Écrire une application web python qui sera appelée par des clients via une API (Fast API). L'application doit exposer une seule fonction d'entrée. Cette fonction d'entrée jouera le rôle de dispatcher : elle recevra le nom de la logique applicative à exécuter ainsi qu'une charge utile au format JSON. En fonction du nom de la fonction demandée, elle redirigera l'appel vers la fonction interne appropriée. Si la fonction appelée ne nécessite pas d'arguments, le JSON d'entrée pourra être vide. Une fois la fonction exécutée, elle devra retourner une réponse au format JSON qui sera transmise au client via l'API.

Le projet a pour objectif de gérer les objetcs/classes/éléments/entités suivants :

1. **Gestionnaire** (id, nom, code, tel, contactPrincipal, email ...): Un gestionnaire peut gérer un ou plusieurs fonds.
2. Region1(id, code, nom) : Une region contient un ou plusieurs pays
3. Pays (id, code, nom, idRegion, continent, idDevise) : Pays a une seule devise (clé etrangerer de la table Devise : idDevise) et fait partie d’une region (clé etrangere de la table Region : idpays)
4. Devise (id, code, nom, idPays) : devise par pays
5. Secteur(id, codeGics, codeBics, nom, …)
6. SousTypeActif1(id, idTypeAcif1, nom,…)
7. TypeActif1(id, type, …) : un typeActif1 contient zero ou plusieurs sousTypeActif1 (clé etrangere de la table SousActif1 : idTypeActif1)
8. SousClassif1(id, idClassif1, nom,…)
9. Classif1(id, nom, …) : un Classif1 contient zero ou plusieurs sousClassif1 (clé etrangere de la table SousClassif1: idClassif1)
10. **Titre** (id, nom, cusip, isin, ticker, emetteur, idTypeTitre1, idSousTypeTitre1, idTypeTitre2, idSecteur, idClassification1, idSousClassification1, classification2, idNotation, idPays, …) : un Titre a des liens avec des cles etrangere avec les tables : Region1, Pays, secteur, TypeActif1, SousTypeActif1, SousClassif1, Classif1…
11. **Indice** (id, nom)
12. **Fonds** (id, code, nom, ...): Un fonds peut être géré par un ou plusieurs gestionnaires. Chaque fonds a un ou plusieurs Indice. Un fonds peut être de deux types :
    * **FondsSimple : contient que des Titres**
    * **Portefeuille :** Un portefeuille contient des **Titres** et d'autres **FondsSimple**.

Les relations entre ces objets :

1. **Composition de fonds par Gestionnaire** : pour chaque Date, chaque fonds\_gestionnaire est composé de plusieurs Titre. Table composition\_fonds\_gestionnaire : date, id fonds, id gestionnaire, id Titre, id devise, id pays, quantité, prix, valeur marchande, accrued, dividende)
2. **Composition de portefeuille par Gestionnaire** : (Cette table est similaire à la table « **Composition de fonds par Gestionnaire** », mais il se differe de seulement qu’elle contient que les fonds de type portefeuille et non pas des fonds simple). pour chaque Date, chaque portfeuille\_gestionnaire est composé de plusieurs Titre. Table composition\_portfeuille\_gestionnaire : date, id fonds, id gestionnaire, id Titre, id devise, id pays, quantité, prix, valeur marchande, accrued, dividende)
3. Composition indice: pour chaque date, Un indice est composé de plusieurs titres avec des quantités, des prix, valeur marchande, dividende.

Tous les objets (Fonds, Indice, Titre, Gestionnaire, secteur, pays…) seront enregistrés dans une base de données SQL Server, et dans une base de données SQLIte local pour test rapide.

L’application devrait s’organiser avec des répertoires comme suit :

