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

[CUPRINS 2](#_Toc219282305)

[1. DESCRIEREA PROIECTULUI 3](#_Toc219282306)

[1.1 Contextul și motivația realizării proiectului 3](#_Toc219282307)

[1.1 Analiza sursei de date GSMArena 3](#_Toc219282308)

[1.2 Obiectivele aplicației dezvoltate 6](#_Toc219282309)

[2. MEDIUL DE DEZVOLTARE ȘI TEHNOLOGII UTILIZATE 7](#_Toc219282310)

[2.1 Limbajul de programare: Python 7](#_Toc219282311)

[2.2 Biblioteci și tehnologii utilizate 7](#_Toc219282312)

[2.3 Structura aplicației dezvoltate 8](#_Toc219282313)

[2.4 Versionarea codului și repository-ul Git 8](#_Toc219282314)

[3. PROIECTAREA ȘI IMPLEMENTAREA APLICAȚIEI 9](#_Toc219282315)

[3.1 Fluxul de execuție al aplicației 9](#_Toc219282316)

[3.2 Modulul de colectare a datelor 10](#_Toc219282317)

[3.3 Modulul de parsare și extracție a informațiilor 11](#_Toc219282318)

[3.4 Selectarea câmpurilor cheie 12](#_Toc219282319)

[4. REZULTATE ȘI EVALUARE 13](#_Toc219282320)

[4.1 Fișierele generate și rezultatele obținute 13](#_Toc219282321)

[4.2 Observații și evaluare 16](#_Toc219282322)

[5. CONCLUZII ȘI DIRECȚII VIITOARE 17](#_Toc219282323)

[5.1 Concluzii 17](#_Toc219282324)

[5.2 Direcții viitoare de dezvoltare 17](#_Toc219282325)

[BIBLIOGRAFIE 19](#_Toc219282326)

# DESCRIEREA PROIECTULUI

## 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.

## 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.

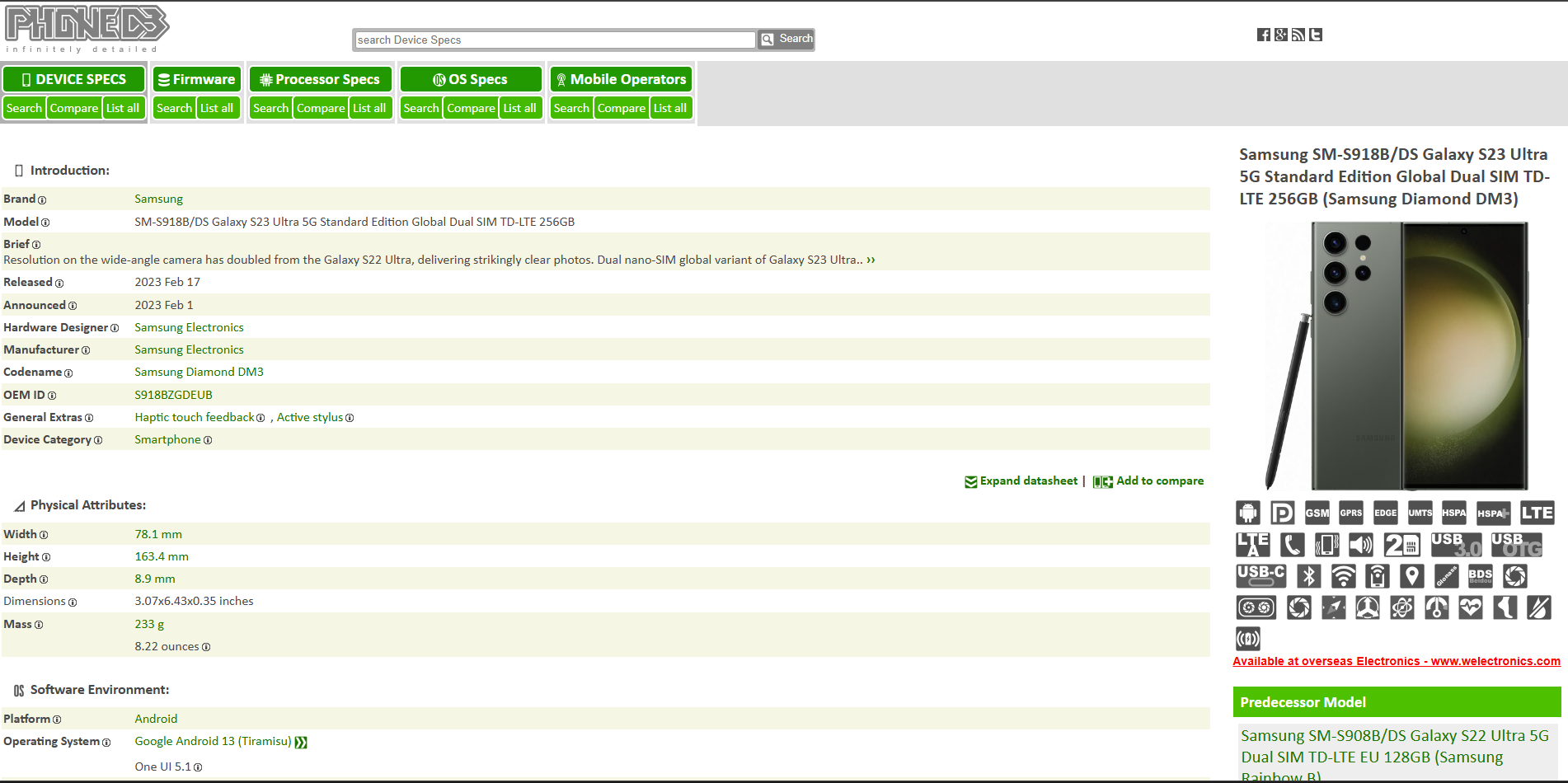
Analiza structurii paginilor GSMArena a arătat că informațiile sunt organizate într-un mod clar și relativ constant. Fiecare brand are o pagină de listă cu modelele disponibile, iar fiecare model are o pagină individuală unde specificațiile sunt structurate în tabele, pe secțiuni precum Network, Display, 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.

