Abry Maxime
Vaglio Lisa
Demeulenaere Bastien
Egloff Baptiste
Simon Théo

Documentation Projet Chauffage

SOMMAIRE

Partie test :

Chaque fonction test possède un docstring ainsi que le nom de cette fonction indique la fonction testée.

La bibliothèque Python utilisée est unittest.
Le lien de la documentation de cette bibliothèque est :
https://docs.python.org/3/library/unittest.html

Chaque fichier.py possède une classe test dans le fichier “test_uni.py”. Chaque classe a pour but de tester les fonctions présentes dans le fichier.py correspondant ainsi lors de modifications ou d'ajouts dans le code. Si on lance les tests, cela nous prévient directement quel test de quelle classe rencontre un problème. Ainsi, on peut directement voir en regardant le nom de la classe le nom du fichier qui possède un dysfonctionnement suite à une modification, ce qui permet de corriger l'erreur plus facilement.

Pour lancer les tests, se référer à la partie Docker.

Partie Docker :

Projet GitHub : https://github.com/simotheo/Projet_Chauffage
Télécharger le projet GitHub.

Deux fichiers : Dockerfile et requirements.txt

Dockerfile : les informations pour le Docker
requirements.txt : les bibliothèques nécessaires et leurs versions

Dans le fichier Docker :

Définition des arguments de construction :

```
ARG nom_de_votre_variable
```

Pour chaque variable à relier avec le code.

Utiliser les arguments de construction pour définir les variables d'environnement :

```
ENV nom_de_votre_variable = ${nom_de_votre_variable}
```

Pour choisir l'image du fichier python que vous voulez lancer, modifier la commande :

```
CMD ["python", "votre_fichier.py"]
```

En modifiant "votre_fichier.py" par le nom de votre fichier.

Dans le fichier requirements.txt :

Mettre le nom des bibliothèques et leurs versions.

Dans le fichier main :
Pour faire le lien entre les variables python et celles du Dockerfile :

```
import os

nom_variable_dockerfile = os.getenv("nom_variable_dockerfile", "nom_variable_python")
```

Pour chaque variable configurable.

Pour permettre ou non la modification de la variable lors du build sur l'invite de commande :

Pour les entiers :

```
if nom_variable_dockerfile != '':
    nom_variable_python = int(nom_variable_dockerfile)
```

Pour les string :

```
if nom_variable_dockerfile != '':
    nom_variable_python = nom_variable_dockerfile
```

Dans une invite de commandes, déplacez vous à l'emplacement du projet GitHub et rentrez les commandes suivantes :

Pour lancer le programme sans modifier les paramètres, il faut :

- build avec la commande :

```
docker build -t nom_de_votre_image .
```

- run avec la commande :

```
docker run nom_de_votre_image
```

Pour lancer le programme en choisissant tous les paramètres, il faut :

- build avec la commande :

```
docker build --build-arg VAR1="value1" --build-arg VAR2="value2" -t nom_de_votre_image .
```

En modifiant les VAR et les "value" selon toutes les données voulues

- run avec la commande :

```
docker run nom_de_votre_image
```

Pour lancer le programme en choisissant seulement certains paramètres, il faut :

- build avec la commande :

```
docker build --build-arg VAR1="value1" --build-arg VAR2="value2" -t nom_de_votre_image .
```

Seulement pour les paramètres à modifier, pas besoin de noter les variables à ne pas modifier

Partie Programme :

Description des Fichiers Python

-variables.py:

Ce fichier contient les variables globales et les configurations utilisées dans le projet, centralisant ainsi les paramètres modifiables.

Les variables modifiables sont :

-**liste_vannes**={"simu-vanne-01": [5465,"8D-001"] } # liste des vannes à gérer avec le numéro et le nom de la salle. Le dictionnaire fonctionne ainsi : "Nom de la vanne" : [numéro de salle, "nom de la salle"]

temperature_occupee, un entier # température de consigne pour une salle occupée

temperature_non_occupee, un entier # température de consigne pour une salle inoccupée

temps_prechauffage, temps en minutes pour préchauffer la salle# temps de préchauffage en minutes

temps_arret, entier # temps d'arrêt en minutes

#Variable pour gérer le délai d'appel à ADE

refresh, un entier # temps d'appel au scheduler en secondes

Variables pour la connexion à la base de données InfluxDB

host, une chaine de caractère # adresse du serveur InfluxDB

org, une chaine de caractère # nom de l'organisation

token, une chaine de caractère # token d'accès

bucket, une chaine de caractère # nom du bucket

Variables pour la connexion au broker MQTT

broker, une chaine de caractère # adresse du broker MQTT

port, un entier # port du broker MQTT

client_id, une chaine de caractère # identifiant du client

username, une chaine de caractère # nom d'utilisateur

password, une chaine de caractère # mot de passe

-connexion.py :

Ce fichier gère la connexion à la base de données influxDB pour le projet.

-create_file_icalendar.py :

Ce fichier contient les fonctions nécessaires pour créer des fichiers iCalendar, permettant de gérer des événements et des calendriers.

-genereJSon.py:

Ce fichier génère des fichiers JSON à partir des données fournies, assurant un format standardisé pour l'échange de données.

