Universitatea Politehnica Timișoara Master –
Securitatea Informațiilor și a Sistemelor Cibernetice
An universitar: 2025-2026

Dezvoltarea unei aplicații de tip web scraper pentru colectarea și analiza datelor de pe platfoma GSMArena

Repository GitHub: https://github.com/CorinaOxani/gsmarena-scraper

- Securitatea Aplicațiilor Mobile -

Student: Oxani Corina
Profesor Coordonator :
Prof. Dr.Ing. Habil. Marius Marcu

Grupa 1.2, anul I, AC, IS- master SISC
Anul universitar 2025-2026

CUPRINS

1. DESCRIEREA PROIECTULUI

1.1 Contextul și motivația realizării proiectului

În prezent, piața dispozitivelor mobile este într-o continuă dezvoltare, iar utilizatorii au la dispoziție o gamă foarte variată de telefoane. Pentru a putea face comparații sau analize corecte, este necesar accesul la informații tehnice precum tipul ecranului, performanța hardware, capacitatea bateriei sau caracteristicile camerelor.

GSMArena este una dintre cele mai populare platforme online care oferă specificații detaliate și actualizate despre dispozitive mobile [1]. Informațiile sunt însă disponibile doar sub forma paginilor web, fiind destinate în principal utilizatorilor finali, fără a exista un API oficial care să permită accesul programatic la aceste date.

În acest context, utilizarea tehnicilor de web scraping reprezintă o soluție practică pentru colectarea automată a informațiilor de pe platforma GSMArena. Web scraping-ul permite extragerea și structurarea datelor din pagini web, astfel încât acestea să poată fi utilizate ulterior pentru analiză sau comparare.

Motivația realizării acestui proiect este dezvoltarea unei aplicații capabile să colecteze automat informații relevante despre dispozitive mobile de pe GSMArena și să le salveze în formate ușor de utilizat, precum JSON și CSV. Prin acest proiect se urmărește, de asemenea, înțelegerea pașilor necesari pentru analiza structurii unei pagini web și implementarea unei soluții funcționale de colectare a datelor în lipsa unui acces programatic oficial.

1.1 Analiza sursei de date GSMArena

Pentru realizarea aplicației a fost necesară analizarea modului în care pot fi accesate datele disponibile pe platforma GSMArena. Au fost luate în considerare două variante: utilizarea unui API pentru acces programatic sau colectarea directă a datelor din paginile web.

În urma unei căutări pe platforme precum Google, GitHub și RapidAPI, au fost identificate mai multe API-uri neoficiale care oferă acces la datele GSMArena. Aceste soluții permit obținerea informațiilor în format JSON, însă nu sunt dezvoltate sau susținute oficial de GSMArena și funcționează, în realitate, tot prin tehnici de web

scraping. Unele dintre ele impun și limitări de utilizare, cum ar fi accesul pe bază de cheie API sau un număr limitat de cereri. [6][7]

Platformă / Sursă	Link / Descriere	Statut
nordmarin/gsmarena-api (GitHub)	API care citește date direct de pe GSMArena și returnează rezultate în format JSON. https://github.com/nordmarin/gsmarena-api	Neoficial, proiect în dezvoltare
RapidAPI – GSM Arena Parser	Serviciu comercial care oferă endpoint-uri pentru căutare, detalii despre telefoane, specificații și imagini. https://rapidapi.com/controller2042000/api/gsmarenaparser	Neoficial, acces limitat prin cheie API
Postman API (Kyaw's)	Colecție publică Postman care expune endpoint-uri pentru GSMArena, de exemplu: GET /brands, GET /phones, GET /details. https://www.postman.com/blue-flare-7250/kyaw-s/documentation/9pbl2cg/gsm-arena-api	Neoficial
raffyxyz/api-gsm-arena (GitHub)	API care parsează specificațiile telefoanelor (CPU, camera, battery, etc.) direct din HTML. https://github.com/raffyxyz/api-gsm-arena	Neoficial, cod open-source

