Outil de Génération de Jeux de Données

Introduction

L'outil de génération de jeux de données a été conçu pour faciliter et automatiser la création de données de test pour les environnements Cleva. Il se compose de deux composants principaux:

1. **Un programme en Python** qui génère un fichier CSV contenant les données.
2. **Une collection de requêtes API Postman** qui prend en entrée le fichier CSV généré par le programme Python et injecte les données dans l'environnement cible.

Vidéo de présentation : FSA_Fonctionnement_Jeu_de_données.mp4

Description des Composants

1 - Programme en Python

Le programme en Python a été développé pour générer des données structurées en fonction des besoins spécifiques des recetteurs. Les principales caractéristiques du programmes incluent :

Personnalisation des données :

Possibilité de configurer le volume de données à générer, les préfix à appliquer pour retrouver facilement ses données, les typologies de contrat, etc…

Génération de CSV :

Le programme crée un fichier CSV structuré qui contient les données générées et qui pourra ensuite être exploité par une collection Postman.

Il génère également un fichier au format XLSX épuré de plusieurs données techniques afin de faciliter sa lecture par un recetteur.

Facilité d'utilisation :

Le script Python peut soit être exécuté simplement depuis la ligne de commande ou bien être exécuté directement depuis un IDE.

2 - Collection de Requêtes API Postman

La collection de requêtes Postman est utilisée pour lire le fichier CSV généré par le programme Python et injecter les données dans l' environnement cible. Les points clés de ce composant sont :

Importation des données :

La collection est configurée pour lire le fichier CSV et mapper les champs aux paramètres des requêtes.

Automatisation de l'injection :

L'exécution des requêtes API Postman permet d'automatiser le processus d'insertion des données, réduisant ainsi le temps et les efforts manuels.

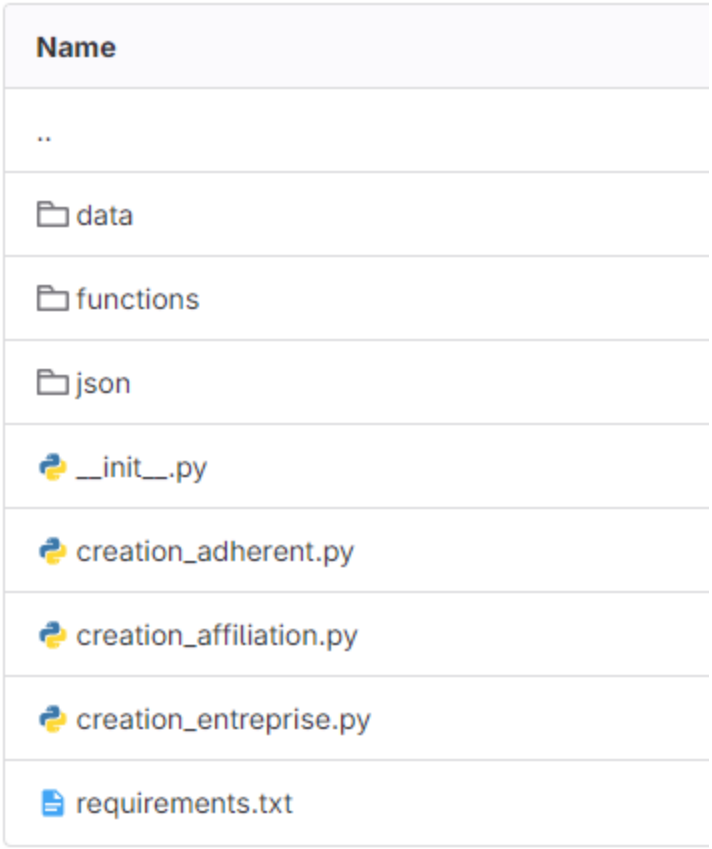
Configuration de l'environnement :

Les variables d'environnement de Postman peuvent être utilisées pour s'adapter à différents environnements cibles (alpha, beta-current, …).

