

PubliExploit

Guide d'utilisation de l'application



Réalisé par :
Antoine LELOIR

Pour :
Nicolas SOKOLOFF

Version 1.0

11 juillet 2024

Table des matières

1	Contexte	1
2	Présentation de l'application existante	1
3	Brève Description de PubliExploit	3
4	Objectifs de l'application	4
5	Fonctionnement général de PubliExploit	5
5.1	Schéma	5
5.2	Surveillance des boîtes et traitement des mails	5
5.3	Décompression du dossier	8
5.4	Base de de données	8
5.5	Vérifier la cohérence des données	10
5.6	Gestion des erreurs	11
6	Arborescence	13
7	Exécution	25
7.1	Exécution à intervalles réguliers	25
7.2	Exécution manuelle	25
7.3	Rattrapage	26
7.4	Exécution des tests	26
8	Déroulement des tests	27
9	Configuration de l'application	28
9.1	Fichiers de configuration	28
	column_mapping.json	28
	attribute_mapping.json	29
	main_config.ini	29
	sites.ini	30
9.2	Modification du code source	30
9.3	Configuration de l'exécution automatique	34

10 Interface d'ajout d'un site dans sites.ini	37
10.1 Fonctionnalités de l'interface	37
10.2 Fonctionnement de l'interface	37
10.2.1 Téléchargement et mise à jour du fichier de configuration . . .	37
10.2.2 Ajout d'un site	37
10.2.3 Suppression d'un site	38
10.2.4 Visualisation des sites	38
10.2.5 Quitter l'interface	38
10.3 Exécution de l'interface	38
11 Paramétrage serveur et base	40
11.1 Serveur virtuel distant	40
11.2 Base de données	40
Annexes	43
11.3 Connexion au serveur virtuel distant	43
11.4 Connexion à la base de données	43
11.5 Connecter Power BI à la base MySQL	44

1 Contexte

Pour garantir le bon fonctionnement des systèmes de mesure installés, des infrastructures étudiées, ainsi que des dispositifs d'autoproduction et d'autoconsommation, le service exploitation de l'entreprise assure un suivi rigoureux des données de ses clients. Ce suivi permet d'obtenir une vue d'ensemble complète sur les outils déployés et d'assurer leur efficacité et leur performance. Grâce à cette approche proactive, le service exploitation veille à ce que chaque composant fonctionne de manière optimale, contribuant ainsi à maximiser les bénéfices de l'autoconsommation collective pour les clients.

2 Présentation de l'application existante

Cohérence Energies reçoit chaque mois des données de la part d'ENEDIS, ce qu'on appelle des publications. Celles-ci sont reçues par e-mail par chacun des sites producteurs d'une boucle. Le fournisseur envoie un premier e-mail avec un dossier en pièce jointe, protégé par un mot de passe, contenant un certain nombre de fichiers .csv et .xlsx. Le second e-mail envoyé par le fournisseur contient le mot de passe permettant de décompresser le dossier.

Le chargé d'exploitation plaçait alors le dossier dans un répertoire et le décompressait manuellement pour avoir accès aux fichiers de données. Ceux-ci se décomposent en différents types : les fichiers Courbe De Charge, dits CDC, et les fichiers dits Énergie. Le premier est composé d'un ensemble de valeurs horodatées de la puissance électrique (en Watt) mesurée pendant un intervalle de temps défini pour un PRM (Point de Raccordement Mesurage) – un identifiant utilisé pour désigner un compteur spécifique et enregistrer les flux d'électricité qui y transitent.


Le second type de fichier propose, à travers une seule valeur, l'énergie (en kWh) mesurée pour un PRM sur l'entièreté d'un mois. Dans le dossier, on retrouve des fichiers CDC et Énergie sur la consommation, la production, l'autoconsommation, l'autoproduction et le surplus d'énergie. En d'autres termes, les fichiers CDC fournissent un

détail plus fin par rapport aux fichiers Énergie, qui représentent une agrégation des données sur une période plus longue.

Une fois le dossier décompressé, le chargé d'exploitation va chercher à vérifier que les données envoyées par ENEDIS sont cohérentes. Par cela, il faut comprendre que les données Courbe De Charge (CDC) et les données Énergie sont liées : on peut retrouver les valeurs du fichier Énergie à partir des valeurs d'un fichier CDC. En effet, l'un possède des données de puissance (en Watt), tandis que l'autre contient des données d'énergie (en kWh). Un simple calcul permet de vérifier si les valeurs correspondent : on calcule la moyenne de la puissance (Watt) pour chaque heure afin d'obtenir les Wh, puis on fait la somme de ces moyennes que l'on divise par 1000 pour obtenir l'énergie en kWh sur le mois.

01/03/2024 10:00	0	0
01/03/2024 11:00	0	1
01/03/2024 12:00	1	81
01/03/2024 13:00	688	723
01/03/2024 14:00	354	612
01/03/2024 15:00	1049	324
01/03/2024 16:00	620	112
01/03/2024 17:00	22	0

**Mesure pour la plage horaire
17h00 – 17h30**



**Mesure pour la plage horaire
17h30 – 18h00**




FIGURE 1 – Exemple CDC plage horaire

Le chargé d'exploitation effectuait alors ce calcul manuellement sur Excel pour chaque paire de fichiers CDC / Énergie.

D'autres fichiers de type feuille de calcul Excel constituent le dossier. Le fichier Synthèse, constitué de trois pages rassemble les données d'énergie des fichiers Energie. La première page du fichier donne les indicateurs généraux de la boucle d'autoconsommation (Consommation globale, production globale, ...). La seconde

page indique la quantité d'énergie produite par chaque producteur ainsi que l'énergie consommée par l'ensemble des consommateurs associés à chaque producteur. Enfin, la dernière page donne des détails sur les données de consommation et autoconsommation pour les consommateurs de la boucle. Aussi, un fichier « Périmètre de participation » répertorie les PRM des utilisateurs de la boucle (producteurs et consommateurs), ainsi que la date de début et la date de fin de l'autoconsommation.

Une fois que le chargé d'exploitation avait vérifié la cohérence des données, il alimentait des tableaux de données dans l'outil PowerBI à partir de tables qu'il avait préalablement créées. Ainsi, il importait les données manuellement jusqu'aux tables suivantes :

- La table "CONSO_Synthèse_ENEDIS" tire ses informations de la troisième page du fichier synthèse en appliquant de légères modifications telles que l'encodage des variables.
- La table "PROD_Synthèse_ENEDIS (Autoconso)" récupère ses informations de la deuxième page du fichier synthèse. De plus, elle est couplée avec une table extérieure pour obtenir les noms des consommateurs et des producteurs.
- La table "Particulier CDC ENEDIS" tire ses informations des fichiers Courbe De Charge (CDC) d'autoconsommation des particuliers. Les données sont regroupées par jour pour réduire la dimension de la table, mais l'objectif est de conserver les données par demi-heure.

Les données insérées dans PowerBI ne pouvaient être utilisées que dans PowerBI. Le processus d'insertion des données dans Power BI devait être répété autant de fois qu'il y avait de boucles d'ACC dans la clientèle de l'entreprise.

3 Brève Description de PubliExploit

Les processus d'importation et de traitement des données ont été programmés en Python. Une base de données a été créée pour y stocker les données énergétiques. L'application a été déployée sur un serveur virtuel distant afin de l'exécuter automatiquement à intervalles réguliers.

4 Objectifs de l'application

- Détection des e-mails d'ENEDIS
- Extraction des données des publications et insertion automatique dans la base de données
- Vérification automatique de la corrélation des données
- Sauvegarde des logs pour s'assurer du bon fonctionnement de l'application

5 Fonctionnement général de PubliExploit

5.1 Schéma

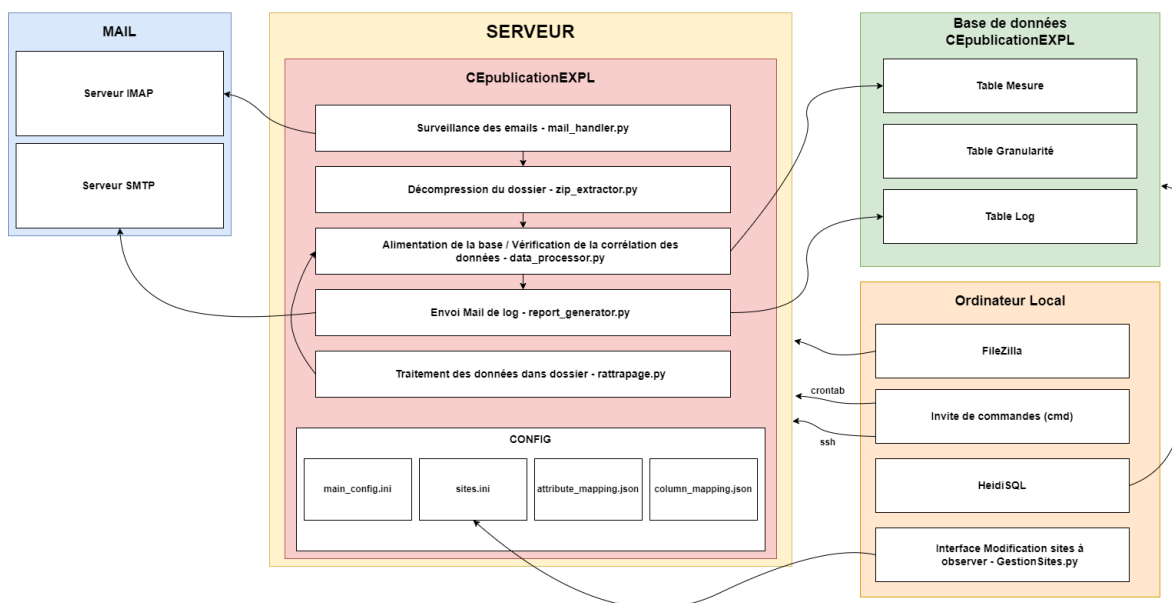


FIGURE 2 – Schéma de l'application

5.2 Surveillance des boîtes et traitement des mails

La première étape de l'automatisation de l'importation des données dans une base consiste à capturer les mails envoyés par ENEDIS. L'objectif est d'identifier et d'analyser les mails reçus dans les différentes boîtes mail associées aux boucles d'auto-consommation, afin d'en extraire le contenu pertinent, à savoir le dossier de publication contenant les données et le mot de passe permettant de déverrouiller ce dossier.

