UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

"Web Scraping: Un'analisi sul recupero automatico di informazioni sul retrocomputing"

Relatore: Prof. Giorgio Maria Di Nunzio

Laureando: Filippo Franceschini

ANNO ACCADEMICO 2023 – 2024

Data di laurea 23/09/2024

Sommario

Nel contesto dinamico della rapida evoluzione tecnologica, la conservazione e la comprensione del patrimonio informatico del passato diventano aspetti fondamentali. Questa tesi si propone di esplorare e approfondire il ruolo del web scraping come metodologia avanzata per il recupero automatico di informazioni legate al Retrocomputing. Esso si focalizza sull'analisi dei sistemi informatici del passato e richiede un approccio innovativo per reperire in modo sistematico dati storici, documenti e informazioni pertinenti. Il web scraping d'altra parte, con la sua capacità di estrarre dati da pagine web in modo efficiente, si presenta come uno strumento cruciale in questo contesto.

Attraverso un'analisi dettagliata, questa tesi esplorerà le sfide tecniche associate all'estrazione di dati dal web inerenti il Retrocomputing, comprese le variazioni nella struttura dei siti web nel corso del tempo, l'analisi della loro struttura e le loro limitazioni. Saranno esaminati anche gli aspetti etici legati al recupero automatico di dati online, considerando la gestione della privacy e la conformità alle normative vigenti. Al fine di fornire un quadro completo, saranno presentati esempi pratici che illustrano l'efficacia del web scraping nella raccolta sistematica di informazioni sui sistemi informatici storici, enfatizzando il suo ruolo non solo come strumento di ricerca, ma anche come mezzo essenziale per la preservazione e la valorizzazione del nostro ricco patrimonio informatico. L'obiettivo è dunque quello di descrivere la pratica del web scraping nel contesto del Retrocomputing, evidenziando le sue implicazioni, sfide e prospettive future.

Indice

Introduzione

Nell'epoca digitale in cui ci troviamo immersi, la rapidità con cui le tecnologie informatiche si evolvono è spaventosa, e ciò rende molti dispositivi obsoleti in poco tempo. È per questo che nel contesto del Retrocomputing, che si dedica all'analisi dei sistemi informatici storici, le principali sfide sono legate al recupero accurato e sistematico di dati, documenti e informazioni rilevanti prima che esse vadano perdute. Queste informazioni sono quindi in continuo aumento e necessitano di robuste basi di dati per essere salvate in maniera permanente.
In risposta a questa esigenza, il web scraping si presenta come una metodologia innovativa, che consente l'acquisizione automatizzata di informazioni in maniera rapida e sicura, attraverso metodologie e strumenti scelti in relazione alla struttura di siti web su cui vengono applicati. Questa pratica però è spesso limitata o addirittura bandita da alcune pagine, in quanto potrebbe portare alla diffusione di dati sensibili o protetti, e per questo è necessario applicarla con opportuni accorgimenti.
Questa tesi si propone di esplorare il connubio tra web scraping e Retrocomputing, analizzando in dettaglio come questa pratica possa contribuire in modo significativo alla preservazione e all'approfondimento della nostra eredità informatica. Attraverso l'indagine di sfide tecniche, opportunità e considerazioni etiche legate al web scraping nel contesto specifico del Retrocomputing, si cercherà di delineare una prospettiva completa su questa sinergia.
Il viaggio attraverso questa ricerca comprenderà innanzitutto una spiegazione approfondita sul termine Retrocomputing, partendo dalla definizione e dalla sua origine per poi approfondire guardando alle sue prospettive future. Successivamente, si analizzerà la pratica del web scraping, definendone le modalità e gli strumenti che esso impiega, dando anche una panoramica su quelle che sono le implicazioni legali e gli accorgimenti che vanno adottati durante il suo utilizzo.
Si passa poi alla descrizione di un esempio pratico, rappresentato da un programma in linguaggio Python che svolga l'estrazione di dati e immagini da alcuni archivi web opportunamente selezionati. Viene definita in primo luogo la progettazione teorica di esso, la strategia e i mezzi per costruirlo e per definire poi la base dati che andrà a contenere tutte le informazioni di interesse. Si prosegue poi con un'analisi dettagliata del codice e delle varie parti che lo compongono, descrivendone anche la procedura per eseguirlo. Infine si andranno ad analizzare i risultati e a trarre le opportune considerazioni finali. Riassumendo, questa tesi si propone di gettare luce su un'innovativa frontiera della conservazione digitale, esplorando come il web scraping possa diventare un elemento chiave nella preservazione del vasto patrimonio inerente il Retrocomputing.

