UNIVERSITÀ DEGLI STUDI DI BERGAMO

Dipartimento di
Ingegneria Gestionale, dell'Informazione e della Produzione

Corso di laurea in
Ingegneria Informatica

Classe n. L-8 – Ingegneria dell'informazione

Soluzione web scraping scalabile e in cloud: per un'acquisizione dei dati più semplice ed efficace

Candidato:
Alessandra Cortinovis

Matricola n.
1073344

Relatore:
Chiar.mo Prof. Mauro Pelucchi

Anno Accademico
2023/2024

INDICE

Abstract

Con web scraping si indica l'estrazione dei dati dalle pagine web, che una volta raccolti e messi in database o tabelle, possono essere analizzati e utilizzati per diversi scopi. Uno dei più semplici metodi di raccolta dati è l'elementare copia-incolla, ma visto il numero di risorse e pagine web che si potrebbero andare ad analizzare (quasi due miliardi di siti web, di cui cinquecento milioni attivi) sarebbe laborioso e dispendioso.

In questo documento si andrà infatti a progettare e implementare una soluzione per il web scraping, scalabile e basata in cloud, per facilitare e rendere più efficiente l'acquisizione dei dati.

Il progetto si concentrerà sull'implementazione di una architettura serverless e cloud agnostic, capace di gestire grandi volumi di dati provenienti da diverse fonti. Per garantire scalabilità e affidabilità, verranno usate risorse di cloud computing per gestire l'estrazione dei dati.

Il risultato finale sarà una soluzione di scraping robusta, ospitata su AWS (Amazon Web Services), piattaforma di Amazon che offre servizi in cloud per la creazione e gestione di applicazioni, dotata di funzionalità per la scalabilità e facilità di gestione.

Capitolo 1

Introduzione

Con l'avvento del World Wide Web e grazie alla crescita e avanzamento tecnologico, c'è stato un aumento esponenziale di richieste di informazioni e, di conseguenza, di strumenti per mettere a disposizione tali informazioni.
Spesso il primo passo è andare sul motore di ricerca preferito e cliccare sul primo link che sembra offrire la soluzione ai nostri problemi, ma in un mondo dove la domanda e l'offerta di informazioni è in costante aumento, risulta sempre più difficile raccogliere tutti i dati, o almeno la maggior parte, in maniera veloce ed efficace.
Una delle soluzioni la possiamo trovare nell'uso del web scraping, una delle tante tecniche di estrazione dei dati dalle pagine web, se non quella più efficace e veloce per la raccolta delle informazioni, visto soprattutto il numero crescente di fonti e pagine da analizzare, quasi due miliardi di siti di cui cinquecento milioni attivi al 2022 (*Libero Tecnologia*): un semplice copia incolla da tempo non è il metodo più appropriato per salvare e immagazzinare dati nel minor tempo possibile.

1.1 Storia del web scraping

Il desiderio e la crescente necessità di condividere informazioni sempre più velocemente tra le varie università e istituti, ha portato verso la fine del XX secolo al bisogno di creare un sistema di rete universale proprio con lo scopo di poter scambiare risorse tra i diversi enti.

Si sviluppano allora diversi strumenti, tra cui ARPANET: progetto finanziato dall'Agenzia per i Progetti di Ricerca Avanzata (ARPA), è stata la prima connessione vera e propria tra i computer di diverse università nordamericane a chilometri di distanza e rappresenta un primo passo per la nascita dell'internet come lo conosciamo oggi.

Grazie alla sua evoluzione e diffusione oltreoceano, nel 1989 nel CERN, Tim Berners-Lee propone un sistema per evitare la perdita di informazioni, salvandole in un sistema distribuito. Da questa idea nasce il World Wide Web, e con la conseguente creazione del primo sito web due anni dopo, nel 1991, si sono poste le basi per gli elementi analizzati nel web scraping: gli URL dei siti specifici, gli hyperlink e i vari dati (testo, immagini, audio, video, ecc.) all'interno delle pagine. Il web scraping non sarebbe esistito senza il World Wide Web.

Infatti, uno dei primi web crawler, The Wanderer[1] di Matthew Gray, nasce poco dopo nel 1993. Nonostante il suo scopo fosse soltanto misurare la dimensione della rete, è stato il punto di partenza per lo sviluppo degli strumenti come JumpStation[2], il primo motore di ricerca basato sul web crawler, nato lo stesso anno, che ha effettivamente reso l'internet una piattaforma open-source e ha reso possibile la realizzazione dei motori di ricerca più usati, come Google, Bing e Yahoo.

Nel 2004 si inizia sfruttare la struttura basata sul linguaggio HTML (Hypertext Markup Language) per estrarre i dati dai siti web utilizzando BeautifulSoup[3], un parser HTML costruito con librerie scritte nel linguaggio di programmazione Python. La facilità con cui chiunque in possesso di un computer e una connessione internet potesse estrarre informazioni, ha spinto al desiderio di ottenere tutte quelle disponibili e ha portato alla necessità di accelerare la raccolta dei dati e di conseguenza alla sua automatizzazione.

Nasce così il web scraping come lo conosciamo oggi: uno strumento per raccogliere dati e salvare le informazioni raccolte in tabelle, facile da usare e accessibile a tecnici e non tecnici. (*webscraper.io, 2021*)

---

[1] *World Wide Web Wanderer*, https://en.wikipedia.org/wiki/World_Wide_Web_Wanderer, *Brief History of Web Scraping*, https://webscraper.io/blog/brief-history-of-web-scraping [12]

[2] *JumpStation*, https://en.wikipedia.org/wiki/JumpStation, *Brief History of Web Scraping*, https://webscraper.io/blog/brief-history-of-web-scraping [12]

[3] *BeautifulSoup Documentation*, https://www.crummy.com/software/BeautifulSoup/bs4/doc/, *Brief History of Web Scraping*, https://webscraper.io/blog/brief-history-of-web-scraping [12]

1.2 Web scraping: definizione

Con web scraping si intende una tecnica di estrazione dei dati dalle pagine web, ed è una delle più efficaci per tale scopo.

