# Prezentare generala

Task**:**

- Extracts company info from websites.

- Merges with a company name dataset.

- Indexes the profiles in Elasticsearch.

- Accepts API requests with partial info and returns the best matching company profile.

Deczii**:**

->Am decis sa folosesc python datorita numeroaselor librarii pe care le-as putea utiliza in cod dar si datorita experientei mele in a construii pipelineuri Airbyte in trecut

->Am folosit BeautifulSoup pentru parsarea documentelor html in fisierele crawler

->In primul fisier crawler\_p1 am folosit geturi sincrone apoi in urmatoarele fisiere crawler\_p2, crawler\_p2\_improved, crawler\_p2\_improvedfallback am folosit geturi asyncrone si am modificat timeoutul de get sau cate requresturi in parallel pot rula. Ajustarea finala a fost sa folosesc 20 timeout si 25 requesturi in parallel.

->In fisierul analyse.py se printeaza pentru fiecare crawler: coverage of websites si procentajul datapointurilor extrase.

->Se va folosii pentru API doar crawler\_p2\_improved\_fallback acesta face de asemenea un ‘fallback’ in caz ca https nu functioneaza pentru ca nu exista certificate SSL sau alte cause.

->Am folosit elasticsearch pentru a crea indexul de profil companies cu numele , telefonul , contul facebook , numele si websietul. Elasticsearch ruleaza intr-un container docker am luat aceasta decizie deoarece nu am vrut sa instalez si sa configurez local.

->Folosesc FASTAPI pentru a expune un endpoint /match care primește date despre companie (nume, website, telefon, Facebook) și returnează compania cea mai apropiată din Elasticsearch (indexul de profil), folosind un algoritm de potrivire bazat pe scoruri de similaritate.

->Pentru a face matching folosesc algoritmul RapidFuzz . Algoritmul de potrivire compară datele de intrare ale unei companii (nume, website, telefon, Facebook) cu documentele existente în Elasticsearch, folosind biblioteca RapidFuzz pentru a calcula scoruri de similaritate textuală. Fiecare câmp are o pondere specifică în calculul scorului total: numele companiei (40%), website-ul (30%), telefonul (20%) și pagina de Facebook (10%). Pentru fiecare document, se calculează un scor ponderat pe baza acestor câmpuri, iar algoritmul returnează compania cu cel mai mare scor, considerată cea mai bună potrivire.

## Requirements to run

**Install all dependencies:**

```bash

pip install -r requirements.txt

```

**Need \*\*Elasticsearch\*\* running locally:**

docker run -d --name elasticsearch \

-p 9200:9200 -e "discovery.type=single-node" \

-e "xpack.security.enabled=false" \

docker.elastic.co/elasticsearch/elasticsearch:8.11.3

Rulare script run\_main  
Acest script rulează pașii de îmbinare, indexare, scraping și API matching.

┌────────────────────────────────────────┐

│ run\_main.py │

└────────────────────────────────────────┘

│

▼

[0] Start Elasticsearch în Docker

│

▼

[1] Rulează crawler:

├── crawler\_p2\_improved\_fallback.py

└── analyze.py

│

▼

[2] Mergere date:

└── merge\_to\_api\_format.py

│

▼

[3] Indexare Elasticsearch:

└── index\_api\_input\_format.py

│

▼

[4] Pornește API FastAPI în fundal:

└── final\_company\_match\_api.py

│

▼

[5] Testează API-ul:

└── test\_api\_batch.py

│

▼

**Done + rezultate în `api\_matching\_results.csv`**

## Descriere rulare step by step

1. **Start docker**

docker start elasticsearch || docker run -d --name elasticsearch ^

-p 9200:9200 -e "discovery.type=single-node" ^

-e "xpack.security.enabled=false" ^

docker.elastic.co/elasticsearch/elasticsearch:8.11.3

1. **Running crawler to scrape data...**

python "%BASE\_PATH%\crawler\_p2\_improved\_fallback.py"

1. **Merging scraped data into API input format...**

python "%BASE\_PATH%\merge\_to\_api\_format.py"

1. **Indexing data into Elasticsearch...**

python "%BASE\_PATH%\index\_api\_input\_format.py"

1. **Starting FastAPI server in new terminal...**

start cmd /k "cd %BASE\_PATH% && uvicorn final\_company\_match\_api:app --reload"

1. **Running test script...**

python "%BASE\_PATH%\src\test\_api\_batch.py"

* **Output in`api\_matching\_results.csv`**

## Example Usage API

### Request:

```json

POST /match

{

"name": "Acme Inc.",

"website": "acme.com",

"phone": "+1234567890",

"facebook": "facebook.com/acme"

}

```

### Response:

```json

{

"input": {

"name": "Acme Inc.",

"website": "acme.com",

"phone": "+1234567890",

"facebook": "facebook.com/acme"

},

"best\_score": 92.0,

"best\_match": {

"input name": "ACME Incorporated",

"input phone": "+1234567890",

"input website": "acme.com",

"input\_facebook": "facebook.com/acme"

}

}

```

## Descriere fisiere python

| **Script** | **Descriere** |
| --- | --- |
| crawler\_p1.py | Versiune inițială de crawler sincron pentru extragerea datelor de contact de pe site-uri. |
| crawler\_p2.py | Variantă îmbunătățită asincronă a crawlerului cu aiohttp și BeautifulSoup. |
| crawler\_p2\_improved.py | Variantă asincronă cu mai puține requesturi în paralel pentru stabilitate (semaphore=25). |
| crawler\_p2\_improved\_fallback.py | Crawler robust cu fallback HTTPS → HTTP, extrage telefoane și linkuri sociale. Este folosit în pipeline. |
| analyze.py | Analizează datele extrase (ex. număr de rezultate lipsă, distribuții etc.). |
| merge\_to\_api\_format.py | Îmbină fișierul cu datele extrase cu cel ce conține numele companiilor → pregătește datele finale pentru API. |
| index\_api\_input\_format.py | Indexează fișierul de profiluri de companii în Elasticsearch pentru a permite căutarea și potrivirea. |
| final\_company\_match\_api.py | Server FastAPI care expune endpointul /match pentru a face matching între input și companiile indexate. |
| test\_api\_batch.py | Trimite în batch date către API și salvează răspunsurile în api\_matching\_results.csv. |
| run\_main.py | Rulează întregul pipeline (scraping, analiză, merge, indexare, pornire API, testare API). |

## Descriere fisiere CSV

| **Fișier CSV** | **Scop** |
| --- | --- |
| sample-websites.csv | Domeniile web care trebuie crawl-uite (un domeniu per linie). |
| scraped\_data\_async\_improved.csv | Output din crawler: conține domenii, numere de telefon și linkuri sociale. |
| sample-websites-company-names.csv | Conține perechi domain - company\_name pentru potrivirea cu datele extrase. |
| company\_profiles\_for\_api.csv | Output rezultat din merge\_to\_api\_format.py, gata de indexare în Elasticsearch. |
| API-input-sample.csv | Setul de companii care trebuie potrivite cu cele din Elasticsearch. |
| api\_matching\_results.csv | Outputul testării API-ului, cu rezultatele potrivirii pentru fiecare rând din API-input-sample.csv. |