'B_213', 'WIP' => true],
      'b214' => [ 'nom' => 'B_214', 'WIP' => true],
      'b215' => [ 'nom' => 'B_215', 'WIP' => true],
      'b216' => [ 'nom' => 'B_216', 'WIP' => true],
      'b219' => [ 'nom' => 'B_219', 'range' => [ 'min' =>
20, 'max' => 5000]],

      'b220' => [ 'nom' => 'B_220', 'WIP' => true],
      'b221' => [ 'nom' => 'B_221', 'WIP' => true],
      'b222' => [ 'nom' => 'B_222', 'WIP' => true],
      'b223' => [ 'nom' => 'B_223', 'WIP' => true],
      'b224' => [ 'nom' => 'B_224', 'WIP' => true],
      'b225' => [ 'nom' => 'B_225', 'WIP' => true],
      'b226' => [ 'nom' => 'B_226', 'WIP' => true],
      'b227' => [ 'nom' => 'B_227', 'WIP' => true],
      'b228a' => [ 'nom' => 'B_228a', 'WIP' => true],
      'b228b' => [ 'nom' => 'B_228b', 'WIP' => true],
      'b228c' => [ 'nom' => 'B_228c', 'WIP' => true],
      'b229' => [ 'nom' => 'B_229', 'WIP' => true],
      'b230' => [ 'nom' => 'B_230', 'WIP' => true],
      'b231' => [ 'nom' => 'B_231', 'WIP' => true],
      'b232' => [ 'nom' => 'B_232', 'WIP' => true],
      'b233' => [ 'nom' => 'B_233', 'WIP' => true],
      'b234' => [ 'nom' => 'B_234', 'WIP' => true],
      'b235a' => [ 'nom' => 'B_235', 'WIP' => true],
      'b235b' => [ 'nom' => 'B_235', 'WIP' => true],
    ]
  ]
]

```

```

        'san' => [ 'nom' => 'Sanitaires', 'WIP' => true],
        'rgt' => [ 'nom' => 'Rangement', 'WIP' => true],
        'balcon' => [ 'nom' => 'Balcon', 'WIP' => true],
    ]
]
],
];

```

**NOTE** Nous pouvons remarquer la mention "WIP" ↗ true par endroits dans le code ci-dessus. Cette mention sert à indiquer qu'une entrée est toujours "en construction". L'objectif est de pouvoir créer toutes les entrées possibles pour un niveau mais sans nécessairement que celles-ci soient prises en compte.

## 3.2. Configurer l'authentification

Le dossier /sensible contient plusieurs fichiers :

- .htaccess ↗ un fichier contenant les règles d'accès au répertoire /sensible`
- .htpasswd ↗ un fichier contenant les login/hash d'accès au répertoire /sensible`
- .salt ↗ un fichier contenant le "grain de sel" du site qui permet de rendre les hash vraiment spécifiques au site
- .mdp.json ↗ un fichier json contenant un dictionnaire tel :

```
{
    "login1" : "hash1",
    "login2" : "hash2",
    ...
}
```

- genHash.php ↗ une petite page qui contient un formulaire permettant de générer une nouvelle ligne pour .mdp.json.

Nou pouvons ajouter de nouveaux utilisateurs au site en ajoutant une ligne à .mdp.json. Si nous changeons le salt alors tous les hash devrons êtres regénérés.

Pour salt et hash voir <https://www.php.net/manual/fr/function.crypt.php>

```
"admin_I_strateur" : "$2y$15$zerjiocAptEleczeqn464OF3Wi0SN.HkrwxUcZ5RqhSsxFV9vhwya",
```

admin_I_strateur
mdpAdmin
générer

Figure 8. Page de génération de hash pour nouveau login (genHash.php)

## 4. Mise en place du serveur

### 4.1. Introduction

Nous travaillons sur une machine Ubuntu Server LTS 20.04 dont nous ne détaillerons pas l'installation ici. Vous pouvez vous reporter au lien suivant : <https://www.linuxtechi.com/ubuntu-20-04-lts-server-installation-guide/>.

Pour les tests, toutes les URLs sont données comme locales (127.0.0.1). Bien évidemment, il faut remplacer cela par l'adresse de votre site (serveur).

Nous allons effectuer beaucoup d'installations donc nous commençons par mettre à jour les dépôts et applications déjà présentes.

```
sudo apt update  
sudo apt upgrade
```

### 4.2. Installation de NodeRed

#### 4.2.1. Installation

##### 4.2.1.1. Node

Nous commençons par installer Node.js dans sa version 14.x comme recommandé ici <https://nodered.org/docs/faq/node-versions>

*Installation de Node.js 14.x*

```
curl -fsSL https://deb.nodesource.com/setup_14.x | sudo -E bash -  
sudo apt-get install -y nodejs
```

#### 4.2.1.2. Node-RED

*Installation de Node-RED via npm (gestionnaire de packages de Node)*

```
sudo npm install -g --unsafe-perm node-red
```

Nous pouvons vérifier que tout soit bien installé en :

- ¬ regardant la version de Node.js
- ¬ regardant la version de npm
- ¬ démarrant le service Node-RED

```
v14.18.1
user@vm-CaptElec:~$ npm -v
6.14.15
user@vm-CaptElec:~$ node-red -v
3 Nov 20:03:04 - [info]

Welcome to Node-RED
=====
3 Nov 20:03:04 - [info] Node-RED version: v2.1.3
3 Nov 20:03:04 - [info] Node.js version: v14.18.1
3 Nov 20:03:04 - [info] Linux 5.11.0-34-generic x64 LE
3 Nov 20:03:05 - [info] Loading palette nodes
3 Nov 20:03:05 - [info] Settings file : /home/user/.node-red/settings.js
3 Nov 20:03:05 - [info] Context store : 'default' [module=memory]
3 Nov 20:03:05 - [info] User directory : /home/user/.node-red
3 Nov 20:03:05 - [warn] Projects disabled : editorTheme.projects.enabled=false
3 Nov 20:03:05 - [info] Flows file : /home/user/.node-red/flows.json
3 Nov 20:03:05 - [info] Creating new flow file
3 Nov 20:03:05 - [warn]

-----
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
-----

3 Nov 20:03:05 - [info] Server now running at http://127.0.0.1:1880/
3 Nov 20:03:05 - [info] Starting flows
3 Nov 20:03:05 - [info] Started flows
```

Figure 9. Vérifications de la bonne installation de Node

**NOTE**

Avec le service démarré nous pouvons dès à présent ouvrir Node-RED dans le navigateur à l'url <http://127.0.0.1:1880/>.

**NOTE**

CTRL+C pour arrêter le service.

#### 4.2.1.3. Node-RED Admin et Configuration user

À des fins de sécurité, nous allons maintenant installer l'outil Node-RED Admin. Nous allons également générer le hash de mot de passe pour le compte admin

##### *Installation Node-RED Admin*

```
sudo npm install -g --unsafe-perm node-red-admin  
node-red admin hash-pw
```

**WARNING**

Le login/mdp admin actuel pour NodeRed est admin/admin. Il devra donc être changé lors de la mise en production.

**NOTE**

Le hash :  
  
\$2b\$08\$VG0y4gKFB50Quai75WOfzOs7fjVVRCO5d3W1sYDWsdvalbU  
41ZqoK

Nous plaçons ce hash dans le fichier de configuration de NodeRed (settings file dans la capture précédente) `/home/user/.node-red/settings.js` . Nous décommentons la section suivante, et nous la modifions avec les données qui nous intéressent :

## Fichier settings.js

```
*****
 * Security
 * - adminAuth
 * - https
 * - httpsRefreshInterval
 * - requireHttps
 * - httpNodeAuth
 * - httpStaticAuth
*****
*/
/** To password protect the Node-RED editor and admin API, the
following
 * property can be used. See
http://nodered.org/docs/security.html for details.
 */
adminAuth: {
    type: "credentials",
    users: [
        {
            username: "admin",
            password:
"2b$08$VG0y4gKFB50Quai75WOfzOs7fjVVRCO5d3W1sYDWsdvalbU41ZqoK",
            permissions: "*"
        },
        {
            username: "guest",
            password:
"2b$08$1z5p7eB9qn037dN4r0QrouPhwpV0juvcYBMZo26KzHxKk1Rcinnou",
            permissions: "read"
        }
    ],
}
```

- Nous avons bien notre utilisateur admin.
- Nous avons également ajouter un utilisateur guest (mdp: hello) avec les droits de lecture uniquement

Maintenant, si nous redémarrons le service nodeRED et que nous l'ouvrons dans un navigateur (<http://127.0.0.1:1880/>), nous sommes maintenant accueillis par une page de connexion.

#### 4.2.1.4. Packet Node-RED pour InfluxDB

Nous avons node-RED, il est correctement installé et sécurisé avec des users. Toutefois nos chartFlows ne peuvent toujours pas communiquer avec une base influxDB car nous n'avons pas installé le packet nécessaire.

Pour réaliser cette commande, il faut se placer à la racine de notre installation Node-RED.

**WARNING** Ici `/home/user/.node-red`. Pour trouver le chemin nous pouvons nous servir de `settings.js` qui se trouve à la racine souhaitée (cf [\[trouverFichierSettings\]](#))

→*Installation du packet pour la communication influxDB <→ Node-RED*

```
cd /path/.node-red  
npm install node-red-contrib-influxdb
```

#### 4.2.1.5. Lancer Node-RED au démarrage

Il faut créer un nouveau fichier `systemd nodered.service`

*Création du fichier*

```
sudo nano /etc/systemd/system/nodered.service
```

Ensute nous copions le contenu suivant dans le fichier :

### *Contenu du fichier nodeRed.service*

```
[Unit]
Description=Node-RED
After=syslog.target network.target

[Service]
ExecStart=/usr/local/bin/node-red --max-old-space-size=128 -v
Restart=on-failure
KillSignal=SIGINT