La mise en place d'un programme de traitement des mails permettant de surveiller à intervalles réguliers les boîtes mail a permis de répondre au besoin de capture des mails. Ainsi, nous pouvons détecter automatiquement la réception de nouveaux emails en provenance d'ENEDIS contenant des dossiers compressés, extraire ces fichiers de manière automatisée et les stocker dans un répertoire spécifié.

Pour ce faire, le programme de traitement des mails en arrière-plan est hébergé sur un serveur virtuel distant, en utilisant un outil de planification de tâches. Cela permet

de lancer le traitement des emails à des moments précis de la journée. Un serveur VPS d'OVH a été utilisé pour héberger l'application. Le système d'exploitation du serveur étant Ubuntu, l'outil 'crontab' permet de planifier l'exécution du programme de traitement des mails.

Pour surveiller les boîtes mail, le programme doit se connecter au serveur IMAP avec les informations requises, telles que l'adresse mail et le mot de passe de la boîte concernée. IMAP est un protocole qui permet aux utilisateurs d'accéder à leurs emails depuis un serveur distant. L'application doit pouvoir observer les boîtes mail de toutes les boucles d'autoconsommation (ACC). Ainsi, les éléments de connexion aux boîtes à surveiller ont été mis dans un fichier de configuration. Cela permet de modifier la liste des boîtes à surveiller sans modifier le code. Pour appliquer ces changements aux éléments de connexion aux boîtes, une interface simple a été créée, permettant de consulter la liste des boîtes mail, d'ajouter et de supprimer une boucle d'ACC, facilitant ainsi la gestion des boîtes mail à surveiller.

Le programme boucle sur chaque boîte mail associée aux boucles d'autoconsommation (ACC). Une fois connecté, il lance périodiquement un processus d'analyse des nouveaux emails entrants (les mails non lus). Les emails qui nous intéressent sont ceux provenant d'ENEDIS, et plus particulièrement de l'adresse 'noreply.nova@enedis.fr'. De plus, les emails contenant le dossier de fichiers de données et le mot de passe sont facilement identifiables par un objet de mail spécifique. Le dossier compressé que nous recherchons porte un nom distinct : 'export.zip', ce qui permet de l'identifier. Le mot de passe, quant à lui, se trouve dans le corps du texte d'un des deux emails.

Un programme a été créé pour traiter le corps du message et le décoder, car il est souvent encodé selon des normes telles que l'UTF-8. Une fois le texte décodé, une règle spécifique permet d'identifier le mot de passe parmi les autres éléments du texte. L'adresse du destinataire, le nom du dossier compressé et le texte permettant d'identifier le mot de passe ont été ajoutés à un fichier de configuration. Cela offre la possibilité de modifier les critères de déclenchement sans avoir à modifier le code source du programme.

Grâce à cette étape, le dossier compressé et le mot de passe sont identifiés.

Un répertoire temporaire va être créé afin d'y placer le dossier compressé. Ce répertoire est nommé en utilisant l'identifiant de la boucle d'ACC observée et la période sur laquelle les mesures ont été prises. Ainsi, si le dossier comporte les données pour la boucle ACC00000033, mesurées entre le 1er mars 2024 et le 31 mars 2024, le nom du dossier sera "ACC00000033_01032024_31032024".

Le cas où les deux mails envoyés par ENEDIS ne seraient pas reçus simultanément a été pris en compte. Ainsi, si nous ne recevons qu'un seul mail, nous n'aurons qu'une partie de l'information, rendant impossible la décompression du dossier. Pour pallier ce problème, un tableau a été mis en place rassemblant les informations suivantes : identifiant de la boucle d'ACC, période de mesure, mot de passe, et répertoire contenant le dossier compressé.

Pour chaque mail reçu, une ligne est ajoutée au tableau avec les informations collectées. Pour le mail contenant le dossier compressé, la colonne du mot de passe est laissée à « NULL » puisqu'il n'y a pas de mot de passe dans ce mail. De la même manière, pour le mail contenant le mot de passe, la colonne du répertoire est laissée à « NULL ». Un programme spécifique est chargé de comparer les couples identifiés dans le tableau, basés sur l'identifiant de la boucle et la période. Ainsi, si un couple est identifié, nous pourrions procéder à l'étape de décompression du dossier en utilisant les informations stockées dans le tableau. En revanche, si aucune correspondance n'est trouvée, le processus sera interrompu, et nous devrions attendre qu'un nouveau mail vienne compléter le tableau pour créer un nouveau couple.

BOUCLE	DATE_DEBUT	DATE_FIN	MDP	REPertoire	STATUS	DATE_ENTREE
ACC00000033	01/03/2024	31/03/2024	wxB9D5+429Ad	/home/ubuntu/CEpublicationEXPL/data/output\ACC00000033_01032024_31032024	NULL	2024-06-24 10:38:34
ACC00000033	01/03/2024	31/03/2024	NULL	/home/ubuntu/CEpublicationEXPL/data/output\ACC00000033_01032024_31032024	NULL	2024-06-24 10:38:35

FIGURE 3 – Exemple d'un couple dans la table

5.3 Décompression du dossier

Ayant réussi à obtenir à la fois le dossier zippé et le mot de passe associé grâce aux informations stockées dans la table précédemment mentionnée, le dossier peut être décompressé. Avec ces données en main, nous pouvons nous positionner dans le répertoire dédié et décompresser le dossier en utilisant le mot de passe. Ainsi, les fichiers de données deviennent accessibles.

Une fois le dossier décompressé, le couple traité est automatiquement supprimé du tableau pour éviter un traitement répété.

5.4 Base de données

La base de données est structurée des tables suivantes :

La table 'Mesure' stocke les données énergétiques provenant de tous les fichiers de données, notamment les fichiers Courbe de Charge et Énergie. Elle comporte 11 attributs permettant d'obtenir toutes les informations nécessaires sur une mesure :

- Id_boucle : identifiant de la boucle
- PRM_prod : PRM producteur
- PRM_conso : PRM consommateur
- Attribut : prend les valeurs Consommation, Autoconsommation, Production, Autoproduction, Surplus
- Date de début (avec heure)
- Date de fin (avec heure)
- Valeur : valeur de la mesure
- Granularité : fait référence à l'identifiant de la granularité
- Responsable : correspond au responsable d'équilibre
- Classe : prend les valeurs Base, Heures Pleines, Heures Creuses, etc.
- Etat : prend les valeurs « Vérifié », « Non vérifié »

La table 'Granularité' relie un identifiant à une temporalité. Par exemple, une mesure avec une granularité de 1 correspond à une donnée mesurée sur un mois. Une granularité de 2 indique que la donnée a été mesurée sur une période de 30 minutes, et ainsi de suite. Ainsi la table 'Granularité' contient les attributs suivants :

- Id_Granu : identifiant de la granularité
- Granularité : temps (ex : 15 minutes, 30 minutes, 1 mois, etc.)

L'identifiant de la granularité est ensuite référencé dans la table 'Mesure'.

Il est important de faire un suivi des différentes actions du programme pour s'assurer que tout fonctionne correctement du début à la fin. Ainsi, une table SQL dédiée aux logs a été créée, avec les attributs suivants :

- Id_boucle : identifiant de la boucle
- Periode : periode de mesure
- DateHeure : Date et heure du message de log
- NomFichier : Nom du fichier concerné par le message de log
- Description : le message de log en lui-même
- Importance : prend les valeurs 1 et 2, où 2 fait référence à un message important
- Type : prend les valeurs « INFO », « WARNING » ou « ERROR »

Les données étant maintenant accessibles dans le répertoire et la base de données créée, elle est prête à être alimentée. Pour ce faire, il a fallu se placer dans le répertoire contenant les fichiers et parcourir tous les fichiers qui nous intéressent : les fichiers CD et le fichier 'Synthèse'.