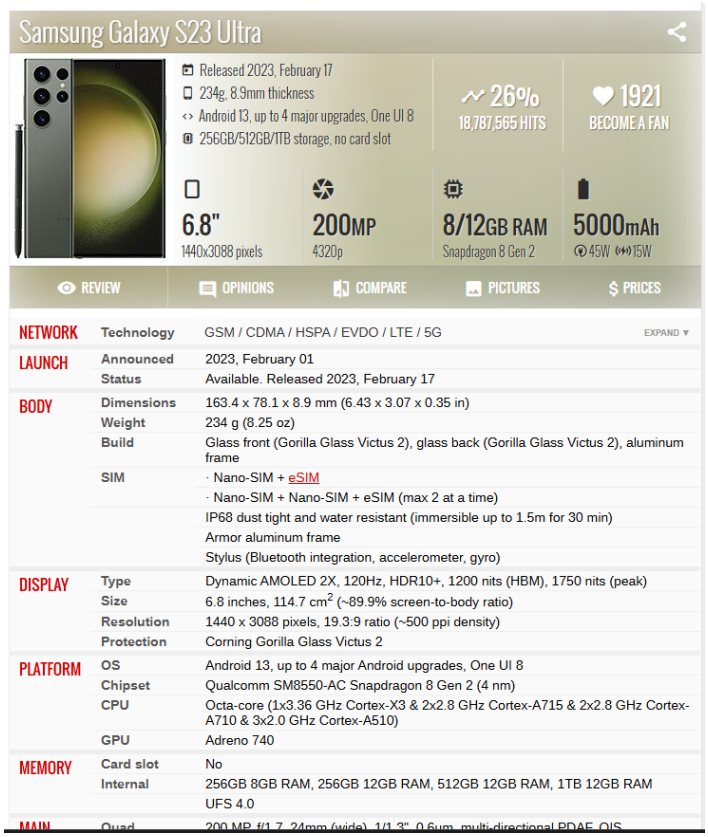
În urma acestei analize, s-a constatat că GSMArena nu oferă un API oficial, iar utilizarea API-urilor neoficiale nu reprezintă o soluție stabilă pe termen lung. Din acest motiv, pentru acest proiect a fost aleasă implementarea unui scraper propriu, care oferă control complet asupra modului de colectare și procesare a datelor.

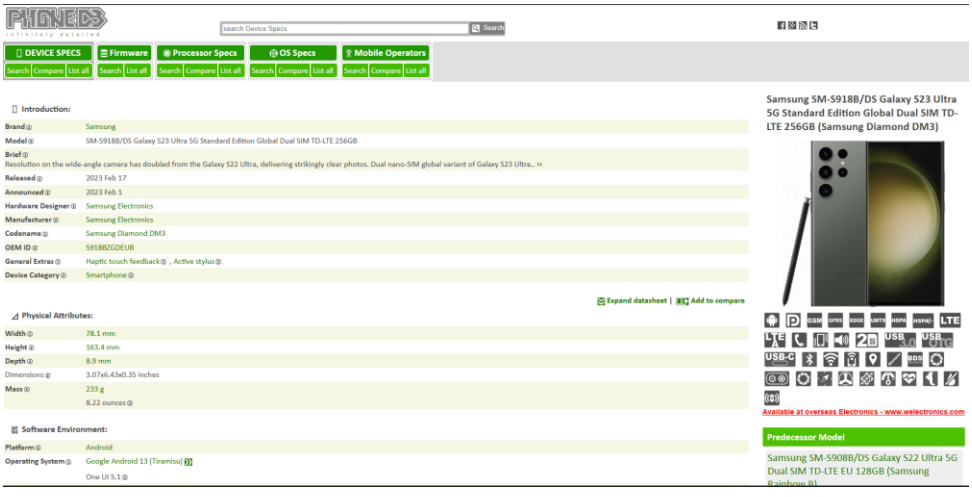
Analiza structurii paginilor GSMArena a arătat că informațiile sunt organizate într-un mod clar și relativ constant. Fiecare brand are o pagină de listă cu modelele disponibile, iar fiecare model are o pagină individuală unde specificațiile sunt structurate în tabele, pe secțiuni precum Network, Display, Platform, Memory, Camera și Battery. Această structură facilitează extragerea automată a datelor. [5]

De asemenea, a fost analizată aplicația terță PhoneDB, care oferă specificații similare cu cele de pe GSMArena. În urma comparării aceluiași model de telefon, s-a observat că datele principale sunt foarte apropiate, diferențele apărând în special la nivel de formatare și detaliere. [1][8]

Acest lucru arată că aplicația utilizează informații publice, pe care le organizează într-un mod diferit față de GSMArena.

În următorul exemplu se poate observa că, GSMArena prezintă datele într-un mod vizual și compact, iar PhoneDB le descompune în câmpuri tehnice (mm, Hz, GiB, Wh) și adaugă detalii suplimentare precum frecvențe, tensiune baterie sau coduri hardware.





În concluzie, GSMArena reprezintă o sursă de date potrivită pentru acest proiect, iar lipsa unui API oficial justifică utilizarea tehnicilor de web scraping. Structura clară a paginilor permite colectarea și salvarea datelor într-un format structurat, precum JSON sau CSV, pentru analiză ulterioară.