# log output to syslog as 'node-red'
SyslogIdentifier=node-red
StandardOutput=syslog

# non-root user to run as
WorkingDirectory=/home/user/
User=user
Group=user

# if using a root user
#WorkingDirectory=/root/
#User=root
#Group=root

[Install]
WantedBy=multi-user.target
```

#### **NOTE**

Il faut penser à remplacer user par le nom d'utilisateur qui exécute nodeRED ou décommenter la partie root si c'est root qui l'exécute.

Il faut maintenant activer le lancement automatique

### *Activation du lancement automatique de NodeRED*

```
sudo systemctl enable nodered.service
```

## **4.2.2. Sources**

<https://www.arubacloud.com/tutorial/how-to-install-node-red-on-ubuntu-20-04.aspx>  
<https://nodered.org/docs/getting-started/local>  
<https://github.com/nodesource/distributions/blob/master/README.md#debinstall>  
<https://flows.nodered.org/node/node-red-contrib-influxdb>

<https://randomnerdtutorials.com/access-node-red-dashboard-anywhere-digital-ocean/>

## 4.3. Installation de InfluxDB

Maintenant que NodeRED est installé nous devons pouvoir stocker les données qu'il génère, pour cela nous devons installer une base InfluxDB.

### 4.3.1. Installation

Avant de procéder à l'installation, il nous faut ajouter le package InfluxDB

#### Ajout du package influxDB

```
wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add -
source /etc/lsb-release
echo "deb https://repos.influxdata.com/${DISTRIB_ID,,}
${DISTRIB_CODENAME} stable" | sudo tee
/etc/apt/sources.list.d/influxdb.list
```

Nous pouvons maintenant procéder à l'installation. Ici, nous mettons à nouveau à jour les packets apt puisque ils viennent d'en ajouter.

#### Installation d'influxDB

```
sudo apt-get update
sudo apt-get install influxdb
sudo service influxdb start
```

**NOTE** Nous finissons par démarrer influxDB pour vérifier sa bonne installation.

### 4.3.2. Configuration

Nous pouvons afficher la configuration d'influx

#### Configuration influxDB

```
influxd -config /etc/influxdb/influxdb.conf
```

```

user@vm-CaptElec:~$ influxd -config /etc/influxdb/influxdb.conf

88888888          .d888 888          88888888b. 8888888b.
 888      d88P" 888          888  "Y88b 888  "88b
 888      888 888          888 888 888 .88P
 888 888888b. 8888888 888 888 888 888 888 8888888K.
 888 888 "88b 888 888 888 Y8bd8P' 888 888 888 "Y88b
 888 888 888 888 888 X88K 888 888 888 888
 888 888 888 888 888 Y88b 888 .d8""8b. 888 .d88P 888  d88P
88888888 888 888 888 888 "Y88888 888 888 8888888P" 8888888P"
2021-11-04T14:15:44.789433Z  info  InfluxDB starting      {"log_id": "0XbpjTaG000", "version": "1.8.10", "branch": "1.8", "commit": "688e697c51fd"}
2021-11-04T14:15:44.789475Z  info  Go runtime       {"log_id": "0XbpjTaG000", "version": "go1.13.8", "maxprocs": 8}
run: open server: listen: listen tcp 127.0.0.1:8088: bind: address already in use
user@vm-CaptElec:~$ 

```

Figure 10. Résultats de configuration influxDB

### TIP

Nous pouvons remarquer que le fichier de configuration que nous pourrons modifier se trouve à `/etc/influxdb/influxdb.conf`

#### 4.3.2.1. Réseau

Par défaut InfluxDB utilise les ports :

- TCP 8086 pour la communication client-serveur par l'API HTTP d'influxDB
- TCP 8088 pour les services de backup/restoration RPC

Nous allons ouvrir ces ports aux connexions extérieures vers InfluxDB.

#### Ouverture du port 8086

```

sudo ufw enable
sudo ufw allow 8086/tcp

```

#### 4.3.2.2. User et base de données

Nous allons maintenant nous connecter, créer un nouvel utilisateur et une nouvelle base de données.

#### Création d'un nouvel utilisateur influxDB

```

curl -XPOST "http://localhost:8086/query" \
--data-urlencode "q=CREATE USER admin WITH PASSWORD 'admin' WITH ALL
PRIVILEGES"

```

**WARNING**

Ici, nous créons par requête http un nouvel utilisateur admin/admin qui devra être modifié lors de la mise en production.

Nous pouvons maintenant nous connecter avec notre utilisateur fraîchement créé.

*Connexion à un user InfluxDB*

```
influx -username 'admin' -password 'admin'
```

Nous avons maintenant accès au shell d'influxDB

```
user@vm-CaptElec:~$ influx -username 'admin' -password 'admin'  
Connected to http://localhost:8086 version 1.8.10  
InfluxDB shell version: 1.8.10  
> █
```

Figure 11. Capture shell influxDB

Enfin nous pouvons créer une nouvelle base

*Création d'une base dans InfluxDB*

```
CREATE DATABASE CaptElec_sim  
SHOW DATABASES
```

### 4.3.3. Sources

<https://otodiginet.com/database/how-to-install-influxdb-on-ubuntu-20-04-lts/>

## 4.4. Installation Grafana

### 4.4.1. Installation

Pour installer Grafana nous allons d'abord devoir ajouter le package software-properties-common, mais également ajouter la clé GPG de Grafana à apt (sinon nous n'aurons pas les dernières mises à jour).

## Ajout de software-properties-common et de la clé GPG grafana

```
sudo apt-get install -y gnupg2 curl software-properties-common  
curl https://packages.grafana.com/gpg.key | sudo apt-key add -  
sudo add-apt-repository "deb https://packages.grafana.com/oss/deb  
stable main"
```

Nous pouvons maintenant procéder à l'installation (nous mettons à jour apt puisque nous venons d'ajouter la clé GPG de grafana).

## Installation de Grafana

```
sudo apt-get update  
sudo apt-get install grafana
```

Afin de vérifier que l'installation est bonne, nous pouvons maintenant lancer le service grafana

## Démarrage du service Grafana

```
sudo systemctl start grafana-server
```

## Supervision du service grafana

```
systemctl status grafana-server.service
```

```
user@vm-CaptElec:~$ sudo systemctl start grafana-server  
user@vm-CaptElec:~$ sudo systemctl status grafana-server  
● grafana-server.service - Grafana instance  
   Loaded: loaded (/lib/systemd/system/grafana-server.service; enabled; vendor preset: enabled)  
   Active: active (running) since Thu 2021-11-04 18:32:33 CET; 28s ago  
     Docs: http://docs.grafana.org  
    Main PID: 15388 (grafana-server)  
      Tasks: 12 (limit: 4649)  
     Memory: 28.4M  
    CGroup: /system.slice/grafana-server.service  
           └─15388 /usr/sbin/grafana-server --config=/etc/grafana/grafana.ini --pidfile=/run/grafan  
  
nov. 04 18:32:33 vm-CaptElec grafana-server[15388]: t=2021-11-04T18:32:33+0100 lvl=info msg="Path Pr>  
nov. 04 18:32:33 vm-CaptElec grafana-server[15388]: t=2021-11-04T18:32:33+0100 lvl=info msg="App mod>  
nov. 04 18:32:33 vm-CaptElec grafana-server[15388]: t=2021-11-04T18:32:33+0100 lvl=info msg="Connect>  
nov. 04 18:32:33 vm-CaptElec grafana-server[15388]: t=2021-11-04T18:32:33+0100 lvl=info msg="Startin>  
nov. 04 18:32:33 vm-CaptElec grafana-server[15388]: t=2021-11-04T18:32:33+0100 lvl=info msg="migrati>  
nov. 04 18:32:33 vm-CaptElec grafana-server[15388]: t=2021-11-04T18:32:33+0100 lvl=info msg="Startin>  
nov. 04 18:32:33 vm-CaptElec grafana-server[15388]: t=2021-11-04T18:32:33+0100 lvl=info msg="Registe>  
nov. 04 18:32:33 vm-CaptElec grafana-server[15388]: t=2021-11-04T18:32:33+0100 lvl=info msg="Live Pu>  
nov. 04 18:32:33 vm-CaptElec grafana-server[15388]: t=2021-11-04T18:32:33+0100 lvl=info msg="Writing>  
nov. 04 18:32:33 vm-CaptElec grafana-server[15388]: t=2021-11-04T18:32:33+0100 lvl=info msg="HTTP Se>
```

Figure 12. Visualisation du service grafana

## 4.4.2. Configuration

### 4.4.2.1. Boot

Pour que grafana se lance au démarrage, nous utiliserons la commande suivante

*Lancement automatique de grafana au boot*

```
sudo systemctl enable grafana-server
```

### 4.4.2.2. Réseau

Par défaut grafana utilise le port 3000. Nous allons l'ouvrir aux connexions extérieures.

*Ouverture du port 3000 pour grafana*

