Vincent Rapp
rapp.vincent.pro@gmail.com

MECALIST

Mode opératoire

V1.0

14/04/2023

1 Table des matières

Vincent Rapp
rapp.vincent.pro@gmail.com

MECALIST

Mode opératoire

V1.0
14/04/2023

1. Description

MECALIST est un programme écrit en *Python* fonctionnant sous *Windows*.
Microsoft Excel/Word sont nécessaires pour utiliser MECALIST.
MECALIST est un script qui exécute une succession de requêtes SQL définis par l'utilisateur, sur des données provenant de fichiers Excel afin d'en sortir des tables.
Le système de base de données utilisé est *SQLite*.

2. Arborescence

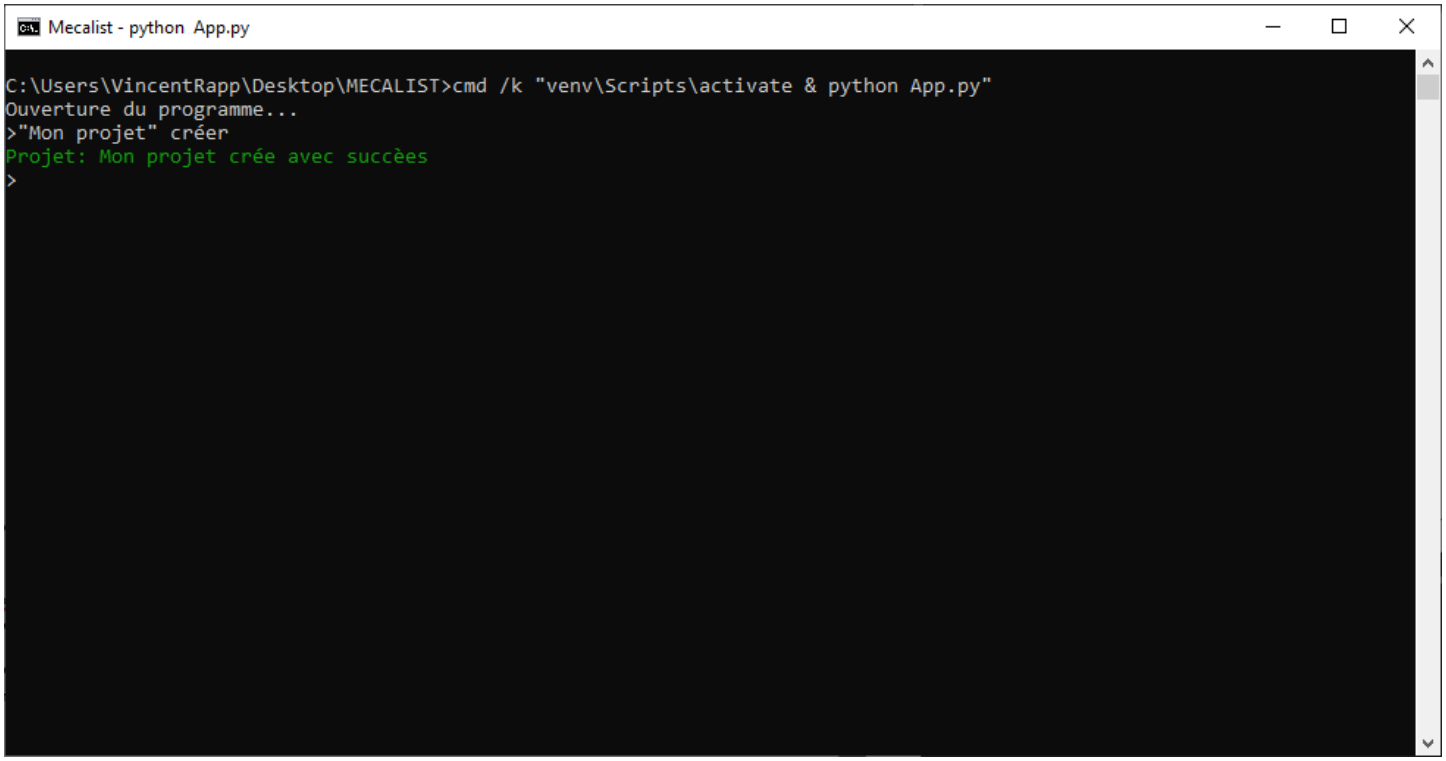
```
MECALIST
│   App.py
│   Run.cmd
│   Mecalist.lnk
│   pdf.ico
│   png.ico
│   docx.ico
│   xlsx.ico
│   zip.ico
│
├───Mon projet
│   │   batch.cmd
│   │   bdd.db
│   │
│   ├───0-conversion
│   │   │   fichiers convertis.xlsx
│   │   │
│   │   └───xlsx
│   │           config.json
│   │           fichiers à convertir.xlsx
│   │
│   ├───0-unification
│   │   │   fichiers unifiés.csv
│   │   │
│   │   └───csv
│   │           config.json
│   │           fichiers à unifier.csv
│   │
│   ├───1-import
│   │   ├───csv
│   │   │       config.json
│   │   │       fichiers à importer.csv
│   │   │
│   │   └───sql
│   │           1_traitement_après_import.sql
│   │
│   ├───2-analyse
│   │       1_mes_requettes.sql
│   │       calendrier.json
│   │
│   └───3-export
│       │   fichiers exportés.xlsx
│       │
│       ├───selection
│       │       1_selection_des_tables_à_exporter.sql
│       │       config.json
│       │
│       └───trames
│               config.json
│               fichiers trame.xlsx
│
└───venv
```

