MINISTERUL EDUCAȚIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi Microelectronică

Departamentul Informatică şi Ingineria Sistemelor

RAPORT

Sarcina nr 1

la cursul „***Procesarea informațiilor***”

Tema: Extragerea informațiilor

A efectuat : **St. gr. SD-241, Căpățînă Sorin**

A verificat: **Asis.univ. Elena GOGOI**

Chișinău 2026

CUPRINS

INTRODUCERE

În era digitală, volumul informației online crește rapid, iar imaginile au un rol central în documentare și analiză. Scopul acestui proiect este colectarea automată a imaginilor de pe o sursă online selectată, împreună cu metadatele asociate, și stocarea lor într-o bază de date JSON, structurată și ușor de procesat.

Sarcina:

1. De selectat sursa de informație online din lista prezentată în Anexa 1.
2. De creat un script care va extrage imagini de pe sursa online cu articole/postări.
3. De stocat informația în formatul json. Json trebuie să conțină următoarele câmpuri:
    - Descrierea imaginii
    - Autorul/sursa
    - Data/ora
    - Tematica/Tematici

Teorie generală

Colectarea se realizează prin web scraping sau API-uri, metode care permit extragerea sistematică a datelor de pe pagini web sau din fluxuri structurate. Fiecare intrare în JSON conține: descrierea imaginii, autorul/sursa, data/ora publicării și tematica, dacă aceste informații sunt disponibile.

Scriptul asigură extragerea tuturor imaginilor existente și poate fi configurat să ruleze zilnic pentru capturarea postărilor noi. Tehnologiile folosite includ Python și biblioteci specializate precum requests, BeautifulSoup sau Selenium, garantând colectarea corectă și stocarea fiabilă a datelor.

1. ANALIZA STRUCTURII HTML

Pentru realizarea acestei sarcini, procesul de rezolvare este împărțit în mai multe etape succesive. În prima etapă se analizează structura paginii web pentru a identifica zonele care conțin informația relevantă.

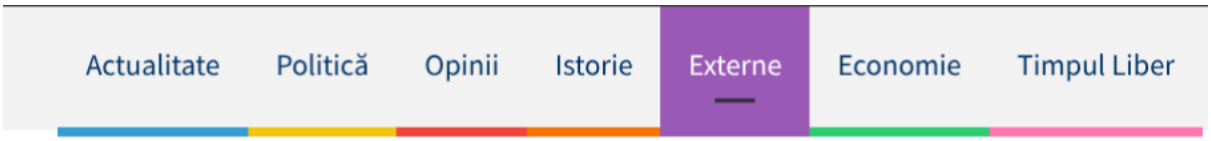

Figura 1.1 Tematica

Pagina principală nu oferă imagini unice organizate clar, de aceea se utilizează meniul de navigare („nav”), unde sunt selectate rubricile tematice (de exemplu „Actualitate”).