```
sudo ufw enable  
sudo ufw allow 3000/tcp
```

### 4.4.2.3. Première connexion

Nous pouvons maintenant ouvrir grafana dans notre navigateur à l'url suivante <http://127.0.0.1:3000/>. Nous sommes accueillis par une page de connexion.

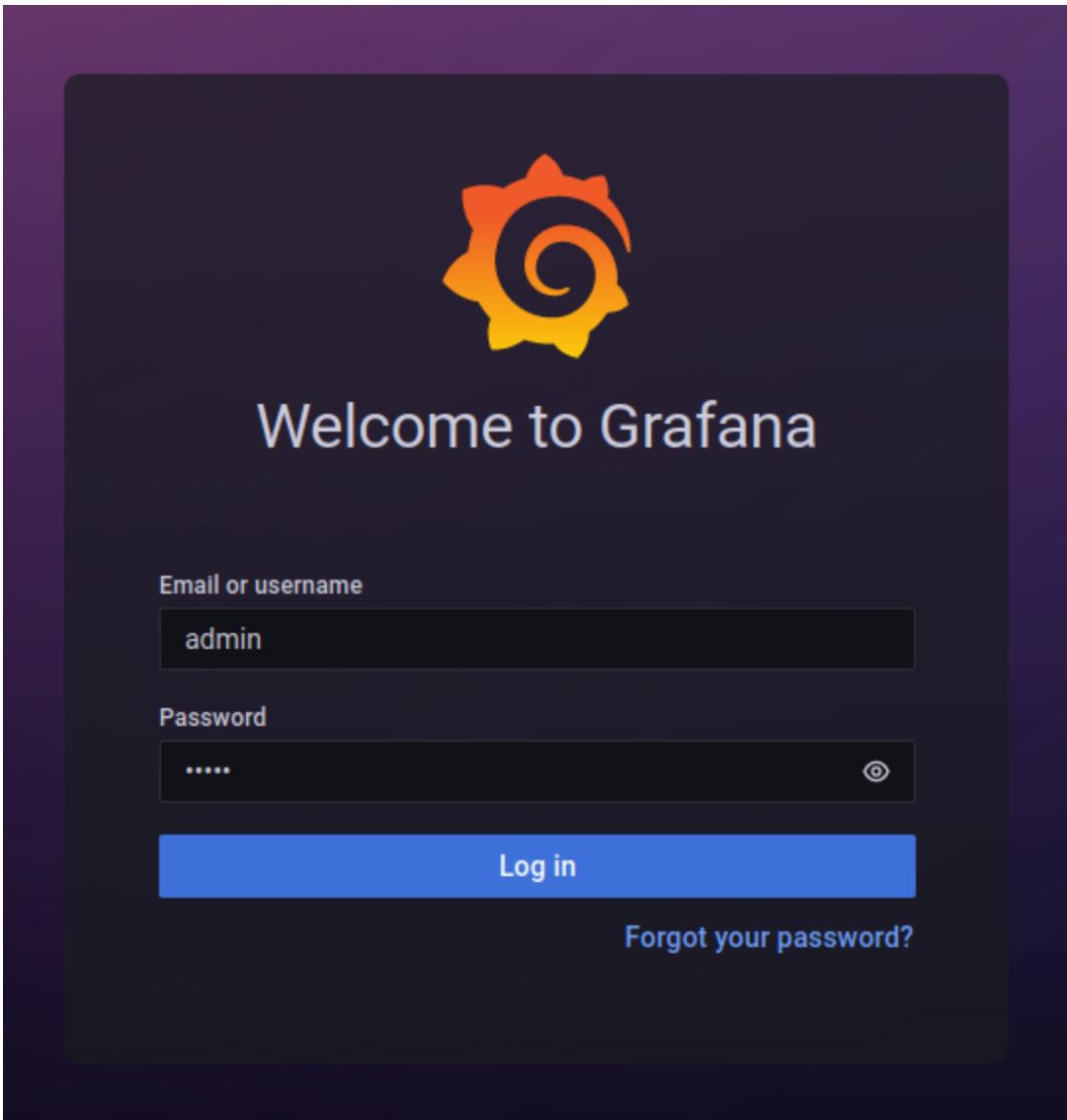


Figure 13. Connexion grafana

**NOTE** Par défaut grafana crée un compte admin/admin.

Puisqu'il s'agit de notre première connexion, il nous est proposé de changer le mot de passe.

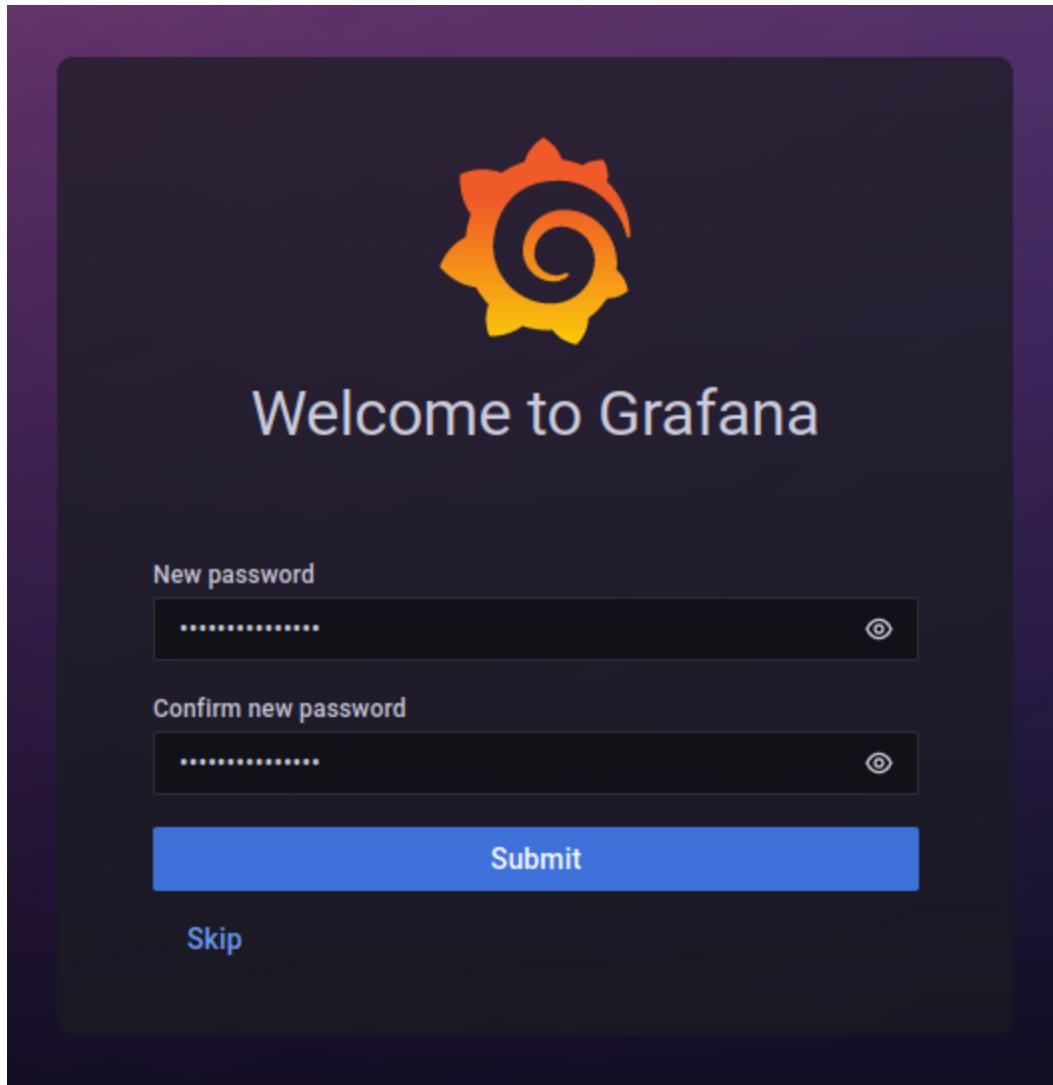


Figure 14. Changement mot de passe première connexion Grafana

**NOTE** Le compte est maintenant admin/motdepasseadmin.

#### 4.4.2.4. Source de données

Maintenant, il faut ajouter notre base influxDB comme source de données dans grafana.

Pour cela il faut aller dans Configuration(engrenage) > Data Sources

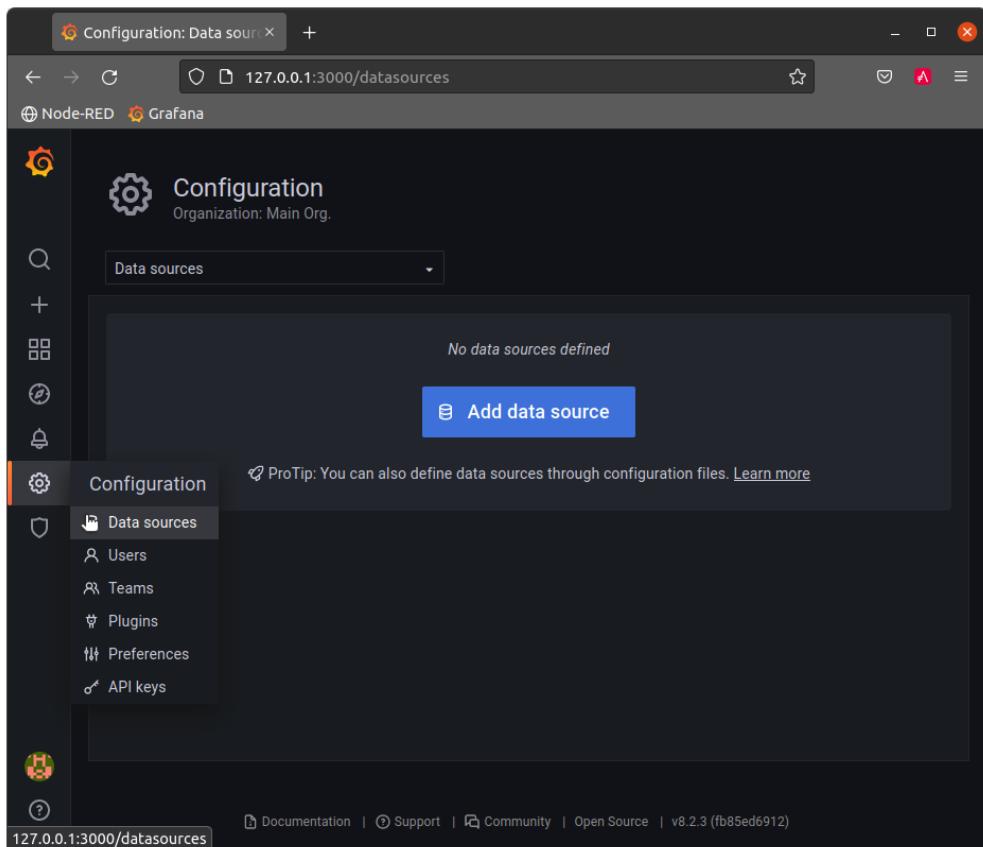


Figure 15. Accès aux sources de données

Nous choisissons une nouvelle source de données type influxDB

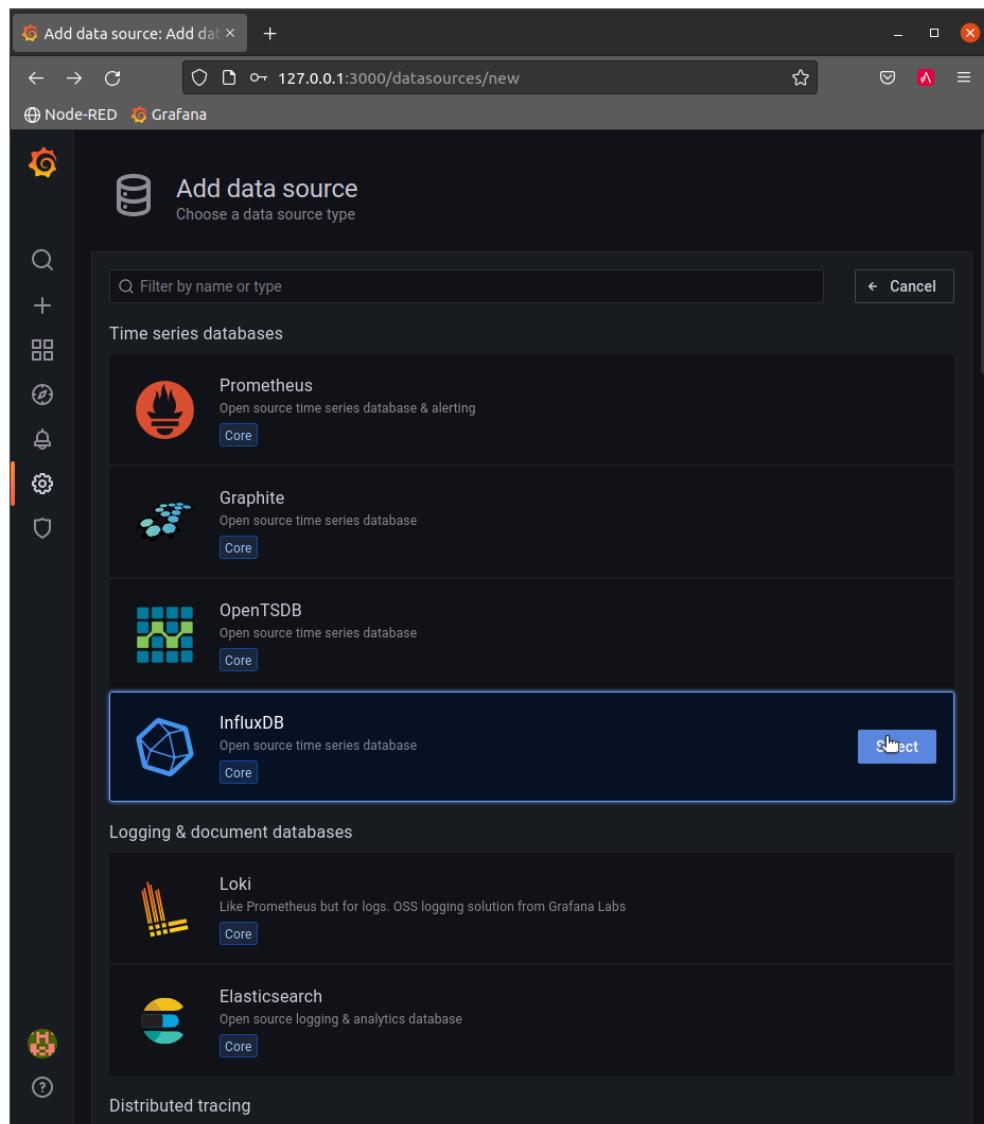


Figure 16. Choix nouvelle source de données

Maintenant nous pouvons paramétriser cette nouvelle source de données.

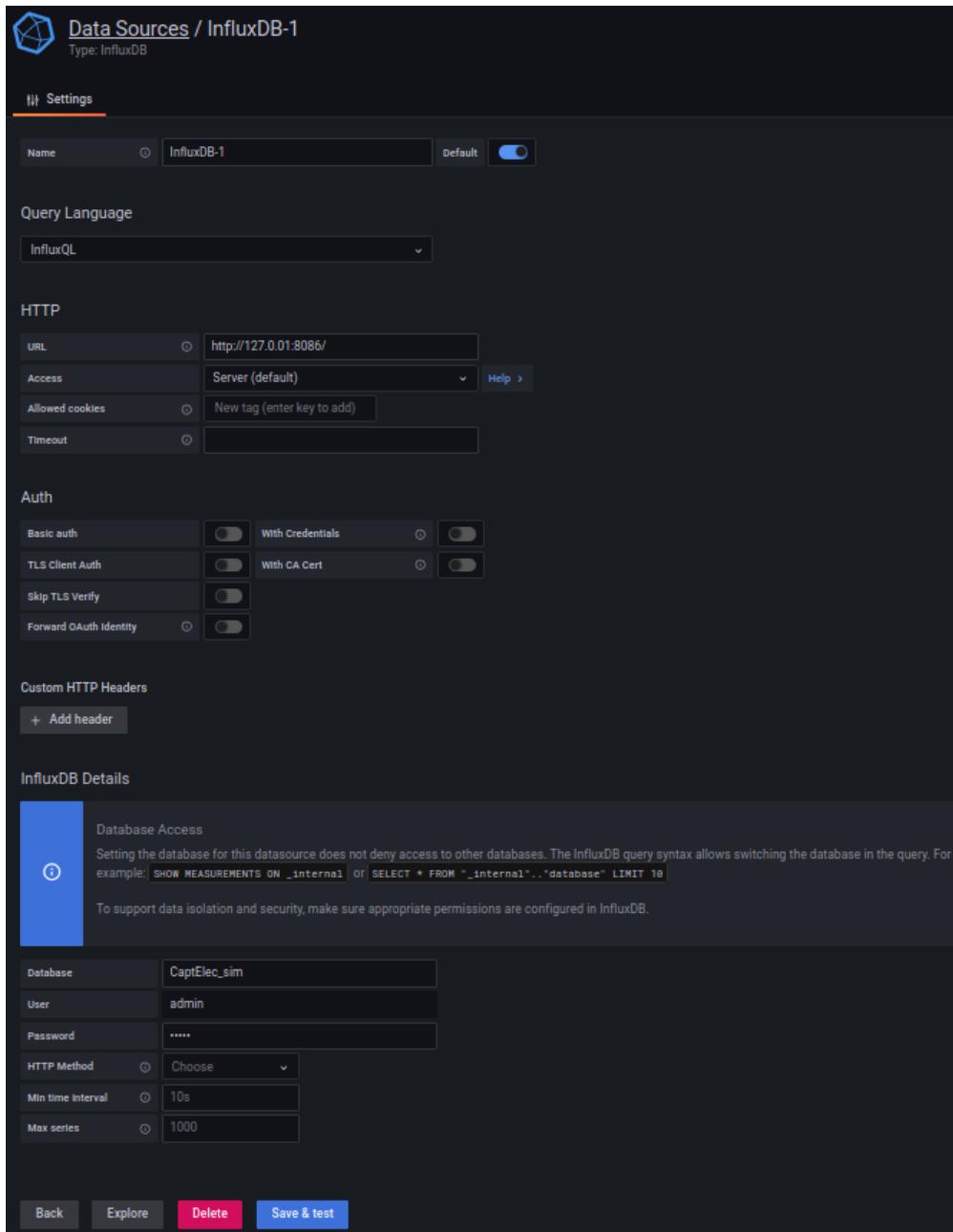


Figure 17. Paramétrage de la source de données

Les paramètres sont :

Name	Nom de la source de données dans Grafana. Cela a peu d'importance, nous laisserons le nom par défaut.
------	---

Query Language	Méthode de requête avec la base (Langage de requête/flux) Nous choisirons InfluxQL qui est le langage de requête d'influxDB.