-recuperation.py:

Ce fichier gère la récupération des données depuis ADE et le traitement de ces données sources et assure leur intégration dans le projet.

-gestionADE.py

Ce fichier est responsable de la gestion des données amenant à la gestion de la consigne en lien avec les résultats avec le site ADE, souvent utilisé pour la planification des emplois du temps.

-influxDB.py:

Ce fichier contient les fonctions nécessaires pour interagir avec une base de données InfluxDB, utilisée pour le stockage et la gestion de données temporelles.

-recup_mqtt.py:

Ce fichier est dédié à la récupération des messages MQTT et à leur traitement.

-gestionMqtt.py:

Ce fichier gère la communication via le protocole MQTT, utilisé pour la communication machine à machine.

-threadVanne.py:

Ce fichier gère les threads pour la manipulation des vannes, permettant un contrôle asynchrone et concurrent des dispositifs. Une vanne correspond à un Thread qui est à l'écoute de son topic.

-plannificateur.py:

Ce fichier contient les fonctions nécessaires pour la planification et la gestion des tâches ou des événements. Il permet de mettre en place le planificateur toute les 5 minutes

-test_uni.py:

Ce fichier contient les tests unitaires pour les différents fichiers `.py` du projet. Chaque classe de test est dédiée à un fichier spécifique, facilitant l'identification des erreurs et leur correction.

-main.py:

Ce fichier principal orchestre l'exécution du projet, en utilisant les différentes fonctions et modules définis dans les autres fichiers.

Problèmes possibles :

- Le fichier .ics récupéré d'ADE peut contenir des cours "fantômes" qui ne s'affichent pas dans l'emploi du temps. Cela peut perturber la fonction qui cherche les différents cours car elle ne peut différencier ceux qui s'affichent de ceux qui ne s'affichent pas.

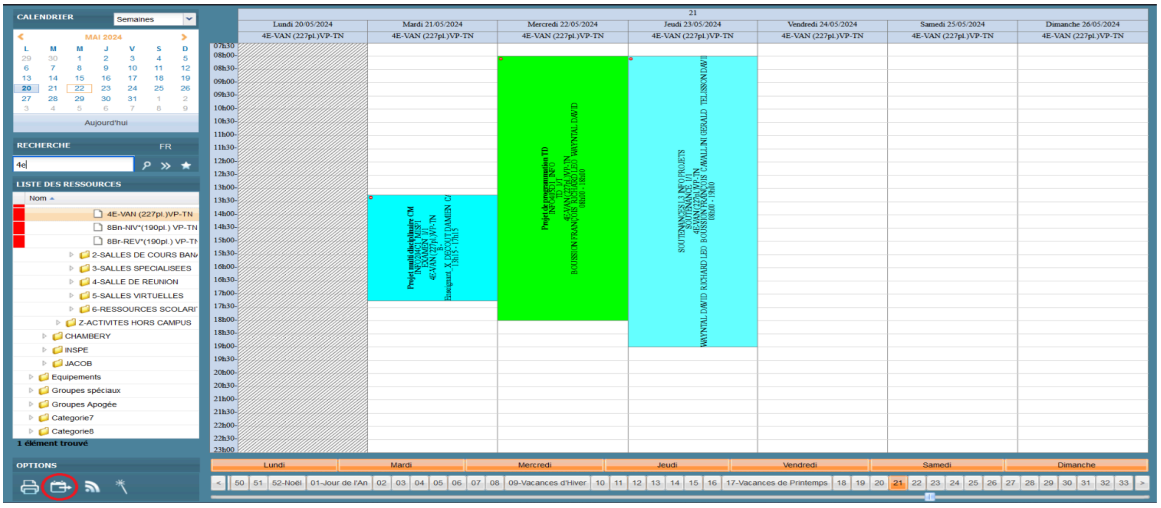
- Le fichier ics a, certaines fois, des évènements entrés d'une manière qui n'est pas détectée par l'api de lecture de ceux-ci. Cela peut amener des cours à ne pas apparaître.

- Si la connexion internet saute ou est mauvaise et que la salle n'a jamais eu de consigne dans la base de donnée, cela peut causer un problème d'écriture dans celle-ci. Si la vanne demande sa nouvelle consigne avant l'écriture dans la base de donnée cela provoque une erreur.