Processus de Génération et d'Injection de Données

1. **Récupération du programme Python**

- Se rendre sur le projet Gitlab Cleva FSA QA dans la branche de la version Cleva cible pour récupérer le dossier 'generation_jdd' qui contient le programme Python.
- Dans l'arborescence des fichiers projet, le programme python est à cet emplacement : cleva/test/newman/generation_jdd
- Dossier generation_jdd du programme Python - Exemple :



1. **Installation des packages requis par le programme**

Installez les packages depuis le fichier requirements.txt :

```
pip install -r requirements.txt
```

1. **Configuration des données**

Ouvrir le fichier python creation_affiliation/adherent/entreprise.py à l'emplacement /generation_jdd puis modifier aux choix le nombre de données à générer, les préfixes de configuration, les typologies de contrat, dates, etc…

```
########################################
# CONFIGURATION

# Les contrats santé et prévoyance des entreprises sont définis par pair (exemple: SCSPS-ARCHI avec PCSPS-ARCHI)
# En l'absence de pair existante, seul le contrat santé ou prévoyance est défini

nombre_affiliation = 25 # Indiquer le nombre d'affiliation santé et prévoyance à générer, ce nombre est à appliquer pour chacun des prefix
prefix_list = ["APE", "CBO", "JMO", "AGA", "AFO", "PAC", "EFL", "RBO", "CMO"] # Indiquer une liste de prefix à appliquer aux noms des adhérents
contrat_sante_code = ["SCSPS-ARCHI", "SCSPS-IMMO", "SCSPS-PIMMO", "SCSPS-CARTO", "SCSPS-SXPY"] # Indiquer un code contrat santé de base
contrat_prev_code  = ["PCSPS-ARCHI", "PCSPS-IMMO", "PCSPS-PIMMO", "PCSPS-CARTO", ""] # Indiquer un code contrat prev de base
contrat_sante_extension = False # Indiquer True ou False pour ajouter ou non le contrat extension lié au contrat de base
contrat_prev_extension  = False # Indiquer True ou False pour ajouter ou non le contrat extension lié au contrat de base
contrat_sante_date = "" # Indiquer une date de début d'effet pour le contrat santé ("" par défaut pour une date réglementaire à - 2 ans)
contrat_prev_date  = "" # Indiquer une date de début d'effet pour le contrat prev ("" par défaut pour une date réglementaire à - 3 ans)
nom_fichier = "jdd_affiliations"

########################################
```

1. **Exécution du programme Python**

Lancer le script Python pour générer le fichier CSV avec les données nécessaires.

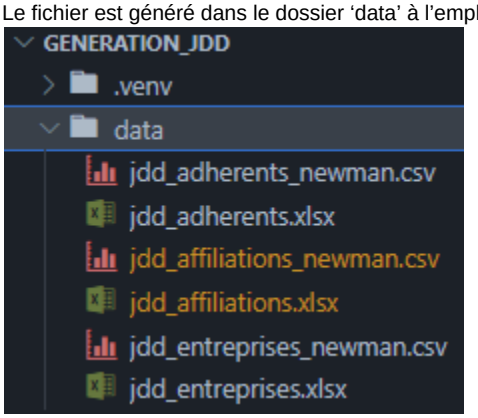
Via CLI :

```
python creation_affiliation.py
```

Via IDE et fichier python :

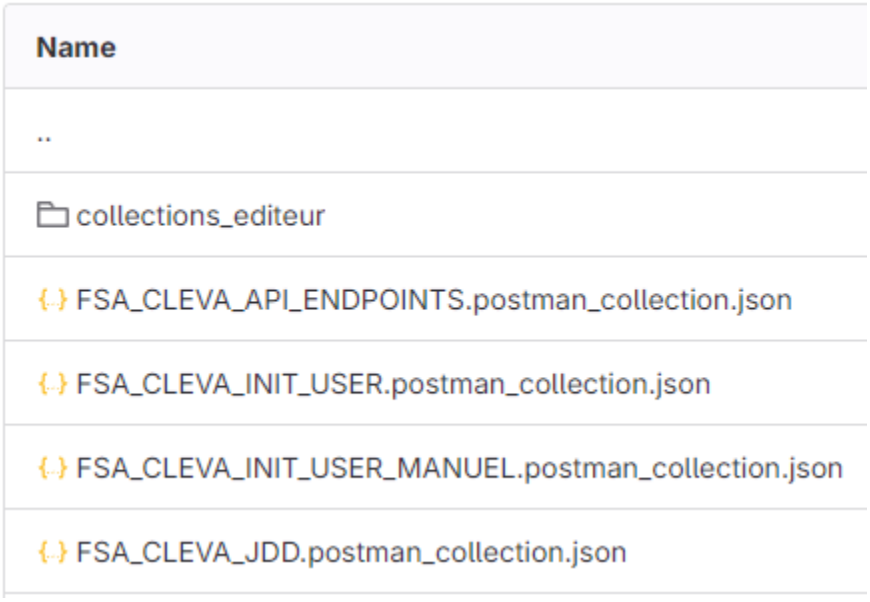
1. **Récupération du fichier CSV généré**