HTTP > URL	Adresse pour accéder à la base : <a href="http://127.0.0.1:8086">http://127.0.0.1:8086</a> .
HTTP > Access	Point depuis lequel sont faites les requêtes (navigateur/serveur grafana). Dans ce choix, nous retrouvons des enjeux de sécurité et de traitement de multiples sources de données. Nous choisirons le Serveur puisque cela permet de garder la base inaccessible depuis l'extérieur.
InfluxDB Details > Access	Base à laquelle nous accédons lors des requêtes. Le langage de requête permet toutefois de facilement changer de base si nécessaire
InfluxDB Details > User	User utilisé pour faire les requêtes
InfluxDB Details > Password	Mot de passe du "User" précédent
InfluxDB Details > HTTP Method	Méthode de requêtes (POST/GET) GET est limité en nombre de caractères nous préfèrons donc utiliser POST

Enfin nous pouvons cliquer sur *Sava & test* si tout est vert nous avons fini.

#### 4.4.3. Sources

<https://linuxways.net/ubuntu/how-to-install-grafana-on-ubuntu-20-04/>

<https://computingforgeeks.com/how-to-install-grafana-on-ubuntu-debian/>

<https://grafana.com/docs/grafana/v7.5/datasources/influxdb/>

## 4.5. Installation Apache et PHP

Maintenant que nous avons InfluxDB, Grafana, Node-RED il nous reste à

installer apache et php pour réaliser certaines parties de notre tableau de bord.

#### 4.5.1. Instalation

##### 4.5.1.1. Apache

Nous allons commencer par installer apache2

*Installation de apache2*