Le programme d'insertion des données va alors parcourir tous les fichiers Courbe de Charge (CDC) ainsi que le fichier Synthèse. Ces deux types de fichiers demandent un traitement différent en raison de leur structure distincte. Le fichier CDC est relativement simple car il ne contient que trois colonnes : date (et heure) de la mesure

et deux valeurs. Pour chaque fichier, je vais transformer chaque valeur en une ligne de ma base de données. Avant de l'insérer dans ma base, je vais d'abord l'intégrer dans un dataframe. Un dataframe est une structure de données en forme de tableau utilisée pour organiser et manipuler des données, couramment utilisée en analyse et traitement de données. Cela me permettra d'effectuer un nettoyage des données si nécessaire, comme supprimer les valeurs vides ou formater les colonnes, par exemple. Ainsi, je me retrouve avec un dataframe contenant toutes mes mesures CDC.

Le fichier 'Synthèse' est plus complexe, comportant déjà trois feuilles, mais seules les deux dernières nous intéressent. Les noms de colonnes dans ce fichier peuvent varier en fonction de la date de publication. Pour résoudre ce problème, un mapping des colonnes a été mis en place pour les renommer dynamiquement. Cela consiste à associer automatiquement les colonnes du fichier aux colonnes de la base de données. Le mapping est stocké dans un fichier de configuration, ce qui permet de le modifier facilement sans toucher au code source. De plus, les données à extraire de chaque page sont identifiées grâce à des critères de regex, des motifs de recherche permettant de sélectionner des colonnes spécifiques. Par exemple, l'expression `.Autoconso.` permet de récupérer des colonnes telles que 'Autoconsommation' ou 'Autoconsommation individuelle'. Les données sont ensuite ajoutées à un dataframe et nettoyées.

Les deux dataframes obtenus sont concaténés pour former un seul dataframe unifié.

Le jeu de données est prêt à être inséré dans la base, mais une dernière étape est nécessaire : la vérification de la cohérence des données.

5.5 Vérifier la cohérence des données

Comme expliqué précédemment, un lien unit les données des fichiers CDC aux données du fichier Synthèse : on doit pouvoir vérifier à l'aide d'un calcul que la somme des puissances calculées à partir des fichiers CDC correspond à l'énergie totale indiquée dans le fichier Synthèse, après avoir pris en compte les différences d'unités

(Watt pour la puissance et kWh pour l'énergie). Pour ce faire, il a fallu parcourir pour chaque PRM et pour chaque attribut (consommation, production, autoconsommation, etc.) les données correspondant aux fichiers CDC en triant le dataframe, puis sommer les valeurs de toutes les lignes. Ensuite, comparer ce nombre à la somme obtenue en triant le dataframe pour obtenir les données correspondant aux données du fichier Synthèse pour ce même PRM et attribut. Si les sommes étaient égales (avec une certaine marge d'erreur), alors les lignes du dataframe correspondant aux fichiers Synthèse pour le PRM et pour l'attribut étaient marquées comme « Vérifié » dans la colonne « État » du tableau. Sinon, les lignes étaient marquées comme « Non vérifié ».

Une fois cela fait, les données peuvent être insérées dans la base, même si certaines valeurs sont considérées comme « Non vérifié ». Ensuite, un traitement est appliqué sur la base de données pour supprimer tous les doublons.

5.6 Gestion des erreurs

Du début à la fin du processus, des messages sont insérés dans la table Log pour informer l'utilisateur de l'application du déroulement du processus. Ces messages fournissent un niveau de détail sur les informations, erreurs ou warnings rencontrés, indiquent les fichiers concernés par les messages et le niveau d'importance à accorder au message de log. Par exemple, lors de la vérification de la cohérence des données, si un écart est constaté entre les deux sommes calculées, alors une erreur est relevée avec la ligne et l'écart spécifiés dans le message.

Cette table SQL permet de centraliser les logs et d'offrir une meilleure traçabilité des actions réalisées, facilitant ainsi le suivi et le diagnostic en cas de problème.

Une fois que les données ont été importées dans la base et que la justesse des données a été vérifiée, un mail est créé à destination de la boîte mail associé à la boucle d'ACC. Les messages de log avec une forte importance (caractérisé par importance = 2 dans la base) sont copiés dans le corps du mail, le dossier de fichier est ajouté en pièce jointe et le mail est envoyé. Le répertoire temporaire créé au début du pro-

cessus est supprimé.

6 Arborescence

L'application est déposée sur un serveur virtuel distant afin d'être exécutée à intervalles réguliers tout au long de la journée. Elle est contenue dans le dossier « CEpublicationEXPL ». Voici l'arborescence de ce répertoire :

```
CEpublicationEXPL
├── README.md
├── config/
│   ├── main_config.ini
│   ├── mappings/
│   │   ├── attribute_mapping.json
│   │   └── column_mapping.json
│   ├── sites.ini
│   └── templates/
├── data/
│   ├── input/
│   │   ├── rattrapage/
│   │   └── tableMail.xlsx
│   ├── output/
│   └── temp/
├── docs/
│   ├── Guide_Utilisation_PubliExploit.pdf
│   ├── DiagrammeActivite.pdf
│   └── SchemaAppli.pdf
├── requirements.txt
├── src/
│   ├── __init__.py
│   ├── __pycache__/
│   ├── cli/
│   │   ├── data_processor_cli.py
│   │   ├── mail_processor_cli.py
│   │   ├── rattrapage_cli.py
│   │   ├── report_generator_cli.py
│   │   └── zip_extractor_cli.py
│   ├── core/
│   │   ├── __pycache__/
│   │   ├── mail_handler.py
│   │   ├── zip_extractor.py
│   │   ├── data_processor.py
│   │   ├── report_generator.py
│   │   ├── rattrapage.py
│   │   └── GestionSites.py
│   └── utils/
│       ├── __pycache__/
│       ├── config_utils.py
│       ├── database_utils.py
│       ├── df_utils.py
│       └── email_utils.py
```



```
├── excel_utils.py
├── file_utils.py
├── imap_utils.py
├── regex_utils.py
├── tests/
│   ├── __init__.py
│   ├── __pycache__/
│   ├── fixtures/
│   │   ├── sample_emails/
│   │   │   ├── password_mail.txt
│   │   │   ├── sample_email.eml
│   │   │   └── sample_email_PJ.eml
│   │   ├── test_config.ini
│   │   ├── test_config_sites.ini
│   │   ├── test_directory_df/
│   │   │   ├── 01305499214693_01032024_31032024_Autoconso_CDC.csv
│   │   │   └── ACC00000066_01032024_31032024_synthese.xlsx
│   │   └── test_verif.xlsx
│   └── unit/
│       ├── test_utils/
│       │   ├── __init__.py
│       │   ├── __pycache__/
│       │   ├── test_config_utils.py
│       │   ├── test_df_utils.py
│       │   ├── test_email_utils.py
│       │   ├── test_excel_utils.py
│       │   ├── test_file_utils.py
│       │   ├── test_imap_utils.py
│       │   ├── test_logging_utils.py
│       │   └── test_regex_utils.py
```

Le dossier `src` contient tous les codes de programmation Python qui constituent l'application.

Dans le sous-répertoire `core`, on retrouve les programmes principaux :

- **mail_handler.py** : Détecte les e-mails d'ENEDIS, identifie le dossier compressé et le mot de passe, stocke les informations dans un tableau de couples d'e-mails, identifie les couples d'e-mails dans le tableau.
- **zip_extractor.py** : Récupère les informations du couple identifié (mot de passe et répertoire où se trouve le dossier), décompresse le dossier, supprime le couple d'e-mails du tableau.
- **data_processor.py** : Parcourt les fichiers CDC et le fichier Synthèse pour extraire les données dans un DataFrame, vérifie la justesse des données, insère les données dans la base de données, supprime les doublons de la table Mesure de la base.
- **report_generator.py** : Récupère les messages de log associés à la boucle et à la période concernées et les insère dans un e-mail qui fait état de l'insertion des données dans la base et de la vérification de la cohérence des données. Ajoute le dossier de données (compressé mais non protégé) et envoie l'e-mail. Supprime les logs anciens de plus de trois mois.
- **rattrapage.py** : Permet d'insérer les données des publications stockées dans des répertoires dans la base de données. Appelle le programme `Data_processor.py`.
- **GestionSites.py** : Permet de modifier le fichier de configuration `sites.ini`.

Dans le sous-répertoire `utils`, on retrouve les programmes de fonctions utilitaires, regroupés par thème :

- **config_utils.py** : Contient les fonctions relatives à la gestion des fichiers de configuration. Ce module inclut des fonctions permettant de lire les paramètres de configuration depuis un fichier spécifié, facilitant ainsi la configuration et la personnalisation des applications. Les principales fonctionnalités incluent :

- **read_config(config_file)** : Lit et renvoie les paramètres de configuration depuis un fichier donné.
- **database_utils.py** : Contient les fonctions relatives à la gestion des interactions avec la base de données MySQL. Ce module inclut des fonctions pour établir des connexions, insérer des données, enregistrer des logs, supprimer des données anciennes, et éliminer les doublons. Les principales fonctionnalités incluent :
 - **connect_to_database()** : Établit une connexion à la base de données MySQL en utilisant les paramètres de connexion spécifiés et retourne un objet de connexion.
 - **log_to_database(id_alim, periode, date_heure, nom_fichier, description, importance, type_log)** : Écrit un log dans la base de données, permettant de suivre les opérations effectuées et les erreurs rencontrées.
 - **delete_old_logs(num_boucle, periode)** : Supprime les logs de la table "Log" qui sont plus anciens de trois mois.
 - **remove_duplicates(num_boucle, periode)** : Supprime les doublons de la table "Mesure" de la base de données.
 - **delete_test_logs(num_boucle, periode)** : Supprime les lignes de la table 'Log' qui correspondent aux messages de journal générés par les fonctions de test.
- **df_utils.py** : Contient les fonctions relatives au traitement et à la manipulation des DataFrames pour le traitement des données. Ce module inclut des fonctions pour créer la structure de la base de données en dataframe, traiter les fichiers CDC et de synthèse, harmoniser les attributs, et vérifier la cohérence des données. Les principales fonctionnalités incluent :
 - **create_database_structure(num_boucle, periode)** : Crée un DataFrame vide avec la structure de la base de données pour stocker les données.
 - **process_files_in_directory(directory, id_boucle, date_debut, date_fin, periode)** : Traite tous les fichiers dans un répertoire donné, en distinguant les fichiers CDC et les fichiers de synthèse.