1 Retrocomputing

1.1 Definizione

Il Retrocomputing, è un termine della lingua inglese indica un'attività che consiste nel recuperare e intervenire su computer di vecchie generazioni e utilizzarli nuovamente per scopi storico-culturali e hobbistici. Si può definire come un intricato intreccio di storia e passione tecnologica, un'incursione affascinante nel cuore dell'evoluzione informatica. Questo capitolo si propone di illuminare i sentieri intricati che hanno portato alla formazione di questo fenomeno, esplorando le sue origini e il contesto culturale che ha alimentato la sua crescita. Nel corso del tempo, il termine *"Retrocomputing"* ha assunto significati diversi, riflettendo la continua metamorfosi di un movimento che va oltre la semplice nostalgia per il passato. Dai primi passi di appassionati individui determinati a conservare i primi computer personali, fino alla crescita di comunità online dedicate, è diventato un fenomeno che attraversa confini geografici e temporali. Questa esplorazione si immerge nelle radici profonde di un movimento che non solo celebra l'hardware e il software del passato, ma che funge anche da ponte tra le generazioni, unendo appassionati, ricercatori e curiosi in un dialogo continuo con il nostro patrimonio informatico.
Dall'altra parte della medaglia, il capitolo esamina l'hardware e il software che hanno definito l'epoca d'oro dell'informatica personale. Sistemi iconici come il Commodore 64, l'Amiga, l'Atari, e molti altri, hanno plasmato il paesaggio del Retrocomputing, diventando i simboli tangibili di un'era in cui la tecnologia stava iniziando a trovare la sua strada nelle case di milioni di persone. Oltre a essere considerati semplicemente "vecchi computer", questi sistemi sono diventati portatori di memorie, testimoni delle prime fasi della rivoluzione digitale.
In questo panorama ricco di storie e innovazioni, ci si sofferma anche sulla dimensione sociale che esso comporta, analizzando le comunità che hanno fiorito online e offline. Queste comunità fungono da centri di condivisione di conoscenze, esperienze e risorse, svolgendo un ruolo fondamentale nella preservazione e nell'espansione di esso. La creazione di archivi digitali, il supporto tecnico reciproco e l'organizzazione di eventi dedicati sono solo alcune delle manifestazioni di un movimento che si nutre della passione collettiva per il passato informatico.
Personalmente, trovo che osservare i vecchi dispostivi informatici ci porta poi ad apprezzare ulteriormente quelli disponibili attualmente, in quanto si può notare la base di partenza dalla quale, tramite sviluppo di nuovi materiali e migliorie generali, si è arrivati agli ultimi modelli.

1.2 Origini e Sviluppo

L'origine del termine *"Retrocomputing"* può essere fatta risalire al desiderio di preservare e comprendere l'eredità dei primi giorni dell'informatica personale. Gli appassionati iniziarono a raccogliere e preservare computer considerati ormai obsoleti, vedendoli non solo come strumenti tecnologici del passato, ma come veri e propri artefatti culturali. Questo movimento prese piede soprattutto negli anni '80 e '90, un periodo in cui la tecnologia stava facendo passi da gigante e i primi personal computer stavano diventando accessibili al grande pubblico. In questo contesto, il Retrocomputing nacque come risposta al rapido obsolescenza dei dispositivi e al desiderio di conservare la storia informatica prima che andasse perduta.

Con il passare del tempo, questo fenomeno ha attraversato diverse fasi di sviluppo. Da una nicchia di appassionati entusiasti, è cresciuto fino a diventare un movimento più ampio e strutturato. Il termine ha assunto significati più ampi, andando oltre la mera raccolta di hardware per includere la preservazione del software, la documentazione e la condivisione di conoscenze. Le organizzazioni e le comunità online hanno svolto un ruolo chiave nello sviluppo del movimento, fornendo piattaforme dove gli appassionati possono scambiare idee, risorse e esperienze.

Tra queste sono presenti numerosi siti web, ma anche pagine su social come *Facebook, Reddit* e *Instagram* che diffondo anche ai più giovani il fascino delle vecchie tecnologie.

Inoltre, nel contesto delle origini del Retrocomputing, è importante esaminare i primi progetti e le iniziative che hanno contribuito a definire il movimento. La creazione di musei virtuali, archivi digitali e la catalogazione di hardware e software storici hanno svolto un ruolo essenziale nella preservazione di questa parte della storia informatica. Progetti come il *"Homebrew Computer Club"* degli anni '70, che fu uno dei primi gruppi a riunire appassionati di computer, rappresentano un punto di partenza significativo per comprendere come il Retrocomputing ha iniziato a prendere forma come movimento organizzato.

Esplorare le origini e lo sviluppo di questo affascinante movimento non solo offre un quadro storico delle sue radici, ma getta anche le basi per comprendere come questo movimento si è evoluto nel tempo, trasformandosi da una passione di pochi a un fenomeno culturale più ampio e interconnesso.

1.3 Sfide e prospettive future

Il viaggio nel Retrocomputing non è privo di sfide uniche e prospettive intriganti per il futuro. Un aspetto cruciale da esplorare è la sfida della disponibilità e manutenzione dell'hardware originale. Con il passare degli anni, trovare e mantenere in funzione computer e periferiche d'epoca diventa sempre più complesso. Questo solleva interrogativi sulla sostenibilità a lungo termine del movimento e sulle strategie per affrontare l'inevitabile degrado del materiale fisico.

Un'altra sfida rilevante riguarda l'accesso e la preservazione del software. Molti programmi e sistemi operativi delle prime epoche informatiche sono diventati obsoleti, portando a un rischio reale di perdita di software chiave. La sfida è dunque trovare modi innovativi per preservare il software storico, consentendo alle generazioni future di esplorare e comprendere appieno le esperienze informatiche del passato. Parallelamente, emergono anche considerazioni etiche riguardo alla conservazione digitale e alla presentazione accurata del passato; delineare confini chiari tra il restauro autentico e l'adattamento moderno diventa essenziale per mantenere l'integrità storica, rispettando al contempo i principi etici e la trasparenza nel processo di preservazione. Riuscire a reperire pezzi di ricambio per vecchi dispositivi danneggiati è sicuramente difficile e ciò porta ad usare componenti magari più recenti, che però tolgono originalità e modificano l'integrità dell'apparecchio. Con l'avanzare delle tecnologie moderne, sorgono nuove opportunità per ricreare o emulare ambienti Retrocomputing su hardware più recente e questo solleva domande stimolanti sulla combinazione di autenticità storica e innovazioni tecnologiche, aprendo la strada a nuove forme di interazione con il passato informatico.

Infine, il ruolo del Retrocomputing nella trasmissione di conoscenze e nell'educazione riveste un'importanza crescente. Esitono infatti numerosi progetti didattici che coinvolgono questo argomento e possono fornire ai giovani una prospettiva preziosa sulle radici dell'informatica, facilitando la comprensione di come le tecnologie attuali abbiano avuto origine e sviluppato la loro complessità nel corso del tempo. Uno tra questi è sicuramente il *Ctrl+Alt Museum* a Pavia, aperto recentemente, nel 2022, e gestito dall'associazione *comPVter*. Con la sua superficie di 600m^2 mette in mostra una vasta collezione di informatica funzionante che comprende oltre 2000 pezzi tra computer digitali, macchine calcolatrici, periferiche, programmi, documenti storici, disegni tecnici, fotografie.