## ****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.

# MEDIUL DE DEZVOLTARE ȘI TEHNOLOGII UTILIZATE

## 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]

## 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.

## 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.

## 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]

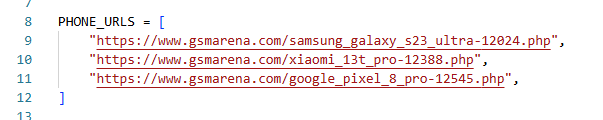
# PROIECTAREA ȘI IMPLEMENTAREA APLICAȚIEI

## 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:

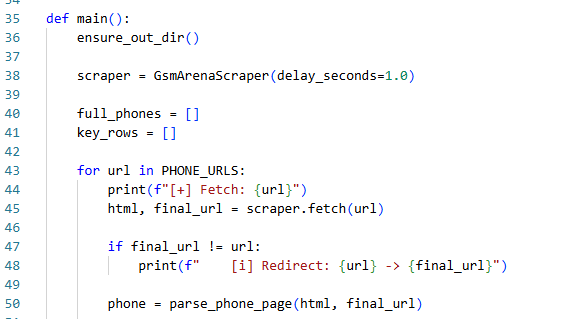
* colectare
* parsare
* selecție câmpuri
* export

Lista de pagini analizate este definită explicit în cod:

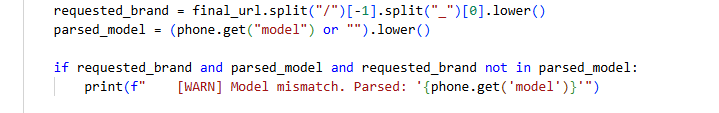


În funcția main(), aplicația:

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



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

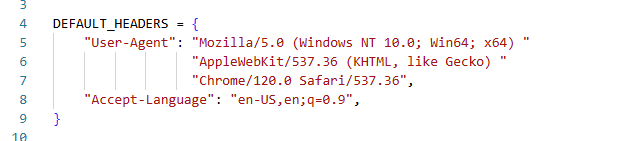


Datele complete și cele filtrate sunt adunate în două liste separate.