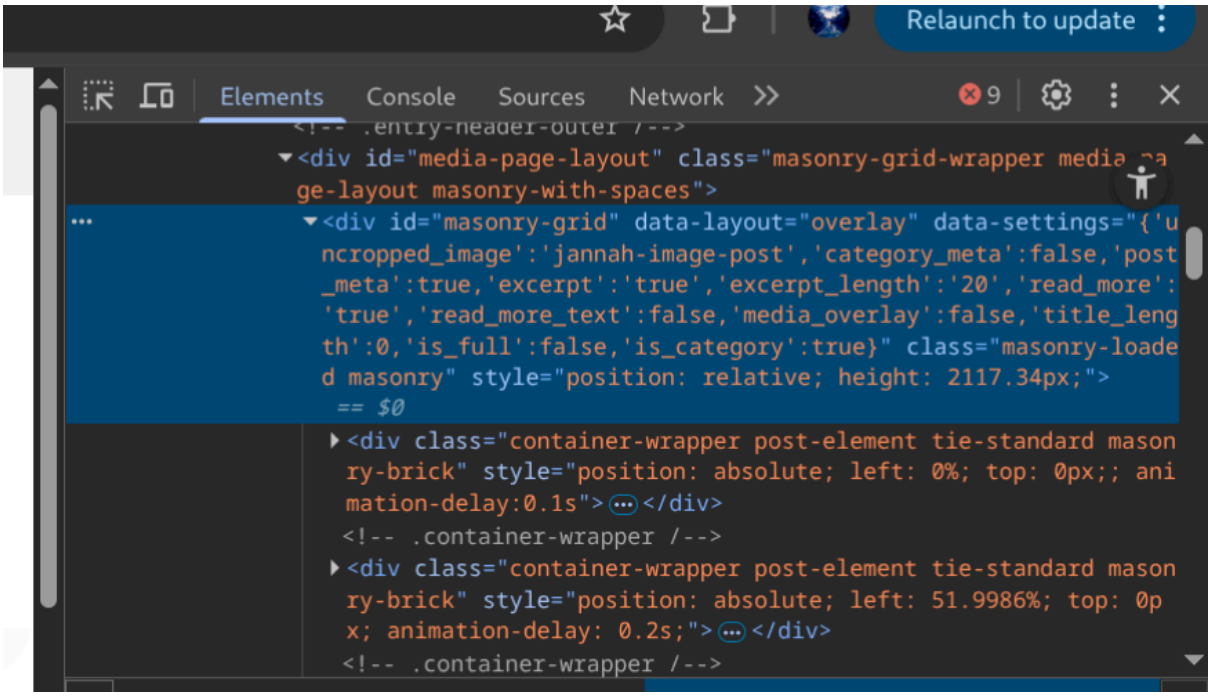

Figura 1.2 divul principal

În aceste pagini, articolele sunt grupate într-un container de tip div-masonry-grid, care conține elemente individuale (div-slide) ce includ imaginea, titlul, descrierea și metadatele asociate.

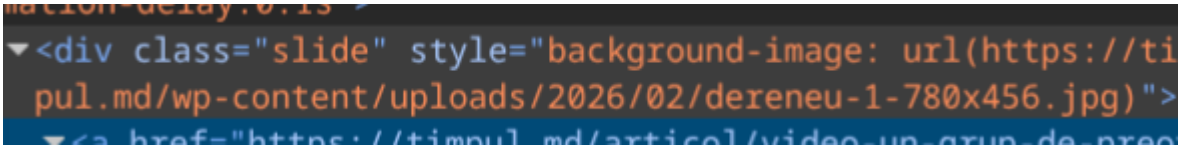

Figura 1.3 Articol individual

În interiorul containerului principal, fiecare articol este reprezentat printr-un element individual de tip div-slide, care conține toate datele asociate unui singur material: imaginea, titlul, descrierea și metadatele.

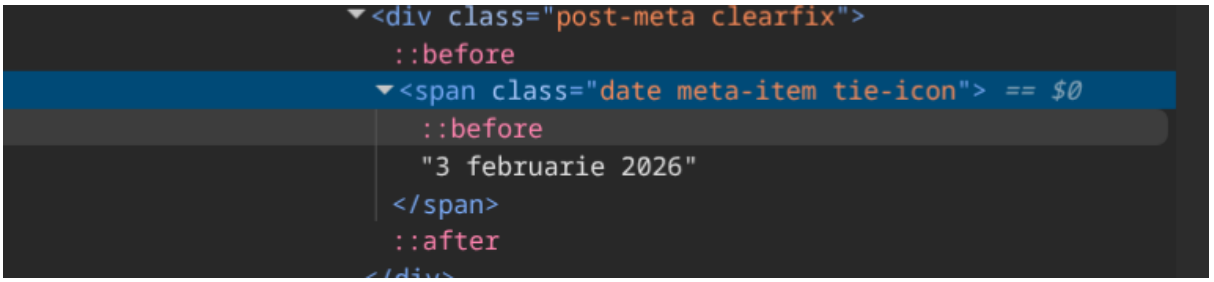

Figura 1.4 Data

În etapa următoare se identifică elementele HTML din care vor fi extrase datele: data publicării articolului din elementul care conține data publicării imaginii