3. Commandes

Double cliquer sur Mecalist pour ouvrir le programme

7 commandes sont disponibles :

```
"Mon projet" créer
"Mon projet" convertir
"Mon projet" unifier
"Mon projet" effacer
"Mon projet" importer
"Mon projet" analyser
"Mon projet" exporter
```



3.1. Créer

Exemple : `"Mon projet" créer`

Crée un projet nommé *Mon projet*

3.2. Convertir

Exemple : `"Mon projet" convertir`

Converti en CSV les fichiers Excel présents dans `0-conversion\xlsx`
Les fichiers convertis seront déposés dans `0-conversion`
Pour convertir uniquement certains fichiers Excel, ajouter un argument texte à la commande.
Exemple : `"Mon projet" convertir "fic"`
Seuls les fichiers Excel contenant dans leur nom le texte "fic" seront convertis.

3.3. Unifier

Exemple : `"Mon projet" unifier`

Unifie les fichiers CSV présents dans `0-unification\csv`
Les fichiers unifiés seront déposés dans `0-unification`
Exemple 2 : `"Mon projet" unifier "fic"`
Seront unifiés uniquement les fichiers CSV contenant dans leur nom le texte "fic"

3.4. Effacer

Exemple : `"Mon projet" effacer`

Efface les tables de la base de données `bdd.db`

3.5. Importer

Exemple : `"Mon projet" importer`

Importe dans la base de données bdd.db les fichiers CSV présents dans `1-import\csv`
Exécute ensuite par ordre de leur nom les requêtes présentent dans `1-import\sql`
Pour importer uniquement certains fichiers CSV, ajouter un argument texte à la commande.
Exemple : `"Mon projet" importer "fic"`
Seuls les fichiers CSV contenant dans leur nom le texte "fic" seront importés.

3.6. Analyser

Exemple : `"Mon projet" analyser`

Lance par ordre de leur nom les requêtes SQL présentes dans le dossier `2-analyse`
Pour analyser uniquement certains fichiers SQL, ajouter un argument texte à la commande.
Exemple : `"Mon projet" analyser "fic"`
Seuls les fichiers SQL contenant dans leur nom le texte "fic" seront exécutés.

3.7. Exporter

Exemple : `"Mon projet" exporter`

Exporte en fichier les requêtes SQL présentes dans `3-export\selection`
Les fichiers exportés seront déposés dans `3-export`
Pour exporter uniquement certaines tables (alias), ajouter un argument texte à la commande.
Exemple : `"Mon projet" exporter "tab"`
Seuls les requêtes de tables ayant dans leur alias le texte "tab" seront exportés.

Nota Bene : possibilité de créer un dossier de projets, pour ce faire, déplacer le projet dans un dossier.
Prenons l'exemple de `"Mon projet"`, si nous le déplaçons dans un dossier nommé `"Dossier"`, dans ce cas la commande deviendra : `"Dossier\Mon projet" convertir`

4. Configuration

4.1. Les fichiers `config.json`

Le fichier `config.json` est présent dans les dossiers suivants :

- `0-conversion\xlsx`
- `0-unification\csv`
- `1-import\csv`
- `3-export\selection`
- `3-export\trames`

Il permet de définir le mode de lecture et d'écriture des fichiers d'import/export.
Il est écrit au format JSON

4.1.1. Conversion

Ci-dessous le format du fichier `0-conversion\xlsx\config.json`

Ce fichier permet de configurer la conversion des fichiers Excel en CSV.

```
{
    "fichier_1.xlsx(1)": {
        "onglet_1(2)": {
            "codage des caractères": "utf-8",
            "convertir cet onglet": "Oui",
            "liste des colonnes à convertir": [
                "*"
            ],
            "nommage du fichier exporté": "fichier_1_onglet_1",
            "saut de ligne": 0,
            "scinder par": "",
            "supprimer les colonnes sans nom": "Oui",
            "séparateur": ";"
        },
        "onglet_2": {
            "codage des caractères": "utf-8",
            "convertir cet onglet": "Oui",
            "liste des colonnes à convertir": [
                "*"
            ],
            "nommage du fichier exporté": "fichier_1_onglet_2",
            "saut de ligne": 0,
```

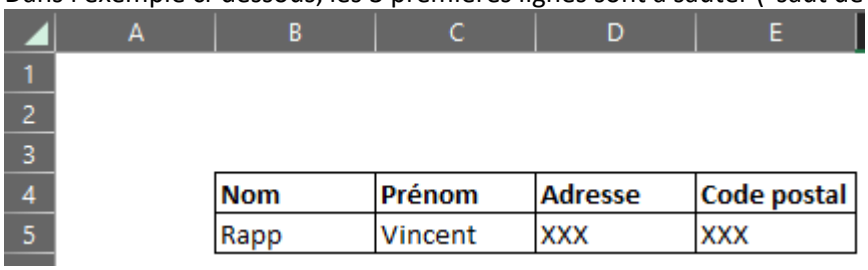
```
            "scinder par": "",
            "supprimer les colonnes sans nom": "Oui",
            "séparateur": ";"
        },
        "onglet_3": {
            "codage des caractères": "utf-8",
            "convertir cet onglet": "Oui",
```