1.2 Obiectivele aplicației dezvoltate

Obiectivul principal al aplicației este dezvoltarea unui scraper web capabil să colecteze automat informații despre telefoanele mobile de pe platforma GSMArena.

Obiectivele specifice ale aplicației sunt:

- accesarea paginilor individuale ale telefoanelor pe baza unor linkuri predefinite;
- extragerea informațiilor esențiale, precum denumirea telefonului, ecranul, procesorul, memoria, camera, bateria, sistemul de operare și prețul;
- structurarea datelor colectate într-un format ușor de utilizat;
- salvarea rezultatelor în fișiere de tip JSON și CSV, pentru analiză ulterioară; [10][9]
- testarea aplicației pe un set de telefoane predefinite.

Aplicația are ca scop demonstrarea modului în care pot fi utilizate tehnicile de web scraping pentru colectarea și organizarea automată a datelor din surse web care nu oferă acces programatic oficial.

2. MEDIUL DE DEZVOLTARE ȘI TEHNOLOGII UTILIZATE

2.1 Limbajul de programare: Python

Aplicația a fost dezvoltată utilizând limbajul de programare Python, ales datorită simplității sintaxei și a suportului foarte bun pentru aplicații de tip web scraping. Python permite dezvoltarea rapidă a aplicațiilor și oferă biblioteci mature pentru accesarea paginilor web, procesarea conținutului HTML și salvarea datelor într-un format structurat. [2]

În cadrul acestui proiect, Python a fost utilizat pentru implementarea întregului flux de colectare a datelor: trimiterea cererilor HTTP către platforma GSMArena, analizarea structurii paginilor web, extragerea informațiilor relevante despre telefoane și salvarea rezultatelor în fișiere JSON și CSV. De asemenea, limbajul a permis organizarea codului într-o structură modulară, ușor de înțeles și extins. [2][10][9]

2.2 Biblioteci și tehnologii utilizate

Pentru realizarea aplicației de tip scraper au fost utilizate mai multe biblioteci Python, fiecare având un rol bine definit în procesul de colectare și procesare a datelor.

Biblioteca requests este utilizată pentru trimiterea cererilor HTTP către paginile GSMArena [3]. Aceasta permite descărcarea conținutului HTML al paginilor și gestionează automat redirecționările. În aplicație, cererile sunt realizate prin intermediul unei sesiuni (requests.Session), la care sunt atașate headere HTTP personalizate, inclusiv un User-Agent, pentru a simula comportamentul unui browser real. De asemenea, între cereri este introdus un timp de așteptare (delay_seconds) pentru a evita supraîncărcarea serverului.

Biblioteca BeautifulSoup este utilizată pentru parsarea conținutului HTML și extragerea informațiilor relevante. Aceasta permite identificarea elementelor HTML specifice paginilor GSMArena, precum titlul telefonului, imaginea principală și tabelele care conțin specificațiile tehnice. Structura relativ constantă a paginilor permite navigarea ușoară prin aceste elemente și colectarea datelor dorite. [4][5]

Pentru salvarea rezultatelor, sunt utilizate bibliotecile standard json și csv, care permit exportarea informațiilor extrase în formate ușor de utilizat și analizat ulterior.

2.3 Structura aplicației dezvoltate

Aplicația este organizată într-o structură modulară, fiecare fișier având o responsabilitate clară în procesul de colectare și procesare a datelor.

Fișierul scraper.py conține clasa GsmArenaScraper, responsabilă de accesarea paginilor web GSMArena. Aceasta gestionează trimiterea cererilor HTTP, setarea headerelor, tratarea redirecționărilor și introducerea unui interval de timp între cereri. Metoda fetch() returnează atât conținutul HTML al paginii, cât și adresa finală după eventualele redirecționări. [3]

Fișierul parser.py este responsabil de analizarea conținutului HTML și de extragerea informațiilor despre telefoane. Funcția parse_phone_page() extrage datele generale ale telefonului (brand, model, imagine) și construiește o structură de tip dicționar care conține specificațiile organizate pe secțiuni (Network, Display, Platform, Memory, Camera, Battery etc.). Funcția pick_key_fields() selectează un set de câmpuri relevante, precum tipul ecranului, procesorul, memoria, camera principală, bateria, sistemul de operare și prețul, utilizând o logică robustă care ține cont de secțiunile corecte din care trebuie preluate valorile. [4][5]

Fișierul main.py coordonează execuția aplicației. Acesta definește lista de adrese URL analizate, inițializează scraperul, apelează funcțiile de parsare pentru fiecare telefon și gestionează salvarea rezultatelor. Datele complete sunt salvate într-un fișier JSON, iar câmpurile esențiale sunt exportate atât în format JSON, cât și CSV, în directorul output. [10][9]

Această structură modulară face aplicația ușor de înțeles, testat și extins, permițând adăugarea de noi câmpuri sau analizarea altor modele de telefoane fără modificări majore ale codului existent.