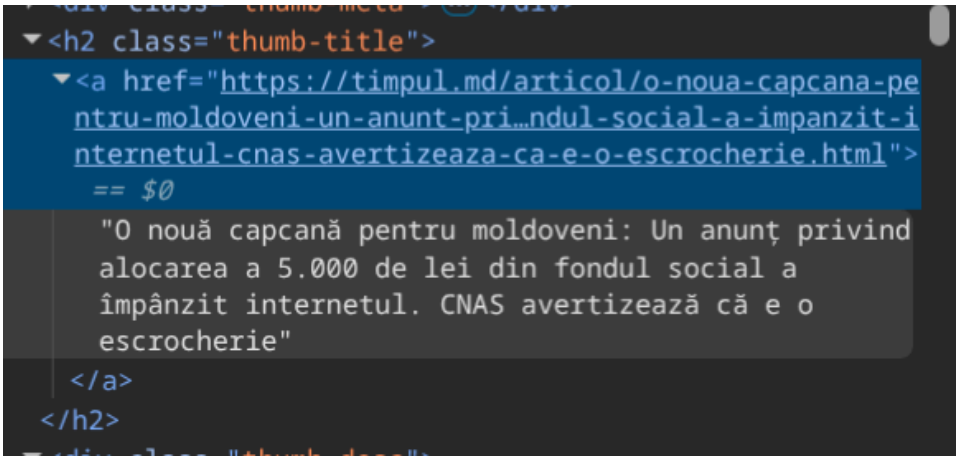

Figura 1.5 Titlul

Titlul articolului este localizat în elementul HTML <h2 class="thumb-title">. Scriptul identifică acest element pentru fiecare articol și extrage conținutul textual,

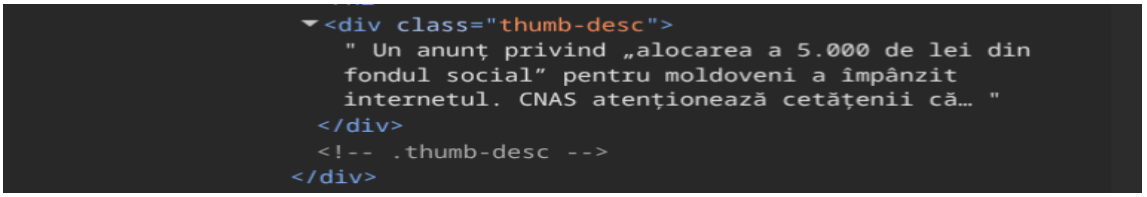

Figura 1.6 Descrierea imaginii

Descrierea articolului se află în elementul . Scriptul extrage textul din acest element pentru fiecare articol și îl stochează în câmpul „descriere” al fișierului JSON.

2. EXTRAGEREA DATELOR

Procesul de extragere a datelor este realizat în mai multe etape, utilizând limbajul Python și bibliotecile Requests, BeautifulSoup, re, json și os. Scriptul automatizează colectarea informațiilor despre articolele ce conțin imagini, descărcarea imaginilor și salvarea datelor într-un fișier JSON.

Obținerea linkurilor categoriilor

În prima etapă sunt identificate linkurile către rubricile tematice ale site-ului. Scriptul trimite o cerere HTTP către pagina principală https://timpul.md, după care conținutul HTML este analizat cu BeautifulSoup. Din meniul de navigare sunt selectate elementele li care conțin clase de tip *menu-item-type-taxonomy*, acestea reprezentând categoriile site-ului

```
import requests
from bs4 import BeautifulSoup
import re

def get_links():
    url = "https://timpul.md"
    headers = {"User-Agent": "Mozilla/5.0"}

    response = requests.get(url, headers=headers)
    soup = BeautifulSoup(response.text, "lxml")

    lists = soup.find("ul")
    elements = lists.find_all("li",
class_=re.compile(r"menu-item-type-taxonomy"))

    links = {}
    for el in elements:
        links[el.text] = el.find("a").get("href")

    return links
```

Pentru fiecare element este extras textul (denumirea tematicii) și atributul href al elementului a, care reprezintă adresa paginii respective. Rezultatul este memorat într-un dicționar Python, unde cheia este tematica, iar valoarea este linkul categoriei. Aceste linkuri vor fi ulterior utilizate de scriptul principal pentru a accesa paginile articolelor și pentru a extrage imaginile.

Extragerea adresei imaginii

Adresa imaginii nu este prezentă direct într-un atribut src, ci într-un atribut de tip stil (data-lazy-style). Funcția extract_image_url() utilizează expresii regulate pentru a extrage adresa URL din textul stilului.

```
def extract_image_url(style):
    if not style:
        return None
    match = re.search(r"url\((['\"]?)(.*?)\1\)", style)
    return match.group(2) if match else None
```

Această funcție returnează adresa imaginii care va fi ulterior descărcată.