Il web scraping è una tecnica particolare di crawling. Come visto prima, un crawler è un software che raccoglie tutte le informazioni per indicizzare in automatico le pagine di un sito, analizza i link ipertestuali presenti e classifica i termini di ricerca, che viene utilizzato dalla maggior parte dei motori di ricerca. (*Fumagalli, 2021*)

Il web scraping, pur essendo correlato all'indicizzazione del crawler, si concentra di più sulla trasformazione dei dati non strutturati presenti nella rete in metadati che possano essere salvati in tabelle locali o database e successivamente analizzati. La raccolta dei dati avviene attraverso la simulazione dell'attività umana sul web facendo richieste HTTP (Hypertext Transfer Protocol). (*Wikipedia, Web Scraping*)

Questa tecnica può essere utilizzata per diversi scopi e in diversi campi, grazie principalmente al fatto che gli strumenti di web scraping possono essere realizzati con linguaggi di programmazione differenti e messi a disposizione sotto forma di applicazioni o estensioni per il browser.

1.3 Perché si fa web scraping

Web scraping può essere utilizzato per molteplici scopi (*Sirisuriya, 2015*), tra cui:

- Raccolta dati;
- Controllo prezzi su siti di e-commerce;
- Indicizzazione motori di ricerca;
- Monitoraggio dati metereologici;
- Rilevamento cambiamenti in siti web;
- Integrazione di molteplici risorse;
- Raccolta di offerte e sconti;
- Analisi di mercato;

- Ottenimento dettagli aziendali, da pagine gialle per esempio;
- Ricerca;
- Raccolta informazioni dei listini di vendita del mercato immobiliare.

Anche organizzazioni internazionali usano tecniche di scraping per migliorare le loro policy, un esempio è il Web Intelligence Hub di EUROSTAT (*https://www.wih-con.eu/en/index*).

1.4 Obbiettivi del web scraping

L'obbiettivo principale di un sistema di web scraping è rendere la raccolta dei dati e il loro salvataggio in tabelle o database più semplice e meno laborioso, soprattutto per grandi quantitativi di dati, quindi estrarli dai siti e trasformarli in un formato leggibile e comprensibile.

1.5 Etica del web scraping

Visto l'utilizzo del web scraping per la raccolta di dati, analisi di mercato e indagini statistiche, non è improbabile che venga adoperato, per esempio, con lo scopo di acquisire vantaggi competitivi da aziende che vogliono saper in tempo reale le strategie dei loro concorrenti, i loro prezzi e i loro prodotti.

Il web scraping è legale solo nei casi in cui i siti che vengono utilizzati siano liberamente accessibili e adoperabili per scopi statistici o di analisi. Nei casi in cui si faccia web scraping per ottenere informazioni, per scopi commerciali o di lucro, all'insaputa e senza il consenso dell'utente, o si raccolgano dati protetti da privacy, il web scraping viene considerato illegale. (*Densmore, Ethics in Web Scraping, 2017*)

Alcune attività di scraping non etiche sono:

- Raccolta di dati personali e dati sensibili da siti o risorse online, come dati bancari, medici o persino credenziali di accesso, che non devono mai essere diffusi pubblicamente e sono protette da leggi che garantiscono la privacy degli utenti.

- Estrazione di dati con diritto d’autore, come testo, immagini, foto e video.
- Utilizzo di web scraping per raccogliere informazioni sensibili con lo scopo di utilizzarle per attuare attività di phishing o spam.
- Raccolta di dati da siti di attività concorrenti per ottenere vantaggi competitivi.

Il web scraper etico deve quindi estrarre solo i dati necessari e da siti che hanno un’area pubblica o che rendono disponibili APIs, rendere le proprie intenzioni chiare quando effettua le richieste di scraping ed estrarre i dati a una velocità ragionevole, senza intasare il traffico e portare a pericoli, non passare i dati estratti come se fossero i propri e creare valore con i dati raccolti.

Capitolo 2

Stato dell'arte

Quando ci sono informazioni che si vogliono ottenere da un qualunque sito web si può fare da sé, estraendo i dati manualmente, se si è più esperti si può pensare di sviluppare un programma che lo faccia al posto nostro, oppure più semplicemente usare uno strumento di web scraping già pronto all'uso.

Esistono diverse strategie per effettuare il web scraping, ognuna specifica per la sua area di specializzazione. Queste strategie combinate a soluzioni per una migliore gestione della grande mole di dati presenti nel web sono alla base del funzionamento di un sistema di web scraping.

2.1 Metodi di web scraping

I metodi per eseguire il web scraping si sono evoluti nel tempo, migliorandosi e adattandosi alle esigenze attuali.

Esistono diversi metodi di web scraping, tra cui:

- Hypertext Markup Language Document Object Model (HTML DOM) parsing, dove la struttura nelle pagine web rende facile l'individuazione dei dati da estrarre, specialmente se ci si basa sulla libreria fornita da schema.org, che permette una ricerca più dettagliata dei dati grazie ai suoi tag specifici;
- XPath, per analizzare pagine e documenti contenenti XML;
- API (Application Programming Interface), che permettono di accedere ai siti e estrapolare anche dati che non sono strutturati con gli standard HTML, rispondere a richieste http tipiche restituendo risultati strutturati (es. CSV o XML).
- scraping manuale, da usare nei casi in cui il numero di dati da raccogliere è esiguo, specialmente se la raccolta stessa richiede meno tempo della creazione di

un processo di scraping automatizzato, o se la struttura delle pagine non permette l'automazione della raccolta dei dati. (*Sharma et al., 2023*)

2.2 Approcci al web scraping

Vista la dimensione dell'internet, sorge la necessità di riuscire a gestire il grande quantitativo di dati da raccogliere, sia in termini di dove andare a salvare questi dati, che di trovare un modo per non impattare negativamente sulle risorse messe a disposizione per effettuare l'attività di scraping.