[etc…]

```
}
```

Pour chaque onglet[2] de chaque fichier[1], définir les informations suivantes :

- `codage des caractères`
  - Codage des caractères du fichier CSV exporté à la suite de la conversion de l'onglet.
  - Choix possibles :
    - `"utf-8"`
    - `"ainsi"`
    - Autres codages vraisemblablement possibles mais non testés
- `convertir cet onglet`
  - Indique si le tableau présent dans l'onglet doit être converti
  - Choix possibles :
    - `"Oui"`
    - `"Non"`
- `liste des colonnes à convertir`
  - Liste des noms de colonnes du tableau présent dans l'onglet qui seront prises en compte par la conversion
  - Exemple 1 : `["Nom", "Prénom", "Adresse", "Code postal"]`
    - Exportera uniquement les 4 colonnes indiquées.
    - Si l'une des 4 colonnes est absente, le programme s'arrête.
    - Toutes les autres colonnes seront exclues de la conversion
    - Pour des raisons de bonne pratique, il est conseillé d'indiquer les colonnes. Cela permet de contrôler la bonne présence des colonnes.
  - Exemple 2 : `["*"]`
    - Exportera toutes les colonnes de la table présente dans l'onglet, sans vérification des colonnes.
- `nommage du fichier exporté`
  - Précise le nom du fichier qui résultera de l'exportation
- `saut de ligne`
  - Précise au programme le nombre de lignes à sauter dans l'onglet pour être aligné avec les colonnes du tableau.
    Dans l'exemple ci-dessous, les 3 premières lignes sont à sauter ("saut de ligne": 3).



Nom	Prénom	Adresse	Code postal
Rapp	Vincent	XXX	XXX

- `scinder par`
  - Permet d'exporter les données du tableau en le scindant par les valeurs présentes dans la colonne indiquée.
  - Exemple : `"Code postal"`
- `supprimer les colonnes sans nom`

  - Indique s'il le programme doit conserver les colonnes sans noms lors de la conversion
  - Choix possibles :
    - `"Oui"`
    - `"Non"`
- `séparateur`
  - Indique le séparateur à utiliser pour le CSV
  - Exemple : `";"`

4.1.2. Unification

Ci-dessous le format du fichier `0-unification\csv\config.json`

Ce fichier permet de configurer l'unification des fichiers CSV.
L'unification consiste à fusionner les CSV de même format en un seul fichier CSV.

```
{
    "fichier_1": {
        "séparateur": ";"
    },
    "fichier_2": {
        "séparateur": ";"
    },
    "fichier_3": {
        "séparateur": ";"
    }

[etc…]

}
```

Pour unifier les fichiers de même format, il faut qu'une partie de leur nom soit commun entres eux (*pattern*).

Exemple :
- `"Export_liste_2022-09-04"`
- `"Export_liste_2022-09-05"`
- `"Export_liste_2022-09-06"`

Ces 3 fichiers ont pour nom en commun le *pattern* : `"Export_liste"`

Pour chaque *pattern*, définir l'information suivante :

- `séparateur :`
  - Indique le séparateur utilisé dans le CSV
  - Exemple : `";"`

Le fichier unifié se verra ajouter une nouvelle colonne nommée `"source"` indiquant pour chaque ligne le nom du fichier dont les données proviennent.

4.1.3. Importation

Ci-dessous le format du fichier `1-import\csv\config.json`

Ce fichier permet de configurer l'import des fichiers CSV dans la base de données du projet (`bdd.db`).

```
{
    "fichier_1.csv(1)": {
        "liste des colonnes à importer": [
```

```
            "*"
        ],
        "nom de la table dans la bdd": "Table",
        "parser les colonnes date": "Oui",
        "séparateur": ";"
    },

    "fichier_2.csv(1)": {
        "liste des colonnes à importer": [
            "*"
        ],
        "nom de la table dans la bdd": "Table",
        "parser les colonnes date": "Oui",
        "séparateur": ";"
    }
```

[etc…]

}

Pour chaque fichier CSV(1), définir les informations suivantes :

- `liste des colonnes à importer`
  - Liste des noms de colonnes du tableau présent dans le fichier CSV(1) qui seront prises en compte par l'importation
  - Exemple 1 : `["Nom", "Prénom", "Adresse", "Code postal"]`
    - Importera uniquement les 4 colonnes indiquées.
    - Si l'une des 4 colonnes est absente, le programme s'arrête.
    - Toutes les autres colonnes seront exclues de la conversion
    - Pour des raisons de bonne pratique, il est conseillé d'indiquer les colonnes. Cela permet de contrôler la bonne présence des colonnes.
  - Exemple 2 : `["*"]`
    - Importera toutes les colonnes de la table présente dans l'onglet, sans vérification des colonnes.
- `nom de la table dans la bdd`
  - Indique le nom que prendra la table présente dans le fichier CSV dans la base de données du projet (`bdd.db`).
- `parser les colonnes date`
  - Indique au programme si les données des colonnes *date* doivent être enregistrées en base comme une date (DATETIME) ou comme du texte (TEXT).
  - Une colonne date est une colonne dont le nom contient le mot `"date"`
- `séparateur`
  - Indique le séparateur utilisé dans le fichier CSV à importer

4.1.4. Analyse

Pas de fichier `config.json` pour ce dossier.

