Compte rendu d'activités

Réalisées lors du stage de 2e année dans l'entreprise Meduza store

Projet :
L'entreprise revend des articles de marque de seconde main en ligne, une des plateformes qu'elle utilise est Vinted, cependant il n'existe pas d'outil permettant d'automatiser la mise en ligne des produits, tous doit être fait à la main, l'entreprise m'a donné comme mission de créer en Python un programme permettant de gérer la mise en ligne automatique de produit, mais aussi d'actualiser automatiquement la base de donnée en fonction des actions faite.

Durée du projet : 5 semaines

Outils professionnel utilisé : Pycharm et **DB Browser for SQLite**

Point tous les 2/3 jours sur l'avancée du projet

Lien Github : https://github.com/AxelusSS/ProjectVictor/blob/main/LICENCE

Problème rencontré pour la réalisation du bot :
Le site possède un système qui détecte les mouvements de souris et le scrolling, il faut donc agir au maximum comme un humain.

Creer un aléatoire dans les mouvements de souris et ne pas scroll directement vers la zone que l'on veut atteindre

Ses bouts de code nous serons utile :

```
def rdmMouse(x,y):
    i = random.randint(10,15)
    j = random.randint(1,9)
    z = round(((i*j)/10)+1)
    #print(z)
    vit = random.randint(1,10)
    if z==2:
        pyautogui.moveTo(x, y, 5, pyautogui.easeInQuad)
    elif z==3:
        pyautogui.moveTo(x, y, vit, pyautogui.easeInQuad)
    elif z==4:
        pyautogui.moveTo(x, y, 3, pyautogui.easeOutQuad)
    elif z==5:
        pyautogui.moveTo(x, y, 5, pyautogui.easeOutQuad)
    elif z==6:
        pyautogui.moveTo(x, y, vit, pyautogui.easeOutQuad)
    elif z==7:
        pyautogui.moveTo(x, y, 2, pyautogui.easeInOutQuad)
    elif z==8:
        pyautogui.moveTo(x, y, 4, pyautogui.easeInOutQuad)
    elif z==9:
        pyautogui.moveTo(x, y, vit, pyautogui.easeInOutQuad)
    elif z==10:
        pyautogui.moveTo(x, y, 4, pyautogui.easeInBounce)
    elif z==11:
        pyautogui.moveTo(x, y, 8, pyautogui.easeInBounce)
    elif z==12:
        pyautogui.moveTo(x, y, vit, pyautogui.easeInBounce)
    elif z==13:
        pyautogui.moveTo(x, y, 3, pyautogui.easeInElastic)
    elif z==14:
        pyautogui.moveTo(x, y, 7, pyautogui.easeInElastic)
    elif z==15:
        pyautogui.moveTo(x, y, vit, pyautogui.easeInElastic)
    else:
        pyautogui.moveTo(x, y, 2, pyautogui.easeInQuad)
```

```python
import random
import time

import pyautogui as pg
from selenium import webdriver

def autosrcoll(pixel):
    i = 0
    while i != pixel:
        if pixel < 0 :
            pg.scroll(-1)
            i = i - 1
        else:
            pg.scroll(1)
            i = i + 1
        time.sleep(random.uniform(0.02, 0.05))

def autoscrollend(driver):
    # driver.get("https://www.meduzastore.com/")
    time.sleep(4)
    for i in range(0, 25):
        autosrcoll(-2)
        time.sleep(random.uniform(0.000001, 0.005))

def autoscrolldeb(driver):
    # driver.get("https://www.meduzastore.com/")
    time.sleep(4)
    for i in range(0, 25):
        autosrcoll(1)
        time.sleep(random.uniform(0.000001, 0.005))
```

Ils permettront de fluidifier les mouvements de souris avec un touche d'aléatoire et de scroll a une vitesse plutôt humaine.

En plus de cela, l'ajout de deux bouts de code permettrons de rendre l'écriture et les zone ou les click sont effectué plus aléatoire :

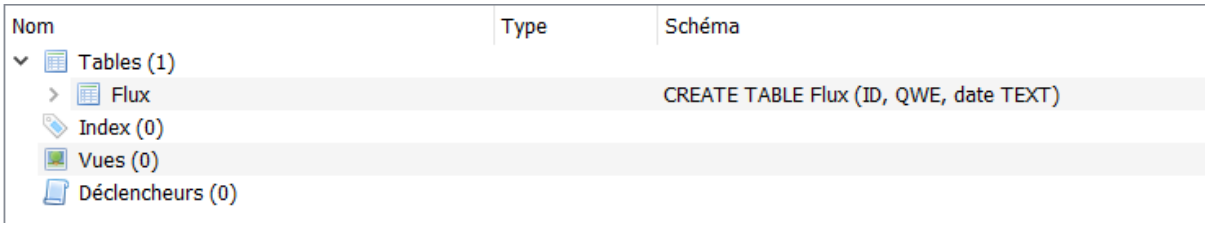
```
for L in prix :
    r = random.randint(1,5)
    if r == 1 or r == 3:
        driver.find_element(By.ID,'price').send_keys(L)
        i = random.uniform(0.0001 , 0.3)
        time.sleep(i)
        pg.press("backspace")
        driver.find_element(By.ID,'price').send_keys(L)
        i = random.uniform(0.0001 , 0.3)
        time.sleep(i)
    else:
        driver.find_element(By.ID,'price').send_keys(L)
        i = random.uniform(0.0001 , 0.3)
        time.sleep(i)
```

```
def rdmcreateco():
    x = 1730
    y = 134
    rm.rdmMouse(x,y)
    pg.click()
def rdmtitleco():
    x = random.randint(1281,1733)
    y = random.randint(628,650)
    rm.rdmMouse(x,y)
    pg.click()
def rdmdescco():
    x = random.randint(1281,1733)
    y = random.randint(664,771)
    rm.rdmMouse(x,y)
    pg.click()
```