2.2.1 Scraping in cloud

Si potrebbe pensare che, quando si fa web scraping sia meglio salvare i dati estratti in un database locale, per avere un accesso più immediato alle informazioni e soprattutto per averle a portata di mano e non correre nel rischio che vengano perse. In realtà questo risultare difficile quando la quantità di dati raccolti aumenta e rischia di occupare la maggior parte dello spazio di memoria del dispositivo che sta effettuando l'attività di scraping.

Quindi per rendere più efficiente la raccolta dati, senza dipendere sulla performance del proprio dispositivo, è consigliato basare la propria applicazione di web scraping in cloud, come fanno la maggior parte degli strumenti di scraping. Infatti, questa soluzione non solo permette di risparmiare tempo e concentrarsi sulla raccolta dei dati, proteggendo al contempo la privacy del cliente (*Kumar, 2024*), ma permette anche la ripetizione, anche all'infinito, della richiesta di scraping e di effettuare queste richieste contemporaneamente su più siti.

2.2.2 Web scraping e big data

Un'altra priorità per i servizi di web scraping è la gestione di dati con grandi volumi e strutture differenti. Come scrive infatti Plamen Milev in *Conseptual Approach for Development of WEB Scraping Application for Tracking information* (2017),

indipendentemente dalla diversità delle varie di strutture di dati, l'approccio concettuale deve essere uniforme per tutti i sistemi e deve includere i seguenti passi:

- definizione della fonte, ovvero dei siti richiesti;
- analisi delle fonti, assicurandosi che i siti possano essere raschiati;
- estrazione dei dati, salvandoli nella loro forma originale;
- salvataggio dei dati estratti in un database temporaneo;
- trasformazione dei dati in un formato specifico alla data warehouse scelta;
- caricare i dati trasformati nella data warehouse;
- se necessario, cercare parole chiave tra i dati immagazzinati;
- pulire le risorse usate per il web scraping, in quanto i dati necessari sono già stati salvati e immagazzinati.

Oltre a questi step gli algoritmi di scraping devono essere innanzitutto facili da usare per i non esperti; devono garantire l'aggiornamento dell'elenco dei dati da analizzare in intervalli brevi e frequenti, la categorizzazione dei dati in base alla loro fonte e l'indicizzazione dei grandi volumi di dati per agevolarne la consultazione; devono avere algoritmi specifici per gruppi di dati particolari e offrire risultati in base alla complessità della ricerca, alla data o periodo, in base alla loro popolarità o in base alla lingua utilizzata.

Per riuscire a gestire i grandi volumi di dati servono anche soluzioni adeguate per immagazzinarli. I database tradizionali sono sì ben strutturati e supportano query complesse, però all'aumentare del volume dei dati si presentano ostacoli nella performance e aumentano i rischi di deadlock, senza contare lo spazio di storage limitato che influisce sulla scalabilità del sistema. I sistemi di storage che riescono a gestire grandi volumi di dati, come i sistemi di big data storage, sono più affidabili e altamente scalabili, garantendo performance migliori soprattutto con elevate quantità di dati.

È ottimo avere grandi quantità di informazioni e dati da analizzare e un sistema specializzato per gestirli, però le informazioni cambiano velocemente e si aggiornano di continuo in quest'era digitale richiedendo quindi, come conferma Milev, conoscenze e competenze professionali per adoperare al meglio questi sistemi.

2.2.3 Web Scraping e l'Intelligenza Artificiale (IA)

E i sistemi che riescono al meglio soddisfare i suggerimenti del professor Milev (*2017, pag. 475-485*) sono quelli che utilizzano l'intelligenza artificiale nel loro processo di web scraping.

L'IA permette di automatizzare al massimo la raccolta dati, specialmente a grandi volumi, trasformando radicalmente l'efficienza del processo di web scraping. Questi sistemi possono agire autonomamente all'estrazione dei dati da siti con layout differenti e interagire con gli elementi dinamici e interattivi (come menu a tendina e form dinamici), senza ricorrere a riconfigurazioni frequenti, grazie alla loro veloce capacità di apprendimento. Anche i vari tipi di dati (testo, immagini e video) sono salvati e immagazzinati in vari formati, come JSON, Excel e CSV, per facilitarne l'analisi. (*Chen, 2024*)

L'unico lato negativo di utilizzare sistemi basati sull'intelligenza artificiale è la costruzione e manutenzione del meccanismo di estrazione, molto costosa e da effettuare regolarmente. Tuttavia, è l'unico costo consistente da sostenere nella fase di creazione di un sistema di web scraping.

Il web scraping diventa quindi più preciso e personalizzabile, più scalabile e veloce, capace anche di elaborare il linguaggio naturale, dando così una sfumatura in più ai dati raccolti e analizzati, e meno propenso a commettere errori per via del continuo aggiornamento dei parametri interni dei suoi algoritmi. (*Bergonzi, 2024*)

2.3 Servizi di web scraping

Sul mercato sono già disponibili strumenti che offrono servizi di web scraping e raccolta dati, e nonostante la maggior parte dei servizi siano molto simili, col rischio talvolta di apparire generici, ogni strumento ha il suo ambito di utilizzo. (*Sirisuriya, 2015*)

Ci sono ovviamente strumenti che offrono solo servizi di scraping base (es. Scrapy, ScraperAPI, ScrapingBee), ma diversi offrono un servizio di scraping dati preciso e ambienti di sviluppo integrati (strumenti di questo tipo sono per esempio Octoparse o

ParseHub) o si concentrano sulla raccolta dei dati relativa a e-commerce e social media (come ad esempio la piattagforma SocioViz). Molti altri offrono ulteriori servizi come la trasformazione dei siti in dati facilmente consultabili, manutenzione gratuita e estrazione facile dei dati da pagine sia semplici che complesse. Alcuni strumenti offrono anche la possibilità di personalizzare e implementare i codici nelle proprie applicazioni personali.