- **process_cdc_file(file_path, id_boucle, prm, attribut, periode)** : Traite un fichier CDC spécifique et retourne un DataFrame structuré avec les données extraites.
 - **process_synthese_file(file_path, data_frames, id_boucle, date_debut, date_fin, periode)** : Traite un fichier de synthèse Excel, en extrayant les données pertinentes des différentes feuilles.
 - **rename_columns(df, num_boucle, periode)** : Renomme les colonnes d'un DataFrame en utilisant un mapping défini dans un fichier JSON.
 - **harmonize_attribut(attribut)** : Harmonise les noms d'attributs en utilisant un dictionnaire de correspondance.
 - **calculer_sommeSynt_page2(df, granu, prm, harmonized_att, sommeCDC, num_boucle, periode)** : Calculer et vérifier la cohérence des données de la page 2 du fichier 'Synthèse'.
 - **calculer_sommeSynt_page3(df, granu, prm, att, harmonized_att, sommeCDC, num_boucle, periode)** : Calculer et vérifier la cohérence des données de la page 3 du fichier 'Synthèse'.
 - **verif_autoprod(df, num_boucle, periode)** : Vérifie la cohérence des données d'autoproduction.
 - **verif_surplus(df, num_boucle, periode)** : Vérifie la cohérence des données de surplus en comparant avec les données d'autoproduction et de production.
-
- **email_utils.py** : Contient les fonctions relatives à la gestion des e-mails, incluant l'extraction, la validation, et l'envoi d'e-mails. Ce module fournit des outils pour traiter les e-mails entrants et sortants dans le contexte de l'application. Les principales fonctionnalités incluent :
 - **extract_email_body(email, num_boucle, periode)** : Extraît le corps textuel d'un e-mail, qu'il soit multipart ou non, en gérant les différents encodages possibles.
 - **decode_email_subject(encoded_subject)** : Décode le sujet d'un e-mail qui peut être encodé de différentes manières, assurant une lecture correcte du sujet.

- **fetch_emails_since_last_check(mail)** : Récupère les e-mails non lus depuis la dernière vérification dans la boîte de réception.
 - **is_email_valid(email, num_boucle, periode)** : Vérifie la validité d'une adresse e-mail en utilisant la bibliothèque email_validator.
 - **send_email(smtp_server, smtp_username, smtp_password, sender_email, receiver_email, subject, num_boucle, periode, zip_file_path)** : Envoie un e-mail contenant les logs de la base de données et un fichier ZIP en pièce jointe. Cette fonction gère également la création du sujet de l'e-mail en fonction du nombre d'erreurs rencontrées lors du traitement.
-
- **excel_utils.py** : Contient les fonctions relatives à la manipulation et à la gestion des fichiers Excel dans le contexte de l'application. Ce module fournit des outils pour écrire, lire et mettre à jour des informations dans des fichiers Excel. Les principales fonctionnalités incluent :
 - **write_to_excel(excel_file, num_boucle, date_debut, date_fin, password, folder_path)** : Écrit une nouvelle ligne d'informations dans le tableau de couple de mails existant. Le ligne d'informations contient le numéro de boucle, les dates de début et de fin, le mot de passe, et le chemin du dossier.
 - **find_matching_entries(excel_file, num_boucle, periode)** : Recherche dans le tableau les couples de mails par rapport aux un numéro de boucle et une période spécifiques. Met à jour le statut des lignes correspondantes et renvoie le mot de passe et le répertoire associés si un couple est trouvé.
 - **remove_ok_status_rows(excel_file, num_boucle, periode)** : Supprime les couples de mails traités. Cette fonction permet de nettoyer le fichier des entrées qui ont déjà été traitées.
-
- **file_utils.py** : Contient les fonctions relatives à la manipulation et à la gestion des fichiers et répertoires dans le contexte de l'application. Ce module fournit des outils pour créer, déplacer, compresser et supprimer des fichiers et répertoires. Les principales fonctionnalités incluent :
 - **create_directory(folder_path)** : Crée un nouveau répertoire si celui-ci n'existe pas déjà.

- **move_zip_to_directory(email, folder_path, zip_filename, num_boucle, periode)** : Déplace un fichier ZIP attaché à un email vers un répertoire spécifié.
 - **get_first_directory_name(directory)** : Retourne le nom du premier sous-répertoire trouvé dans le répertoire spécifié.
 - **list_zip_files(directory)** : Renvoie une liste de tous les fichiers ZIP présents dans le répertoire spécifié et ses sous-répertoires.
 - **compresser_dossier(dossier_path, archive_path, num_boucle, periode)** : Comprime un dossier spécifié en un fichier ZIP.
 - **supprimer_fichier(fichier_path, num_boucle, periode)** : Supprime un fichier spécifié et enregistre l'action dans le journal.
 - **get_first_zip_file(directory)** : Retourne le nom du premier fichier ZIP trouvé dans le répertoire spécifié.
 - **delete_directory(directory, num_boucle, periode)** : Supprime un répertoire spécifié et tout son contenu.
-
- **imap_utils.py** : Contient les fonctions relatives à la connexion et à la déconnexion d'un serveur IMAP (Internet Message Access Protocol). Ce module fournit des outils pour établir et terminer une connexion sécurisée à un serveur de messagerie. Les principales fonctionnalités incluent :
 - **connect_to_imap(config, config_sites)** : Établit une connexion sécurisée à un serveur IMAP. Cette fonction : - Récupère les informations de configuration (serveur, nom d'utilisateur, mot de passe) à partir des objets de configuration fournis. - Établit une connexion SSL au serveur IMAP. - Authentifie l'utilisateur avec les identifiants fournis. - Retourne l'objet de connexion IMAP en cas de succès. - En cas d'erreur, affiche un message d'erreur et termine le programme.
 - **disconnect_from_imap(mail)** : Ferme proprement une connexion IMAP existante. Cette fonction : - Tente de se déconnecter du serveur IMAP. - Affiche un message de confirmation en cas de succès. - Affiche un message d'erreur en cas d'échec de la déconnexion.

- **regex_utils.py** : Contient les fonctions utilisant des expressions régulières pour extraire des informations spécifiques à partir de chaînes de caractères, notamment des sujets d'e-mails, du contenu d'e-mails et des noms de fichiers. Les principales fonctionnalités incluent :
 - **extract_boucle_num(subject)** : Extrait le numéro de boucle du sujet de l'e-mail. Recherche le motif 'Convention N° suivi du numéro.
 - **extract_dates(subject, num_boucle)** : Extrait les dates de début et de fin à partir du sujet de l'e-mail. Recherche le motif 'Période du « date1 » au « date2 »'.
 - **extract_password(email_content, num_boucle, periode)** : Extrait le mot de passe du contenu de l'e-mail. Recherche une chaîne de 12 caractères contenant des minuscules, majuscules, chiffres et caractères spéciaux.
 - **extract_prm(file_name, num_boucle, periode)** : Extrait le PRM (Point de Référence Mesure) du nom de fichier. Récupère la partie du nom de fichier avant le premier underscore.
 - **extract_attribut(file_name, num_boucle, periode)** : : Extrait l'attribut du nom de fichier. Recherche un motif spécifique entre deux underscores suivis de '_CDC'.

Dans le sous-répertoire `cli` on retrouve les programmes d'interface en ligne de commande (CLI) qui servent à exécuter les différentes fonctionnalités de l'application :

- `mail_processor_cli.py` : Permet de lancer le programme `mail_processor.py`
- `zip_extractor_cli.py` : Permet de lancer le programme `zip_extractor.py`
- `data_processor_cli.py` : Permet de lancer le programme `data_processor_cli.py`
- `report_generator_cli.py` : Permet de lancer le programme `report_generator.py`
- `rattrapage_cli.py` : Permet de lancer le programme `rattrapage_cli.py`

Le dossier `config` contient tous les différents fichiers de configuration pour que l'application fonctionne correctement.

