FICHE DE SYNTHÈSE

SAE 1.01 - Implémentation d'un besoin client

COUDIN Ulrich

COQUERELLE Mélissa

LAMARQUE Noé

MOLINIER Hugo

IUT de Créteil-Vitry

BUT Informatique : Groupe B

23/01/2023

Logiciel ADAV (Analyse de Données à Vitry-sur-Seine) :

- **Dossier 'src' composé de 9 modules différents :**
  - **web_scraping.py**
  - **lectureCSV.py**
  - **visualisation.py**
  - **graphs.py**
  - **app.py**
  - **run.py**
  - **pdfFiles.py**
  - **web.py**
  - **main.py**

web_scraping.py :

Description :

Ce module utilise les bibliothèques Python "requests" et "BeautifulSoup" pour établir des requêtes HTTP et extraire des données à partir de pages web.

Il utilise trois URLs différentes pour récupérer les données de polluants atmosphériques en 2021, 2022 et 2023, en utilisant la bibliothèque "requests" pour récupérer le contenu de chaque page et la bibliothèque "BeautifulSoup" pour parser et manipuler ces données. Il stocke ces informations dans des variables soup1, soup2 et soup3.

Ce module ne possède aucune fonction crée par nous-même mais uniquement des variables et des fonctions pré-définies.

lectureCSV.py :

Description :

Ce module utilise les bibliothèques Python "pandas" et "**web_scraping**" pour lire et extraire des données à partir de fichiers CSV. Il définit des fonctions pour trouver tous les liens vers des fichiers CSV sur la base de données de l'AASDAQ, lire le dernier fichier CSV d'une année donnée, et lire un fichier CSV spécifique en fonction d'une date donnée. La fonction "**find_allCSV**" prend en entrée une page web et renvoie une liste de liens vers des fichiers CSV de cette page. La fonction "**readLastCSV**" utilise la fonction précédente pour trouver tous les liens vers des fichiers CSV sur une page web donnée, puis lit le dernier fichier CSV de la liste. La fonction "**readDateCSV**" prend en entrée une date au format AAAA-MM-JJ et utilise les fonctions précédentes pour trouver et lire le fichier CSV correspondant à cette date.

Nom de la fonction : **find_allCSV**

Paramètres : site - objet de type BeautifulSoup correspondant à une page web parsée.

Rôle : Trouve tous les liens vers des fichiers .csv présents sur la page et les range dans une liste.

Valeur de retours : list[string] - liste contenant tous les liens (chainés) vers des fichiers .csv trouvés sur la page

Nom de la fonction : **findDate**

Paramètres : date,liste - objets de type string (AAAA-MM-JJ) et list[string]

Rôle : Trouve la date passée en paramètre dans la liste des fichiers .csv et retourne le DataFrame correspondant au .csv trouvé

Valeur de retours : DataFrame - Conversion du fichier .csv de la liste correspondant à la date passée en paramètre puis lecture du fichier en DataFrame.

Nom de la fonction : **readDateCSV**

Paramètres : date - objet de type string (AAAA-MM-JJ)

Préconditions : La fonction findDate doit exister

Rôle : Renvoie le DataFrame correspondant à la date passée en paramètre.

Valeur de retours : DataFrame - Conversion du fichier .csv de la liste correspondant à la date passé en paramètre puis lecture du fichier en DataFrame.

visualisation.py :

Description :

Ce module utilise les bibliothèques Python "**lectureCSV**" pour manipuler des données à partir de fichiers CSV. Il définit des fonctions pour récupérer des données spécifiques à partir d'un DataFrame, telles que les données correspondant à la ville de Vitry-sur-Seine, les mesures brutes de polluants spécifiques, les dates de mesures correspondant à ces mesures et créer un nouveau DataFrame avec ces données. Il utilise aussi des fonctions pour créer des DataFrame détaillés avec des moyennes, des minimas, des maximas, des écarts types et des quantiles. Il utilise la fonction "**donneesAVitry**" pour récupérer les lignes d'un DataFrame correspondant à la ville de "Vitry-sur-Seine", la fonction "**mesuresBrutesPolluants**" pour récupérer les mesures brutes des polluants spécifiques à cette ville, la fonction "**DateDebutMesure**" pour récupérer les dates de mesure correspondant à ces mesures et la fonction "**CreateDataframe**" pour créer un nouveau DataFrame avec les données récupérées.