Per scegliere lo strumento che fa di più al caso nostro dobbiamo considerare diversi fattori (*Özşahan, 2024*):

- Facilità di utilizzo: l'interfaccia di alcuni sistemi di scraping è intuitiva e user-friendly, mentre altre possono richiedere conoscenze avanzate per essere adoperate.
- Scalabilità: in base alla dimensione del progetto di web scraping, serve selezionare lo strumento che permetta di gestire il volume di dati su cui si andrà a lavorare.
- Capacità di estrazione dei dati: serve scegliere il proprio strumento di web scraping anche in base alla tipologia di dati che si andrà ad estrarre, in quanto non tutti i sistemi riescono ad estrarre gli stessi tipi di dati.
- Robustezza e affidabilità: uno sistema di scraping affidabile deve aggiornarsi con frequenza per stare al passo coi cambiamenti nell'ambiente circostante e sapersi adattare alle evoluzioni del sito che si sta analizzando.
- Costi: alcuni strumenti sono gratuiti mentre altri richiedono un abbonamento per accedere ai loro servizi.
- Supporto Proxy: per accedere ai contenuti dei siti evitando di essere bloccati per le richieste di accesso frequenti.

Tra gli strumenti presenti sul mercato, i seguenti sono i più gettonati:

- Bright Data: creato per gli sviluppatori, offre funzioni e template di scraping già pronte, riducendo il tempo di sviluppo. È scalabile e adopera funzioni di proxy e sbloccaggio. Adatta per ogni tipo di utilizzo, perfetta per piattaforme di e-

commerce e raccogliere dati dai social media. Il costo di utilizzo è per ogni chiamata di scraping.

- Oxylabs Scraper API: soluzione semplice ed efficace per l'estrazione dei dati, anche in tempo reale. Ideale per ricerche di mercato, protezione e verifica del marchio, controllo dei prezzi e molto altro. Il costo di partenza è di $49 al mese.
- Webscraper.io: sviluppato per estrarre dati e automatizzare task in parallelo, è un sistema di web scraping robusto. Offre rotazione degli indirizzi IP per rendere l'estrazione dati più affidabile e versatile. Permette di esportare i dati ottenuti in formati CSV, XLSX e JSON. È gratuito sottoforma di estensione browser con semplici funzioni a disposizione. Per funzioni più avanzate offre pacchetti che vanno dai $50 ai $300.
- Apify: lo strumento più facile da usare per i non esperti di informatica. Offre centinaia di funzioni pronte all'uso e non richiede codice per essere adoperato. Utilizza servizio di proxy per accedere ai siti web. Disponibile come estensione browser, Apify è perfetta per estrarre dati da pagine web, applicazioni mobile e piattaforme di e-commerce. Sono disponibili pacchetti gratuiti e un piano personale a partire da $49.

Un sistema di web scraping ideale deve quindi ottenere informazioni provenienti da diverse fonti in tempo reale e utilizzare metodi speciali di estrazione, trasformazione e immagazzinamento dei dati, che tengano conto della quantità di dati, piccola o grande che sia, e delle diverse strutture che i dati possano avere, ancor meglio se basato in cloud, scalabile e utilizza tecniche basate sull'intelligenza artificiale per rendere più efficace ed efficiente l'attività di raccolta dei dati.

Capitolo 3

Metodologia in cloud

Si può quindi iniziare a costruire la struttura base del funzionamento di un sistema di scraping facile da usare, che permetta di essere scalabile e che segua questi step fondamentali: ottenere lista dei siti da analizzare, estrarre i dati dal sito in questione, qualora fosse possibile, e trasformare e immagazzinare le informazioni acquisite in ambienti di storage sicuri.

3.1 Requisiti

Un progetto di web scraping deve essere facile da utilizzare e intuitivo, principalmente per gli utenti poco esperti; preferibilmente basato in cloud e scalabile, per poter gestire al meglio sia i piccoli che i grandi volumi di dati che si andranno a estrarre e poi analizzare; deve salvare i dati estratti in maniera sicura e efficiente, per evitare la perdita delle informazioni ottenute; mettere a disposizione dell'utente finale i risultati estratti in maniera chiara e leggibile; deve anche essere aggiornato abbastanza frequentemente, per stare al passo con le richieste degli utenti e per tenere d'occhio gli aggiornamenti stessi delle pagine web che va ad analizzare. Ovviamente le informazioni ottenute devono essere usate per scopi leciti, trasformate per dargli nuovo valore, e il tutto deve essere fatto tenendo presente il consenso all'estrazione dei dati da parte dei proprietari dei siti web: i siti che non permettono l'estrazione dei propri dati non devono essere estratti.

3.2 Diagramma delle attività

Per andare più nel dettaglio si può vedere il funzionamento nel diagramma delle attività [*Figura 1*].

Si parte dall'argomento che si vuole analizzare e, tramite una parola chiave, per esempio, si ottiene la lista dei siti da analizzare, ovviamente se è possibile lo scraping di quest'ultimi.