Dans le sous-répertoire `mappings` on retrouve les mappings utilisés dans '`df_utils.py`' pour renommer les colonnes et les attributs :

```
{
  "column_mapping": {
    "Id_boucle": [],
    "PRM_prod": [".*PRM.*Prod(ucteur)?.*", "PRM\\s*Prod", ".*Prod(ucteur)?.*"],
    "PRM_conso": [".*PRM.*Cons(ommateur)?.*", "PRM\\s*Conso"],
    "Attribut": [],
    "Date_debut": ["Date.*Debut", "Debut"],
    "Date_fin": ["Date.*Fin", "Fin"],
    "Valeur": [],
    "Granularite": [".*Granu(larite)?.*"],
    "Respo": [".*Respo.*"],
    "Classe": ["Classe"],
    "Etat": ["Etat"]
  }
}
```

FIGURE 4 – Fichier column_mapping.json

Ce fichier JSON définit un mapping entre des noms de colonnes "types" (ou standardisés) et les différentes expressions qui peuvent être trouvées dans les fichiers de données et qui doivent être renommées en ces colonnes types. Les clés à gauche (comme "Id_boucle", "PRM_prod", "PRM_conso", etc.) représentent les noms de colonnes standardisés que le programme utilisera internement. Les tableaux à droite contiennent des expressions régulières ou des chaînes de caractères qui correspondent aux différentes façons dont ces colonnes pourraient être nommées dans les fichiers de données d'entrée. Par exemple, pour "PRM_prod", le programme cherchera des en-têtes de colonne qui correspondent à l'une des expressions listées (".PRM.Prod(ucteur)?.", "PRM\\sProd", ".*Prod(ucteur)?.") et les renommera en "PRM_prod".

```
{
  "Autoconsommation": "Autoproduction",
  "Surplus": "Surplus",
  "Surplus Réparti": "Surplus",
  "Consommation": "Consommation",
  "Complément": "Complément",
  "Production": "Production"
}
```

FIGURE 5 – Fichier attribute_mapping.json

C'est le même fonctionnement pour les attributs.

A la racine du dossier `config`, on retrouve aussi deux fichiers de configuration :

- **main_config.ini** : est un fichier de configuration général qui contient des paramètres essentiels pour différentes parties de l'application. Il est structuré en sections et inclut des informations pour se connecter aux serveurs de messagerie et gérer les e-mails. Voici un aperçu des sections et de leurs contenus :

1. **[IMAP]** : Configuration pour la connexion au serveur IMAP (pour lire les e-mails).

- `server` : Adresse du serveur IMAP permettant de se connecter au serveur est donc détecter l'arrivée de mail.

2. **[SMTP]** : Configuration pour la connexion au serveur SMTP (pour envoyer les e-mails).

- `server` : Adresse du serveur SMTP permettant de se connecter au serveur est donc de pouvoir envoyé des mails.

3. **[Mail]** : Paramètres relatifs au traitement des e-mails.

- `sender` : Adresse e-mail de l'expéditeur d'un mail. C'est l'adresse mail de la personne qui envoie un mail. Ici, l'adresse qui nous intéresse est celle d'ENEDIS : 'noreplynova@enedis.fr'.
- `subject_keyword` : Mot-clé à rechercher dans l'objet des e-mails. Va permettre de déterminer si le mail concerné contient ou non le mot de passe pour déverrouiller le dossier protégé.
- `compressed_filename` : Nom du dossier protégé que l'on cherche

4. **[Destination]** : Paramètres pour le stockage des fichiers traités.

- `folder` : Chemin du répertoire de destination pour les fichiers de sortie. C'est à cet endroit que les dossiers temporaires sont déposés pour traiter la donnée. Les dossiers étant temporaires, c'est normal si le répertoire est vide.

- **sites.ini** est un fichier de configuration qui contient les informations d'authentification des différents à observer. Chaque section représente un compte distinct avec ses propres identifiants. Voici la structure du fichier :

1. **[NomDuSite1]** : Informations d'authentification pour le compte principal.
 - username : Adresse e-mail du compte
 - password : Mot de passe associé
2. **[NomDuSite2]** : Informations d'authentification pour le compte principal.
 - username : Adresse e-mail du compte
 - password : Mot de passe associé
3. **[Etc...]**

Des sites peuvent être ajoutés ou supprimer manuellement en accédant à ce fichier. Ou bien en utilisant l'interface créée à cet effet. **Cliquez ici** pour en apprendre plus sur l'interface.

Le dossier `data` contient tous les fichiers de données différents nécessaires au bon fonctionnement de l'application.

Dans le sous-répertoire `input`, on trouve 'tableMail.xlsx', qui est le tableau des couples d'adresses e-mail. La feuille de ce fichier est protégée par un mot de passe (1234) pour éviter toute altération. Il existe également un dossier 'rattrapage' où peuvent être déposés des dossiers contenant des fichiers .csv et .xlsx.

Comme mentionné précédemment, le sous-répertoire `output` est utilisé pour contenir temporairement les dossiers où les données sont traitées. Ces dossiers sont ensuite supprimés.

Le sous-répertoire `temp` peut servir à stocker des données quelconques.

Le dossier `docs` contient le guide d'utilisation et le diagramme d'activité de l'application.

Le dossier `tests` contient des tests du code de programmation Python qui constitue l'application.

Dans le sous-répertoire `unit/test_utils` on retrouve les tests des fonctions utilitaires :

- **Test_config_utils.py** : Test les fonctions du programme **config_utils.py**
- **Test_df_utils.py** : Test les fonctions du programme **df_utils.py**
- **Test_email_utils.py** : Test les fonctions du programme **email_utils.py**
- **Test_excel_utils.py** : Test les fonctions du programme **excel_utils.py**
- **Test_file_utils.py** : Test les fonctions du programme **file_utils.py**
- **Test_imap_utils.py** : Test les fonctions du programme **imap_utils.py**
- **Test_regex_utils.py** : Test les fonctions du programme **regex_utils.py**

Dans le sous-répertoire `fixtures` on retrouve les fichiers de test nécessaire aux tests des fonctions utilitaires :

- Le dossier `'sample_emails'` contient un échantillon de mails contenant un mot de passe, un échantillon de mail contenant une PJ `'export.zip'`, et un fichier texte `'password_mail'` contenant le corps d'un mail avec un mot de passe.
- Le dossier `'test_directory_df'` contient un fichier de données CDC et un fichier de données `'Synthèse'`.
- À la racine du répertoire, on trouve deux fichiers de configuration : `'test_config.ini'` et `'test_config_sites.ini'`. Ils imitent respectivement les fichiers de configuration `'main_config.ini'` et `'sites.ini'`. De plus, il y a un fichier `'test_verif.xlsx'` qui permet d'effectuer des tests sur la vérification de la corrélation des données.

Le fichier `requirements.txt` est un fichier texte utilisé pour spécifier les dépendances nécessaires à l'exécution du code. Chaque ligne de ce fichier liste un module Python spécifique avec une version particulière, assurant ainsi que tous les composants du projet peuvent être reproduits avec précision sur différents environnements.

7 Exécution

Pour lancer l'application, vous devez d'abord vous connecter au serveur virtuel distant. Voici comment procéder :

1. Ouvrir un terminal ou une invite de commandes.
2. Entrer les lignes suivantes :

```
ssh ubuntu@51.254.35.169
password : AvMzBKeBkzCT
```

Vous arriverez à la racine du serveur virtuel distant.

7.1 Exécution à intervalles réguliers

Actuellement, sur le serveur virtuel distant, l'application est exécutée deux fois par jour : à 7h30 et à 12h30 durant la pause méridienne. L'exécution de l'application est programmée par l'outil `crontab` sur Ubuntu.

Pour consulter la configuration de l'exécution de l'application, vous pouvez taper `crontab -l` dans le terminal. Pour modifier la configuration de l'exécution, utilisez `crontab -e`.

7.2 Exécution manuelle

Vous pouvez lancer l'application depuis la racine du serveur en utilisant la ligne de commande suivante (se connecter au serveur via terminal préalablement, [cliquez-ici](#) pour avoir les instructions) :

```
python3 /home/ubuntu/CEpublicationEXPL/src/cli/mail_processor_cli.py /home/
ubuntu/CEpublicationEXPL/config/main_config.ini /home/ubuntu/CEpublicationEXPL/
config/sites.ini
```

Remarquez que cette commande lance le script Python `mail_processor_cli.py` qui prend en entrée les deux fichiers de configuration `main_config.ini` et `sites.ini`.

7.3 Rattrapage

Avant de lancer le rattrapage des fichiers de données qui n'ont pas été traités, placez-les dans l'application. Déposez-les dans le répertoire `CEpublicationEXPL/data/input/rattrapage` prévu à cet effet. Ensuite, exécutez la commande suivante depuis la racine du serveur :

```
python3 /home/ubuntu/CEpublicationEXPL/src/cli/rattrapage_cli.py /home/ubuntu/CEpublicationEXPL/data/input/rattrapage adresse_mail MDP_adresse
```

Cette commande lance le script Python `rattrapage_cli.py` qui prend en entrée un chemin de répertoire correspondant à l'emplacement des dossiers à rattraper. Assurez-vous que le chemin pointe directement vers le répertoire contenant les dossiers à rattraper. Par exemple, si les dossiers à rattraper sont placés dans `CEpublicationEXPL/data/input/rattrapage/site1`, utilisez ce chemin dans la commande. Les noms des dossiers à rattraper doivent suivre le format spécifique : `NumeroBoucle_Personne Morale_DateDebut_DateFin`. D'autres arguments sont également passés à ce script : une adresse mail et un mot de passe (associé à l'adresse). Cela permet de se connecter au serveur IMAP et d'envoyer un rapport par mail lorsque le processus est terminé.