2.4 Versionarea codului și repository-ul Git

Codul sursă al aplicației dezvoltate este versionat folosind sistemul de control al versiunilor Git și este disponibil într-un repository public GitHub. Repository-ul conține întregul cod sursă al aplicației, structura modulară a proiectului, precum și fișierele de configurare necesare pentru rulare. [11]

3. PROIECTAREA ȘI IMPLEMENTAREA APLICAȚIEI

3.1 Fluxul de execuție al aplicației

Aplicația este implementată ca un script Python care rulează secvențial și procesează un set predefinit de pagini GSMArena. Punctul de pornire este fișierul main.py, unde se definește lista de URL-uri și se coordonează întregul flux:

- colectare
- parsare
- selecție câmpuri
- export

Lista de pagini analizate este definită explicit în cod:

```
8   PHONE_URLS = [
9       "https://www.gsmarena.com/samsung_galaxy_s23_ultra-12024.php",
10      "https://www.gsmarena.com/xiaomi_13t_pro-12388.php",
11      "https://www.gsmarena.com/google_pixel_8_pro-12545.php",
12  ]
```

În funcția main(), aplicația:

- creează folderul de output,
- inițializează scraperul,
- parcurge fiecare URL, descarcă HTML-ul, parsează datele și construiește rezultatele:

```
35  def main():
36      ensure_out_dir()
37
38      scraper = GsmArenaScraper(delay_seconds=1.0)
39
40      full_phones = []
41      key_rows = []
42
43      for url in PHONE_URLS:
44          print(f"[+] Fetch: {url}")
45          html, final_url = scraper.fetch(url)
46
47          if final_url != url:
48              print(f"    [i] Redirect: {url} -> {final_url}")
49
50          phone = parse_phone_page(html, final_url)
```

Pentru consistență, aplicația face și o verificare simplă dacă modelul rezultat pare să corespundă cu brandul din URL:

```
requested_brand = final_url.split("/")[-1].split("_")[0].lower()
parsed_model = (phone.get("model") or "").lower()

if requested_brand and parsed_model and requested_brand not in parsed_model:
    print(f"    [WARN] Model mismatch. Parsed: '{phone.get('model')}'")
```

Datele complete și cele filtrate sunt adunate în două liste separate.

3.2 Modulul de colectare a datelor

Colectarea datelor este realizată în fișierul scraper.py, prin clasa GsmArenaScraper. Modulul are rolul de a trimite cereri HTTP către paginile GSMArena și de a întoarce conținutul HTML care va fi procesat ulterior.

Pentru a imita comportamentul unui browser real, se folosește un User-Agent și Accept-Language, setate în headere:

```
3
4   DEFAULT_HEADERS = {
5       "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
6                     "AppleWebKit/537.36 (KHTML, like Gecko) "
7                     "Chrome/120.0 Safari/537.36",
8       "Accept-Language": "en-US,en;q=0.9",
9   }
10
```

Scraperul folosește o sesiune requests.Session() pentru a păstra setările și a avea cereri mai stabile:

```
10
11  class GsmArenaScraper:
12      def __init__(self, delay_seconds: float = 1.0, timeout: int = 20):
13          self.delay_seconds = delay_seconds
14          self.timeout = timeout
15          self.session = requests.Session()
16          self.session.headers.update(DEFAULT_HEADERS)
17
```

Metoda fetch() descarcă pagina, tratează redirecționările și introduce un delay între cereri:

```
    def fetch(self, url: str) -> tuple[str, str]:
        """
        Descarcă HTML-ul paginii și întoarce:
        - html (string)
        - final_url (după redirect, dacă există)
        """
        resp = self.session.get(url, timeout=self.timeout, allow_redirects=True)
        resp.raise_for_status()
        time.sleep(self.delay_seconds)
        return resp.text, resp.url
```

Rezultatul acestei etape este tuplul:

- resp.text (HTML-ul paginii)
- resp.url (URL-ul final, după redirect, dacă există)

3.3 Modulul de parsare și extracție a informațiilor

Parsarea este implementată în parser.py folosind BeautifulSoup. Scopul funcției parse_phone_page() este să extragă:

- modelul (numele telefonului),
- brandul (derivat din nume sau din URL),
- imaginea principală,
- specificațiile organizate pe secțiuni (Network, Display, Platform etc.).