Fig. 1.1 – Una parete del museo

2 Web scraping

2.1 Definizione

Il web scraping, conosciuto anche come estrazione dati da pagine web, è una tecnica informatica di estrazione dei dati da un sito web per mezzo di programmi software. Riveste un ruolo centrale nella caccia alle informazioni nell'era digitale, in un mondo in cui la vastità e la diversità di dati online sono diventate una risorsa senza precedenti. Esso emerge come un alleato indispensabile per coloro che cercano di navigare attraverso l'oceano informativo e selezionare solo i dati rilevanti e utili. Questa pratica si distingue per la sua capacità di automatizzare il processo di acquisizione di dati da pagine web, rendendo possibile l'estrazione di informazioni strutturate o non strutturate in modo rapido ed efficiente.

Nella sua essenza, il web scraping supera le limitazioni della raccolta manuale di dati, consentendo agli utenti di recuperare informazioni da una varietà di fonti online con una precisione e una velocità che sarebbero altrimenti impossibili da ottenere manualmente e ciò lo rende uno strumento indispensabile per coloro che si occupano di analisi di mercato, ricerca accademica, monitoraggio delle tendenze o qualsiasi altra attività che richieda un accesso sistematico e tempestivo a dati provenienti da pagine web.

In questo capitolo quindi, ci proponiamo di esplorare i fondamenti di questa pratica, comprendendo il suo impatto e la sua rilevanza nel contesto più ampio della ricerca e dell'estrazione dati. Oltre a delineare la definizione di base del web scraping, approfondiremo le sue applicazioni pratiche, evidenziando come questa tecnica diventa uno strumento versatile in molteplici settori e scenari. Navigando tra le sue possibilità e potenzialità, avremo modo di comprendere come il web scraping si sia affermato come un elemento chiave nel panorama digitale contemporaneo.

Personalmente, ritengo che questa pratica sia di estrema utilità perché fornisce i dati di base per ricerche e analisi, il tutto risparmiando tempo e risorse umane.

2.2 Applicazioni pratiche

Il web scraping, con la sua capacità di estrarre dati in modo efficiente da svariate fonti online, riveste un ruolo chiave in numerose applicazioni pratiche che spaziano trasversalmente attraverso diversi settori.

Nel contesto aziendale, si configura come uno strumento essenziale per monitorare i prezzi dei concorrenti, fornendo alle imprese una panoramica in tempo reale delle dinamiche di mercato. Raccogliere recensioni dei clienti da diverse piattaforme online consente inoltre alle aziende di

valutare la soddisfazione del cliente e adattare di conseguenza le proprie strategie di prodotto e servizio. Questa tecnica offre inoltre la possibilità di analizzare le tendenze di mercato, identificare opportunità emergenti e adattare rapidamente le strategie aziendali in risposta alle mutevoli condizioni del settore.

Nell'ambito della ricerca e dell'analisi dei dati, il web scraping assume un ruolo fondamentale nella raccolta di informazioni utili per formulare ipotesi e convalidare teorie. Gli accademici possono impiegare questa tecnologia per raccogliere dati di ricerca da fonti online, accelerando il processo di acquisizione di informazioni e consentendo un'analisi più approfondita. L'estrazione dati da fonti eterogenee permette inoltre di integrare informazioni provenienti da diverse discipline, agevolando la creazione di modelli predittivi e l'individuazione di pattern complessi.

Nel campo dell'automazione, lo scraping web contribuisce in modo significativo alla riduzione del lavoro manuale e consente alle organizzazioni di integrare informazioni in tempo reale nei propri sistemi aziendali, migliorando l'efficienza operativa e garantendo la tempestiva disponibilità di dati aggiornati. In questo modo, lo scraping web emerge come uno strumento versatile, in grado di migliorare la precisione delle decisioni aziendali, fornire un vantaggio competitivo e stimolare le innovazioni in molteplici settori.

2.3 Aspetti etici e legali

Il web scraping, pur essendo una tecnica potente per l'estrazione di dati online, presenta una serie di sfide etiche e legali che richiedono un approccio ponderato e responsabile. Questo sottoparagrafo si propone di esplorare gli aspetti etici nell'uso di questa pratica e di fornire una panoramica delle leggi e delle normative pertinenti.

In termini etici, il web scraping solleva questioni legate alla privacy e alla gestione delle informazioni personali. Mentre alcuni siti web possono essere di accesso pubblico e consentirlo, altri possono richiedere l'autorizzazione o addirittura proibire esplicitamente questa pratica. È fondamentale, dunque, considerare attentamente la moralità e l'etica in relazione al contesto specifico, cercando di minimizzare l'impatto negativo sulla privacy degli utenti e rispettando le norme etiche stabilite. Nel nostro caso di studio andremo ad analizzare siti che permettono esplicitamente l'estrazione di dati da essi per scopi didattici e di ricerca.

Da un punto di vista legale, la pratica del web scraping può variare notevolmente a seconda della giurisdizione. Alcuni paesi hanno leggi che lo regolamentano specificatamente, mentre altri si affidano a leggi più generiche sulla protezione dei dati e sulla proprietà intellettuale. Comprendere le leggi locali e rispettarle è cruciale per evitare potenziali problemi legali derivanti dal suo uso

improprio ed è importante essere consapevoli dei Termini di Servizio dei siti web, in quanto possono contenere disposizioni specifiche sul consenso al suo uso.
Per promuovere un approccio etico e legale al web scraping quindi, è consigliabile adottare pratiche che rispettino la trasparenza e la buona fede. L'ottenimento di consenso o l'adesione ai protocolli specifici di un sito web possono contribuire a ridurre i rischi legali e di carattere etico e si possono attuare contattando per esempio gli amministratori delle pagine web.