```
sudo apt install apache2
```

**WARNING** Ici, aucune configuration de pare-feu/port (ufw) n'a été effectuée cela sera nécessaire lors de la mise en production. Aucun hôte virtuel n'a également été mis en place.

##### 4.5.1.2. PHP

Ici, nous installons php et le package apache2 pour gérer php. Nous vérifions la version pour voir si php a bien été installé.

*Installation PHP*

```
sudo apt install php libapache2-mod-php  
php -v
```

```
user@vm-CaptElec:~$ php -v  
PHP 7.4.3 (cli) (built: Oct 25 2021 18:20:54) ( NTS )  
Copyright (c) The PHP Group  
Zend Engine v3.4.0, Copyright (c) Zend Technologies  
    with Zend OPcache v7.4.3, Copyright (c), by Zend Technologies  
user@vm-CaptElec:~$ █
```

Figure 18. Vérification d'installation de php

#### 4.5.2. Configuration Apache

Nous allons également configurer apache pour qu'il serve en priorité les pages index.php plutôt que les pages index.html.

## *Accéder à la configuration des priorités*

```
sudo nano /etc/apache2/mods-enabled/dir.conf
```

```
GNU nano 4.8                               /etc/apache2/mods-enabled/dir.conf
<IfModule mod_dir.c>
    DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm
</IfModule>
```

Figure 19. Contenu de dir.conf

### 4.5.3. Source

<https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-20-04-fr>

## 4.6. Configuration réseau

Nous avons installé de nombreux services dans le but de mettre en place une application Web. Il est maintenant nécessaire de faire en sorte que ces services soient disponibles.

Outre l'attribution d'un domaine de nom et d'une adresse routable sur internet. Nous allons devoir rediriger les ports utilisés par nos différents services.

Table 2. Liste des ports à rediriger (NAT/PAT)

Numéro de Port	Service correspondant
22	ssh
80	Web — Apache
1880	Node-Red
3000	Grafana
8086	InfluxDB

## 5. Simulation de données avec Node-Red

## 5.1. Introduction

Le projet n'étant qu'une preuve de concept, nous n'allons connecter qu'un seul capteur au bus MQTT de l'IUT. De ce fait, la quantité de données affichées par le tableau de bord sera très limitée. À des fins de démonstrations, nous allons donc simuler des données qui pourraient être envoyées par un capteur connecté.

## 5.2. Valeurs de simulation

Les valeurs que nous allons simuler n'ont qu'un objectif démonstratif : leur précision ou leur cohérence ne sont donc pas des facteurs très importants.

Toutefois, nous allons tenter de nous approcher de quelque chose de réel, tant en terme de valeurs, qu'en fluctuation de celles-ci pour que les données affichées aient l'air crédibles.

Nous parlerons en poste de travail (Ecran + Machine). Nous considérons que ces postes sont utilisés pour faire de la bureautique.

- La consommation d'un poste de travail est fixée à 200 watts +- 15 (pour apporter un peu de fluctuation).
- Un poste de travail est considéré comme actif ou éteint (nous ignorons le mode veille).
- Une session de travail sur un poste dure entre 10 et 90 minutes.
- Des données seront simulées en permanence (sans prendre en compte les heures de fermeture/ouverture de l'IUT).
- Ces données seront ajoutées toutes les 30 secondes.

Pour l'instant nous simulerons 9 salles

Nom de Salle	Nombre de Poste (vérifié)
B_007	21
B_101	15
B_102	15
B_103	14

B_104	15
B_105	13
B_106	15
B_117	16
B_219	34

### 5.3. Code

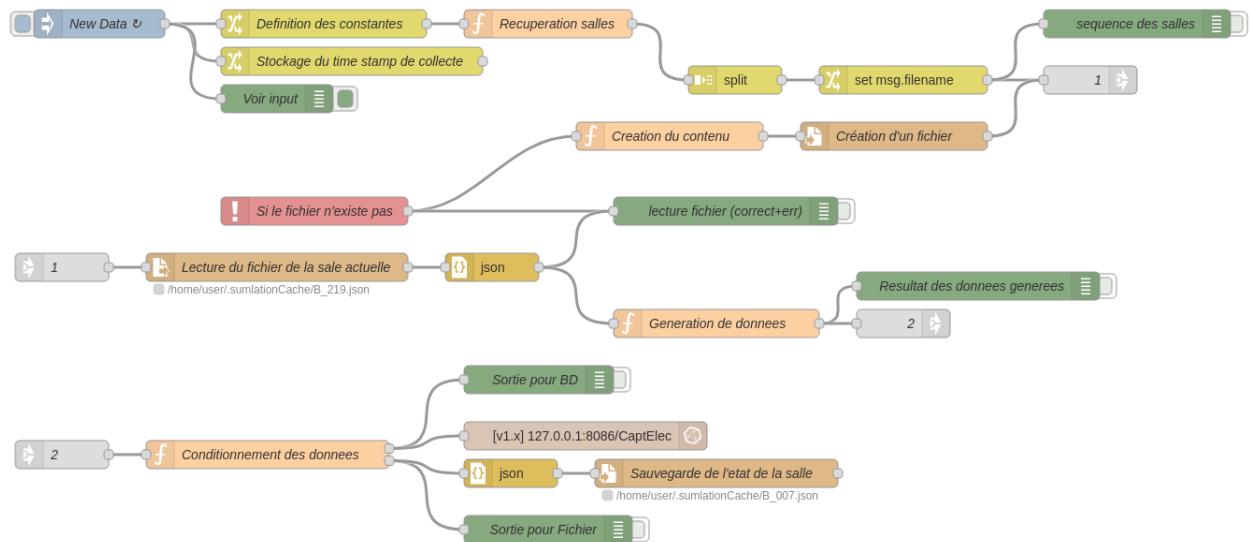


Figure 20. Capture du Flow de simulation de des données dans Node-Red

Pour la simulation des données vers grafana, nous avons utilisé NodeRED. C'est un outil low-code constitué de blocs. Il repose sur NodeJS et permettant de traiter des flux de données d'une base InfluxDB.

Chaque bloc de couleur possède une spécialité :

- Le bloc bleu (New Data) est un point d'activation à partir du lancement de NodeRED qui créé un message toutes les x secondes (paramétrable : ici 30) contenant des données.
- Les blocs jaunes (Définition des constantes, etc.) permettent la manipulation des variables créées par des blocs bleus, variable globales ou variables de message.
- Les blocs saumons/corail permettent de manipuler du code JavaScript personnalisé.
- Les blocs verts (Voir input, etc.) effectue des affichages dans la console debug pour s'assurer du bon fonctionnement du flow.

- Les blocs oranges (json) sont des parsers : ils permettent de récupérer une chaîne de caractère json afin de créer un objet en JavaScript, ou bien récupérer les données d'un objet JavaScript afin d'écrire dans le fichier json.
- Les blocs gris numérotés sont des passerelles afin de rendre le flow plus lisible et esthétique.
- Les blocs marrons permettent de travailler sur un fichier en faisant des opérations de lecture ou d'écriture.