Extragerea numelui modelului se face din titlul paginii (h1.specs-phone-name-title):

```
    # Nume telefon
    name_tag = soup.select_one("h1.specs-phone-name-title")
    model_name = _clean(name_tag.get_text()) if name_tag else None
```

Brandul este dedus fie din primul cuvânt al modelului, fie din URL (fallback):

```
    # Brand
    brand = None
    if model_name:
        brand = model_name.split()[0]
    else:
        path = urlparse(url).path.rsplit("/", 1)[-1]
        brand = path.split("_", 1)[0] if "_" in path else None
```

Imaginea principală este extrasă din zona dedicată (div.specs-photo-main img):

```
    # Imagine principală
    img_tag = soup.select_one("div.specs-photo-main img")
    image_url = img_tag.get("src") if img_tag else None
```

Apoi se parsează tabelul de specificații din #specs-list. Pentru fiecare tabel se ia numele secțiunii și se extrag rândurile (cheie–valoare):

```
    # Specs:
    specs = {}
    specs_root = soup.select_one("#specs-list")

    if specs_root:
        for table in specs_root.select("table"):
            th = table.select_one("th")
            section = _clean(th.get_text()) if th else "Other"
            specs.setdefault(section, {})

            for row in table.select("tr"):
                key_cell = row.select_one("td.ttl")
                val_cell = row.select_one("td.nfo")
                if not key_cell or not val_cell:
                    continue

                key = _clean(key_cell.get_text())
                val = _clean(val_cell.get_text(" ", strip=True))
                if key and val:
                    specs[section][key] = val
```

Rezultatul final al acestei funcții este o structură de tip dicționar, cu date generale împreună cu specificațiile pe secțiuni.

3.4 Selectarea câmpurilor cheie

După ce pagina este parsată și toate specificațiile sunt salvate pe secțiuni (Display, Platform, Battery etc.), aplicația extrage un set mai mic de informații „esențiale”, utile pentru comparații rapide între telefoane. Această selecție este realizată în funcția pick_key_fields() din fișierul parser.py.

Pentru a evita erori, logica de selecție ține cont de faptul că anumite denumiri de câmpuri se repetă în secțiuni diferite. De exemplu, cheia „Type” apare atât la Display (tipul ecranului), cât și la Battery (tipul bateriei). Din acest motiv, aplicația nu caută global în tot dicționarul, ci extrage valorile din secțiunea corectă.

Exemplu: tipul ecranului

```
    # Display: strict din Display -> Type
    display = specs.get("Display", {}).get("Type")

```

Bateria este luată strict din Battery, ca să nu fie confundată cu „Type” din alte secțiuni:

```
# Battery: strict din Battery
battery = specs.get("Battery", {}).get("Type")
charging = specs.get("Battery", {}).get("Charging")
```

Pentru OS și chipset, aplicația caută cu prioritate în secțiunea Platform, folosind funcția find_value() cu parametrul preferred_sections. Această abordare face selecția mai robustă dacă structura paginii diferă ușor sau dacă unele câmpuri apar și în alte secțiuni.

```
# Platform: OS + Chipset (prefer Platform)
os_ = find_value({"os"}, preferred_sections=["Platform"])
chipset = find_value({"chipset"}, preferred_sections=["Platform"])
```

4. REZULTATE ȘI EVALUARE

4.1 Fișierele generate și rezultatele obținute

La finalul rulării, aplicația generează 3 fișiere în directorul output, definite în main.py:

```
14  OUT_DIR = "output"
15  OUT_JSON_FULL = os.path.join(OUT_DIR, "phones_full.json")
16  OUT_JSON_KEYS = os.path.join(OUT_DIR, "phones_key_fields.json")
17  OUT_CSV_KEYS = os.path.join(OUT_DIR, "phones_key_fields.csv")
18
```

phones_full.json - conține pentru fiecare telefon:

- brand, model, url, image_url
- toate specificațiile din tabel, grupate pe secțiuni (Network, Display, Platform, Battery etc.)


```
  },
  {
    "url": "https://www.gsmarena.com/xiaomi_13t_pro-12388.php",
    "brand": "Xiaomi",
    "model": "Xiaomi 13T Pro",
    "image_url": "https://fdn2.gsmarena.com/vv/bigpic/xiaomi-13t-pro.jpg",
    "specs": {
      "Network": {
        "Technology": "GSM / HSPA / LTE / 5G",
        "2G bands": "GSM 850 / 900 / 1800 / 1900",
        "3G bands": "HSDPA 800 / 850 / 900 / 1700(AWS) / 1900 / 2100",
        "4G bands": "1, 2, 3, 4, 5, 7, 8, 12, 13, 17, 18, 19, 20, 25, 26, 28, 32, 38, 39, 40, 41, 42, 48, 66",
        "5G bands": "1, 3, 5, 7, 8, 20, 28, 38, 40, 41, 66, 75, 77, 78 SA/NSA",
        "Speed": "HSPA, LTE, 5G"
      },
      "Launch": {
        "Announced": "2023, September 26",
        "Status": "Available. Released 2023, September 26"
      },
      "Body": {
        "Dimensions": "162.2 x 75.7 x 8.5 mm (6.39 x 2.98 x 0.33 in)",
        "Weight": "200 g or 206 g (7.05 oz)",
        "Build": "Glass front (Gorilla Glass 5), glass back or silicone polymer back, plastic frame",
        "SIM": "· Nano-SIM + eSIM · Nano-SIM + Nano-SIM"
      },
      "Display": {
        "Type": "AMOLED, 68B colors, 144Hz, Dolby Vision, HDR10+, 1200 nits (HBM), 2600 nits (peak)",
        "Size": "6.67 inches, 107.4 cm 2 (~87.5% screen-to-body ratio)",
        "Resolution": "1220 x 2712 pixels, 20:9 ratio (~446 ppi density)",
        "Protection": "Corning Gorilla Glass 5"
      },
      "Platform": {
        "OS": "Android 13, up to 4 major Android upgrades, HyperOS",
        "Chipset": "Mediatek Dimensity 9200+ (4 nm)",
        "CPU": "Octa-core (1x3.35 GHz Cortex-X3 & 3x3.0 GHz Cortex-A715 & 4x2.0 GHz Cortex-A510)",
        "GPU": "Immortalis-G715 MC11"
      },
      "Memory": {
        "Card slot": "No",
        "Internal": "256GB 12GB RAM, 512GB 12GB RAM, 1TB 16GB RAM"
      },
      "Main Camera": {
        "Triple": "50 MP, f/1.9, 24mm (wide), 1/1.28\", 1.22µm, PDAF, OIS 50 MP, f/1.9, 50mm (telephoto), 1/2.88\", 0.61µm, PDAF, 2x opt
        "Features": "Leica lens, color spectrum sensor, LED flash, HDR, panorama",
        "Video": "8K@24fps, 4K@24/30/60fps, 4K/1080p@30fps HDR10+, 1080p@30/60/120/240fps; 10-bit LOG, gyro-EIS"
```


phones_key_fields.json - Acest fișier conține doar câmpurile selectate pentru analiză/comparație:

- brand, model, url
- display, chipset, memory
- main_camera, selfie_camera
- battery, charging
- os, price

put > {} phones_key_fields.json > ...