7.4 Exécution des tests

Les tests doivent être exécutés un par un. Vous pouvez les exécuter depuis la racine du serveur avec la commande suivante :

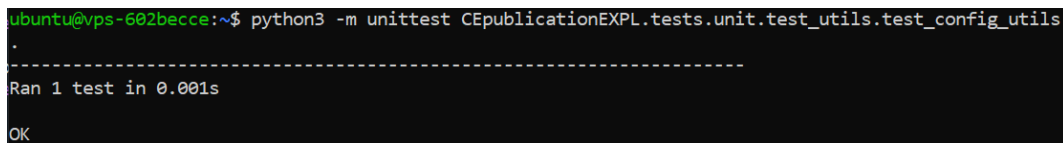
```
python3 -m unittest /home/ubuntu/CEpublicationEXPL/tests/unit/test_utils/test_config_utils.py
```

Remplacez `test_config_utils.py` par le nom du fichier de test que vous souhaitez exécuter pour les autres tests.

8 Déroulement des tests

Dans le cadre du développement logiciel, les tests unitaires jouent un rôle crucial pour vérifier le bon fonctionnement des composants individuels du code. Le processus de test unitaire commence par la préparation de l'environnement, où le framework de test, comme unittest dans notre cas, charge les classes de test définies. Ensuite, chaque méthode de test est exécutée de manière isolée. Ces méthodes comprennent généralement la mise en place de l'environnement spécifique au test (setUp), l'appel de la fonction à tester, et des assertions pour vérifier que les résultats obtenus correspondent aux attentes.

Après l'exécution de chaque test, un éventuel nettoyage est effectué pour restaurer l'environnement à son état initial (tearDown). Une fois tous les tests terminés, le framework génère un rapport synthétique qui résume les résultats.

A terminal window with a black background and green text. The command 'python3 -m unittest CEpublisherEXPL.tests.unit.test_utils.test_config_utils' is entered. The output shows a single dot '.' on the first line, followed by a dashed line '-----', then 'Ran 1 test in 0.001s', and finally 'OK' on the last line.

```
ubuntu@vps-602becce:~$ python3 -m unittest CEpublisherEXPL.tests.unit.test_utils.test_config_utils
.
-----
Ran 1 test in 0.001s
OK
```

FIGURE 6 – Test fonctions du fichier config_utils.py

La lecture de la sortie des tests est relativement simple. Dans l'exemple fourni, nous voyons une ligne avec un point, indiquant qu'un test a été exécuté avec succès. Ce point serait remplacé par un 'E' en cas d'erreur ou un 'F' en cas d'échec du test. Ensuite, un résumé indique le nombre total de tests exécutés et leur durée. Dans ce cas, un seul test a été effectué en 0.001 secondes. La mention "OK" à la fin confirme que tous les tests ont réussi.

Cette sortie concise permet aux développeurs d'avoir un aperçu rapide de l'état des tests. En cas d'échec, des informations détaillées seraient affichées pour aider à identifier et résoudre les problèmes. L'interprétation de ces résultats est essentielle pour maintenir la qualité du code et guider les améliorations nécessaires.

9 Configuration de l'application

L'application a été conçue pour être aussi configurable que possible, offrant ainsi une flexibilité optimale selon les besoins spécifiques de l'utilisateur. Nous allons maintenant explorer comment configurer cette application pour tirer le meilleur parti de ses fonctionnalités et l'adapter à vos exigences particulières.

9.1 Fichiers de configuration

Le dossier `config` contient tous les différents fichiers de configuration pour que l'application fonctionne correctement.

Dans le sous-répertoire `mappings` on retrouve les mappings utilisés dans `'df_utils.py'` pour renommer les colonnes et les attributs :

```
{
  "column_mapping": {
    "Id_boucle": [],
    "PRM_prod": [".*PRM.*Prod(ucteur)?.*", "PRM\\s*Prod", ".*Prod(ucteur)?.*"],
    "PRM_conso": [".*PRM.*Cons(ommateur)?.*", "PRM\\s*Conso"],
    "Attribut": [],
    "Date_debut": ["Date.*Debut", "Debut"],
    "Date_fin": ["Date.*Fin", "Fin"],
    "Valeur": [],
    "Granularite": [".*Granu(larite)?.*"],
    "Respo": [".*Respo.*"],
    "Classe": ["Classe"],
    "Etat": ["Etat"]
  }
}
```

FIGURE 7 – Fichier `column_mapping.json`

Ce fichier JSON définit un mapping entre des noms de colonnes "types" (ou standardisés) et les différentes expressions qui peuvent être trouvées dans les fichiers de données et qui doivent être renommées en ces colonnes types. Les clés à gauche (comme `"Id_boucle"`, `"PRM_prod"`, `"PRM_conso"`, etc.) représentent les noms de colonnes standardisés que le programme utilisera internement. Les tableaux à droite contiennent des expressions régulières ou des chaînes de caractères qui corres-

pondent aux différentes façons dont ces colonnes pourraient être nommées dans les fichiers de données d'entrée. Par exemple, pour "PRM_prod", le programme cherchera des en-têtes de colonne qui correspondent à l'une des expressions listées (".PRM.Prod(ucteur) ?.", "PRM\sProd", ".Prod(ucteur) ?.") et les renommera en "PRM_prod".

```
{
  "Autoconsommation": "Autoproduction",
  "Surplus": "Surplus",
  "Surplus Réparti": "Surplus",
  "Consommation": "Consommation",
  "Complément": "Complément",
  "Production": "Production"
}
```

FIGURE 8 – Fichier attribute_mapping.json

C'est le même fonctionnement pour les attributs.

A la racine du dossier `config`, on retrouve aussi deux fichiers de configuration :

- **main_config.ini** : est un fichier de configuration général qui contient des paramètres essentiels pour différentes parties de l'application. Il est structuré en sections et inclut des informations pour se connecter aux serveurs de messagerie et gérer les e-mails. Voici un aperçu des sections et de leurs contenus :

1. **[IMAP]** : Configuration pour la connexion au serveur IMAP (pour lire les e-mails).
 - **server** : Adresse du serveur IMAP permettant de se connecter au serveur est donc détecter l'arrivée de mail.
2. **[SMTP]** : Configuration pour la connexion au serveur SMTP (pour envoyer les e-mails).
 - **server** : Adresse du serveur SMTP permettant de se connecter au serveur est donc de pouvoir envoyé des mails.
3. **[Mail]** : Paramètres relatifs au traitement des e-mails.
 - **sender** : Adresse e-mail de l'expéditeur d'un mail. C'est l'adresse mail

de la personne qui envoie un mail. Ici, l'adresse qui nous intéresse est celle d'ENEDIS : 'noreplynova@enedis.fr'.

- `subject_keyword` : Mot-clé à rechercher dans l'objet des e-mails. Va permettre de déterminer si le mail concerné contient ou non le mot de passe pour déverrouiller le dossier protégé.
- `compressed_filename` : Nom du dossier protégé que l'on cherche

4. **[Destination]** : Paramètres pour le stockage des fichiers traités.

- `folder` : Chemin du répertoire de destination pour les fichiers de sortie. C'est à cet endroit que les dossiers temporaires sont déposés pour traiter la donnée. Les dossiers étant temporaires, c'est normal si le répertoire est vide.

- **sites.ini** est un fichier de configuration qui contient les informations d'authentification des différents à observer. Chaque section représente un compte distinct avec ses propres identifiants. Voici la structure du fichier :

1. **[NomDuSite1]** : Informations d'authentification pour le compte principal.

- `username` : Adresse e-mail du compte
- `password` : Mot de passe associé

2. **[NomDuSite2]** : Informations d'authentification pour le compte principal.

- `username` : Adresse e-mail du compte
- `password` : Mot de passe associé

3. **[Etc...]**

Des sites peuvent être ajoutés ou supprimer manuellement en accédant à ce fichier. Ou bien en utilisant l'interface créée à cet effet. [Cliquez ici](#) pour en apprendre plus sur l'interface.

9.2 Modification du code source

Dans la fonction `find_matching_entries`, le programme cherche les couples de mails dans le tableau `tableMail.xlsx`. Si un mail n'a pas trouvé de mail complémentaire depuis quelques jours, un message d'avertissement est enregistré dans la

table Log de la base de données. Par défaut, le message est relevé si le mail attend depuis 5 jours. Cependant, cette valeur peut être modifiée en ajustant simplement ce chiffre dans la fonction du fichier `excel_utils.py`.

```
if status == "NULL":
    # Vérifier si la ligne a été ajoutée il y a plus de 5 jours
    if (datetime.datetime.now() - date_entree).days > 5:
        log_to_database(num_boucle, periode, datetime.datetime.now(), f"{excel_name}", f"Ligne ajoutée il y a plus de 5 jours : {boucle}, {date_
```

FIGURE 9 –

Pour changer cette valeur, ouvrez le fichier `excel_utils.py`, modifiez le nombre de jours selon vos besoins, enregistrez le fichier, puis poussez la nouvelle version sur le serveur en remplaçant l'ancienne. Vous pouvez réaliser ce transfert via FileZilla.