4.1.5. Export

Deux fichiers `config.json` pour ce dossier.

Ci-dessous le format du premier fichier `3-export\selection\config.json`

Ce fichier permet de configurer l'exportation des requêtes de sélection (de type SELECT) présentent dans les fichiers SQL du même dossier.

```
{
    "alias_1[1]": {
        "exporter au format csv": "Non",
        "scinder par": ""
    },
    "alias_2": {
        "exporter au format csv": "Non",
        "scinder par": ""
    },
    "alias_3": {
        "exporter au format csv": "Non",
        "scinder par": ""
    },
    "alias_4": {
        "exporter au format csv": "Non",
        "scinder par": ""
    }
```

[etc…]

```
}
```

Pour chaque ALIAS[1] des requêtes de sélection (les requêtes de sélection doivent obligatoirement avoir un ALIAS : `SELECT <liste des colonnes> FROM <table> AS <nom de la trame>`), définir les informations suivantes :

- `exporter au format csv`
  - Indique s'il le programme doit exporter la table en CSV ou en Excel
  - Choix possibles :
    - `"Oui"`
      - Table exporter en CSV
    - `"Non"`
      - Table exporter en Excel
- `scinder par`
  - Permet d'exporter la table en la scindant par les valeurs présentes dans la colonne indiquée.
  - Exemple : `"Code postal"`

Ci-dessous le format du second fichier `3-export\trames\config.json`

Ce fichier permet de définir les informations de configuration des trames Excel d'exportation (s'il y en a).

```
{
    "trame_1.xlsx[1]": {
        "adresse de la cellule de départ où charger la table": "A1",
        "nom de l'onglet où charger la table": "table"
    },
    "trame_2.xlsx": {
        "adresse de la cellule de départ où charger la table": "A1",
        "nom de l'onglet où charger la table": "table"
    }
}
```

Pour chaque Trame Excel [1], définir les informations suivantes :

- `adresse de la cellule de départ où charger la table`
  - Indique la cellule Excel de départ à partir de laquelle la table écrite (avec la ligne des noms des colonnes)
  - Exemple :
    - `"A1"`
- `nom de l'onglet où charger la table`
  - Indique le nom de l'onglet du fichier Excel où sera écrite la table
  - Exemple :
    - `"Feuil1"`

4.2. Le fichier `calendrier.json`

Ce fichier permet de définir le mode de création de la table `calendrier`.
Il est écrit au format JSON.
Ci-dessous le format du fichier `2-analyse\calendrier.json`

```
{
    "calendrier": {
        "créer la table calendrier": "Oui",
        "date pivot": "SELECT DATE('now') AS date_pivot",
        "nombre de mois avant": 120,
        "nombre de mois après": 120
    }
}
```

Définition des informations :

- `créer la table calendrier`
  - Indique si la table `calendrier` doit être créé ou non
  - 2 choix possible :
    - `"Oui"`
    - `"Non"`
- `date pivot`
  - Indique la date pivot (date du milieu) de la table `calendrier`.
  - Sous forme de requête, la date pivot doit être l'unique champs de sélection avec pour alias `date_pivot`
  - Exemples :
    - `"SELECT DATE('now') AS date_pivot"`
    - `"SELECT DATE('now','-1 days') AS date_pivot"`
  - La requête peut faire référence à un champ `date` provenant d'une table importée
    - `"SELECT DATE(<champ>) AS date_pivot FROM <table>"`

Données de la table `calendrier`:

date	numjoursemaine	nomjoursemaine	quarter	trimestre	semestre	année	nummois	numjourmois	nommois	compteurjour	compteursemaine	lundi	dimanche
Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
2023-01-10 00:00:00	1	Mardi	1	1	1	2023	01	10	Janvier	-7	-1	2023-01-09 00:00:00	2023-01-15 00:00:00
2023-01-11 00:00:00	2	Mercredi	1	1	1	2023	01	11	Janvier	-6	-1	2023-01-09 00:00:00	2023-01-15 00:00:00
2023-01-12 00:00:00	3	Jeudi	1	1	1	2023	01	12	Janvier	-5	-1	2023-01-09 00:00:00	2023-01-15 00:00:00
2023-01-13 00:00:00	4	Vendredi	1	1	1	2023	01	13	Janvier	-4	-1	2023-01-09 00:00:00	2023-01-15 00:00:00
2023-01-14 00:00:00	5	Samedi	1	1	1	2023	01	14	Janvier	-3	-1	2023-01-09 00:00:00	2023-01-15 00:00:00
2023-01-15 00:00:00	6	Dimanche	1	1	1	2023	01	15	Janvier	-2	-1	2023-01-09 00:00:00	2023-01-15 00:00:00
2023-01-16 00:00:00	0	Lundi	1	1	1	2023	01	16	Janvier	-1	0	2023-01-16 00:00:00	2023-01-22 00:00:00
2023-01-17 00:00:00	1	Mardi	1	1	1	2023	01	17	Janvier	0	0	2023-01-16 00:00:00	2023-01-22 00:00:00
2023-01-18 00:00:00	2	Mercredi	1	1	1	2023	01	18	Janvier	1	0	2023-01-16 00:00:00	2023-01-22 00:00:00
2023-01-19 00:00:00	3	Jeudi	1	1	1	2023	01	19	Janvier	2	0	2023-01-16 00:00:00	2023-01-22 00:00:00
2023-01-20 00:00:00	4	Vendredi	1	1	1	2023	01	20	Janvier	3	0	2023-01-16 00:00:00	2023-01-22 00:00:00
2023-01-21 00:00:00	5	Samedi	1	1	1	2023	01	21	Janvier	4	0	2023-01-16 00:00:00	2023-01-22 00:00:00
2023-01-22 00:00:00	6	Dimanche	1	1	1	2023	01	22	Janvier	5	0	2023-01-16 00:00:00	2023-01-22 00:00:00
2023-01-23 00:00:00	0	Lundi	1	1	1	2023	01	23	Janvier	6	1	2023-01-23 00:00:00	2023-01-29 00:00:00
2023-01-24 00:00:00	1	Mardi	1	1	1	2023	01	24	Janvier	7	1	2023-01-23 00:00:00	2023-01-29 00:00:00
2023-01-25 00:00:00	2	Mercredi	1	1	1	2023	01	25	Janvier	8	1	2023-01-23 00:00:00	2023-01-29 00:00:00