- Le bloc rouge fait office de "try/catch" : si le fichier demandé n'existe pas, le bloc de lecture de fichier lève une erreur, le flow passera par cette branche afin de réparer le problème.
- Le bloc violet ([v1.x]127.0.0.1.8086/CaptElec) insère les données reçues dans la base influxDB.
- Les chemins gris reliant chaque bloc font transiter un objet JavaScript (identique ou non à celui du bloc précédent) afin de réaliser les exécutions du flux voulues. Chaque objet possède un attribut obligatoire : le payload, il contient le contenu principale du message, nous pouvons lui ajouter d'autres champs (comme "filename", etc).

## 5.4. Fonctionnement de la simulation

Lorsque NodeRED se lance, le bloc NewData envoie un message de déclenchement (renouvelé toutes les 30 secondes). Ce message arrive ensuite dans le bloc "Définition des constantes" mettant à jour les constantes de simulation en écrasant les constantes précédentes. Le message arrive ensuite dans le bloc "Récupération salles" il y sera modifié : nous y mettons la liste des chemins de stockage des fichiers de simulation (un par salle). Le message arrive ensuite dans le bloc "Split" qui va séparer la liste pour envoyer x messages : un message par salle. Ces x messages seront alors transmis dans le bloc gris numéro 1 pour rejoindre le bloc "Lecture du fichier de la salle actuelle".

Ce bloc va lire le fichier grâce au chemin que le message contient : si le fichier n'existe pas, un fichier vide sera créé avec le chemin d'accès dicté précédemment, puis le message reviendra à notre bloc "Lecture du fichier de la salle actuelle".

Le contenu du fichier json est donc transformé afin de créer un objet JavaScript

pour travailler sur ses données. Le message est ensuite transmis vers le bloc "Génération de données". Ce bloc va créer les nouvelles données :

- Il va éteindre les pcs qui doivent être éteint (car l'utilisateur a fini de travailler dessus).
- Il va allumer aléatoirement les pcs éteints avec une probabilité de 0,3.
- Pour tous les PCs allumés, il va tirer aléatoirement une valeur correspondant en Watt à la consommation de chaque ordinateur pour additionner le tout et obtenir la consommation totale de la salle.

Le message passe ensuite dans le deuxième bloc gris (le numéro 2) pour rejoindre le bloc "Conditionnement des données". Celui-ci va conditionner les données pour obtenir deux messages corrects pour l'enregistrement de celles-ci. Ce bloc va renvoyer deux messages :

- L'un pour insérer dans la base InfluxDB (consommation électrique totale de la salle)
- L'autre pour mettre à jour les fichiers json de simulation (le nombre de PCs allumés)

Ces données simulées sont ensuite illustrées grâce à l'outil Grafana, qui montrera des graphiques / données en temps réel (consommation maximale, moyenne de consommation, salle illustrée etc.).

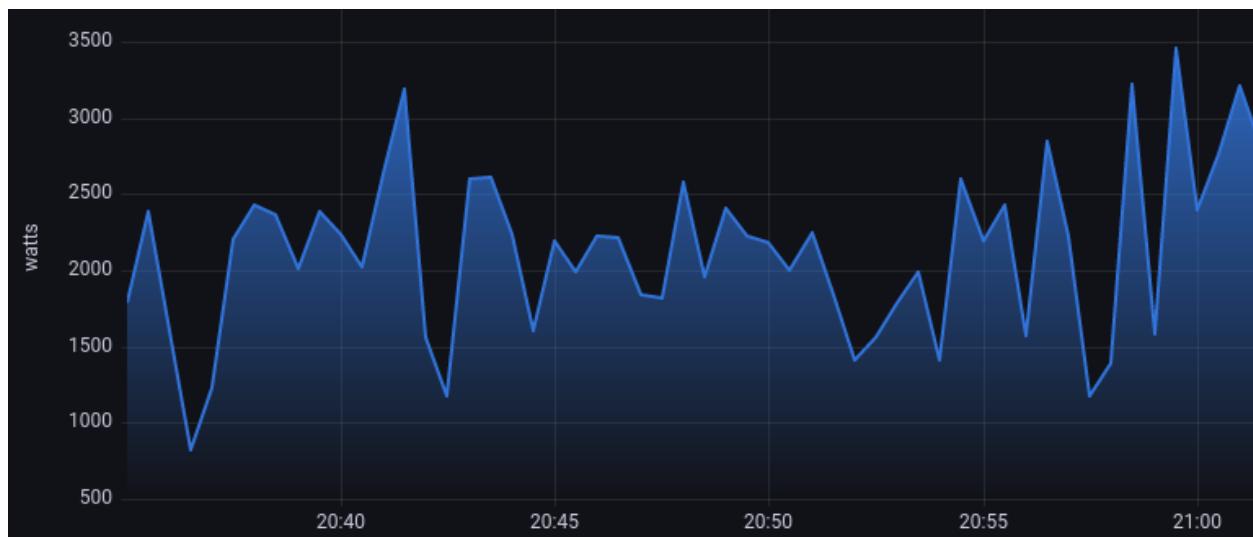


Figure 21. Capture d'un panel Grafana

**NOTE**

Les valeurs simulées sont basées sur des moyennes réelles, cependant leur évolution ne sont pas représentatives de la réalité : des variations aussi grandes et aussi fréquentes de la consommation seront très probablement pas observés lors de la réalisation du projet avec le capteur, ce ne sont que des données simulées. Nous nous laissons nonobstant la possibilité d'améliorer le graphique de simulation.

## 5.5. Sources

<https://www.fournisseurs-electricite.com/guides/consommation/ordinateur>

## 6. Créer de nouveaux SVG

Afin de réaliser des menus en mode images, il nous faut des images dans un format particulier. Nous utilisons des SVG, car la manière de définir les zones de l'image avec des balises nous permet ensuite d'appliquer des classes et des IDs comme pour des éléments HTML. Cela nous donne la possibilité de changer le style grâce à des fichiers CSS, mais également de programmer des événements JavaScript.

**WARNING**

Nous recommandons très vivement l'utilisation d'une tablette graphique pour réaliser ces tâches.

**TIP**

**La question maintenant est : comment créer de nouveaux SVG pour étendre le site ?**

Pour répondre à cette question nous allons simplement détailler les étapes de création. Bien évidemment il ne sagit ici que de notre méthode.

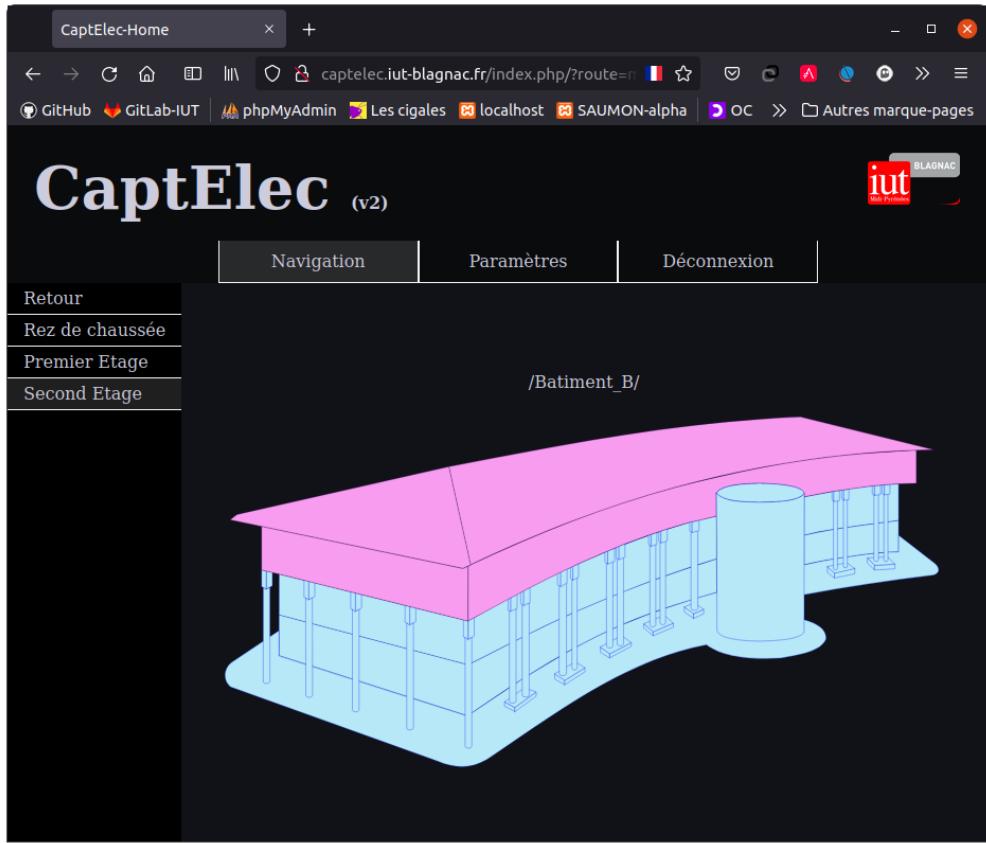


Figure 22. Exemple de menu avec image (ici le Bâtiment B de l'IUT — le second étage est sélectionné)

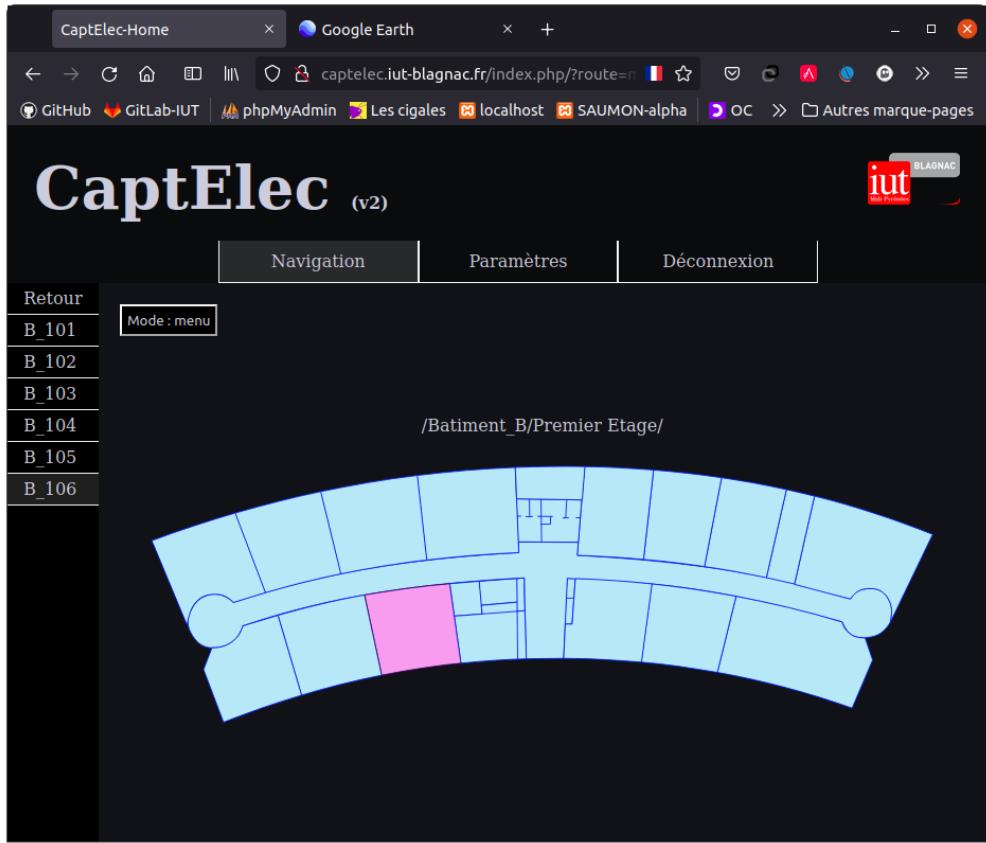


Figure 23. Exemple de menu avec image (ici le premier étage du Bâtiment B de l'IUT — la salle 106 est sélectionnée)

## 6.1. Étape 1 : Choisir un modèle

Nous ne partons pas de zéro. Il nous faut donc un modèle. Pour notre part, nous utilisons les extrapolations 3D de <https://www.google.fr/intl/fr/earth/> pour faire les bâtiments, et les plans d'évacuation pour les plans des étages.