3 Scelte Progettuali

3.1 Obiettivi

Il compito principale del *web scraper* è quello di estrapolare le informazioni dai siti e di salvarne in un database. Un primo passò è la scelta dei siti da analizzare, analizzando il loro contenuto e le informazioni che hanno da offrire. Inoltre è anche importante considerare il formato dei dati e le modalità di navigazione e visualizzazione del sito, fattori importanti nella successiva scelta degli strumenti da utilizzare. Successivamente si passa all'identificazione dei dati di interesse, che nel nostro caso saranno per esempio il produttore del modello, il nome, l'anno e altre specifiche tecniche riportate dal sito al fine di preservare e documentare in modo completo l'eredità dei modelli. Si pianifica poi la creazione di tabelle che riflettano la struttura dei modelli , con campi specifici per ciascuna caratteristica rilevante. Questo assicura che i dati siano organizzati in modo coerente e facilmente navigabile nel database. Infine, una volta individuate le soluzioni più efficaci si passa all'effettiva realizzazione, che comprende la scrittura del codice, la fase di testing e ottimizzazione per migliorarne l'efficienza.

3.2 Analisi dei siti target

In questo paragrafo si andranno ad analizzare i siti scelti per l'estrapolazione, prendendo in considerazione la loro struttura e le modalità con la quale offrono ai navigatori i dati.

3.2.1 Primo sito: www.1000bit.it

È uno tra i primi siti dedicati al Retrocomputing ed è attivo da oltre 20 anni. Inoltre, possiede uno dei più grandi database di dispositivi ed immagini.

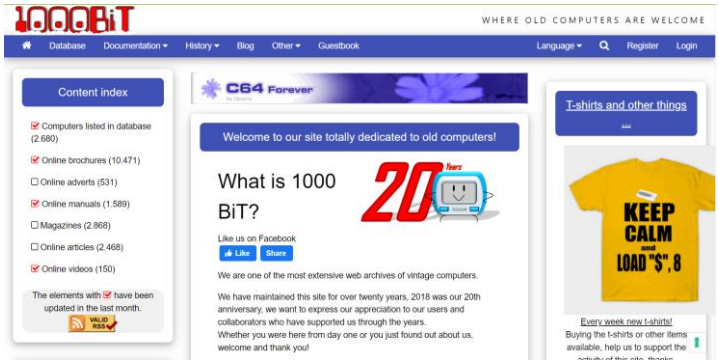

Fig. 3.1 – Schermata iniziale di www.1000bit.it

Il sito inizialmente è in italiano ma è possibile modificare la lingua per ottenere i risultati in inglese. I dispositivi possono essere raggruppati per costruttore, anno, nazione o processore e possono essere visualizzati variando un parametro id nell'URL (evidenziato in rosso di seguito) . Per esempio, al dispositivo Apple II, dell'omonima casa produttrice , viene assegnato l'id 41 e di conseguenza il link alla sua pagina web è il seguente: *https://www.1000bit.it/scheda.asp?id=41*.

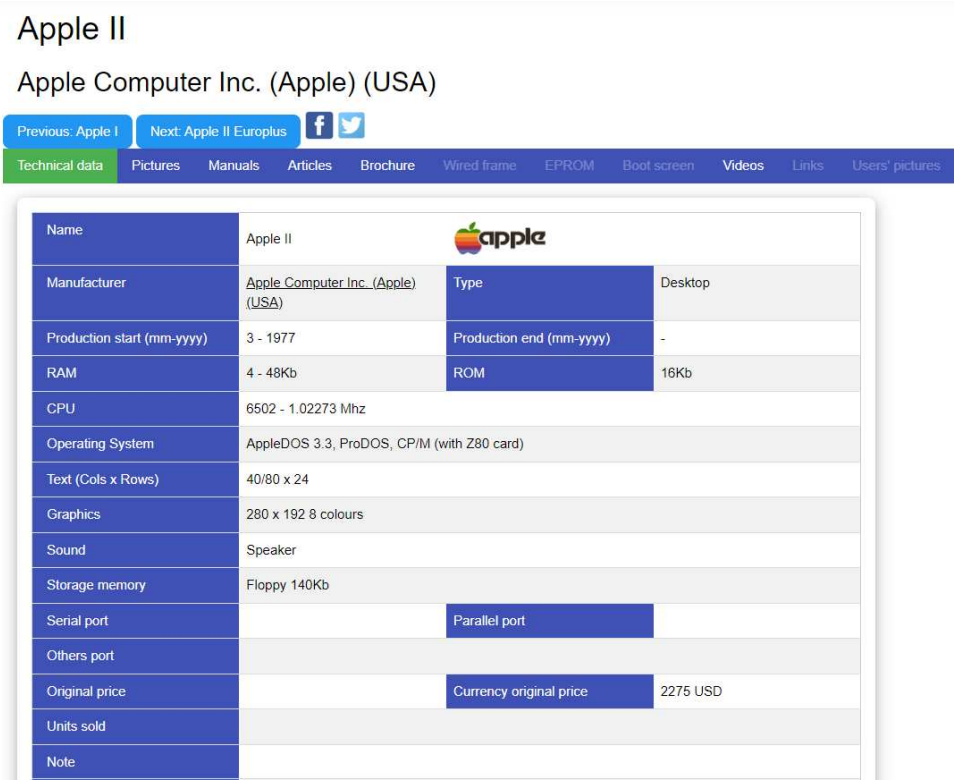

Fig. 3.2 – Schermata del dispositivo *Apple II*

Nella schermata si nota la tabella con i dati tecnici di interesse e una sezione *Pictures* per visualizzare le immagini con le relative caption.

3.2.2 Secondo sito: www.vintage-computer.com

Questo sito include una collezione di dispositivi costruiti prima del 1985 che di conseguenza vengono dunque classificati come antichi.