Nom de la fonction : **donneesAVitry**

Paramètres : data,cle - objets de type DataFrame et string

Rôle : Récupère et retourne un tableau comportant les indices de chaque ligne correspondante à la ville de Vitry-sur-Seine dans un DataFrame.

Valeur de retours : list[int] - liste d'entiers correspondant aux indices parcourus et répondant à la condition dans rôle.

Nom de la fonction : **mesuresBrutesPolluants**

Paramètres : data,cle,polluant - objets de type DataFrame, string et string.

Préconditions : La fonction donneesAVitry(data,city_key) doit exister

Rôle : Récupère et retourne la liste des mesures brutes du polluant passé en paramètre à Vitry-sur-Seine.

Valeur de retours : list[float] - liste correspondant aux mesures brutes du polluant passé en paramètre.

Nom de la fonction : **DateDebutMesure**

Paramètres : data,cle,polluant - objets de type DataFrame, string, string

Préconditions : La fonction donneesAVitry(data,city_key) doit exister

Rôle : Récupère et retourne la liste des dates de mesures du polluant passé en paramètre à Vitry-sur-Seine

Valeur de retours : list[str] - liste correspondant aux dates de mesures du polluant passé en paramètre.

Nom de la fonction : **CreateDataframe**

Paramètres : resultat- objet de type DataFrame

Préconditions : Les fonctions DateDebutMesure et mesuresBrutesPolluants doivent exister

Rôle : Renvoie un nouveau DataFrame représentant les mesures brutes des polluants avec leurs dates correspondantes

Valeur de retours : DataFrame - DataFrame décrit dans Rôle.

Nom de la fonction : **CreateDescribeData**

Paramètres : dataframe - objet de type DataFrame

Rôle : Renvoie un DataFrame détaillé avec minimum, maximum, écart-type et moyenne de données de chaques polluants.

Valeur de retours : DataFrame - DataFrame décrit dans Rôle.

graphs.py :

Description :

Ce module utilise la bibliothèque matplotlib pour créer et visualiser des graphiques à partir de données stockées dans un DataFrame. Il utilise également la bibliothèque numpy pour effectuer des calculs sur les données. Il importe également un module appelé "visualisation" contenant les fonctions liées à la visualisation des données.

Il y a la fonction « **tracerGraphique** » qui prend en entrée une date et un DataFrame de résultats, qui utilise les données pour créer un graphique linéaire représentant les mesures de différents polluants à Vitry-sur-Seine. Il y a également la fonction **indice_mark** pour marquer les seuils de dépassement sur le graphique.

Nom de la fonction : **indice_mark**

Paramètres : dataframe,donnees,name_indices - objets de type DataFrame,numpy,string

Rôle : Marque les seuils de dépassement sur le graphique (plot).

Valeur de retours : None - La fonction ne renvoie rien.

Nom de la fonction : **tracerGraphique**

Paramètres : date,resultat- objets de type str et DataFrame

Préconditions : la fonction CreateDataframe doit exister

Rôle : Trace le graphique correspondant à la mesure de chaque polluant à Vitry-sur-Seine.

Valeur de retours : None - La fonction ne renvoie rien.

app.py :

Description : Ce module importe plusieurs autres modules (web_scraping, lectureCSV, visualisation, graphs, pdfFiles) et définit une fonction "**montrerGraphique**" qui prend en entrée une date et un DataFrame, utilise la fonction **CreateDataframe** pour obtenir un DataFrame représentant les mesures brute des polluants à Vitry-sur-Seine, affiche le DataFrame et ses données statistiques descriptives et affiche un graphique linéaire de la mesure de chaque polluant avec des alertes en cas de dépassement de seuils définis. Il y a également une fonction "**run**" qui exécute l'analyse de données en fonction de la date saisie par l'utilisateur.

Nom de la fonction : **montrerGraphique**

Paramètres : date,resultat- objets de type str et DataFrame

Préconditions : les fonctions CreateDataframe et CreateDescribeData doivent exister

Rôle : Trace et montre le graphique linéaire correspondant à la mesure de chaque polluant à Vitry-sur-Seine.

Valeur de retours : None - La fonction ne renvoie rien.

Nom de la fonction : **run**

Paramètres : None - la fonction ne prend rien en paramètre

Rôle : Exécutes l'analyse de données en fonction de la date saisie par l'utilisateur dans le format respecté, le cas échéant exécutes l'analyse de données pour le dernier relevé présent à ce jour.

Valeur de retours : None - La fonction ne renvoie rien.

run.py :

Description : Ce module utilise la bibliothèque « app.py » afin d'exécuter le logiciel sans interface web.