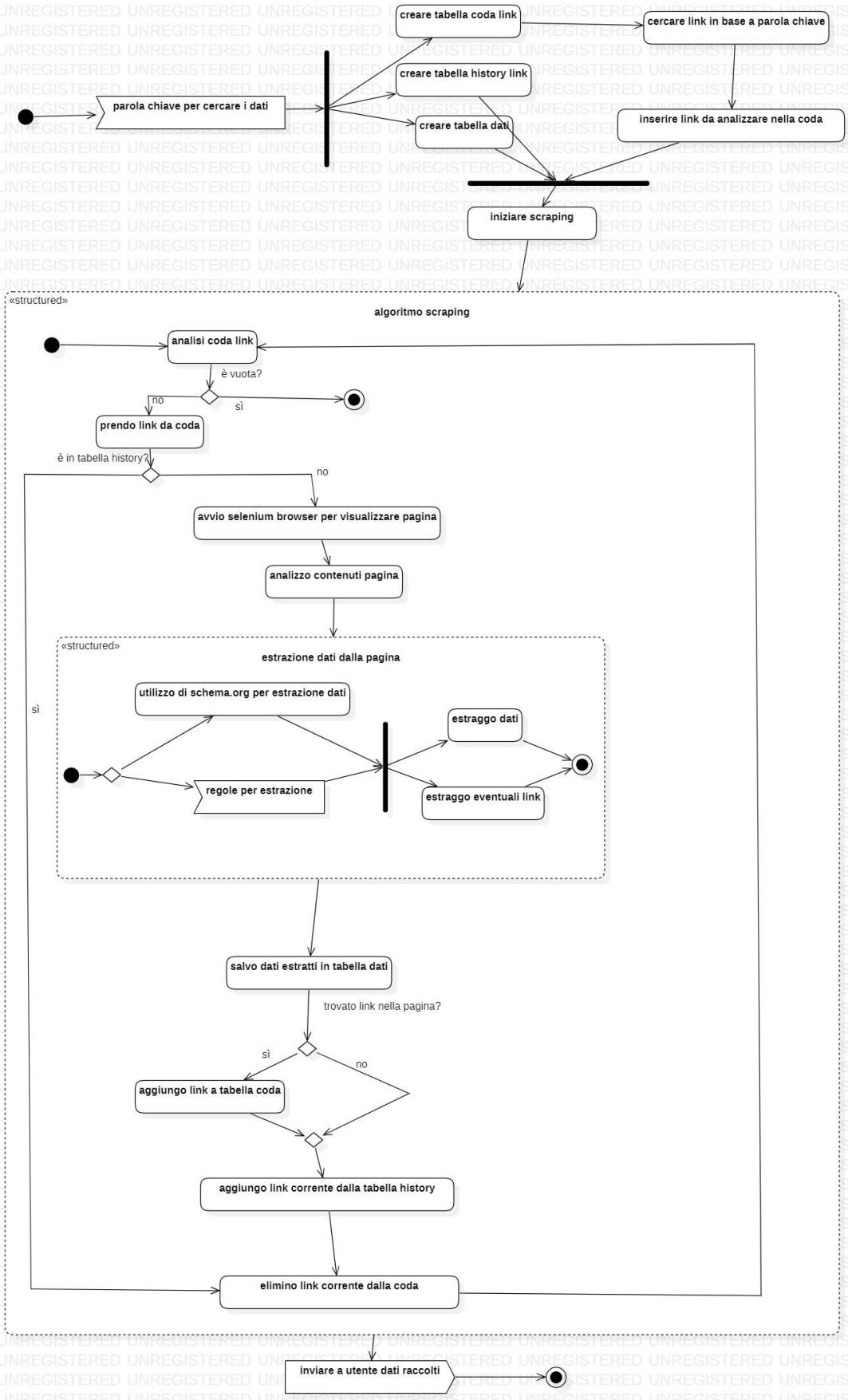

Figura 1: Activity diagram del processo di scraping

Per salvare i dati, che siano l'elenco dei siti o i dati estratti, si ipotizza l'utilizzo di tabelle in database, per facilitare l'inserimento, la modifica e l'eventuale eliminazione dei dati.
Una volta iniziato, il processo di scraping viene ripetuto finché la coda con la lista dei siti ottenuti non sarà vuota.
Il primo passo del processo consiste nel prendere il primo link presente nella coda dei siti da analizzare e successivamente recupere il codice html di quest'ultimo tramite una richiesta GET standard oppure con l'utilizzo di Selenium browser, che consente l'esecuzione della richiesta come se ci fosse un utente fisico dietro. Il primo approccio permette di raccogliere il contenuto delle pagine non facendo distinzione tra le varie strutture, trattando ugualmente pagine create con semplice html e le pagine più interattive. Il secondo approccio, invece, permette di personalizzare la richiesta e ottenere informazioni più dettagliate, specialmente dai siti più interattivi, portando però al bisogno di creare richieste specifiche per ogni sito.

Il codice html ottenuto viene analizzato e i suoi dati sono estratti o tramite regole dettate dalla libreria schema.org o regole prefissate. Utilizzando le regole di schema.org si possono ottenere risultati più generali e più dettagliati, ma si incorre nel rischio di estrarre troppa informazione. Seguendo invece regole prefissate, per esempio dall'utente che fa la domanda di scraping, si ottengono risultati più mirati alle richieste, che ovviamente saranno uniche e specifiche per ogni struttura e sito.

Infine, i dati sono salvati in tabelle o in database scelti dallo sviluppatore e verranno inviati all'utente a processo concluso.

Idealmente i siti una volta lavorati verrebbero tolti dalla lista di link da analizzare e aggiunti alla lista di link analizzati: questa lista è utile per evitare di esaminare più volte lo stesso sito e ottenere risultati duplicati; infatti la lista con gli elementi già estratti dovrebbe essere consultata prima di iniziare il processo di scraping del sito.

3.2.1 Selenium

Selenium[4] è uno strumento open source per la gestione automatica dei browser, utilizzato principalmente come framework di testing, che offre anche servizi per simulare l'attività dell'utente reale all'interno del browser.

Con Selenium IDE l'utente può creare, registrare e fare il debug dei test direttamente sul browser grazie all'estensione Chrome e Firefox, e velocizzare l'operazione utilizzando i comandi presenti in Selenium Builder. (*Selenium Overview*)

Lo strumento per simulare il comportamento di un utente reale è Selenium WebDriver: utilizzando API di automazione del browser che non richiedono di essere compilate con altre applicazioni, WebDriver non è intrusivo e può essere utilizzato sia in locale che su macchine remote, sulla maggior parte dei browser supportati per ricreare l'attività dell'utente. Le sue funzioni possono essere potenziate da Selenium Grid, che permetterebbe l'esecuzione contemporanea dei test su diverse macchine, riducendo notevolmente il tempo.

3.2.2 Schema.org

Schema.org è una iniziativa nata dalla collaborazione di motori di ricerca come Google, Microsoft, Yahoo e Yandex, con lo scopo di creare, mantenere e promuovere schemi standard per i dati strutturati in internet, e-mail e molto altro.

Le categorie dei dati salvati negli schemi o, più precisamente, nei vocabolari di schema.org sono microformati che rappresentano entità, relazioni tra quest'ultime e azioni. Questi vocabolari possono essere usati con diverse codifiche (Microdata, RDFa, JSON-LD) e possono essere estesi senza problemi tramite modelli di estensione. (*Web Scraping Simplified - Scraping Microformats, 2024*)

Il vocabolario condiviso e il suo continuo aggiornamento da parte della comunità, rende schema.org uno degli strumenti più efficaci per semplificare l'automazione del web.