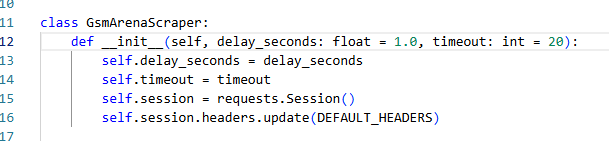
## Modulul de colectare a datelor

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.

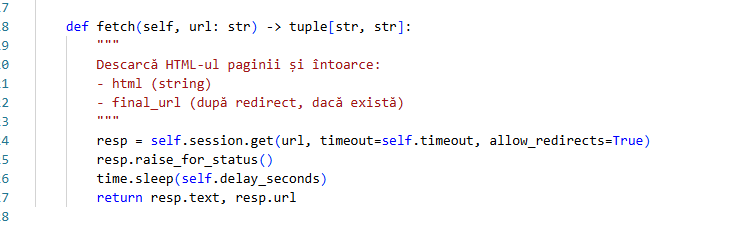
Pentru a imita comportamentul unui browser real, se folosește un User-Agent și Accept-Language, setate în headere:



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



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



Rezultatul acestei etape este tuplul:

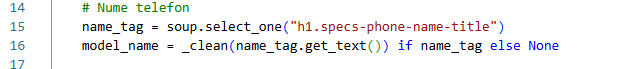
* + resp.text (HTML-ul paginii)
  + resp.url (URL-ul final, după redirect, dacă există)

## Modulul de parsare și extracție a informațiilor

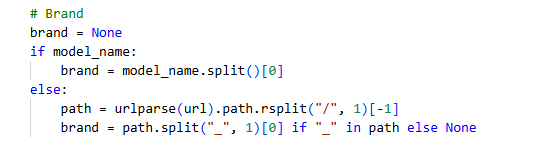
Parsarea este implementată în parser.py folosind BeautifulSoup. Scopul funcției parse\_phone\_page() este să extragă:

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

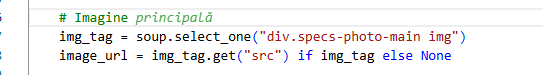
Extragerea numelui modelului se face din titlul paginii (h1.specs-phone-name-title):



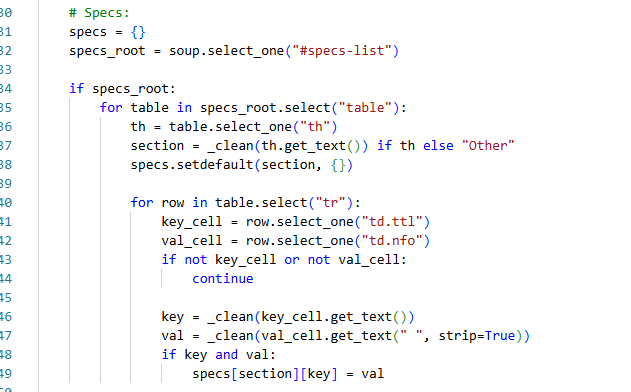
Brandul este dedus fie din primul cuvânt al modelului, fie din URL (fallback):



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



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

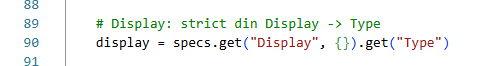


Rezultatul final al acestei funcții este o structură de tip dicționar, cu date generale împreună cu specificațiile pe secțiuni.

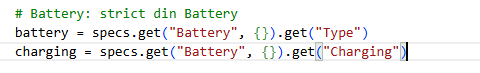
## Selectarea câmpurilor cheie

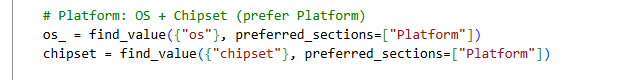
După ce pagina este parsată și toate specificațiile sunt salvate pe secțiuni (Display, Platform, Battery etc.), aplicația extrage un set mai mic de informații „esențiale”, utile pentru comparații rapide între telefoane. Această selecție este realizată în funcția pick\_key\_fields() din fișierul parser.py.