date	Incrément date. Les champs suivants sont calculés en fonction de cette date.
numjoursemaine	Numéro du jour dans la semaine
nomjoursemaine	Nom du jour dans la semaine
quarter	Numéro du quartier de l'année
trimestre	Numéro du trimestre de l'année
semestre	Numéro du semestre de l'année
année	Année
nummois	Numéro du mois dans l'année
numjourmois	Numéro du jours dans le mois
nommois	Nom du mois
compteurjour	Compteur de jour à partir de la date pivot
compteursemaine	Compteur de semaine à partir de la semaine de la date pivot
lundi	Date du lundi de la semaine
dimanche	Date du dimanche de la semaine

5. Trames

5.1. Les trames Excel

Deux formats de trames sont possibles : Excel et Word
Les trames doivent être déposées dans le dossier `3-export\trames`.
Lors de l'export d'une requête de sélection, le programme vérifie dans le dossier `3-export\trames` si un fichier Excel ou Word existe avec pour nom l'alias de table de sélection.

Exemple Excel :

```
└──3-export
   │
   ├──selection
   │      1_selection_des_tables_à_exporter.sql
   │      config.json
   │
   └──trames
```

```
|               config.json
|               nom_de_la_trame.xlsx
```

```
1_selection_des_tables_à_exporter.sql :
      SELECT
            champ1,
            champ2
      FROM
            [nom_de_la_table]
      AS
            [nom_de_la_trame];
```

Le résultat de la requête sera écrit dans le fichier `nom_de_la_trame.xlsx` selon les modalités prévues dans le fichier `3-export\trames\config.json`.
Cela permet de réaliser un traitement Excel sur la table exportée.

5.2. Les trames Word

```
|    └──3-export
|        |
|        |
|        ├──selection
|        |        1_selection_des_tables_à_exporter.sql
|        |        config.json
|        |
|        └──trames
|                 config.json
|                 nom_de_la_trame.docx
```

Exemple 1 :

```
1_selection_des_tables_à_exporter.sql :
      SELECT
            variable1,
            variable2
      FROM
            [nom_de_la_table]
      AS
            [nom_de_la_trame];
```

Dans le cas d'une trame Word, le résultat de la requête de sélection ne doit contenir qu'une ligne.