---

[4] Selenium, https://www.geekandjob.com/wiki/selenium [8]

Capitolo 4

Implementazione

Per realizzare un prototipo di applicazione di web scraping, sono stati utilizzati i servizi di cloud computing forniti da Amazon Web Services (AWS), sussidiaria di Amazon che opera e offre servizi cloud (tra cui servizi di calcolo, elaborazione e computazione, archiviazione e distribuzione dei contenuti, servizi di database, perfino servizi per il machine learning, oltre a servizi per rendere più agile lo sviluppo ai programmatori) in tutto il mondo, oltre a programmi come Visual Studio Code, utilizzati per testare il codice in locale.

4.1 Progetto in AWS

Per il progetto sono stati utilizzati i seguenti servizi AWS:

- AWS Lambda: servizio di elaborazione che esegue codice in risposta a eventi e gestisce le risorse di elaborazione.
- Amazon Simple Storage Service (S3), conosciuto più comunemente come bucket S3, è un servizio che offre soluzioni di storage con diverse caratteristiche per ogni esigenza.
- Amazon Simple Queue Services (SQS): servizio di accodamento di messaggi, utile per microservizi, sistemi distribuiti e applicazioni serverless.
- Amazon CloudWatch: servizio di monitoraggio delle risorse e delle applicazioni AWS.
- AWS CloudFormation: servizio di Amazon per gestire e implementare infrastrutture con codice esterno.

4.1.1 Architettura

I servizi AWS utilizzati nel progetto sono collegati in questo modo [F*igura 2*].

Il programma inizia dall'inserimento in una coda SQS l'URL del sito che si vuole andare ad analizzare, che lo invierà come messaggio alla prima funzione lambda. Si possono inviare più messaggi uno dopo l'altro, e questi verranno accodati in modalità FIFO (First In First Out) all'interno della coda.

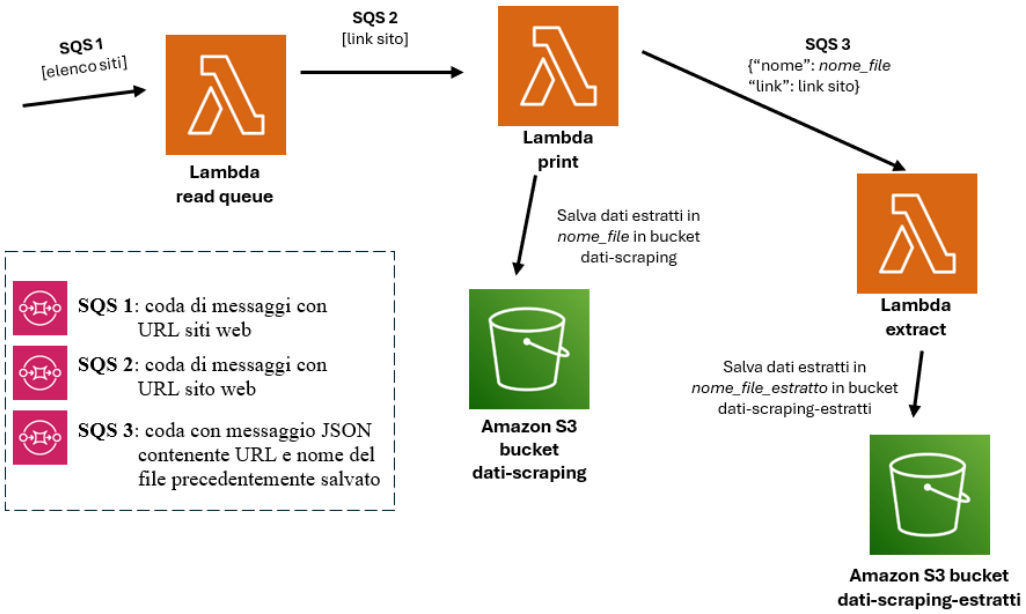

Figura 2: Schema programma web scraping in AWS

La prima funzione (lambda_read_queue) ha lo scopo di leggere i messaggi ricevuti in coda e di inviarli uno alla volta alla funzione successiva tramite una seconda coda, facendo da buffer per l'invio dell'elenco dei siti da analizzare.

La seconda funzione (lambda_print) una volta ricevuto il messaggio inizia ad analizzare il sito effettuando una richiesta GET per ottenere lo script HTML della pagina. Successivamente salva lo script all'interno di un file di testo nel bucket S3 apposito, ma non prima di aver controllato l'esistenza del file: infatti se si salvasse un file con lo stesso nome di un altro file presente nel bucket, l'ultimo sovrascriverebbe il documento precedente. Infine, la funzione, tramite una terza e ultima coda SQS, invia un messaggio

contenente il link del sito analizzato e il nome del file appena salvato alla terza funzione.

La terza funzione (lambda_extract) ha il compito di estrarre i contenuti della pagina ricevuta in ingresso. I dati estratti sono a loro volta salvati in un nuovo file nello spazio di storage apposito.

4.2 Implementazione

La funzione lambda_read_queue, come descritto nella sezione precedente, serve solo da buffer per l'invio dell'elenco dei siti alla funzione lambda_print. È attivata dai messaggi ricevuti in ingresso e a sua volta invia messaggi in una seconda coda. Lo sviluppo della funzione è stato fatto direttamente in AWS, dal pannello della sezione codice della Lambda Function.

Anche la funzione lambda_print è stata realizzata in AWS, e il suo funzionamento è triggerato dalla seconda coda di messaggi popolata dalla prima Lambda Function. Ricevuto il messaggio con l'URL del sito da analizzare, la funzione estrae il codice HTML con codifica utf-8 ed è pronto per salvarlo in un file di testo per la successiva analisi ed estrazione.