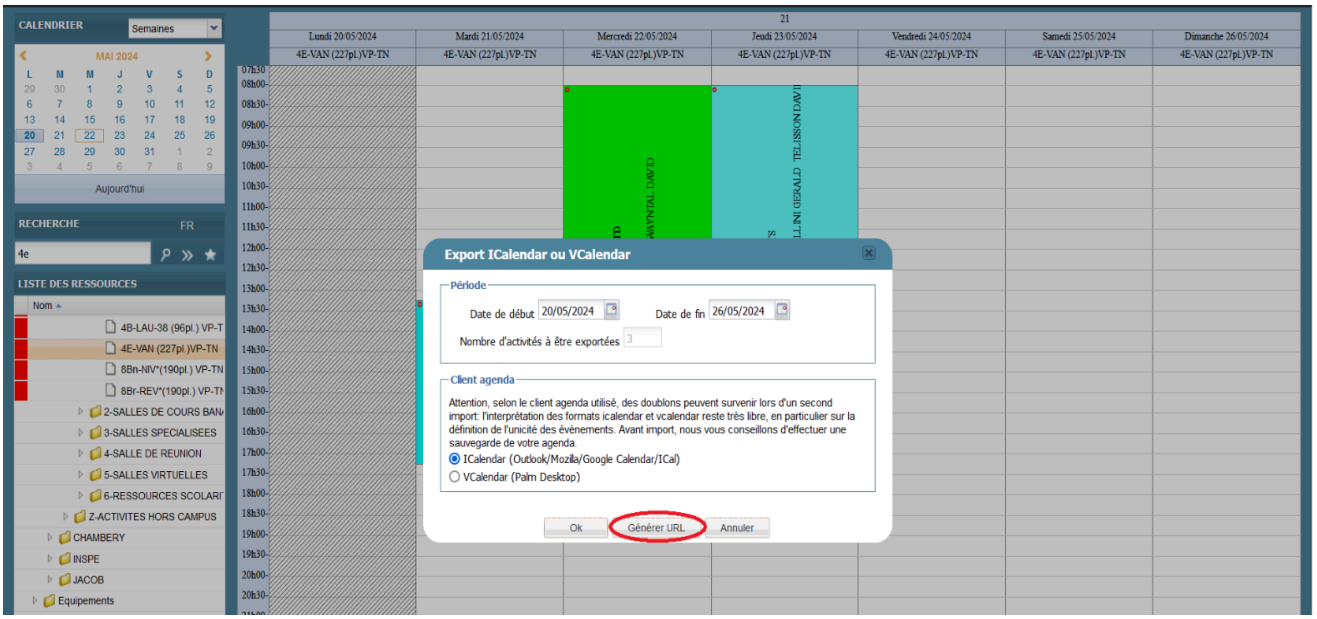
Remarques:

- L’identifiant des salles (ex: 5037 pour la 8C-38) est trouvable grâce à l’url pour trouver le fichier url de l’emploi du temps de la salle ciblée. On le retrouve de la façon suivante :

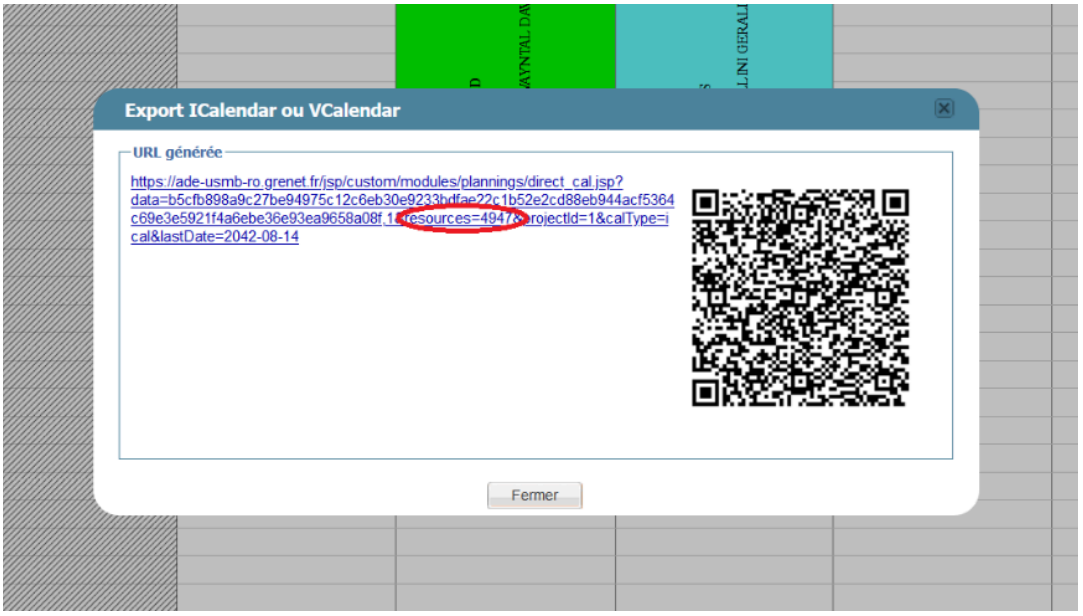
  Rechercher une salle sur ADE et on appuie le deuxième bouton en bas à droite



  Ensuite, nous avons cette fenêtre qui s’affiche et nous devons générer un url



Puis on obtient le lien et le numéro entouré est à relever



- Ajouter la salle à une vanne :

Aller dans le fichier variables.py, la liste des vannes se trouve dans la variable liste_vannes. C'est un dictionnaire où les nouvelles vannes sont enregistrées de la manière suivante :

"nom-vanne" : [numéro de salle (par exemple 4947), "nom de la salle]

Sauvegarder le fichier. La vanne avec le numéro de salle est dorénavant mise à jour.

- Pour que le projet fonctionne, il faut être connecté au VPN de l'Université Savoie Mont-Blanc pour se connecter au serveur LoRaWan