Dans les fonctions de vérification de la corrélation des données, telles que `calculer_sommeSynt_page2`, `calculer_sommeSynt_page3`, `verif_autoprod`, et `verif_surplus`, le programme compare les chiffres correspondant aux fichiers CDC et Synthèse. Pour vérifier si les données sont corrélées, la différence entre ces deux nombres est calculée avec une marge d'erreur. Cette marge d'erreur est initialement fixée à 1 mais peut être modifiée en accédant aux fonctions dans le fichier `df_utils.py`. En ajustant cette marge, vous pouvez personnaliser la sensibilité des vérifications de corrélation en fonction de vos besoins spécifiques.

Pour modifier cette marge d'erreur, ouvrez le fichier `df_utils.py` dans votre répertoire de projet avec un éditeur de texte ou un IDE. Recherchez les fonctions `calculer_sommeSynt_page2`, `calculer_sommeSynt_page3`, `verif_autoprod`, et `verif_surplus`. Dans chaque fonction, identifiez la ligne correspondante et modifiez sa valeur selon vos besoins.

```
# Comparaison entre sommeSynt et sommeCDC (avec un écart accepté)
if abs(sommeSynt - sommeCDC) < 1:
```

FIGURE 10 –

```
# Comparaison entre sommeAutoProd et autoProd_value
if abs(sommeAutoProd - autoProd_value) < 1:
```

FIGURE 11 –

```
# Comparer la différence d'autoproduction avec le surplus
if abs(diffAutoProd - surplus) < 1:
```

FIGURE 12 –

Une fois les modifications effectuées, enregistrez le fichier `df_utils.py`. Ensuite, remplacez l'ancienne version du fichier sur le serveur avec la nouvelle version mise à jour. Vous pouvez utiliser un outil comme FileZilla pour effectuer ce transfert. En suivant ces étapes, vous pouvez ajuster la marge d'erreur utilisée pour la vérification de la corrélation des données, afin de garantir que les vérifications correspondent à vos critères de précision.

Les fonctions permettant d'extraire des caractères grâce à des expressions régulières (regex) sont également paramétrables. Ainsi, les cinq fonctions du fichier `regex_utils.py` peuvent être configurées.

Par exemple, la fonction `extract_boucle_num` permet d'extraire le numéro de boucle d'ACC depuis le sujet d'un e-mail. Actuellement, cette extraction est réalisée par le motif `match=re.search(r'[Cc]onvention N°(\w+)', subject)`. Ce motif recherche la séquence 'Convention N°' suivie d'un ensemble de caractères alphanumériques (`\w+`). Si une correspondance est trouvée, elle est renvoyée comme le numéro de boucle, sinon, un message d'erreur est affiché.

La fonction `extract_boucle_num` est conçue pour extraire le numéro de boucle d'ACC à partir du sujet d'un e-mail. Elle utilise le motif `r'[Cc]onvention N°(\w+)'` pour rechercher la chaîne "Convention N°" suivie d'un numéro. Si ce motif est trouvé dans le sujet de l'e-mail, la fonction retourne le numéro de boucle extrait.

Pour obtenir les dates de début et de fin à partir du sujet d'un e-mail, la fonction `extract_dates` utilise le motif `r'Periode du « (\d{2}/\d{2}/\d{4}) » au « (\d{2}/\d{2}/\d{4}) »'`. Ce motif permet d'identifier les dates au format "jour/-

mois/année" encadrées par les expressions "Période du « " et " au « ". Si les dates sont trouvées, la fonction retourne un tuple contenant les deux dates.

La fonction `extract_password` est utilisée pour extraire un mot de passe du contenu d'un e-mail. Elle utilise un motif regex flexible qui vérifie la présence de lettres minuscules, majuscules, chiffres et caractères spéciaux dans une chaîne de 12 à 14 caractères. Si un mot de passe correspondant est trouvé, la fonction vérifie s'il est entouré d'astérisques et les retire si nécessaire avant de le renvoyer. En cas d'échec de l'extraction, la fonction enregistre une erreur dans la base de données et renvoie `None`.

Pour extraire le PRM d'un nom de fichier, la fonction `extract_prm` utilise le motif `r'^[^_]*[0-9]'`. Le motif commence par chercher le début exact du nom du fichier, grâce au caractère `^`. Ensuite, il accepte une série de caractères, à condition qu'ils ne soient pas des underscores, ce qui est réalisé par la séquence `[^_]*`. Cette partie est cruciale car elle permet de capturer tous les caractères du PRM jusqu'à ce qu'un underscore soit rencontré ou jusqu'à la fin de la chaîne. Enfin, le motif se termine par `[0-9]`, exigeant qu'un chiffre soit présent à la fin de la séquence capturée.

Enfin, la fonction `extract_attribut` est utilisée pour extraire un attribut d'un nom de fichier. Elle utilise le motif `r'_(.*?)_CDC'` pour chercher une sous-chaîne entre deux underscores suivie de `_CDC`. Si l'attribut est trouvé dans le nom du fichier, la fonction le retourne.

Ces fonctions sont toutes configurables par leurs motifs regex respectifs. Chaque motif peut être ajusté pour s'adapter aux variations dans les données sources, offrant ainsi une flexibilité dans l'extraction des informations nécessaires pour le traitement et la gestion des données dans l'application.

Pour en apprendre plus sur le regex, [**cliquez-ici**](#)

9.3 Configuration de l'exécution automatique

Dans ce tutoriel, nous allons expliquer comment utiliser `crontab` pour planifier l'exécution de l'application.

Où Trouver et Comment Éditer le Fichier Crontab

Le fichier `crontab` est utilisé pour spécifier les tâches à exécuter périodiquement par le démon `cron`.

Pour éditer votre fichier `crontab`, ouvrez un terminal et exécutez la commande suivante :

```
crontab -e
```

Cela ouvrira le fichier `crontab` dans l'éditeur de texte par défaut.

Comment Fonctionne une Commande d'Exécution dans Crontab

Le fichier `crontab` contient une liste de tâches à exécuter à des moments spécifiés. Chaque ligne de ce fichier représente une tâche programmée et suit le format suivant :

```
m h dom mon dow command
```

- **m** : Minute (0-59)
- **h** : Heure (0-23)
- **dom** : Jour du mois (1-31)
- **mon** : Mois (1-12)
- **dow** : Jour de la semaine (0-7) (Dimanche est 0 ou 7)
- **command** : La commande à exécuter

Exemple de Ligne Crontab

Voici un exemple de ligne dans crontab :

```
30 7 * * * python3 /home/ubuntu/CEpublicationEXPL/src/cli/mail_processor_
cli.py /home/ubuntu/CEpublicationEXPL/config/main_config.ini /home/ubuntu/
CEpublicationEXPL/config/sites.ini >> /home/ubuntu/cron_log.txt 2>&1
```

— 30 7 * * * : La commande s'exécute tous les jours à 7h30 du matin.

— python3 /home/ubuntu/CEpublicationEXPL/src/cli/mail_processor_cli.py
/home/ubuntu/CEpublicationEXPL/config/main_config.ini /home/ubuntu/CEpublicationEXPL/config/sites.ini : La commande à exécuter.

— >> /home/ubuntu/cron_log.txt 2>&1 : Redirige la sortie standard et la sortie d'erreur vers le fichier /home/ubuntu/cron_log.txt.

Comment Changer l'Heure de Déclenchement

Pour changer l'heure de déclenchement d'une tâche, modifiez les champs `m` et `h` dans la ligne crontab.

Exemple

Supposons que vous voulez changer l'heure d'exécution de 7h30 à 8h45. Voici comment vous pouvez le faire :

Original :

```
30 7 * * * python3 /home/ubuntu/CEpublicationEXPL/src/cli/mail_processor_
cli.py /home/ubuntu/CEpublicationEXPL/config/main_config.ini /home/ubuntu/
CEpublicationEXPL/config/sites.ini >> /home/ubuntu/cron_log.txt 2>&1
```

Modifié :

```
45 8 * * * python3 /home/ubuntu/CEpublicationEXPL/src/cli/mail_processor_
cli.py /home/ubuntu/CEpublicationEXPL/config/main_config.ini /home/ubuntu/
CEpublicationEXPL/config/sites.ini >> /home/ubuntu/cron_log.txt 2>&1
```

Sauvegarde et Fermeture

Après avoir édité votre fichier crontab, il est important de savoir comment sauvegarder vos modifications et fermer l'éditeur de texte.

Sauvegarder vos modifications :

- Appuyez sur Ctrl + X pour commencer le processus de sauvegarde.
- L'éditeur de texte vous demandera si vous souhaitez sauvegarder les modifications. Appuyez sur Y pour dire "Yes".
- Ensuite, l'éditeur de texte vous demandera de confirmer le nom du fichier. Appuyez sur Entrée pour accepter le nom par défaut (qui est correct).

Fermer l'éditeur de texte :

Une fois que vous avez appuyé sur Entrée, l'éditeur de texte sauvegardera vos modifications et quittera l'éditeur.