Il file di testo viene denominato automaticamente grazie a una funzione che, dato l'URL in ingresso, restituisce il nome del sito senza i componenti del percorso o query. Ovviamente prima di restituire il nome del file creato, serve verificare la presenza o meno di altri file con lo stesso nome, per evitare la sovrascrittura e perdita dei dati precedentemente raccolti. Questo viene fatto tramite una seconda funzione che controlla l'esistenza dei file nel bucket S3 e restituisce il valore 1 se esiste il file. Il controllo viene effettuato in una prima istanza sul nome derivato direttamente dal sito e se esistesse, il controllo successivo verrebbe fatto sul nome ricavato a cui viene accodato "_" più un numero contatore, che sarà incrementato (es. _1, _2, _3 e così via) finché non ci saranno corrispondenze tra i file già presenti nel bucket S3, ovvero quando la funzione di controllo restituirà il valore 0.

Prendiamo "https://it.wikipedia.org/wiki/Web_scraping" come esempio. Il nome che avrà il file sarà it_wikipedia_org.txt. Se esistesse un file con lo stesso nome si dovrebbe analizzare it_wikipedia_org_1.txt, e se anche questo nome non dovesse andare bene si dovrebbe analizzare it_wikipedia_org_2.txt e così via.

Funzione di creazione del nome del file:

```
def crea_nome_file(url):

    parsed_url = urlparse(url)
    base_url=parsed_url.netloc

    base_url = base_url.replace(".", "_")
    nome_file = base_url + ".txt"

    controllo = esiste_file(nome_file)
    counter = 0

    while controllo == 1:
        counter += 1
        print(counter)
        num = str(counter)
        nome_file = base_url + "_" + num + ".txt"
        controllo = esiste_file(nome_file)

    return nome_file
```

Funzione per controllo dell'esistenza del file

```
def esiste_file(file_name):
    s3_client = boto3.client('s3')
    s3_bucket = "dati-scraping-test"
    controllo = 0
    ris = s3_client.list_objects(Bucket = s3_bucket)

    for obj in ris.get('Contents'):
        key = obj.get('Key')
        if key == file_name:
            controllo = 1

    return controllo
```

Quando il nome del file è stato scelto, il codice estratto viene salvato nel bucket prescelto.

Come ultimo passo la funzione invia un messaggio in formato JSON grazie a una coda SQS alla funzione per estrarre i dati, contenente il link del sito che è stato analizzato e il nome del file in cui è stato salvato.

{"nome": *nome_del_file_appena_salvato*,
"link": *link_del_file_ricevuto_in_ingresso*}

La funzione lambda_extract dopo aver ricevuto in ingresso il messaggio, salva i dati ricevuti in variabili singole e procede con l'esecuzione dell'estrazione.
L'estrazione dei dati viene effettuata tramite la funzione extract del pacchetto extruct presente in Python. La funzione extruct.extract ricevuti in ingresso il codice html e il link del sito restituisce valori di microdata, json-Id, opengraph, microformat, rdfa e dublincore (*Documentazione Exrtuct 0.17.0*). Per il progetto non si è fatto distinzione, ma si può specificare il tipo di dato che deve essere restituito nel campo syntaxes della funzione.
I valori estratti sono salvati in maniera simile a come effettuato nella funzione precedente: viene aggiunto in coda al nome del file del codice ottenuto la dicitura "_estratto" (facendo diventare nell'esempio precedente it_wikipedia_org_1.txt in it_wikipedia_org_1_estratto.txt) e viene salvato nel bucket predefinito per i file estratti.
La funzione di estrazione è stata sviluppata in Visual Studio Code e successivamente caricata in AWS tramite CloudFormation grazie la creazione di una Docker image, per via di problemi legati alla versione del pacchetto da importare.

Link di GitHub con i codici in dettaglio.

Capitolo 5

Conclusione

5.1 Obbiettivi raggiunti e problemi riscontrati

Il programma di web scraping creato, ricevuto in ingresso un sito o un elenco di siti, alla fine salva nelle cartelle messe a disposizione sia il codice html del sito analizzato, sia i dati estratti dall'ultima funzione, in maniera veloce ed efficace. Grazie allo sviluppo in cloud su AWS, il programma riesce a gestire sia piccoli che grandi volumi di dati, dimostrando la sua scalabilità.

Nello sviluppo della funzione di estrazione dati, tuttavia, è sorto un problema durante il caricamento dei pacchetti necessari per il suo funzionamento: per via della dimensione dei pacchetti da installare, un import standard in linea nel codice impiegava troppo tempo a caricare il pacchetto extruct, rendendo necessario effettuare l'importazione in parallelo e in un ambiente separato dalla funzione ma comunque collegato. Il problema è sorto per via della versione di lxml necessaria per il funzionamento del pacchetto extruct: la versione più recente di lxml non era più compatibile con le richieste del pacchetto extruct, perciò serviva indicare la versione precedente nella fase di importazione dei pacchetti.

Un primo tentativo di risoluzione del problema è stato fatto con l'ausilio dei livelli, ovvero layer, nella funzione lambda: creando un layer contenente il codice per il caricamento dei pacchetti necessari, il programma avrebbe eseguito dapprima il livello, caricando quindi i pacchetti, e successivamente il codice della funzione. Purtroppo, questo metodo non ha funzionato per via dell'impossibilità di specificare le versioni specifiche dei pacchetti necessari.

Per ovviare a questo problema, si è ricorso alla creazione di un'immagine Docker[5]: un container che permette di eseguire il codice in un ambiente separato senza influire direttamente sul programma. Questa immagine è stata creata in prompt dei comandi partendo da un Docker file contenente riferimenti al documento con l'elenco dei pacchetti necessari per lo sviluppo, con le loro versioni corrette, e al file col codice della funzione.

La funzione di estrazione potrebbe essere migliorata aggiungendo ulteriori parametri di estrazione, basandosi di più sulla libreria di schema.org o sui parametri richiesti dall'utente che usufruisce del programma di scraping.

5.2 Implementazioni future

Come descritto nello schema di funzionamento del web scraping, sarebbe utile implementare delle liste di elementi che contengano l'elenco dei siti da analizzare e quelli già analizzati, salvati in file ma idealmente in tabelle, per rendere più facile l'inserimento, la modifica e l'eliminazione dei dati.