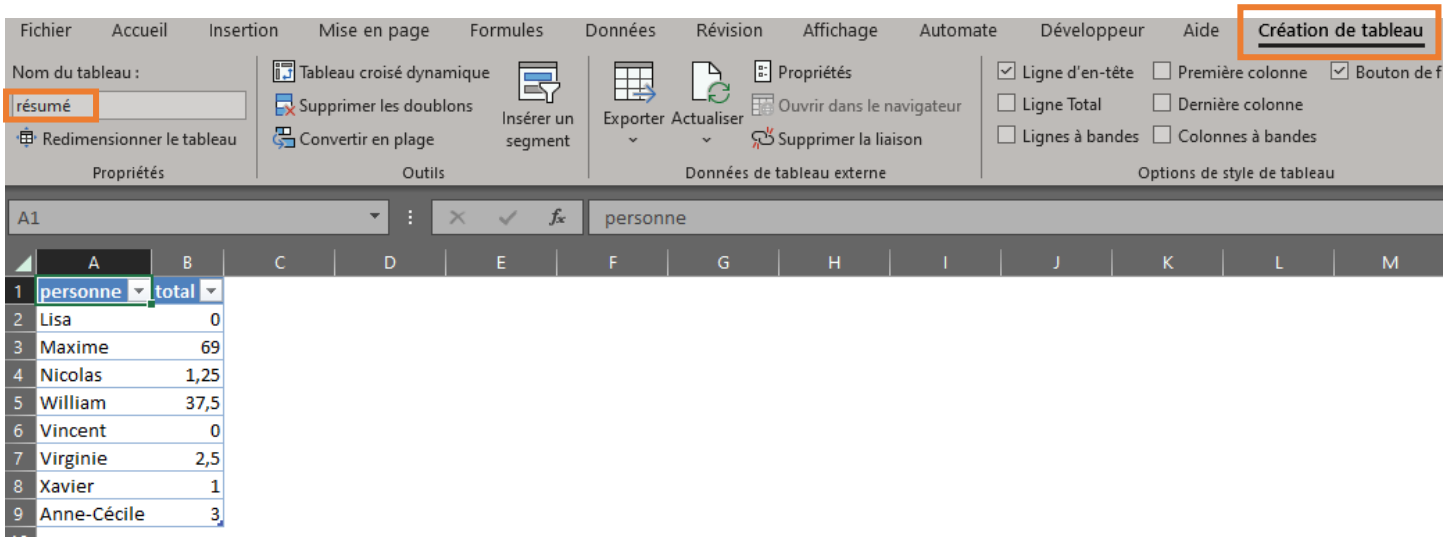
	variable1	variable2
1	Vincent	Rapp

Dans le document Word, les textes `{{champ1}}` et `{{champ2}}` seront respectivement remplacés par '`Vincent`' et '`Rapp`'.

Prénom : `{{variable1}}` Nom : `{{variable2}}`	*devient*	Prénom : `Vincent` Nom : `Rapp`

Exemple 2 :

Il est également possible d'insérer un tableau depuis un fichier Excel dans une trame Word mais pour ce faire celui-ci doit être nommé.



Ci-dessus le tableau `A1:B9` se nomme `résumé`.

Indiquer `<nom_du_fichier.xlsx>[<nom_du_tableau>]` pour faire référence au tableau nommé dans une variable
Le fichier Excel (`nom_du_fichier.xlsx`) doit se trouver dans le dossier `3-export`

```
1_selection_des_tables_à_exporter.sql :
      SELECT
            variable1,
            variable2,
            variable3
      FROM
            [nom_de_la_table]
      AS
            [nom_de_la_trame];
```

Résultat de la requête:

	variable1	variable2	variable3
1	Vincent	Rapp	nom_du_fichier.xlsx[résumé]

Ce qui donne en termes de remplacement de valeur dans la trame Word:

Prénom : `{{variable1}}` Nom : `{{variable2}}` Tableau : `{{variable3}}`	*devient*	Prénom : `Vincent` Nom : `Rapp` Tableau :

personne	total
Lisa	0
Maxime	69
Nicolas	1,25
William	37,5
Vincent	0
Virginie	2,5
Xavier	1
Anne-Cécile	3

Les trames Word ne doivent pas d'être configurer dans le fichier `3-export\trames\config.json`.
Par défaut les requêtes de variables liées à une trame Word ne seront pas exportées, sauf si l'export au format CSV est demandé (`"exporter au format csv": "Non"`)

Exemple 3 :

Il est également possible d'insérer un fichier dans une trame Word. Pour ce faire, ce fichier doit se trouver dans le dossier `3-export` avant la création du fichier Word.

```
1_selection_des_tables_à_exporter.sql :
      SELECT
            variable1,
            variable2,
            variable3,
            variable4
      FROM
            [nom_de_la_table]
      AS
            [nom_de_la_trame];
```

Résultat de la requête :

variable1	variable2	variable3	variable4
Vincent	Rapp	nom_du_fichier.xlsx[résumé]	fichier_compressé.zip

Ce qui donne en termes de remplacement de valeur dans la trame Word:

Prénom : `{{variable1}}`
Nom : `{{variable2}}`
Tableau :
`{{variable3}}`
Fichier :
`{{variable4}}`
	devient	Prénom : `Vincent`
		Nom : `Rapp`
		Tableau :

personne	total
Lisa	0
Maxime	69
Nicolas	1,25
William	37,5
Vincent	0
Virginie	2,5
Xavier	1
Anne-Cécile	3

