**Documentatie API**

**Sava Catalin-Andrei**

Pentru a aborda problema propusa, am ales sa impart sarcinile de rezolvat in 3 parti, cu cate un fisier .js pentru fiecare dintre ele.

Am inceput cu fisierul “scraper.js”, care are scopul de a face web scraping pe toate url-urile aflate in csv-ul dat. Am folosit “csv-parser” pentru a putea citi si interpreta fisierul cu domeniile de input. Apoi am incercat sa accesez aceste domenii. Prima data am facut un request catre domeniile din fisier dupa ce am concatenat cu “ https:// ” utilizand “axios” si am determinat daca website-ul nu exista (eroare 404) sau daca avem eroare de ssl/certificate expirat. Website-urile inexistente sunt numarate utilizand variabila “notExistCount”, iar in cazul erorii de certificat vom incerca si cu “ http:// “ cu un insecure agent.

Mai departe, voi normaliza html-ul obtinut si extrage datele dorite:

* Numerele de telefon sunt extrase utilizand “libphonenumber-js”, iar ca fallback am folosit regex pentru a asigura identificarea numerelor de telefon in format variate.
* Link-urile de social media sunt obtinute prin identificarea link-urilor care contin url-urile date ale platformelor de social media (facebook, Instagram, linkedin etc.)
* Adresele sunt identificate utilizand 3 expresii regex diferite deoarece am observat aceste 3 tipare recurente in paginile analizate. Mai mult de una din aceste verificari poate returna un rezultat, asa ca la final se elimina duplicatele (cu normalizare)

In cazul in care pot accesa website-ul, dar nu obtin date deloc, folosesc ca si fallback “puppeteer” headless browser, pentru a putea extrage date si din acele website-uri care sunt javascript heavy, precum React, Next.js etc.

De asemenea, pentru a obtine si mai multe date relevante, pe langa scraping-ul homepage-urilor, voi cauta si link-uri catre pagini de contact. Daca nu se gasesc, voi incerca sa accesez rute comune pentru aceste pagini (de ex. “/contact”, “/about”, “/about-us” etc).

Fiecare datapoint este masurat dupa numarul de obiecte dintr-un array, daca un array este gol se considera ca acel datapoint nu a putut fi extras. Metrica “percentage” obtinut reflecta ce procentaj din aceste 3 datapoints nu sunt goale (au fost extrase).

Pentru a avea o solutie scalabila si rapida, am folosit “p-limit” pentru a rula functii async concurente si a elimina bottleneck-ul.

La final, returnez niste date analitice care arata numarul de website-uri care au putut fi accesate, care nu au putut fi accesate, care au fost accesate, dar nu au continut date relevante si care nu exista.

Datele au fost stocate intr-un fisier NDJSON, pentru a putea diferentia obiectele de pe fiecare rand.

La urmatoare parte a problemei, am abordat merging-ul de date cu csv-ul dat si apoi indexarea in ElasticSearch.  
Am rulat un container in Docker de ElasticSearch si am creat o conexiune la acesta.

Am accesat csv-ul cu datele obtinute din scraping si csv-ul cu datele furnizate. Am normalizat domeniile si am combinate datele. Dupa care obiectul rezultat a fost indexat in ElasticSearch.

Pentru ultima parte, am creat api-ul in Express, unde am declarant un endpoint “/match” care primeste {name, website, phone, facebook} in body-ul requestului. Am creat un array “should” si am adaugat query-uri “match” in functie de ce am primit in body. Iar la final am facut un query “bool” pe array-ul “should”, care functioneaza ca un SAU logic intre conditii. Cu cat mai multe campuri fac match, cu atat scorul de acuratete va fi mai mare.

Pentru a testa endpoint-ul, se poate face un request la http://localhost:3000/match cu body-ul JSON:  
{

  "name": "Acorn Law P.C.",

  "website": "",

  "phone": "",

  "facebook": "”

}

Rezultatul este de forma:  
{

    "score": 55.073273,

    "profile": {

        "domain": "acornlawpc.com",

        "company\_commercial\_name": "Acorn Law P.C.",

        "company\_legal\_name": "Acorn Law P.C.",

        "company\_all\_available\_names": "Acorn Law P.C. | Acorn Law | Acorn Law P",

        "url": "https://acornlawpc.com",

        "phones": [

            "+18054093878"

        ],

        "socialMedia": [

            "https://www.facebook.com/acornlaw/"

        ],

        "addresses": [

            "310 N. Westlake Blvd, Suite 100Westlake Village, CA 91362",

            "30 p.m.Westlake Village Office310 N. Westlake Blvd, Suite 100Westlake Village, CA 91362"

        ],

        "counts": {

            "phones": 1,

            "socialMedia": 1,

            "addresses": 2

        },

        "percentage": "100.0%"

    }

}

Perspective de dezvoltare:

* Organizarea intr-un pipeline si orchestrarea pentru un flux automat, scalabil si robust
* Integrarea unor servere de proxy de pe care se pot face request-urile blocate de anumite website-uri impotriva scraping-ului
* Message Queue intre scraper si indexer
* Extragere de adrese si numere de telefon globale, nu doar US
* Dashboard