1. fichier main.py : contient l’appel de l’API
2. fichier dispatcher.py : qui reçois l’appel de l’api du main et dispatche les taches en fonction de la fonction demandé et les données input et recevoir les output pour les envoyer au client à travers l’api
3. Un fichier config.env : contient les variables de configuration comme nom de base de données, serveur, mot de passe, et le chemin des fichiers log, les information de SFTP…, le type d’environnement de travail : développement ou déploiement
4. Un dossier « database » : contient les fichiers suivants :
   1. Fichier connexionsqlServer.py : de la classe connexion qui contient l’instantiation des objets connexion, gere les acces à la base de données sql server, et declare les engine de connexion, ouvre les connexion, …
   2. Fichier connexionsqlLiter.py : de la classe connexion qui contient l’instantiation des objets connexion, gere les acces à la base de données sqlite, et declare les engine de connexion, ouvre les connexion, …
   3. Fichier sqlServerCreation.sql : contient les requetes de creation des tables dans la base de données sqlserver
   4. Fichier sqliteCreation.sql : contient les requetes de creation des tables dans la base de données sqlite
5. Dissier schémas : contient le model des objects crees et leurs relations, comme par exemple :
   1. Fonds.py : le model de l’objet fonds
   2. Gestionnaires.py : le modele
   3. Et toutes les autre objets et la relations entre eux
6. Dossier sqlserverOperation qui contient les Fichiers qui geres les operations CRUD, de lectures, écritures dans les tables sqlserver
7. Dossier sqliterOperation qui contient les Fichiers qui geres les operations CRUD, de lectures, écritures dans les tables sqlite
8. Un dossier constantes : contient les fichiers suivants :
   1. Un fichier const1.py : contient les constantes qui vont être visible dans le code entier, comme chaine de connexion dont les variables extraits de fichier config.py
   2. Des constantes hardcodés et qui vont être organisés par type d’utilisations et par quel classe qui les appel et les utilise
   3. Des constantes qui vont être extraites à partir de la base données, comme les noms de colonnes des tables, et les noms de colonnes des fichiers csv, organisés par les noms de fichiers qui les utilises (par classes ou autre organisation optimale)
9. Un dossier STFP : contient les fichiers suivants :
   1. Fichier sftp.py : contient la classe qui gere le sftp, acces au repertoire distant, avec les informations extraites de fichier de config.env
   2. Autre fichiers qui gèrent les operations en liens avec le sftp :
      1. Lister les fichiers
      2. importation les fichiers de repertoire distant vers le repertoire local (toute, ou un nom specifique, ou une partie de leur nom..)
      3. Decompressions des fichiers .zip
10. Un dossier utils : contient les fichiers qui vont être par l’ensemble de projets, et qui sont organisé par leurs fonctions :
    * 1. Fichier csv.py : organisé par classes ou par fonctions partagés et contient toutes les fonctions suivantes :
         1. Chargement de fichiers csv vers la mémoire en dictionnaire, en dataframe
         2. Chargement des fichiers csv à la base de données
         3. Écrire des dictionnaires dans des fichier csv
         4. Extraire des tables sql vers fichier csv
         5. Écrire des dataframe dans des fichier csv
      2. Fichier excel.py : même logique de fichier csv.py, en plus d’autres fonctions comme :
         1. Écrire des fonctions python directement dans des cellules excel specifique
         2. Écrire dans des fichier excel avec des emplacement specifique
         3. Dans des feuilles spécifiques
         4. Vider des feuilles ou des plages specifique dans des classeurs excel
      3. Fichier pdf.py : contient les fonctionnalité suivantes :
         1. Générer des fichiers pdf à partir des csv
         2. Générer des fichiers pdf a partir des feuilles d’Excel
         3. Lire des fichiers pdf dans des listes de lignes de strings
         4. Exporter des dataframe avec des transformation en icones vers des pdf
      4. Fichier io.py : contient les fonctions suivantes :
         1. Lires des fichiers text
         2. Écrire dans des fichers text
         3. Compresser et decompresser des fichiers
      5. Fichier data.py : contient les fonctions qui gere les données :
         1. Charger dataframe à une table sql existante et nouvelle
         2. Charger dictionnaire de donnes à une table sql existnte ou nouvelle
         3. Charger dataframe a partir de table sql
         4. Charger dictionnaire de données, ou liste de dictionnaire de données a partir de table sql
11. Un dossier logic : qui contient des fichiers qui gere les calculs et la logique d’affaire, par exemple :
    1. Fichier qui calcul la valeur marchande (somme des prix \* quantité) d’un fonds à une date donnée
    2. Calcul la valeur marchande par type d’actif, par secteur, par pays, par fonds par pays, par fond par gestionnaires par pays,
    3. Calcul le poids (valeur marchande / valeur marchande total) de chaque titre par fonds, par type d’Actif par fonds, par type d’Actif par gestionnaire…
    4. Calcul des regroupement par object ou par relation d’object, et le mettre dans des dictionnaire ou dans des dataframe
    5. Exporter ces dataframe / dictionnaire dans pdf ou des excel ou dans des table sql

L'application doit être modulaire, extensible et facilement évolutive pour intégrer de nouvelles logiques à l'avenir. Les commentaires des fonctions, des classes doivenent fait en francais.