Exam - Customers & Purchases

Python Software Engineer

Contexte

Nous souhaitons développer un système pour gérer les paiements en vol. Les passagers pourront acheter des produits duty-free ou des articles de restauration (snacks, boissons, etc.) pendant le vol. Cependant, l'avion ne disposant pas toujours d'une connexion internet continue, les données clients et achats seront enregistrées localement dans des fichiers CSV, puis synchronisées avec un serveur au sol après l'atterrissage via une connexion 4G.

Votre mission consiste à construire un service FastAPI qui :

1. Créer un endpoint d'API qui permet d'importer des données CSV, à partir de deux fichiers présents sur le disque local, et de les importer dans une base de données SQLite
2. Créer un endpoint d'API qui permet de déclencher l'envoi de l'ensemble des données à une autre API

Vous devrez :

- Gérer les interruptions réseaux avec une stratégie de résilience pour l'envoi à l'API externe

Exigences

1. Import des CSV dans une base de données

- Lire les fichiers `customers.csv` et `purchases.csv` et les importer dans une base SQLite avec deux tables adaptées aux données fournies dans les fichiers CSV.
- Vous êtes libre de définir les schémas des tables en fonction des besoins du test.

2. API RESTful

Exposez des endpoints RESTful via FastAPI :

- **`POST /import-csv`** :
  - Permet d'importer les fichiers CSV et de les insérer dans la base SQLite.
  - Valide le format des fichiers CSV (par exemple, colonnes obligatoires).
  - L'API prend en argument les chemins absolus des deux fichiers sur le disque en local (là où se trouve le backend de l'API).

```
curl --location 'http://localhost/api/import-csv' \
--header 'Content-Type: application/json' \
```

```
--data '{
   "customers_file_path": "/opt/custexport/customers.csv",
   "purchased_file_path": "/opt/custexport/purchases.csv"
}'
```

- **`POST /send-customers`** :
    - Récupère les données des tables `customers` et `purchases`.
    - Envoie les données au format JSON à une API externe (mockée).

3. Gestion des interruptions réseau (résilience)

- Implémentez une **stratégie de résilience** en cas d'échec lors de l'envoi à l'API externe

Ressources

Fichiers CSV fournis

API externe (Mock) :

Utilisez une URL fictive comme `https://mockapi.example.com/v1/customers`.

Exemple de JSON attendu :

```
[
      {
            "salutation": "xxx",
            "last_name": "xxxx",
            "first_name": "xxxx",
            "email": "xxxxx",
            "purchases": [
                  {
                        "product_id": "xxxx",
                        “price”: 10,
                        “currency”: “dollars”,
                        “quantity”: 1,
                        "purchased_at": "YYYY-MM-DD"
                  },
                  {
                        "product_id": "xxxx",
                        “price”: 10,
                        “currency”: “dollars”
                        “quantity”: 1,
                        "purchased_at": "YYYY-MM-DD"
```

```
                    }
                ]
        },
        {
                (customer 2…)
        },
        (...)
]
```

Livrables attendus

1. Code source :
    - Service FastAPI avec endpoints `/import-csv` et `/send-customers`.
    - Gestion de la résilience réseau.
    - Fichiers CSV d'exemple et script d'initialisation SQLite.
2. Documentation :
    - Un fichier `README.md` contenant :
        - Les instructions pour exécuter l'application et les tests.
        - Des exemples de requêtes API avec `curl` ou équivalent.

Des tests automatisés seront valorisés si présents.

Note importante

Ce test est conçu pour être réalisable en **4 heures maximum**, en se concentrant sur les fonctionnalités essentielles. Assurez-vous d'adopter une approche pragmatique et itérative pour prioriser les livrables.