Le fichier est généré dans le dossier ‘data’ à l’emplacement /generation_jdd/data



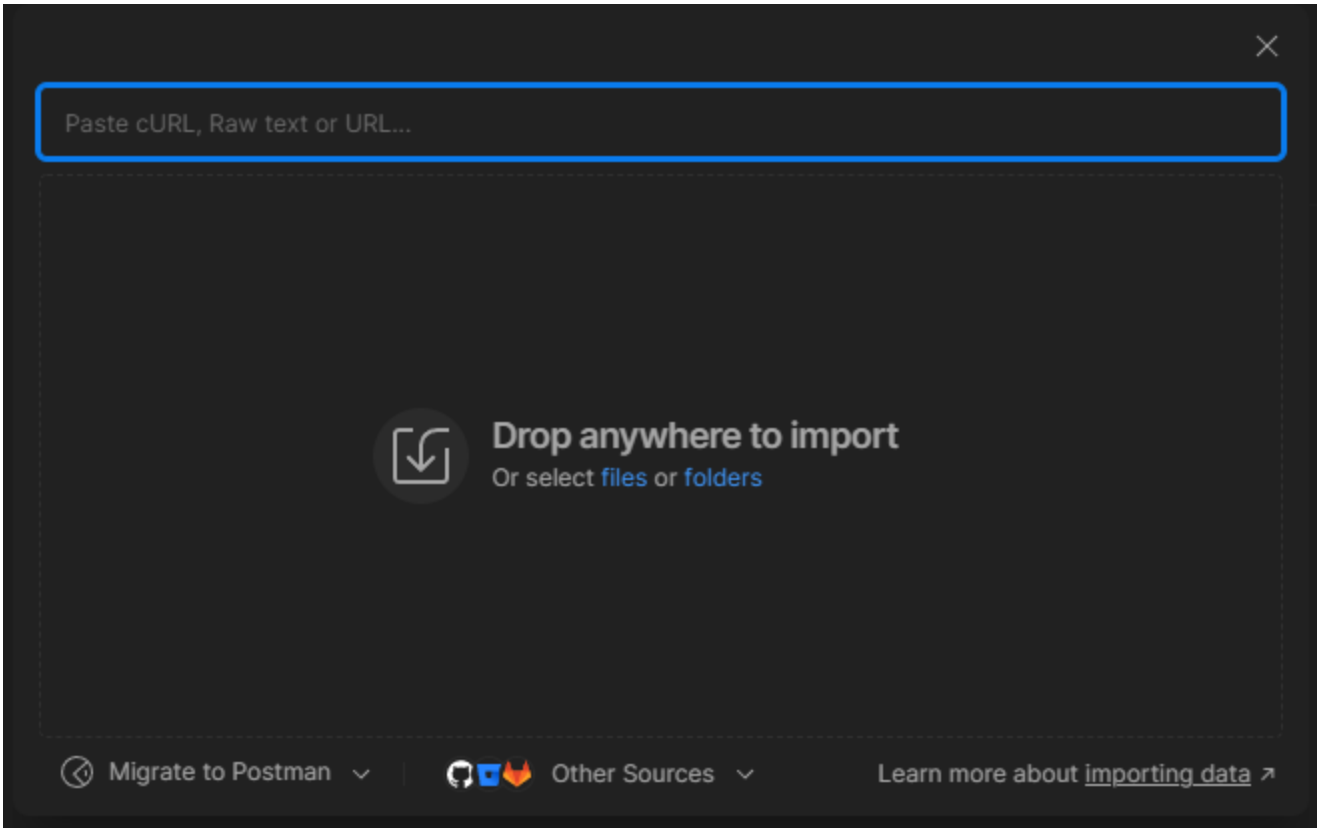
1. **Récupération de la collection API Postman**