La class iziflux.py vas récupérer les donner du flux par le lien du flux

```
    driver = webdriver.Chrome()
    driver.get("https://www.meduzastore.com/?export=iziflux")
    driver.implicitly_wait(5)
    element = driver.find_element(By.CSS_SELECTOR, "body > pre")
    donnees = element.text
    lignes = donnees.split('\n')
# ////////////////////////////////////////////////////////////////////////////////|
# Recupere chaque ligne de iziflux et regarde si l'article est en "Stock" ou "Vendu" |
# ////////////////////////////////////////////////////////////////////////////////|
    conn = sqlite3.connect('msflux.sqlite')
    cursor = conn.cursor()
    for i, ligne in enumerate(lignes):
        sections = ligne.split("|")
        disponibilite = sections[17]
        id_produit = sections[0]
        try:
            qwe_index = ligne.find("ref_qwe:")
            qwe = ligne[qwe_index:].split("#=")[0].split(":")[1]
        except:
            qwe = "BOXZ"
        cursor.execute("SELECT * FROM Flux WHERE ID = ?", (id_produit,))
        conn.commit()
        result = cursor.fetchone()
        print(i)
        print("ID :", id_produit)
        if disponibilite == "S" and "BOXZ" not in qwe and result is None:
```

Il stocke chaque ligne dans la liste lignes, puis regarde pour chaque ligne si, le produit est en stock “S” si oui ou “V” si non, si sont QWE contient “BOXZ” (qui représente les box de vêtement qui ne sont pas vendu sur Vinted par l’entreprise) et si le produit est déjà présent dans la base de données msflux.sqlite dont l’architecture est :



Pour les produits qui ne sont en stock, qui ne sont pas des BOX et qui ne sont pas dans la base de données, le programme récupère les informations nécessaires ;

```
# ////////////////// ID //////////////////
id_produit = sections[0]
print("ID du produit :", id_produit)
# ////////////////// Color //////////////////
couleur_index = ligne.find("couleurDepuisTerm:")
couleur = ligne[couleur_index:].split("#")[0].split(":")[1]
print("Color : ", couleur)
# ////////////////// Title //////////////////
nom_produit = sections[2]
print("Title : ", nom_produit)
# ////////////////// Price //////////////////
prix = sections[18]
print("Prix : ", prix)
# ////////////////// Brand //////////////////
marque_index = ligne.find("marqueDepuisTerm:")
marque = ligne[marque_index:].split("#")[0].split(":")[1]
print("Marque : ", marque)
# ////////////////// State //////////////////
etat_index = ligne.find("Etat du produit:")
etat = ligne[etat_index:].split("#")[0].split(":")[1]
print("Etat : ", etat)
# ////////////////// QWE //////////////////
qwe_index = ligne.find("ref_qwe:")
qwe = ligne[qwe_index:].split("#=")[0].split(":")[1]
print("QWE : ", qwe)
# ////////////////// Sexe,Categorie,Type //////////////////
categorie = sections[14]
c = categorie
lc = c.split(" > ")
lcs = []
for c in lc:
    if any(char.isalpha() for char in c):
        lcs.append(c)
lcs.sort(key=lambda x: ("Homme" not in x, "Femme" not in x, "Enfant" not in x,
sexe = lcs[0]
try:
    subcat = lcs[1]
    typeitem = lcs[2]
except IndexError:
    subcat = ""
    typeitem = lcs[1]
print("Sexe : ", sexe)
print("Sous-categorie : ", subcat)
print("TYpe Article : ", typeitem)
```

L’ID, la couleur, le titre, le prix, la marque, l’état, le QWE, le sexe/la catégorie/le type de produit etc...

Et si l’état de l’article est supérieur à 7, il envoi toute l'information collectée a la class Vinted qui s’occupe de le publier :

```
if '7' in etat or '8' in etat or '9' in etat or '10/' in etat :
    vinted.vinted(nom_produit, qwe, id_produit, prix, sexe, typeitem, taille, subcat,
                  categorie, marque, etat, matiere, couleur, lenght, width, sleeve,
                  tags, photo_urls_list)
```

Une fois le produit publié par le class Vinted (retrouvable avec le lien github : https://github.com/AxelusSS/ProjectVictor/blob/main/LICENCE), iziflux.py va rajouter dans la base de données sqlite le produit avec L'ID, le QWE et la date à la quel l'article a été mis

```
maintenant = datetime.now()
params = (id_produit, qwe, maintenant)
cursor.execute("INSERT INTO Flux (ID, QWE, DATE) VALUES (?, ?, ?)", params)
conn.commit()
```

Ce qui donne dans la BDD

	ID	QWE	date ▲1
	Filtre	Filtre	Filtre
1	70060	#QWE3085	2024-03-08 15:33:20.650540
2	70063	#QWE3075	2024-03-08 15:21:25.322073
3	70064	#QWE3791	2024-03-08 14:22:57.939420
4	70067	#QWE3697	2024-03-08 14:08:03.267256
5	70073	#QWE0450	2024-03-08 14:02:15.250038
6	70075	#QWE3706	2024-03-08 13:56:15.988666
7	70471	#QWE3123	2024-03-08 13:32:19.235427
8	70595	#QWE1864	2024-03-08 13:14:55.855477
9	70596	#QWE0180	2024-03-08 11:07:02.218375