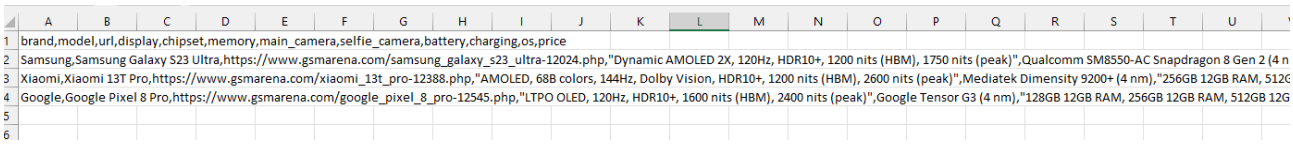

```json
[
  {
    "brand": "Samsung",
    "model": "Samsung Galaxy S23 Ultra",
    "url": "https://www.gsmarena.com/samsung_galaxy_s23_ultra-12024.php",
    "display": "Dynamic AMOLED 2X, 120Hz, HDR10+, 1200 nits (HBM), 1750 nits (peak)",
    "chipset": "Qualcomm SM8550-AC Snapdragon 8 Gen 2 (4 nm)",
    "memory": "256GB 8GB RAM, 256GB 12GB RAM, 512GB 12GB RAM, 1TB 12GB RAM",
    "main_camera": "200 MP, f/1.7, 24mm (wide), 1/1.3\", 0.6µm, multi-directional PDAF, OIS 10 MP, f/2.4, 70mm (tele
    "selfie_camera": "12 MP, f/2.2, 26mm (wide), 1/3.2\", 1.12µm, dual pixel PDAF",
    "battery": "Li-Ion 5000 mAh",
    "charging": "45W wired, PD3.0, 65% in 30 min 15W wireless (Qi) 4.5W reverse wireless",
    "os": "Android 13, up to 4 major Android upgrades, One UI 8",
    "price": "€ 447.48 / $ 437.00 / £ 459.00 / ₹ 89,999"
  },
  {
    "brand": "Xiaomi",
    "model": "Xiaomi 13T Pro",
    "url": "https://www.gsmarena.com/xiaomi_13t_pro-12388.php",
    "display": "AMOLED, 68B colors, 144Hz, Dolby Vision, HDR10+, 1200 nits (HBM), 2600 nits (peak)",
    "chipset": "Mediatek Dimensity 9200+ (4 nm)",
    "memory": "256GB 12GB RAM, 512GB 12GB RAM, 1TB 16GB RAM",
    "main_camera": "50 MP, f/1.9, 24mm (wide), 1/1.28\", 1.22µm, PDAF, OIS 50 MP, f/1.9, 50mm (telephoto), 1/2.88\",
    "selfie_camera": "20 MP, f/2.2, (wide), 0.8µm",
    "battery": "Li-Po 5000 mAh",
    "charging": "120W wired, PD3.0, QC4, 100% in 19 min",
    "os": "Android 13, up to 4 major Android upgrades, HyperOS",
    "price": "€ 287.99 / $ 399.99 / £ 439.67"
  },
  {
    "brand": "Google",
    "model": "Google Pixel 8 Pro",
    "url": "https://www.gsmarena.com/google_pixel_8_pro-12545.php",
    "display": "LTPO OLED, 120Hz, HDR10+, 1600 nits (HBM), 2400 nits (peak)",
    "chipset": "Google Tensor G3 (4 nm)",
    "memory": "128GB 12GB RAM, 256GB 12GB RAM, 512GB 12GB RAM, 1TB 12GB RAM",
    "main_camera": "50 MP, f/1.7, 25mm (wide), 1/1.31\", 1.2µm, dual pixel PDAF, OIS 48 MP, f/2.8, 113mm (periscope
    "selfie_camera": "10.5 MP, f/2.2, 20mm (ultrawide), 1/3.1\", 1.22µm, PDAF",
    "battery": "Li-Ion 5050 mAh",
    "charging": "30W wired, PD3.0, PPS, 50% in 30 min 23W wireless Reverse wireless Bypass charging",
    "os": "Android 14, upgradable to Android 16, up to 7 major Android upgrades",
    "price": "€ 362.00 / $ 299.97 / £ 303.25 / ₹ 62,999"
  }
]
```


phones_key_fields.csv - conține aceleași date ca phones_key_fields.json, dar într-un format potrivit pentru Excel / Google Sheets.

Header-ul generat automat:

brand,model,url,display,chipset,memory,main_camera,selfie_camera,battery,charging,os,price.

Avantajul CSV-ului este că permite compararea rapidă a valorilor între telefoane în mod tabelar.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	brand,model,url,display,chipset,memory,main_camera,selfie_camera,battery,charging,os,price																				
2	Samsung,Samsung Galaxy S23 Ultra,https://www.gsmarena.com/samsung_galaxy_s23_ultra-12024.php,"Dynamic AMOLED 2X, 120Hz, HDR10+, 1200 nits (HBM), 1750 nits (peak)",Qualcomm SM8550-AC Snapdragon 8 Gen 2 (4 n																				
3	Xiaomi,Xiaomi 13T Pro,https://www.gsmarena.com/xiaomi_13t_pro-12388.php,"AMOLED, 68B colors, 144Hz, Dolby Vision, HDR10+, 1200 nits (HBM), 2600 nits (peak)",Mediatek Dimensity 9200+ (4 nm),"256GB 12GB RAM, 512G																				
4	Google,Google Pixel 8 Pro,https://www.gsmarena.com/google_pixel_8_pro-12545.php,"LTPO OLED, 120Hz, HDR10+, 1600 nits (HBM), 2400 nits (peak)",Google Tensor G3 (4 nm),"128GB 12GB RAM, 256GB 12GB RAM, 512GB 12G																				
5																					
6																					

4.2 Observații și evaluare

Pe setul de pagini testate, aplicația a reușit să:

- descarce HTML-ul și să gestioneze redirecționările
- extragă corect modelul, brandul și imaginea principală
- construiască un JSON complet cu specificațiile structurate pe secțiuni
- genereze un set "compact" de câmpuri cheie, util pentru comparații rapide (JSON + CSV)