Pour les plans, il suffit d'une photo ou d'un scan. Mais pour les vues 3D de l'IUT, il est important de bien cadrer ce que vous voulez représenter, et de choisir la bonne inclinaison de caméra car sinon, la perspective risque d'être compliqué à transcrire en dessin par la suite.



Figure 24. Capture d'écran de l'extrapolation 3D google earth du Bâtiment B

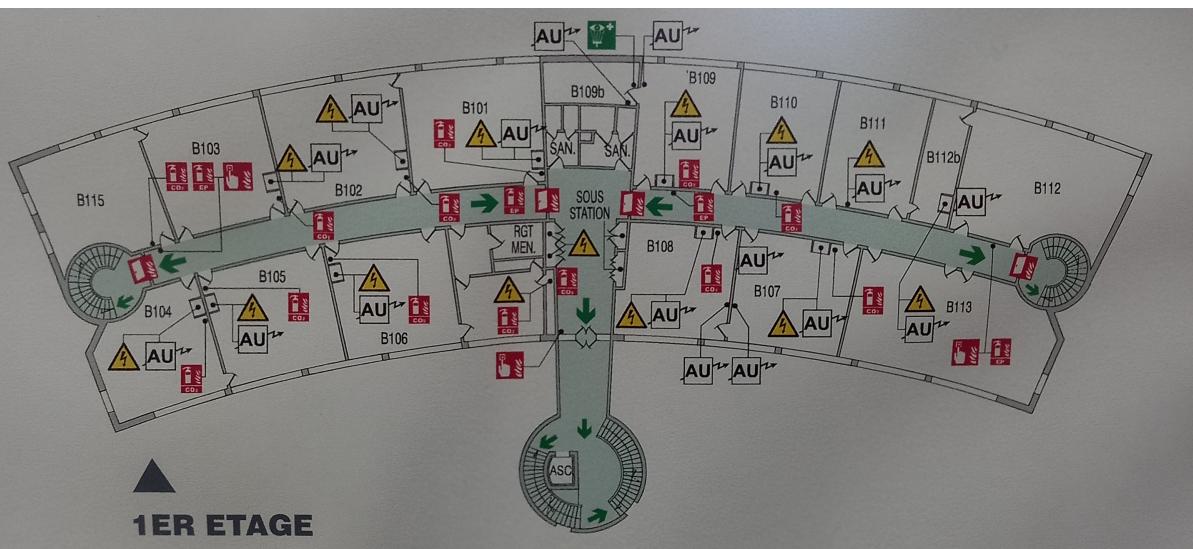


Figure 25. Photo du plan d'évacuation du premier étage du Bâtiment B de l'IUT

## 6.2. Étape 2 : Décalquer (optionnel)

### NOTE

Cette étape n'est pas obligatoire, il est possible de créer les SVG directement sur les modèles. Cependant, nous recommandons vivement d'en passer par là car elle permet de poser un premier filtre sur ce que nous souhaitons représenter ou non. De plus, elle met en évidence les courbes qui sont les éléments les plus complèxes à tracer pour les SVG.

Maintenant que nous avons nos modèles, nous allons passer par un logiciel de dessin tiers pour créer une image intermédiaire en "fil de fer" de nos sujets, c'est à dire créer une image "plate" (pas de distinction de zone .png, .jpg par exemple) qui trace en noir sur fond blanc les arêtes.

Nous avons utilisé Krita qui est parfait pour ce genre de tâche mais GIMP ou encore un produit de la suite Adobe fera aussi bien le travail.



Figure 26. Logo du logiciel krita

Vous devriez obtenir des images dans ce genre :

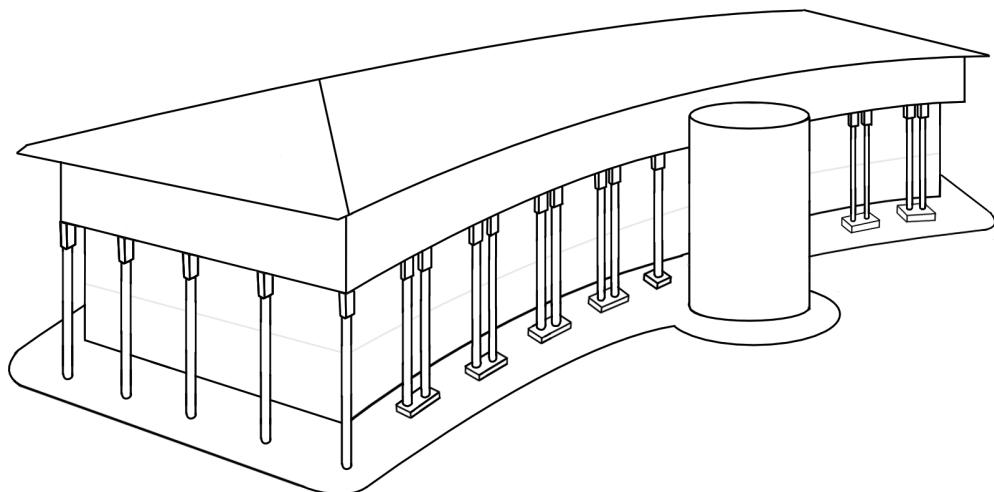


Figure 27. Fil le fer du bâtiment B.

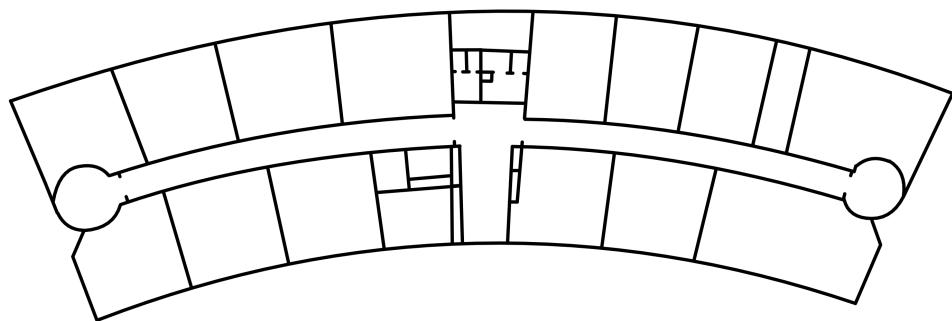


Figure 28. Fil le fer du premier étage du bâtiment B.

### 6.3. Étape 3 : Passage en SVG

Maintenant, nous allons produire nos SVG. Pour cela, il faut avant tout se munir d'un outil de dessin Vectoriel. Nous avons utilisé Ink Scape car c'est un outil spécifique au dessin vectoriel. Mais encore une fois Krita, GIMP, ou la suite Adobe sont

des alternatives envisageables.



Figure 29. logo du logiciel Ink Skape

Dans ink skape :

- Créer un nouveau document (il doit faire au moins la taille de votre modèle pour éviter de perdre de l'information).
- Importer votre modèle.
- Baisser l'opacité du modèle pour pouvoir dessiner dessus plus aisément.
- **Utiliser un nouveau calque pour chaque partie du dessin**

Par exemple :

- Pour le bâtiment B nous avons : Toit, 2<sup>eme</sup> étage, 1<sup>er</sup>, RDC, cylindre, piliers (cf. [image du bâtiment B](#)).
- Pour le premier étage du bâtiment B : Un calque par salle.

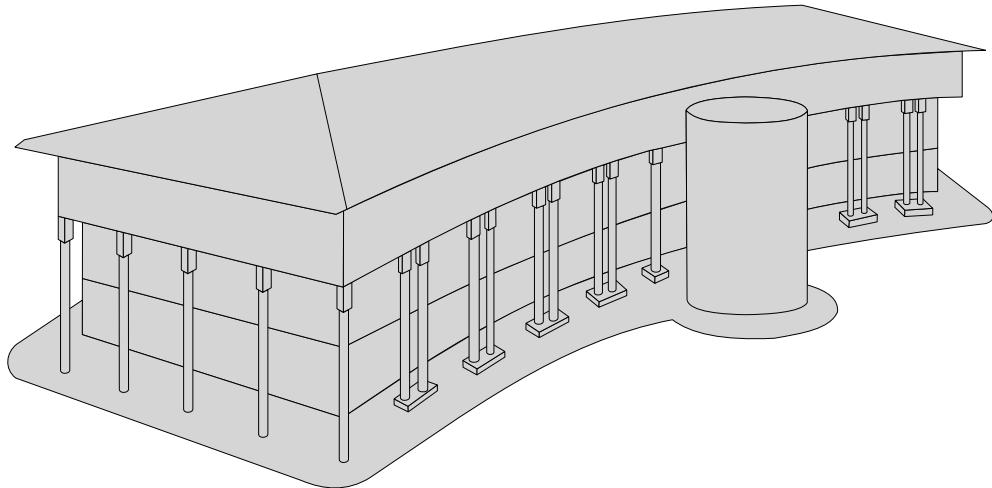
- Avec l'outil chemin de dessin vectoriel, tracez les zones de votre image (une zone est simplement une forme fermée : une salle, un pan de mure).

Une fois terminé, enregistrez au format SVG.

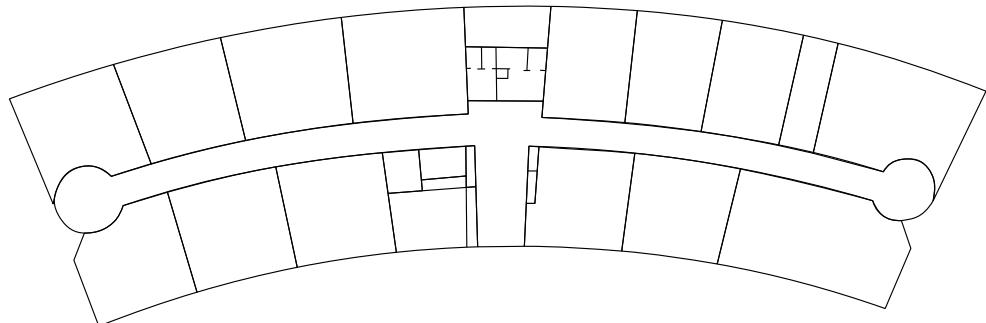
**NOTE** Quand vous tracez vos chemins il est important de garder en tête l'objectif : Nous voulons pouvoir isoler les zones pour pouvoir les sélectionner plus tard dans le site.

**NOTE**

Nous recommandons de mettre le moins de style possible lors de l'édition des SVGs car cela pourrait plus tard interférer avec l'application des styles du site.



→Figure 30. SVG du bâtiment B (avec style → remplissage)



→Figure 31. SVG du permier étage du bâtiment B (sans aucun style → transparent)

## 6.4. Étape 4 : Nettoyage du fichier

Le logiciel d'édition vectoriel a généré automatiquement un fichier SVG. Toutefois, afin qu'il corresponde totalement à ce que le site attend, il est nécessaire de le "nettoyer" et d'y ajouter certains éléments.

Ouvrez votre fichier SVG dans votre éditeur de Code favori. Le début du fichier devrait se présenter comme ci-dessous :

## Début du fichier SVG avant nettoyage

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- Created with Inkscape (http://www.inkscape.org/) -->

<svg
    version="1.1"
    id="svg9"
    width="2141.3333"
    height="1064"
    viewBox="0 0 2141.3333 1064"
    sodipodi:docname="batimentB_ink.svg"
    inkscape:version="1.1.1 (eb90963e84, 2021-10-02)"
    xmlns:inkscape="http://www.inkscape.org/namespaces/inkscape"
    xmlns:sodipodi="http://sodipodi.sourceforge.net/DTD/sodipodi-
0.dtd"
    xmlns="http://www.w3.org/2000/svg"
    xmlns:svg="http://www.w3.org/2000/svg">
<defs
    id="defs13" />
<sodipodi:namedview
    id="namedview11"
    pagecolor="#ffffff"
    bordercolor="#111111"
    borderopacity="1"
    inkscape:pageshadow="0"
    inkscape:pageopacity="0"
    inkscape:pagecheckerboard="1"
    showgrid="false"
    inkscape:snap-intersection-paths="true"
    inkscape:snap-smooth-nodes="true"
    inkscape:object-paths="true"
    inkscape:snap-midpoints="true"
    inkscape:zoom="0.56551269"
    inkscape:cx="1083.1916"
    inkscape:cy="561.45676"
    inkscape>window-width="1848"
    inkscape>window-height="1016"
    inkscape>window-x="72"
    inkscape>window-y="27"
    inkscape>window-maximized="1"
    inkscape:current-layer="poto_1" />
    ...