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   


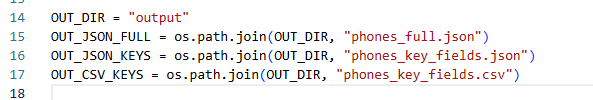
Bateria este luată strict din Battery, ca să nu fie confundată cu „Type” din alte secțiuni:

  
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.



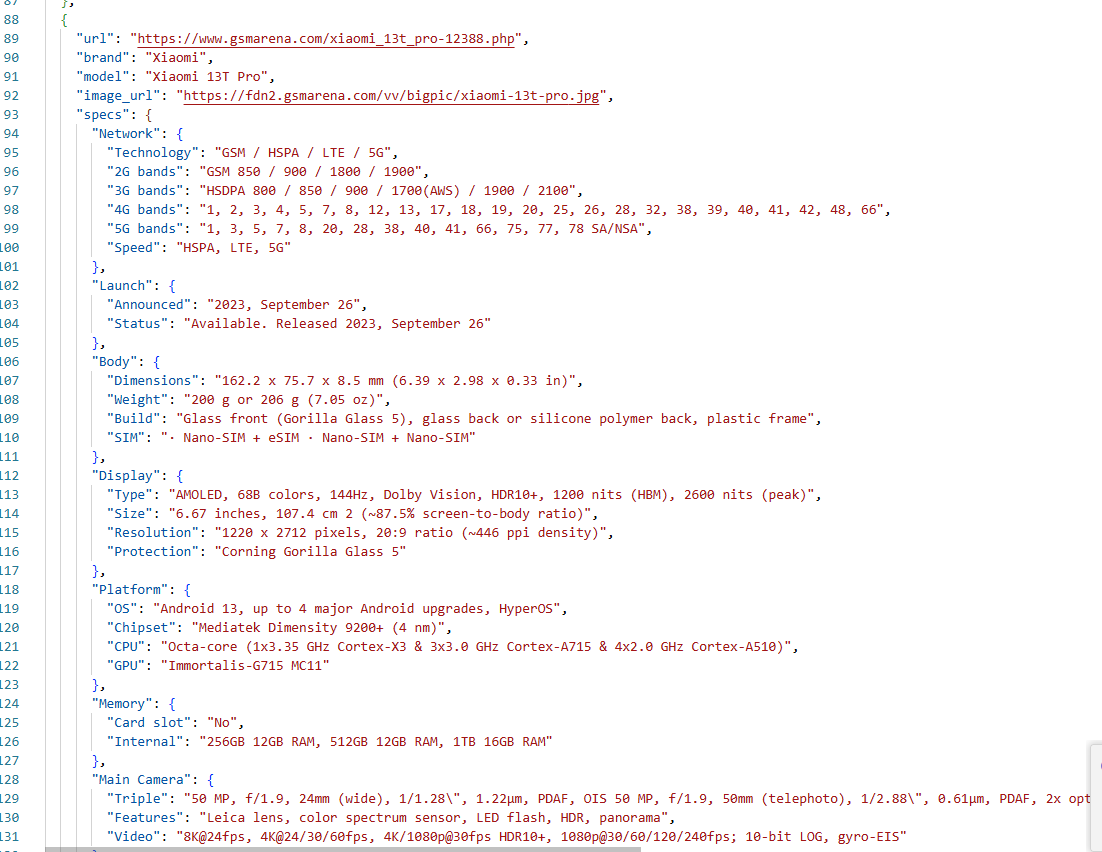
# REZULTATE ȘI EVALUARE

## Fișierele generate și rezultatele obținute

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

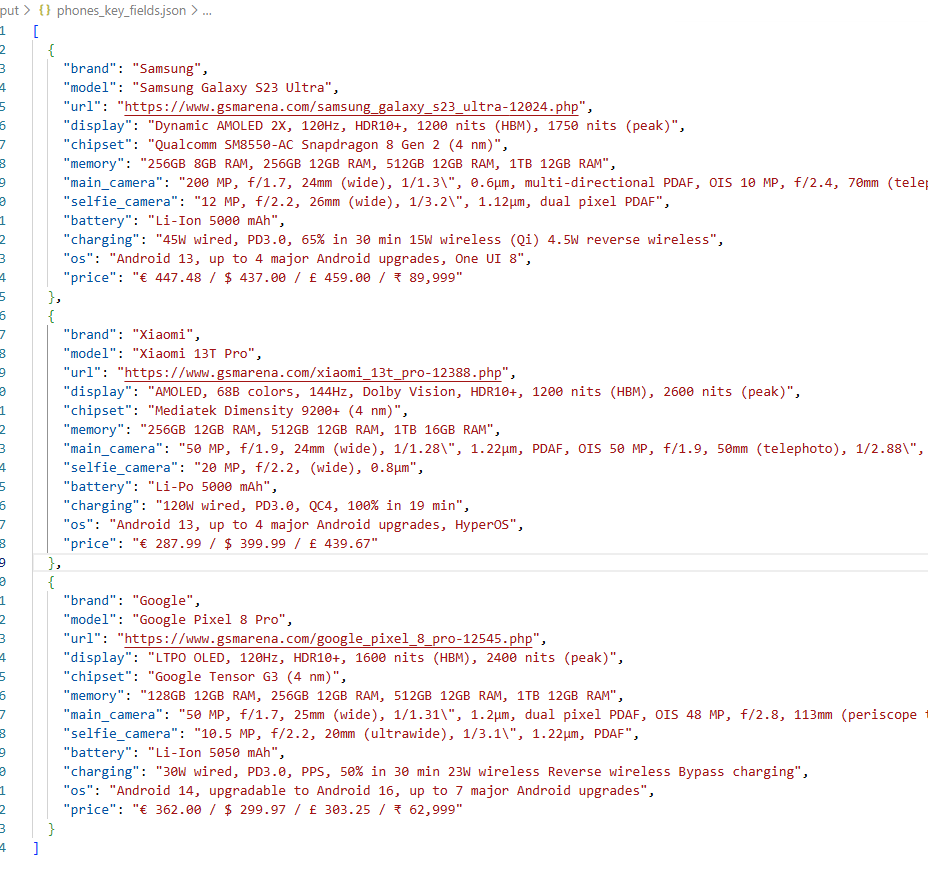
phones\_full.json - conține pentru fiecare telefon:

* brand, model, url, image\_url
* toate specificațiile din tabel, grupate pe secțiuni (Network, Display, Platform, Battery etc.)