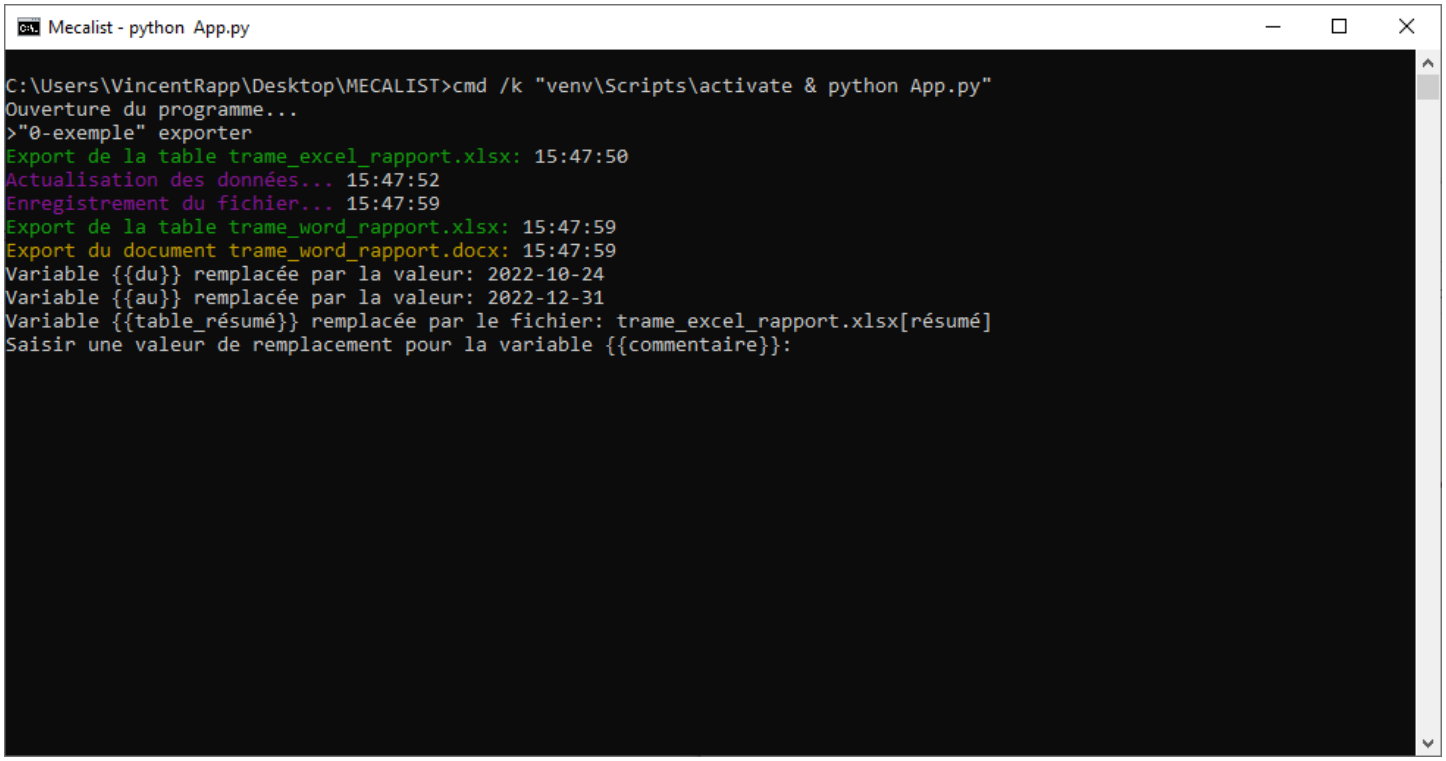
Fichier :

fichier_compressé.zip

Enfin, possibilité de faire appel à la saisie de l'utilisateur lors de l'exportation. Pour ce faire la variable doit être vide (mais non null), exemple :

```
1_selection_des_tables_à_exporter.sql :
      SELECT
            '2022-10-24' AS du,
            '2022-12-31' AS au,
            'trame_excel_rapport.xlsx[résumé]' AS table_résumé,
            '' AS commentaire
      FROM
            [nom_de_la_table]
      AS
            [nom_de_la_trame];
```

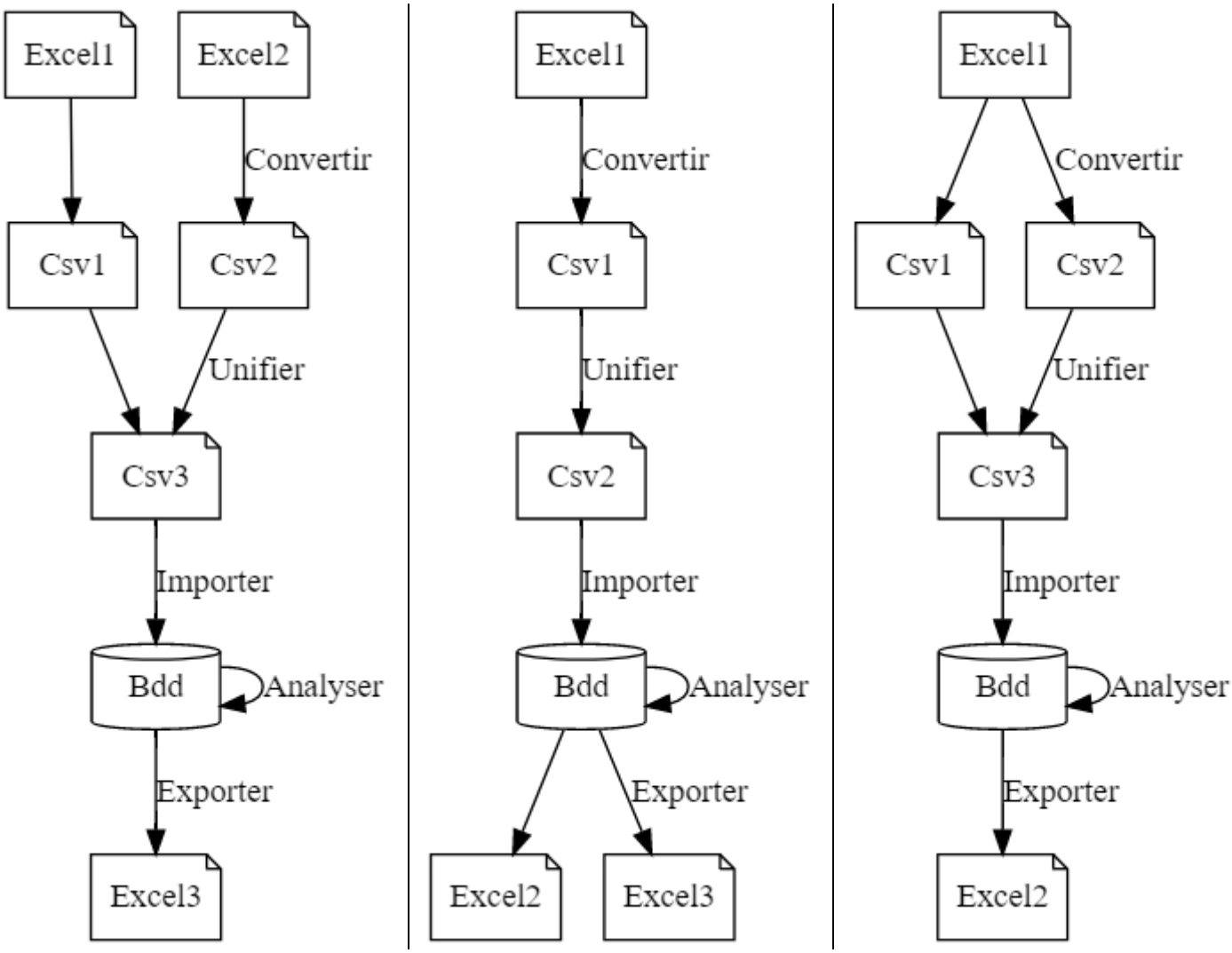
du	au	table_résumé	commentaire
2022-10-24	2022-12-31	trame_excel_rapport.xlsx[résumé]	