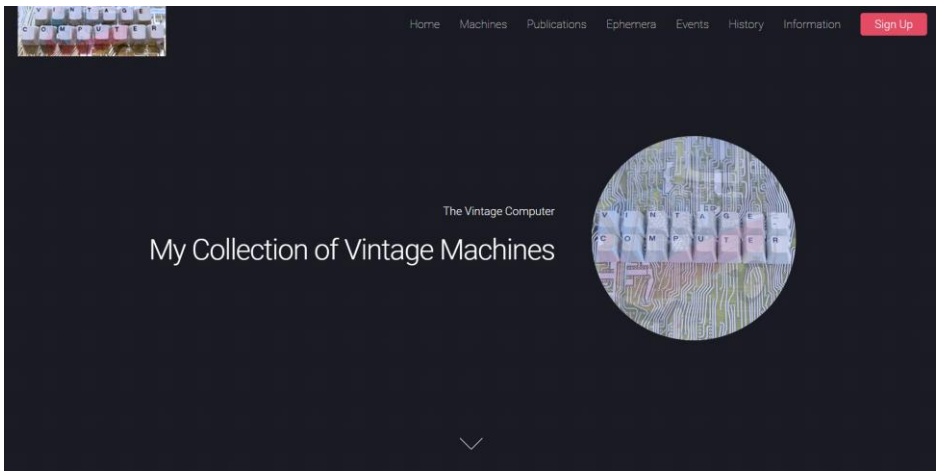

Fig. 3.3 – Schermata iniziale di www.vintage-computer.com

Qui si possono trovare un centinaio di dispositivi e i relativi dati tecnici direttamente in lingua inglese, il che rappresenta una importante aggiunta per la base dati. Ogni apparecchiatura possiede un link comprendente il suo nome, per questo è necessario estrarre direttamente da questa pagina tutti gli indirizzi web di interesse. Una volta fatto ciò, navigando per esempio all'URL *https://www.vintage-computer.com/machines.php?apple2* si presenta la seguente schermata:

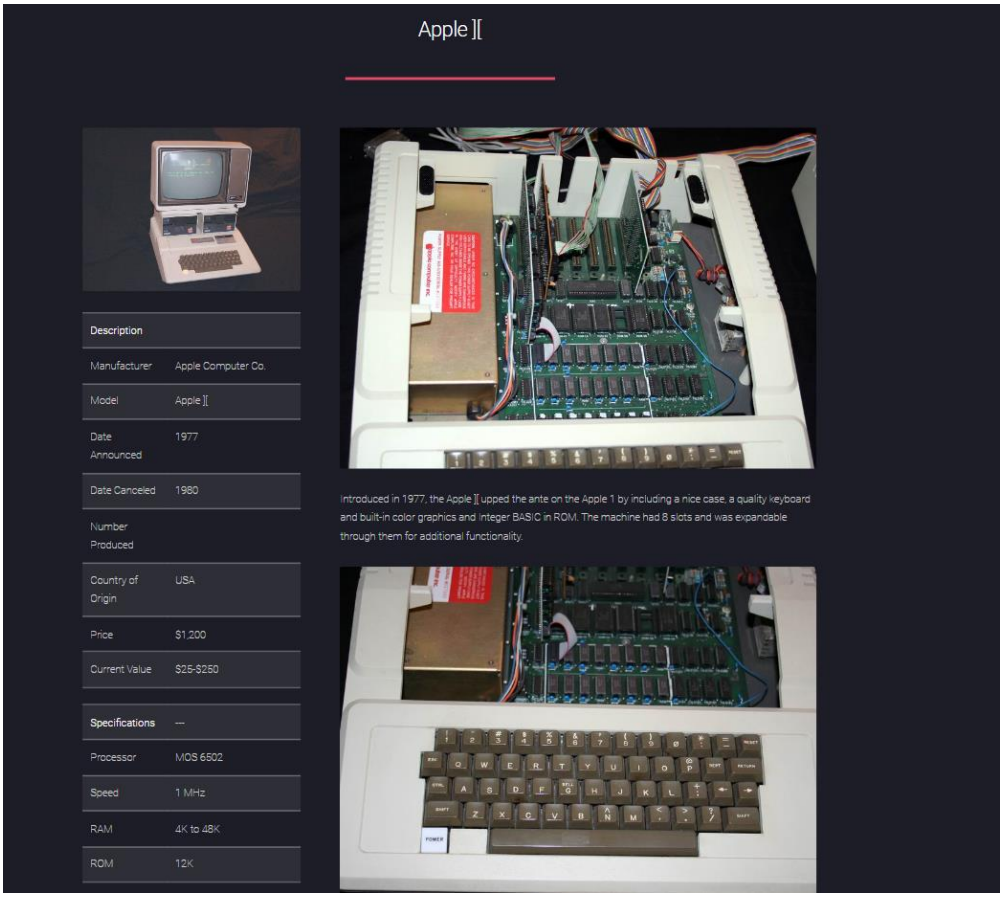

Fig. 3.4 – Schermata del dispositivo *Apple II*

Sulla parte sinistra si trova la tabella con i dati tecnici mentre la restante parte è occupata dalle immagini con le relative descrizioni sottostanti ad esse.

3.2.3 Terzo sito: www.thepcmuseum.net

È uno dei più grandi siti e raccoglie dispositivi costruiti a partire dal 1973 fino al presente.