Extragerea descrierii

Descrierea articolului este localizată în elementul HTML div cu clasa thumb-desc. Scriptul identifică acest element și extrage textul conținut, eliminând spațiile inutile pentru a obține o formă curată a descrierii.

```
desc = img.find("div", class_="thumb-desc")
" ".join(desc.text.split()) if desc else None
```

Extragerea datei publicării

Data publicării articolului este extrasă din elementul span cu clasa date meta-item tie-icon, care conține informația despre momentul apariției materialului pe site.

```
date = img.find("span", class_="date meta-item tie-icon")
date.text.strip() if date else None
```

3. REZULTATE

Stocarea datelor în format JSON

După extragerea informațiilor din paginile web, datele colectate sunt organizate într-o structură de tip listă de dicționare Python, unde fiecare dicționar reprezintă o imagine și metadatele asociate acesteia. Pentru fiecare articol sunt salvate câmpurile: descrierea imaginii, sursa, data publicării, tematica și locația fișierului imaginii descărcate.

Formarea fiecărei înregistrări se realizează prin construirea unui dicționar Python care conține valorile extrase în etapele anterioare:

```
item = {
    "descriere_imagine": " ".join(desc.text.split()) if desc else None,
    "autor_sursa": "Timpul.md",
    "data_ora": date.text.strip() if date else None,
    "tematica": topic,
    "image_file": local_image_path
}

all_data.append(item)
```

Toate înregistrările sunt stocate în lista all_data, care reprezintă colecția finală de date. După finalizarea procesului de colectare, lista este salvată într-un fișier de tip **JSON** utilizând biblioteca json. Salvarea se realizează cu codul:

```
with open(OUTPUT_JSON, "w", encoding="utf-8") as f:
    json.dump(all_data, f, ensure_ascii=False, indent=2)
```

Parametrul ensure_ascii=False permite salvarea corectă a caracterelor speciale, iar indent=2 structurează fișierul într-un format lizibil. În urma acestei etape se obține un fișier JSON care conține toate informațiile extrase, fiind ușor de utilizat ulterior pentru analiză, procesare sau integrare în alte aplicații.

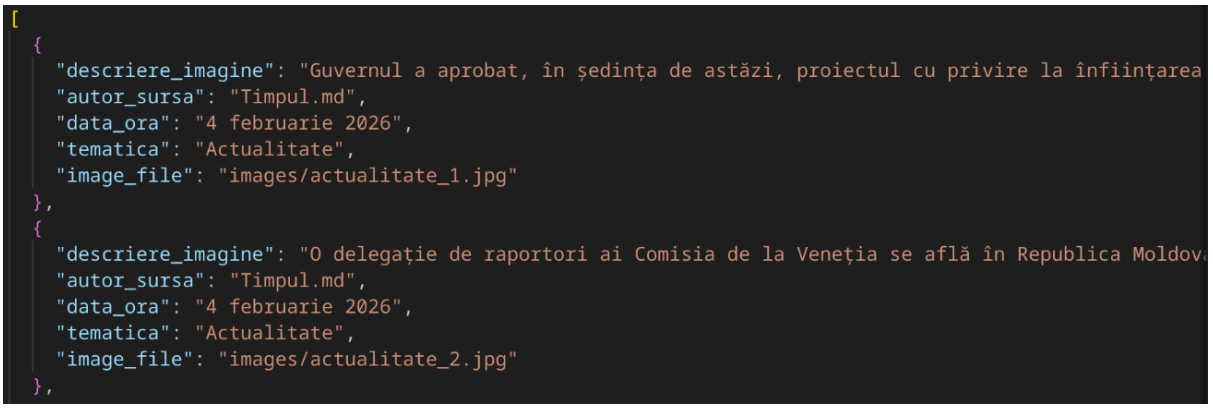

Figura 3.1 Fișierul json

Fiecare obiect din JSON corespunde unei imagini și conține toate informațiile necesare pentru analiză, vizualizare sau integrare în alte aplicații.