10 Interface d'ajout d'un site dans `sites.ini`

Cette interface permet de gérer les sites dans le fichier de configuration `sites.ini`. Elle fournit une interface utilisateur conviviale pour ajouter, supprimer et visualiser des sites. De plus, elle intègre des fonctionnalités pour télécharger et mettre à jour le fichier de configuration sur un serveur distant via SFTP.

10.1 Fonctionnalités de l'interface

L'interface propose les fonctionnalités suivantes :

- **Ajouter un site** : Permet d'ajouter un nouveau site au fichier `sites.ini`. L'utilisateur doit fournir le nom du site, le nom d'utilisateur et le mot de passe.
- **Supprimer un site** : Permet de supprimer un site existant du fichier `sites.ini`. L'utilisateur sélectionne le site à supprimer dans une liste déroulante.
- **Visualiser les sites** : Affiche une liste des sites actuellement enregistrés dans le fichier `sites.ini`, avec les détails du nom d'utilisateur et du mot de passe.
- **Quitter** : Permet de quitter l'application.

10.2 Fonctionnement de l'interface

10.2.1 Téléchargement et mise à jour du fichier de configuration

Lors du lancement de l'application, une connexion SFTP est établie avec le serveur distant. Le fichier de configuration `sites.ini` est alors téléchargé depuis le serveur vers la machine locale. Ce fichier est utilisé pour les opérations de lecture et d'écriture des informations des sites.

10.2.2 Ajout d'un site

Pour ajouter un site, l'utilisateur clique sur le bouton **Ajouter un site**. Une nouvelle fenêtre s'ouvre, invitant l'utilisateur à saisir les informations suivantes :

- **Nom du site** : Le nom unique du site.
- **Nom d'utilisateur** : L'adresse e-mail du site.
- **Mot de passe** : Le mot de passe associé à l'adresse e-mail.

Après avoir rempli les champs et cliqué sur le bouton **Ajouter**, les informations sont validées et enregistrées dans le fichier `sites.ini`. Le fichier mis à jour est ensuite uploadé sur le serveur distant.

10.2.3 Suppression d'un site

Pour supprimer un site, l'utilisateur clique sur le bouton **Supprimer un site**. Une nouvelle fenêtre s'ouvre avec une liste déroulante contenant les noms des sites enregistrés. L'utilisateur sélectionne le site à supprimer et clique sur le bouton **Supprimer**. Le site est alors retiré du fichier `sites.ini` et le fichier mis à jour est uploadé sur le serveur distant.

10.2.4 Visualisation des sites

Pour visualiser les sites enregistrés, l'utilisateur clique sur le bouton **Visualiser les sites**. Une nouvelle fenêtre s'ouvre affichant les sites avec leur nom d'utilisateur et mot de passe.

10.2.5 Quitter l'interface

Lorsque l'utilisateur clique sur le bouton **Quitter**, l'application upload automatiquement le fichier de configuration modifié sur le serveur distant avant de fermer la connexion SFTP et de quitter l'application.

10.3 Exécution de l'interface

Pour exécuter l'interface, placez simplement le fichier `Gestion.py` sur votre ordinateur local. Vous pouvez ensuite double-cliquer sur le fichier Python dans le répertoire pour lancer l'interface. Assurez-vous que les modules Python nécessaires, notamment `paramiko` et `scp`, sont installés pour la gestion des connexions SFTP et le transfert de fichiers.

Pour installer les modules Python requis, suivez les étapes suivantes :

1. Installation de Python :

- Si Python n'est pas déjà installé sur votre système, téléchargez-le à partir du site officiel de Python : <https://www.python.org/downloads/>.

2. Installation des modules Python :

- Ouvrez un terminal (ou une invite de commande) sur votre système.
- Pour installer les modules nécessaires, utilisez `pip`, qui est le gestionnaire de paquets Python standard.
- Exécutez les commandes suivantes pour installer les modules requis :

```
pip install paramiko scp
```

Ces commandes téléchargeront et installeront les modules `paramiko` (utilisé pour les connexions SSH) et `scp` (utilisé pour le transfert de fichiers via SCP) ainsi que leurs dépendances.

11 Paramétrage serveur et base

11.1 Serveur virtuel distant

Pour paramétrer mon serveur virtuel distant et installer les dépendances nécessaires, j'ai utilisé les commandes suivantes :

```
sudo apt install python3-venv
pip3 install requests --break-system-packages
pip3 install sqlalchemy --break-system-packages
pip3 install mysql-connector-python --break-system-packages
pip3 install openpyxl --break-system-packages
pip3 install email_validator --break-system-packages
sudo apt-get update
sudo apt-get install p7zip-full --break-system-packages
pip3 install pandas --break-system-packages
pip3 install pymysql --break-system-packages
```

Ces commandes permettent d'installer et de configurer les outils et bibliothèques nécessaires pour le bon fonctionnement de l'application sur le serveur virtuel distant.

11.2 Base de données

Pour paramétrer et manipuler la base de données, j'ai effectué les étapes suivantes :

```
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

Cette commande ouvre le fichier de configuration de MySQL (mysqld.cnf) dans l'éditeur de texte `nano` pour le modifier.

```
# Modifier la ligne suivante pour permettre les connexions externes :
# bind-address = 127.0.0.1 (adresse locale uniquement)
# à
# bind-address = 0.0.0.0 (toutes les adresses IP)
```

Cette modification permet à MySQL d'accepter des connexions provenant de toutes les adresses IP, pas seulement de l'adresse locale. Cela est nécessaire pour permettre les connexions à la base de données depuis des machines distantes.

```
sudo systemctl restart mysql
```

Cette commande redémarre le service MySQL afin que les modifications dans le fichier de configuration prennent effet.

```
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

Cette commande rouvre le fichier de configuration de MySQL pour effectuer une autre modification.

```
# Modifier la ligne suivante pour changer le port de connexion :  
# port = 3306 (port par défaut de MySQL)  
# à  
# port = 35632 (nouveau port personnalisé)
```

Cette modification change le port sur lequel MySQL écoute pour les connexions. Le port par défaut est 3306, mais vous pouvez le changer pour un autre port (dans cet exemple, 35632) pour des raisons de sécurité ou de configuration réseau.

```
sudo systemctl restart mysql
```

Cette commande redémarre à nouveau le service MySQL afin que la modification du port soit prise en compte.

Pour créer les tables nécessaires dans ma base de données, j'ai utilisé le code SQL suivant :

```
-- Création de la table Granularite
```

```
CREATE TABLE Granularite (  
    Id_Granu INT NOT NULL,  
    Granularite_intitule VARCHAR(50),  
    PRIMARY KEY (Id_Granu)  
);
```

```
-- Création de la table Mesure avec une clé étrangère faisant référence à la table Gra
```

```
CREATE TABLE Mesure (  
    Id_boucle VARCHAR(50) NOT NULL,  
    PRM_prod VARCHAR(50),  
    PRM_conso VARCHAR(50),
```

```
    Attribut VARCHAR(30),
    Date_debut DATETIME,
    Date_fin DATETIME,
    Valeur DECIMAL(10, 2),
    Granularite INT,
    Respo VARCHAR(50),
    Classe VARCHAR(10),
    Etat VARCHAR(30),
    FOREIGN KEY (Granularite) REFERENCES Granularite(Id_Granu)
);

-- Création de la table Log pour enregistrer les journaux
CREATE TABLE Log (
    Id_boucle VARCHAR(50),
    Periode VARCHAR(25),
    DateHeure DATETIME,
    NomFichier VARCHAR(255),
    Description TEXT,
    Importance INT,
    Type VARCHAR(7)
);
```

Annexes

11.3 Connexion au serveur virtuel distant

Pour vous connecter au serveur virtuel distant `vps-602becce.vps.ovh.net`, voici comment procéder :

1. Ouvrir un terminal ou une invite de commandes.
2. Entrer les lignes suivantes :

```
ssh ubuntu@51.254.35.169  
password : AvMzBKeBkzCT
```

Vous arriverez à la racine du serveur virtuel distant. Pour se déplacer vers l'application : `cd CEpublicationEXPL`

OU

1. Ouvrir le logiciel FileZilla.
2. Remplir les champs suivants :

```
Hôte : 51.254.35.169  
Nom d'utilisateur : ubuntu  
Mot de passe : AvMzBKeBkzCT  
Port : 22
```

11.4 Connexion à la base de données

Pour vous connecter à la base de données `da18272-003.eu.clouddb.ovh.net`, voici comment procéder :

1. Ouvrir un terminal ou une invite de commandes.
2. Entrer les lignes suivantes :

```
mysql --host=da18272-003.eu.clouddb.ovh.net --user=sokoloff --port=35632  
--password CEpublicationEXPL
```

password : CEpubliEXPL59

OU

1. Ouvrir la page WEB **PhpMyAdmin**
2. Remplir les champs suivants :

Serveur : da18272-003.eu.clouddb.ovh.net:35632

Utilisateur : sokoloff

Mot de passe : CEpubliEXPL59

11.5 Connecter Power BI à la base MySQL

Pour vous connecter Power BI à la base de données, voici comment procéder :

1. Ouvrir Power BI
2. Insérer des données depuis MySQL
3. Remplir les champs suivants :

Serveur : da18272-003.eu.clouddb.ovh.net:35632

Nom de la base : CEpublicationEXPL

Nom d'utilisateur : sokoloff

Mot de passe : CEpubliEXPL59