Il primo elenco, dei siti da analizzare, potrebbe essere letto da una funzione lambda che invierebbe i link alla prima funzione lambda tramite la prima coda SQS.

L'elenco dei siti già analizzati verrebbe consultato all'interno della funzione lambda_read_queue, per controllare se il sito proposto fosse già stato analizzato e per evitare di duplicare le informazioni analizzate, rendendo la funzione un vero e proprio buffer.

In future implementazioni la funzione di creazione dei nomi dei file potrebbe essere migliorata per rendere più univoci ed esplicativi i nomi dei file, specialmente se il sito principale è lo stesso e cambiano solo le directory.

[5] Software progettato per eseguire processi informatici in ambienti isolabili, minimali e facilmente distribuibili, che ha come obbiettivo semplificare i processi di deployment di applicazioni software. (*https://it.wikipedia.org/wiki/Docker*)

Un altro aspetto da implementare è l'invio dei dati estratti al client che ha fatto la richiesta una volta finito il processo di scraping.

5.3 Considerazioni finali

Visto il numero di campi in cui può essere usato, il sistema di web scraping deve innanzitutto puntare alla rielaborazione dei dati raccolti, dando loro nuova vita, senza sfruttare le sue capacità di estrazione per scopi personali e dannosi. Bisognerebbe anche che il sistema salvi i dati in modo sicuro, senza porre rischi di diffusione non autorizzata.

L'applicazione deve tenersi in costante aggiornamento per restare al passo sempre più veloce dell'evoluzione dell'internet e delle informazioni presenti nel web.

Un sistema di web scraping ideale deve essere quindi un sistema sicuro, per garantire i diritti della privacy e la protezione dei dati sensibili; scalabile, per potersi adattare a qualsiasi volume di dati debba estrarre; basato in cloud, non solo per rendere possibile la scalabilità ma anche per una gestione più semplificata ed efficiente del programma.

Bibliografia

[1] Sharma, Kaajal, and Gautam M. Borkar. "Comparative Analysis of Dynamic Web Scraping Strategies: Evaluating Techniques for Enhanced Data Acquisition."

[2] Chandan Kumar, “14 Popular Cloud-based Web Scraping Solutions”. geekfalre.com, 29 gennaio 2024. Consultato più recentemente il 20 maggio 2024 https://geekflare.com/web-scraping-tools/

[3] Sirisuriya, De S. "A comparative study on web scraping." (2015).

[4] Milev, Plamen. "Conceptual approach for development of web scraping application for tracking information." Economic Alternatives 3 (2017): 475-485.

[5] Andrea Bergonzi, “Come l’AI sta rendendo più efficace il web scraping”. dataskills.it, 23 febbraio 2024. Consultato il 21 maggio 2024 https://www.dataskills.it/come-ai-sta-rendendo-piu-efficace-il-web-scraping/#gref

[6] Chang Chen. “Comparing Top 8 AI Web Scraping Tools (updated 2024)”. brandeen.ai, 16 aprile 2024. https://www.bardeen.ai/posts/web-scraper-tools-2023

[7] Andrea Fumagalli, “Web scraping: cos’è, perché si usa e come difendersi da “intrusioni” indesiderate”. 23 aprile 2021. Consultato più recentemente il 10 giugno 2024. https://www.agendadigitale.eu/sicurezza/web-scraping-cose-perche-si-usa-e-come-difendersi-da-intrusioni-indesiderate/

[8] “Selenium”, consultato più recentemente il 5 giugno 2024, https://www.geekandjob.com/wiki/selenium

[9] “Selenium Overview”, consultato più recentemente il 5 giugno 2024, https://www.selenium.dev/documentation/overview/

[10] “Web Scraping Simplified - Scraping Microformats”, 10 maggio 2024, consultato più recentemente il 7 giugno 2024 https://scrapfly.io/blog/web-scraping-microformats/

[11] “Welcome to schema.org”, 20 maggio 2024, consultato più recentemente il 7 giungo 2024, https://schema.org/

[12] “Brief History of Web Scraping”, 14 maggio 2021, consultato più recentemente il 10 giugno 2024, https://webscraper.io/blog/brief-history-of-web-scraping

[13] “Web scraping”, consultato più recentemente il 11 giugno 2024 https://it.wikipedia.org/wiki/Web_scraping

[14] Hatice Özşahan, “15 Best Web Scraping Tools in 2024 to Extract Online Data”, 2024, consultato più recentemente il 10 giugno 2024 https://popupsmart.com/blog/web-scraping-tools

[15] James Densmore, “Ethics in Web Scraping”, 23 luglio 2017, consultato più recentemente il 13 giugno 2024, https://towardsdatascience.com/ethics-in-web-scraping-b96b18136f01

[16] Wikipedia, “World Wide Web Wanderer”, consultato più recentemente il 20 giugno 2024, https://en.wikipedia.org/wiki/World_Wide_Web_Wanderer,

[17] Wikipedia, “JumpStation”, consultato più recentemente il 20 giugno 2024, https://en.wikipedia.org/wiki/JumpStation

[18] “BeautifulSoup Documentation”, consultato più recentemente il 20 giugno 2024, https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[19] “Exrtuct 0.17.0” (Documentazione pacchetto extruct), consultato più recentemente il 12 giugno 2024, https://pypi.org/project/extruct/

[20] Link codice in GitHub, “Progetto tesi – soluzione scalabile e in cloud per web scraping”, ultimo commit della pagina effettuato il 26 giungo 2024, https://github.com/AleCortinovis/ProgettoTesi.git

[21] Libero Tecnologia, “Sai quanti siti web esistono? Il numero è impressionante”, 24 maggio 2022, consultato più recentemente il 23 giugno 2024, https://tecnologia.libero.it/quanti-siti-web-esistono-numero-impressionante-57018

[22] “Web Intelligence Hub”, consultato più recente il 26 giugno 2024, https://www.wih-con.eu/en/index

[23] Wikipedia, “Docker”, consultato più recentemente il 26 giugno 2024, https://it.wikipedia.org/wiki/Docker