```

Nous ne voulons conserver que l'ouverture de la balise `<svg>` avec seulement l'attribut `viewBox`.

## Début du fichier SVG après nettoyage

```
<svg viewBox="0 0 2141.3333 1064">  
...
```

Selon les versions et les éditeurs, il est possible que l'attribut viewBox soit absent. Dans ce cas il faut le créer.

Pour cela vous pouvez rouvrir votre SVG dans un éditeur d'image. l'objectif est de déterminer 4 entiers :

- $X_{\text{origine}}$
- $Y_{\text{origine}}$
- La largeur
- La hauteur

### WARNING

Pour X et Y vous pouvez mettre 0. Pour la hauteur et la largeur, dans votre éditeur, tracez un rectangle qui contient tout votre dessin en étant aussi petit que possible. Prenez alors la hauteur et la largeur en pixel de ce rectangle. Nous pouvons maintenant écrire la balise SVG tel que :

```
<svg viewBox="X Y largeur hauteur">
```

Pour en savoir plus sur viewBox :  
<https://wattenberger.com/guide/scaling-svg>

La suite de votre SVG est constitué de blocks `<g>` qui représentent vos calques. Nous allons également nettoyer leurs attributs.

## Balise g avant nettoyage

```
...
<g
  inkscape:groupmode="layer"
  id="layer-05-cyl"
  style="display:inline">
  <path
    style="fill:#d5d5d5;fill-opacity:1;stroke:#000000;stroke-
width:1px;stroke-linecap:round;stroke-linejoin:round;stroke-
opacity:1"
    d="m 1462.8373,235.72736 c 9.2665,-16.9624 48.9807,-27.60629
127.2774,-28.79453 102.0212,1.85489 129.922,16.26802
132.4884,30.34378 -20.1337,24.60438 -83.0376,24.0544
-133.9794,25.62044 -51.1156,-1.56985 -105.6764,-2.09544 -125.7864,
-27.16969 z"
    id="path5191"
    sodipodi:nodetypes="cccccc" />
  <path
    style="fill:#d5d5d5;fill-opacity:1;stroke:#000000;stroke-
width:1px;stroke-linecap:round;stroke-linejoin:round;stroke-
opacity:1"
    d="m 1722.6031,237.27661 -0.3909,410.25704 c -14.1432,21.49393
-86.2232,24.81863 -130.1379,25.68983 -46.1326,-0.45692 -123.5105,
-5.24244 -127.2467,-32.37033 l -1.9903,-405.12579 c 27.5006,28.16085
79.191,24.29218 125.7864,27.16969 58.5079,-1.0565 115.7858,-2.77769
133.9794,-25.62044"
    id="path6309"
    sodipodi:nodetypes="ccccccccc" />
</g>
...

```

Dans les attributs de la balise g, nous ne souhaitons conserver que id. Nous allons changer la valeur de l'id pour y mettre quelque chose de plus parlant.

### WARNING

Vous ne pouvez pas choisir n'importe quoi car ces id sont utilisés dans le fichier de configuration. (cf. [Règles sur les Id des SVG](#))

Nous pouvons voir que les blocks <g> contiennent des chemins <path>. Pour ces éléments "chemin", nous n'allons conserver que les attributs id et d.

## Balise g après nettoyage

```
...
<g id="cylindre">
  <path
    id="path5191"
    d="m 1462.8373,235.72736 c 9.2665,-16.9624 48.9807,-
27.60629 127.2774,-28.79453 102.0212,1.85489 129.922,16.26802
132.4884,30.34378 -20.1337,24.60438 -83.0376,24.0544
-133.9794,25.62044 -51.1156,-1.56985 -105.6764,-2.09544 -125.7864,
-27.16969 z"/>
  <path
    id="path6309"
    d="m 1722.6031,237.27661 -0.3909,410.25704 c
-14.1432,21.49393 -86.2232,24.81863 -130.1379,25.68983 -46.1326,
-0.45692 -123.5105,-5.24244 -127.2467,-32.37033 l -1.9903,-405.12579
c 27.5006,28.16085 79.191,24.29218 125.7864,27.16969 58.5079,-1.0565
115.7858,-2.77769 133.9794,-25.62044"/>
</g>
...
...
```

Après ce nettoyage, les SVGs ressemblent à des "ombres". Nous leurs avons retiré tous les styles graphiques (remplissage, couleur, épaisseur de traits, ...). C'est normal, le site ajoutera automatiquement ses styles à l'utilisation.

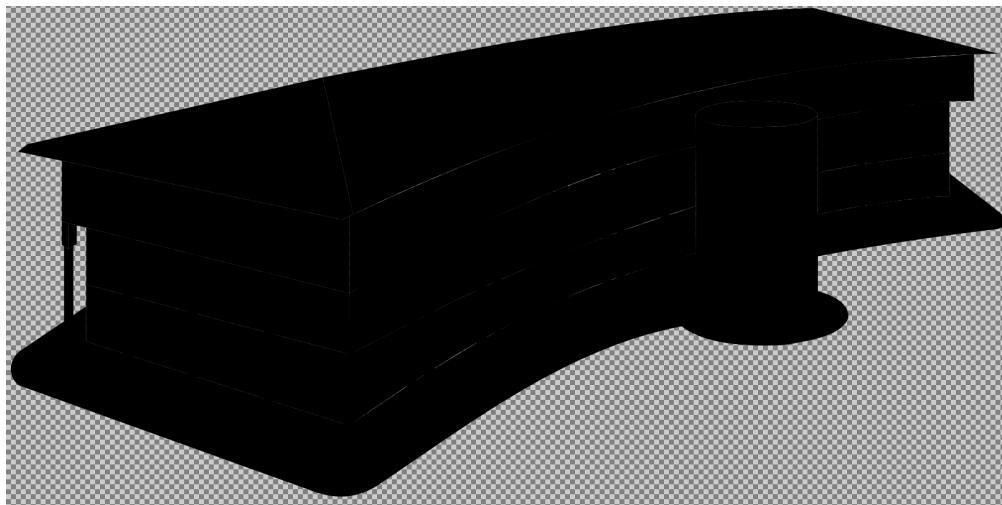


Figure 32. Bâtiment B svg sans style

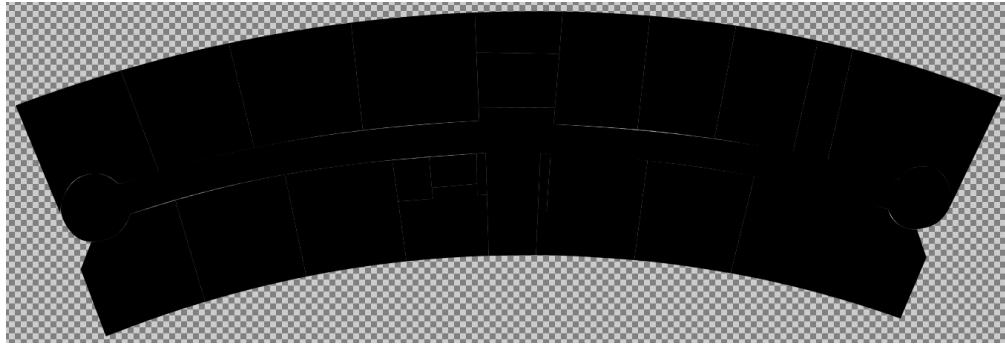


Figure 33. Premier étage du bâtiment B svg sans style

## 6.5. Étape 5 : Ajouter le SVG au site :

Vous devez maintenant rendre votre nouveau SVG disponible sur le site. Pour cela il vous suffit de le placer dans l'arborescence dans : `racineDuSite/static/svg/<nouveauSVG.svg>`.

Pour que le site utilise vos nouveaux SVG dans ses menus il faudra bien entendu modifier le fichier de configuration en conséquence (cf. [Nouveaux menus](#)).