Salvarea imaginilor

Pe lângă salvarea informațiilor în JSON, scriptul descarcă fiecare imagine asociată articolului și o stochează local, într-un director dedicat. Acest lucru permite păstrarea materialului vizual pentru utilizare ulterioară fără a fi nevoie de acces continuu la site-ul web.

```
IMAGES_DIR = "images"
os.makedirs(IMAGES_DIR, exist_ok=True)
```

CONCLUZII

Automatizarea procesului de colectare a imaginilor și a informațiilor asociate a demonstrat cât de mult poate fi eficientizat un astfel de flux de lucru. Prin folosirea unui script bine structurat, am reușit să extragem datele direct din paginile web, să descărcăm imaginile și să păstrăm toate informațiile într-un format organizat, ușor de folosit ulterior. Aceasta elimină nevoia de a face manual fiecare pas și reduce riscul de erori.

Etapele clare ale procesului – identificarea elementelor din HTML, extragerea titlului, descrierii și datei publicării, descărcarea imaginilor și salvarea datelor în JSON – au făcut întregul sistem stabil și ușor de urmărit. Fiecare imagine și informație asociată este păstrată într-un mod organizat, ceea ce permite folosirea lor rapidă în viitor, fără a mai fi nevoie să revenim pe site pentru a colecta datele.

De asemenea, metoda folosită permite extinderea ușoară a scriptului pentru alte categorii sau alte site-uri. Acest lucru arată că un flux bine definit de colectare și stocare a datelor nu doar economisește timp, dar asigură și consistența și calitatea informațiilor.

În concluzie, folosirea unui sistem automatizat de extragere și stocare a datelor web transformă informațiile dispersate în date organizate și accesibile, făcând întregul proces mai rapid, mai sigur și mai eficient.

BIBLIOGRAFIE

Imaginile:https://drive.google.com/drive/folders/1ASSJnHlgwSFo8L5zUbb6ilEvZNmJilSC?usp=sharing
Situl: https://timpul.md/

ANEXE

Anexa 1 Codul Sursă

```
import requests
from bs4 import BeautifulSoup
import re
import json
import os
from links import get_links

headers = {"User-Agent": "Mozilla/5.0"}

OUTPUT_JSON = "images_info.json"
IMAGES_DIR = "images"


def extract_image_url(style):
    if not style:
        return None
    match = re.search(r"url\((['\"]?)(.*?)\1\)", style)
    return match.group(2) if match else None


def download_image(url, filename):
    if not url:
        return None

    try:
        response = requests.get(url, headers=headers, timeout=20)
        response.raise_for_status()
    except requests.exceptions.RequestException:
        return None

    path = os.path.join(IMAGES_DIR, filename)
    with open(path, "wb") as f:
        f.write(response.content)

    return path


def get_image_links():
    links = get_links()
    os.makedirs(IMAGES_DIR, exist_ok=True)

    all_data = []
    image_counter = 1

    for topic, base_url in links.items():
        for page in range(1, 3):  # 🔹 doar 2 pagini
            url = base_url if page == 1 else f"{base_url}/page/{page}"

            response = requests.get(url, headers=headers)
            soup = BeautifulSoup(response.text, "lxml")

            slides = soup.find("div", id="masonry-grid")
            if not slides:
                continue

            images = slides.find_all("div", class_="slide")
```

```
            for img in images:
                image_style = img.get("data-lazy-style")
                image_url = extract_image_url(image_style)

                date = img.find("span", class_="date meta-item tie-icon")
                desc = img.find("div", class_="thumb-desc")

                                    image_name  =  f"{topic.lower().replace('  ',
'_')}_{image_counter}.jpg"
                local_image_path = download_image(image_url, image_name)

                item = {
                       "descriere_imagine": " ".join(desc.text.split()) if desc
else None,
                    "autor_sursa": "Timpul.md",
                    "data_ora": date.text.strip() if date else None,
                    "tematica": topic,
                    "image_file": local_image_path
                }

                all_data.append(item)
                image_counter += 1

    with open(OUTPUT_JSON, "w", encoding="utf-8") as f:
        json.dump(all_data, f, ensure_ascii=False, indent=2)

    print(f"Saved {len(all_data)} records")
    print(f"Images stored in ./{IMAGES_DIR}/")


if __name__ == "__main__":
    get_image_links()
```