CONSTRUCTION D'UNE APPLICATION STRUCTUREE

- La première étape consiste à créer le SparkSessionobjet, afin d'utiliser Spark.

```
>>> from pyspark.sql import SparkSession
>>> spark=SparkSession.builder.appName('structured_streaming').getOrCreate()
22/11/03 10:53:52 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
>>> import pyspark.sql.functions as F
>>> from pyspark.sql.types import *
```

- Ensuite, nous créons des données auto-générées qui peuvent être poussées dans un répertoire local respectif ("csv_folder"),être lu par le Streaming Structuré. Les données que nous allons générer contiennent quatre colonnes et sont au format CSV.Nous pouvons également générer un format Parquet, si nécessaire.
  1. ID utilisateur
  2. Application
  3. Temps passé (secs)
  4. Âge

```
>>> df_1=spark.createDataFrame([("XN203",'FB',300,30),("XN201",'Twitter',10,19),("XN202",'Insta',500,45)],["user_id","app","time_in_secs","age"]).write
.csv("csv_folder",mode='append')
>>>
```

- Une fois que nous avons créé ces dataframes, nous pouvons définir le schéma de ces fichiers, afin de les lire en utilisant le traitement de flux.

```
>>> schema=StructType().add("user_id","string").add("app","string").add("time_in
_secs", "integer").add("age","integer")
>>>
```

- lecture du fichier disponible dans le dossier local comme une trame de données de flux.
  L'API pour lire une trame de données statique est similaire à celle pour lire une trame de en continu, la seule différence étant que nous utilisons readStream.

```
>>> data=spark.readStream.option("sep", ",").schema(schema).csv("csv_folder")
>>>
```

- validation du schéma de la dataframe avec printSchema()

```
>>> data.printSchema()
root
 |-- user_id: string (nullable = true)
 |-- app: string (nullable = true)
 |-- time_in_secs: integer (nullable = true)
 |-- age: integer (nullable = true)

>>>
```