Il appelle la fonction **run()** du module « **app.py** ».

pdfFiles.py :

Description :

Ce module contient des fonctions pour générer des rapports en format PDF à partir de données de pollution de l'air. Il utilise les bibliothèques "tabulate" pour formater les données en tableaux, "fpdf" pour créer les fichiers PDF, et "pdfrw" pour fusionner les fichiers PDF ensemble. La fonction principale "**execute**" utilise la fonction "**tracerGraphique**" pour tracer un graphique des données de pollution de l'air et la fonction "**mergePDF**" pour fusionner les fichiers PDF de chaque rapport ensemble. Les fonctions "**reportDF**" et "**reportDescribeDF**" sont utilisées pour générer les fichiers PDF des données de pollution de l'air et des données statistiques décrivant ces données.

Nom de la fonction : **reportDF**

Paramètres : resultat - objet de type DataFrame

Préconditions : la fonction CreateDataframe doit exister

Rôle : Crée et enregistre le DataFrame passé en paramètre vers un fichier .pdf

Valeur de retours : None - La fonction ne renvoie rien.

Nom de la fonction : **reportDescribeDF**

Paramètres : resultat - objet de type DataFrame

Préconditions : les fonctions CreateDataframe et CreateDescribeData doivent exister

Rôle : Crée et enregistre le DataFrame passé en paramètre et détaillée vers un fichier .pdf

Valeur de retours : None - La fonction ne renvoie rien.

Nom de la fonction : **mergePDF**

Paramètres : resultat - objet de type DataFrame

Préconditions : les fonctions reportDF et reportDescribeDF et les fichiers « data-graph.pdf », « dataframe-report.pdf » et « datadesc-report.pdf doivent exister

Rôle : Fusionne les pdf de chaque rapport entre eux et créer un nouveau fichier .pdf résultant de cette fonction nommé « data-report.pdf »

Valeur de retours : None - La fonction ne renvoie rien.

Nom de la fonction : **execute**

Paramètres : date,resultat - objet de type string et DataFrame

Préconditions : les fonction tracerGraphique et mergePDF doivent exister

Rôle : Trace le graphique et génère le rapport .pdf du DataFrame

Valeur de retours : None - La fonction ne renvoie rien.

web.py :

Description :

Ce module est un serveur web développé avec le framework Flask, une bibliothèque pour créer des applications web en Python. Il a pour fonction principale d'afficher une page web avec un formulaire qui permet à l'utilisateur de saisir une date, lorsque celle-ci est soumise, il utilise cette date pour exécuter une analyse de données avec les fonctions de l'application principale (main) comme **readDateCSV** et **execute** pour générer un graphique et un rapport pdf, puis affiche à nouveau la page web. Il utilise également la bibliothèque request_arg pour récupérer les données soumises par le formulaire. Il y a également un configure pour permettre des requêtes simultanées.

Nom de la fonction : **index**

Paramètres : None - Cette fonction ne prend rien en paramètre.

Préconditions : un fichier « index.html » est présent dans le dossier « /templates »

Rôle : Affiche la page web présent dans le dossier /templates

Valeur de retours : Flask - Objet de type flask

Nom de la fonction : **submit_form**

Paramètres : None - Cette fonction ne prend rien en paramètre.

Préconditions : Un formulaire a été saisie sur la page web et récupéré dans le programme sous forme d'un str et un fichier « index.html » est présent dans le dossier « /templates ».

Les fonctions readDateCSV et execute doivent également exister.

Rôle : Récupère les données soumises par le formulaire et exécute l'analyse de données pour la date requise tout en affichant la page web présent dans le dossier « /templates » à la fin de l'exécution.

Valeur de retours : Flask - Objet de type flask

main.py :

Description :

Ce module est le programme principal de lancement de l'application web.

Il utilise les différents modules crées pour récupérer les données à partir d'un site web, les traiter, les visualiser sous forme de graphique, et les enregistrer sous forme de fichier PDF et PNG. Il utilise la bibliothèque Flask pour créer une interface web permettant à l'utilisateur de saisir une date et de visualiser les mesures de concentrations de polluants atmosphériques à Vitry-sur-Seine, tout en pouvant télécharger le graphique et le rapport de ces mesures.

Il utilise l'instruction **app.run(debug=False)** pour lancer l'application.

L'utilisateur pourra changer le **debug=False** en **debug=True** s'il souhaite consulter les avertissements et erreurs directement sur la console Python.