Ci-dessus l'interface en attente de la saisie de la variable {{commentaire}}.

6. Exemples de fonctionnement

Ci-dessous 3 exemples de processus de fonctionnement.



7. Utilisation manuelle du programme

Ci-dessous les étapes lors de la mise en place du projet.

1- Déposer les fichiers excel dans le dossier `xslx` de `0-convertion`
2- Configurer le fichier `config.json` du dossier `0-convertion`
3- Lancer la commande de conversion
4- Déplacer les fichiers csv résultant de la conversion dans le dossier `csv` de `0-unification`
5- Configurer le fichier `config.json` du dossier `0-unification`
6- Lancer la commande d'unification
7- Déplacer les fichiers csv résultant de l'unification dans le dossier `csv` de `1-import`
8- Configurer le fichier `config.json` du dossier `1-import`
9- Définir les requêtes sql d'importation
10- Lancer la commande d'import
11- Définir les requêtes d'analyse du dossier `sql` de `2-analyse`

12- Lancer la commande d'analyse
13- Définir les requêtes de sélection du dossier sélection de `3-export`
14- Configurer le fichier `config.json` du dossier trames (s'il y en a) de `3-export`
15- Lancer la commande d'exportation

=> les fichiers exportés seront créés dans le dossier `3-export`

Ci-dessous les étapes lorsque le projet est mis en place (config.json, requête sql, trames), les numéros d'étapes ont été conservés pour montrer les "sauts" par rapport aux étapes de mise en place :

1- Déposer les fichiers excel dans le dossier `xslx` de `0-convertion`
3- Lancer la commande de conversion
4- Déplacer les fichiers csv résultant de la conversion dans le dossier `csv` de `0-unification`
6- Lancer la commande d'unification
7- Déplacer les fichiers csv résultant de l'unification dans le dossier `csv` de `1-import`
10- Lancer la commande d'import
12- Lancer la commande d'analyse
15- Lancer la commande d'exportation

8. Utilisation en mode Batch (automation)

Une fois que le processus du projet est mis en place, utiliser le fichier `batch.cmd` pour lancer le processus complet. Les étapes 1, 3, 4, 6, 7, 10, 12, 15 seront alors exécutées à la suite.

Nota Bene : Si le projet se trouve dans un dossier (ex: "`Dossier\Mon projet`"), il faudra éditer le fichier `batch.cmd` comme suit.
Retirer le "`rem`" au niveau des lignes en vert en fonction du nombre de sous-dossier.

```
pause
@echo off
rem # -*- coding: cp1252 -*-
rem Ouvrir et enregistrer le fichier avec l'encodage Windows 1252
chcp 28591 > nul

rem dossier racine (par défaut - à laisser)
for %%I in (.) do set current_dir=%%~nxI
cd..

rem sous-dossier - repeter les 2 lignes ci-dessous autant de fois que de sous-dossiers
rem for %%I in (.) do set current_dir=%%~nxI\%current_dir%
rem cd..
rem for %%I in (.) do set current_dir=%%~nxI\%current_dir%
rem cd..

cmd /k "cd /d .\venv\Scripts & activate & cd /d ../../ & (echo "%current_dir%" convertir && echo q) | python App.py executer runserver & del ".\%current_dir%\0-unification\csv\*.csv" & move ".\%current_dir%\0-conversion\*.csv" ".\%current_dir%\0-unification\csv\"& (echo "%current_dir%" unifier && echo q) | python App.py executer runserver & del ".\%current_dir%\1-import\csv\*.csv" & move ".\%current_dir%\0-unification\*.csv" ".\%current_dir%\1-import\csv\" & (echo "%current_dir%" effacer && echo o && echo "%current_dir%" importer && echo "%current_dir%" analyser && echo "%current_dir%" exporter && echo q) | python App.py executer runserver & cd /d .\venv\Scripts & deactivate & cd /d ../../ & pause"
```

9. Exemple de projet

Le projet `0-exemple` a été créer dans le cadre de ce mode opératoire pour illustrer un cas concret de fonctionnement.