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

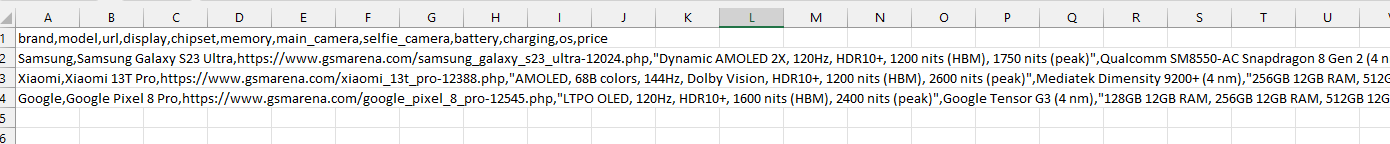


phones\_key\_fields.csv - conține aceleași date ca phones\_key\_fields.json, dar într-un format potrivit pentru Excel / Google Sheets.

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.

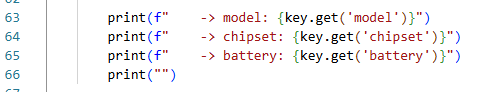


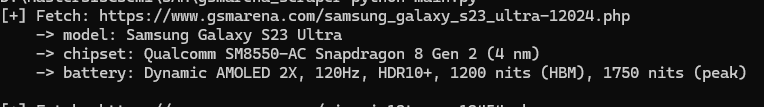
## Observații și evaluare

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:





**Limitări observate**

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

# CONCLUZII ȘI DIRECȚII VIITOARE

## 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.

## 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>