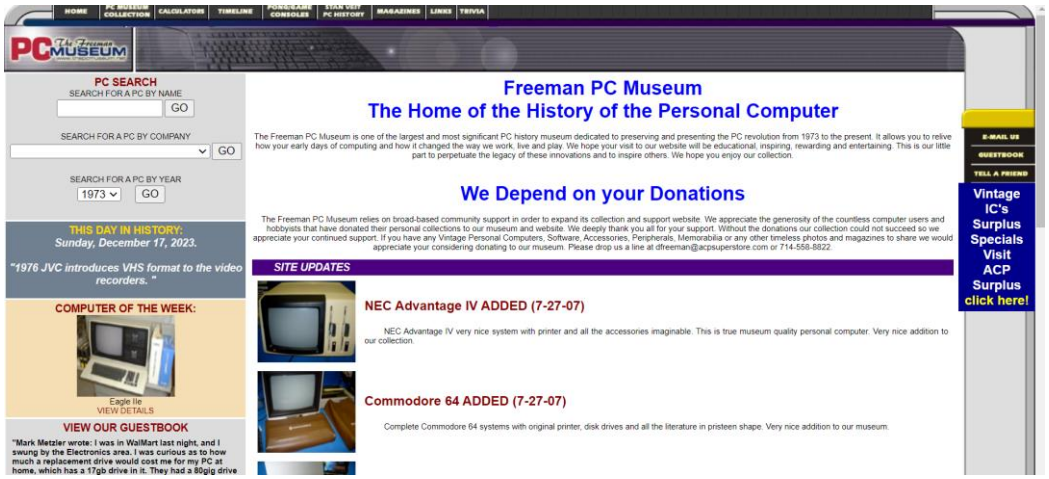

Fig. 3.5 – Schermata iniziale di www.thepcmuseum.net

Il sito offre la possibilità di visualizzare i dispositivi in ordine alfabetico, per costruttore o per anno. Per il nostro scopo è sufficiente individuare il link di un singolo dispositivo e far variare il parametro id (evidenziato in rosso di seguito) contenuto in esso; per il dispositivo Apple II in questo caso l'id è il 15, ottenendo l' URL :
http://www.thepcmuseum.net/details.php?RECORD_KEY%28museum%29=id&id(museum)=15.

Apple
Apple
Apple II

Apple II introduced in April 1977 at the West Coast Computer Faire. This system has a Versa PROM burner and a Videx keypad. Features 6502 cpu, 4Kb RAM, built-in color graphics and (8) Apple II bus expansion slots. First software included Breakout and Pong. The Apple II took off and became one of the most successful computers of its time. Apple II was replaced by the Apple II Plus that shipped significantly more than the Apple II. Note: This item is no longer part of our collection as it has been sent to a new PC museum for display.

MORE INFO

SPECIFICATIONS:

NAME	Apple II
MANUFACTURER	Apple
TYPE	Home Computer
ORIGIN	USA
YEAR	1977
LAST RUN	
QUANTITY BUILT	
OPERATING SYSTEM	BASIC in ROM
CPU	6502
SPEED	1MHz
RAM	4-48Kb
ROM	
TEXT MODES	40 x 24 text
GRAPHIC MODES	280 x 192 (4 colors) 40 x 16 (16 colors)
I/O PORTS	joystick, video composite
POWER SUPPLY	
PRICE	$1298 w/4K

Fig. 3.6 – Schermata del dispositivo *Apple II*

Notiamo la presenza dell'immagine dell'apparecchio nella parte superiore seguita da una breve descrizione e dalla tabella che riporta i suoi dati tecnici.

È interessante condividere il fatto che, come riporta il sito, *"Uno dei consigli sul manuale di istruzioni dell'Apple II per risolvere i problemi di surriscaldamento era di sollevarlo di 5cm e farlo cadere"* .

3.3 Strumenti utilizzati

3.3.1 Python

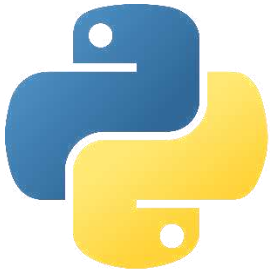

Fig. 3.7 - Logo di Python

Ho scelto Python come linguaggio di sviluppo per diverse ragioni fondamentali.
Innanzitutto è noto per la sua semplicità, chiarezza e flessibilità, fattori chiave che hanno reso questa lingua un'opzione ideale per la realizzazione di uno scraper efficiente e ben strutturato.
Inoltre, Python dispone di una vasta gamma di librerie e framework specificamente progettati per il web scraping, come *Requests e BeautifulSoup*. Questi strumenti semplificano notevolmente le operazioni di navigazione attraverso le pagine web, l'identificazione degli elementi e l'estrazione dei dati, consentendo uno sviluppo più rapido ed efficace del web scraper.

3.3.2 Beautiful Soup

Fig. 3.8 - Logo di Beautiful Soup

Beautiful Soup è una libreria Python che fornisce strumenti per estrarre informazioni da pagine HTML e XML in modo agevole. La sua principale funzione è analizzare documenti HTML o XML, rendendo più semplice la navigazione e l'estrazione di dati specifici da queste pagine. L'ho utilizzata nello scraper per ricavare i dati dal html estratto dalla pagina.

3.3.3 Asyncio

Fig. 3.9 - Logo di Asyncio

È una libreria Python che fornisce supporto per la scrittura di codice asincrono utilizzando il paradigma di programmazione asincrona. Consente di gestire attività concorrenti in modo efficiente senza dover utilizzare thread multipli o processi. Nel codice viene utilizzata per eseguire gli scraper in modo concorrente ottimizzando notevolmente i tempi di esecuzione dell'intero programma

3.3.4 Aiohttp

Fig. 3.10 - Logo di Aiohttp

È una libreria Python che offre supporto per eseguire richieste HTTP asincrone in un ambiente asyncio, progettata per lavorare con il paradigma di programmazione asincrona di Python. Consente di gestire molte richieste contemporaneamente senza bloccare l'esecuzione del programma.
Le sue funzionalità mi hanno permesso di eseguire le richieste internet in modo asincrono, risparmiando tempo e dunque ottimizzando il tempo di esecuzione del codice.