- Se rendre sur le projet Gitlab Cleva FSA QA dans la branche de la version Cleva cible pour récupérer la collection.
- Dans l’arborescence des fichiers projet, la collection Postman est à cet emplacement : cleva/test/newman/collections
- Récupérer la collection nommée ‘FSA_CLEVA_JDD.postman_collection.json’ - Exemple :



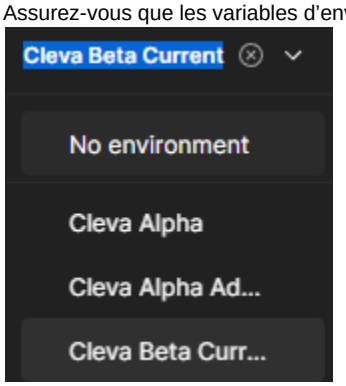
1. **Préparation de Postman**

Ouvrez Postman et importez la collection de requêtes.



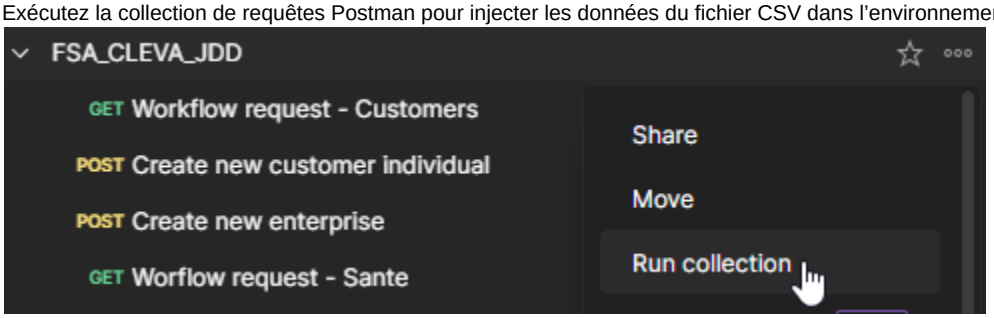
1. **Configuration des variables d'environnement**

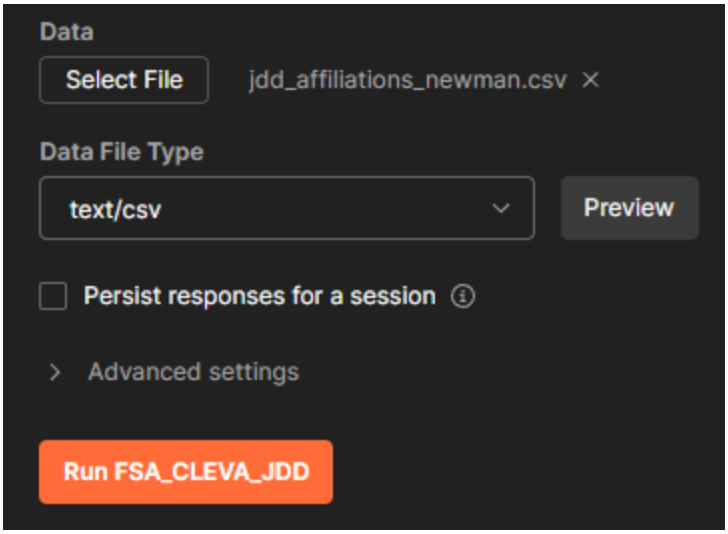
Assurez-vous que les variables d'environnement sont correctement définies pour le contexte cible.



1. **Injection des données depuis CSV**

Exécutez la collection de requêtes Postman pour injecter les données du fichier CSV dans l'environnement.





La collection peut également être exécutée via Newman en local ou dans le cadre d'un pipeline.

```
newman run FSA_CLEVA_JDD.postman_collection.json -e beta_current.
postman_environment.json -k --iteration-data jdd_auto_adherents_newman.
csv --reporters=cli,htmlextra
```