Un element util pentru verificare este log-ul din consolă, care afișează pentru fiecare pagină modelul, chipset-ul și bateria extrase:

```
63          print(f"    -> model: {key.get('model')}")
64          print(f"    -> chipset: {key.get('chipset')}")
65          print(f"    -> battery: {key.get('battery')}")
66          print("")
```

```
[+] Fetch: https://www.gsmarena.com/samsung_galaxy_s23_ultra-12024.php
    -> model: Samsung Galaxy S23 Ultra
    -> chipset: Qualcomm SM8550-AC Snapdragon 8 Gen 2 (4 nm)
    -> battery: Dynamic AMOLED 2X, 120Hz, HDR10+, 1200 nits (HBM), 1750 nits (peak)
```

Limitări observate

Aplicația depinde de structura HTML a paginilor GSMArena; dacă site-ul modifică structura (selectori / clase / tabele), parserul poate necesita adaptări.

5. CONCLUZII ȘI DIRECȚII VIITOARE

5.1 Concluzii

În cadrul acestui proiect a fost dezvoltată o aplicație de tip web scraper care colectează automat informații despre dispozitive mobile de pe platforma GSMArena. Aplicația a fost realizată în limbajul Python și folosește biblioteci cunoscute pentru accesarea paginilor web, parsarea conținutului HTML și salvarea datelor în formate structurate.

Rezultatele obținute arată că, în lipsa unui API oficial oferit de GSMArena, tehnicile de web scraping pot fi utilizate cu succes pentru accesul programatic la datele publice. Structura relativ constantă a paginilor GSMArena a permis extragerea corectă a informațiilor importante, precum specificațiile ecranului, procesorului, memoriei, camerelor, bateriei și sistemului de operare.

Aplicația reușește să colecteze atât date complete, păstrând structura originală a specificațiilor, cât și un set de câmpuri cheie, utile pentru comparații rapide sau analize ulterioare. Salvarea rezultatelor în formate JSON și CSV face posibilă utilizarea datelor în alte aplicații sau instrumente de analiză.

Prin realizarea acestui proiect au fost parcurși pașii principali necesari dezvoltării unei aplicații de tip scraper, de la analiza sursei de date și identificarea elementelor relevante din paginile web, până la implementarea unei soluții funcționale și testarea rezultatelor obținute.

5.2 Direcții viitoare de dezvoltare

Aplicația dezvoltată poate fi îmbunătățită și extinsă în mai multe direcții. O posibilă dezvoltare viitoare este automatizarea colectării listelor de telefoane direct din paginile de brand GSMArena, fără a mai fi nevoie de introducerea manuală a URL-urilor.

De asemenea, aplicația ar putea fi extinsă pentru a analiza un număr mai mare de dispozitive și pentru a include mecanisme mai avansate de tratare a erorilor, cum ar fi reluarea automată a cererilor eșuate sau adaptarea la modificări ale structurii paginilor web.

O altă direcție posibilă este integrarea rezultatelor într-o aplicație web sau desktop, care să permită compararea telefoanelor într-o interfață vizuală. În plus, datele colectate ar putea fi utilizate pentru analize statistice sau studii comparative între diferite modele sau generații de dispozitive mobile.

BIBLIOGRAFIE

[1] GSMArena, “Mobile Phone Specifications Database,” disponibil online: https://www.gsmarena.com/

[2] Python Software Foundation, “Python Documentation,” disponibil online: https://docs.python.org/3/

[3] Kenneth Reitz, “Requests: HTTP for Humans,” documentație oficială, disponibil online: https://requests.readthedocs.io/

[4] Leonard Richardson, “Beautiful Soup Documentation,” disponibil online: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[5] Mozilla Developer Network (MDN), “Introduction to HTML,” disponibil online: https://developer.mozilla.org/en-US/docs/Web/HTML

[6] nordmarin, “gsmarena-api,” GitHub repository, disponibil online: https://github.com/nordmarin/gsmarena-api

[7] RapidAPI, “GSM Arena Parser API,” disponibil online: https://rapidapi.com/controller2042000/api/gsmarenaparser

[8] PhoneDB, “Phone Specifications Database,” disponibil online: https://phonedb.net/

[9] Python Software Foundation, “csv — CSV File Reading and Writing,” disponibil online: https://docs.python.org/3/library/csv.html

[10] Python Software Foundation, “json — JSON encoder and decoder,” disponibil online: https://docs.python.org/3/library/json.html

[11] Oxani Corina, “GSMArena Web Scraper – GitHub Repository,” disponibil online: https://github.com/CorinaOxani/gsmarena-scraper