3.3.5 SqLite3

Fig. 3.11 - Logo di SqLite3

È un modulo Python che fornisce un'interfaccia per interagire con il database SqLite, un motore di database SQL leggero incorporato nella libreria standard di Python con cui possibile creare, manipolare e interrogare database *SqLite* senza la necessità di installare o configurare un server di database separato. L'ho scelto per la creazione del database perché è già incorporato Python e per la sua praticità.

3.3.6 Altre librerie

Le altre librerie che ho utilizzato sono:
- **Datetime:** è un modulo per la creazione di date e la gestione degli orari, che semplifica le operazioni con variabili di tempo. L'ho utilizzato per determinare l'istante di esecuzione delle varie istruzioni e per determinare le tempistiche di esecuzione del programma.
- Logging: fornisce un'infrastruttura flessibile per la stampa a video e la registrazione dei messaggi delle applicazioni. L'ho utilizzato per stampare a video i vari progressi dell'esecuzione del programma con il relativo tempo di esecuzione.

3.4 Progettazione del database

In questa sezione si analizzano le scelte progettuali del database.

3.4.1 Individuazione dei dati

Per individuare i dati da inserire nel database si analizzano i vari siti e si determinano i vari parametri che essi presentano online. In tutti i siti si trovano innanzitutto il nome del produttore, il nome e il tipo di dispositivo, la sua origine, il prezzo, il numero di unità prodotte, l'anno di produzione e una serie di specifiche tecniche tra cui la CPU, RAM, ROM e le porte di Input e Output. Ad ogni dispositivo infine può essere correlata una serie di immagini che possono avere una descrizione.

3.4.2 Progettazione concettuale

Nella progettazione concettuale andiamo a definire le entità con le relative chiavi e attributi per poi definire le relazioni con le rispettive cardinalità e infine realizzare lo schema Entità-Associazione.

- MANUFACTURER: è l'entità che rappresenta il produttore e ha come chiave primaria l'attributo *name* il quale identifica unicamente ogni azienda produttrice di dispositivi. Se un

produttore viene inserito nel database vuol dire che avrà almeno distribuito un modello quindi la relazione tra con l'entità model avrà cardinalità uno a molti (1,n).

- DEVICE: è un'entità debole che rappresenta il dispositivo creato dal produttore. Viene identificata dall'attributo *name* del dispositivo e da *name* del produttore in quanto modelli di produttori diversi potrebbero avere lo stesso nome. Come attributi secondari ha *type, origin, year, price, quantity_built, RAM, ROM, CPU, io_ports.* Ogni dispositivo ha uno ed unico produttore quindi la cardinalità della relazione con il produttore è uno a uno. Infine, ogni dispositivo può essere dotato o meno di una o più immagini e questo fa sì che la cardinalità della relazione con l'entità immagine sia 0 a molti (0,n).
- IMAGE: è l'entità che rappresenta un'immagine ed è identificata dall'attributo *link* che è certamente univoco in quanto un link web porta ad un unico risultato. Inoltre, presenta l'attributo *caption* che rappresenta l'eventuale descrizione dell'immagine e ha cardinalità uno a uno perché ogni immagine è associata ad uno e unico dispositivo.

Lo schema Entità-Relazione risultante è quindi:

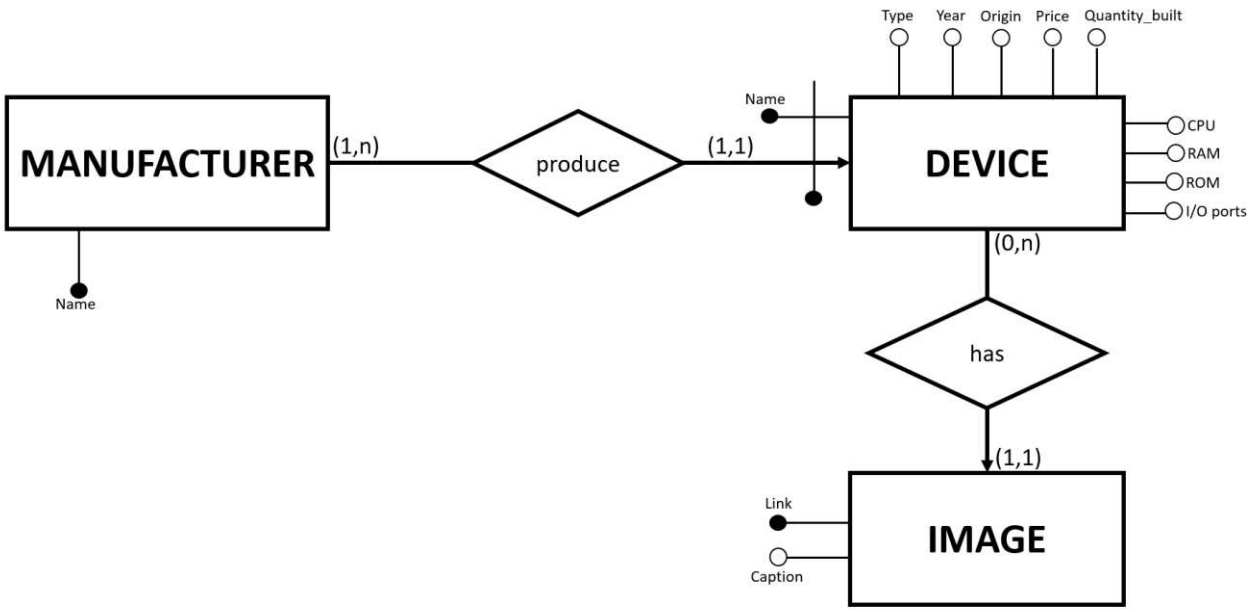

Fig. 3.12 – Schema ER

3.4.3 Progettazione Logica

Lo schema relazionale è il seguente:

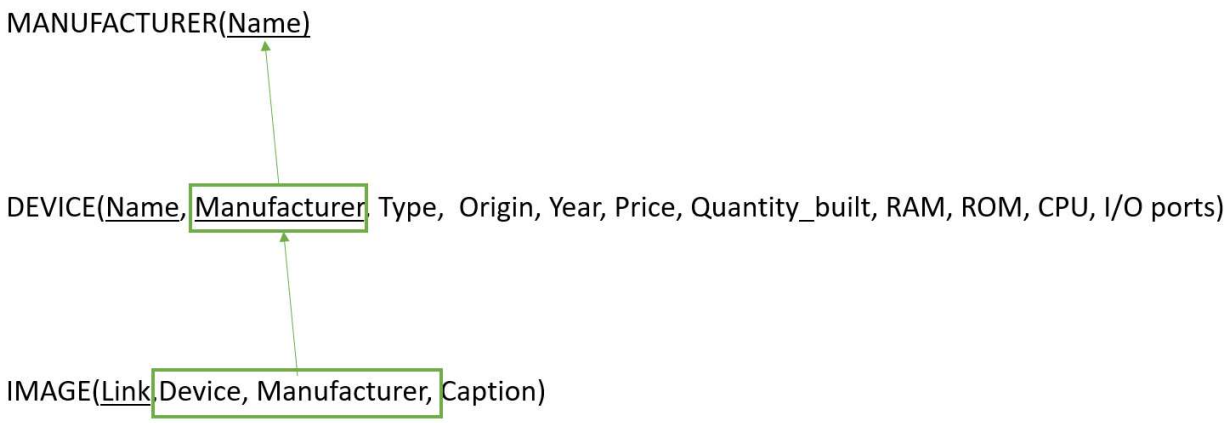

Fig. 3.13 – Schema relazionale

3.4.4 Implementazione del codice

Per realizzare la base dati precedentemente descritta ho utilizzato le seguenti istruzioni SQL.

- *Per creare la tabella manufacturer:*

```
CREATE TABLE IF NOT EXISTS manufacturer (
                name TEXT PRIMARY KEY)
```

- *Per aggiungere un'istanza alla tabella:*

```
INSERT OR REPLACE INTO manufacturer (name)
        VALUES (?)
```

- *Per creare la tabella device con i rispettivi attributi:*

```
CREATE TABLE IF NOT EXISTS device (
            manufacturer TEXT,
            name TEXT,
            type TEXT,
            origin TEXT,
            year TEXT,
```

```
        quantity_built TEXT,
        cpu TEXT,
        ram TEXT,
        rom TEXT,
        io_ports TEXT,
        price TEXT,
        PRIMARY KEY (manufacturer, name) )
```

- *Per aggiungere un'istanza alla tabella:*

```
INSERT OR REPLACE INTO device (
            manufacturer, name, type, origin, year, quantity_built, price, cpu, ram, rom, io_ports
        ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
```

- *Per creare la tabella image con i rispettvi attributi:*

```
CREATE TABLE IF NOT EXISTS image (
            img_link TEXT PRIMARY KEY,
            caption TEXT,
            name TEXT,
            manufacturer TEXT,
            FOREIGN KEY (name, manufacturer) REFERENCES device(name, manufacturer))
```

- *Per aggiungere un'istanza alla tabella:*

```
INSERT OR IGNORE INTO image (
            img_link, caption, name, manufacturer
        ) VALUES (?, ?, ?, ?)
```

4 Creazione del webscraper

4.1 Approccio generale

Nella scrittura del codice del web scraper ho utilizzato un approccio modulare, creando una classe per la gestione del database, una classe per ciascun scraper di un sito, e alcuni classi contenenti funzioni utili per ogni estrattore. Questa tecnica conferisce al codice riusabilità, perché permette di usare lo stesso codice in più parti del programma, semplicità nella fase manutenzione e debugging e infine fornisce un programma più pulito, ordinato e comprensibile.
Un'ulteriore scelta progettuale è stata quella di utilizzare funzione asincrone in modo che i tre estrattori venissero eseguiti contemporaneamente in modo autonomo, aggiungendo i dati estratti al database una volta terminata l'estrazione. Questo ha permesso di ridurre notevolmente le tempistiche di esecuzione dell'intero programma.

4.2 Analisi del codice

In questa sezione si analizzano le varie funzioni che compongono il codice, descrivendo in dettaglio le loro funzionalità. Il codice completo è disponibile a questo link: https://github.com/FilippoFrance/Retrocomputing_scraper e comprende:

-Il file *main.py* contenente l'intero programma da eseguire.

-Il file *requirements.txt* necessario per l'installazione dei moduli Python richiesti per l'esecuzione.

-Il file *retrocomputing.db,* un database di esempio contenente le istanze registrate dopo un'esecuzione del programma.

4.2.1 Classe di gestione del Database

```
class ComputerDatabase:

    def __init__(self, db_name='retrocomputing.db'):
        # Inizializzazione della connessione al database
        self.conn = sqlite3.connect(db_name)
        self.cursor = self.conn.cursor()
        # Creazione delle tabelle nel database
        self.create_tables()
```

```
def create_tables(self):

        #crea la tabella manufacturer
        self.cursor.execute('''
                    CREATE TABLE IF NOT EXISTS manufacturer (
                        name TEXT PRIMARY KEY
                    )
                ''')

        # Crea la tabella device
        self.cursor.execute('''
            CREATE TABLE IF NOT EXISTS device (
                manufacturer TEXT,
                name TEXT,
                type TEXT,
                origin TEXT,
                year TEXT,
                quantity_built TEXT,
                cpu TEXT,
                ram TEXT,
                rom TEXT,
                io_ports TEXT,
                price TEXT,
                PRIMARY KEY (manufacturer, name)
                FOREIGN KEY (manufacturer) REFERENCES manufacturer(name)
            )
        ''')

        # Crea la tabella images
        self.cursor.execute('''
                CREATE TABLE IF NOT EXISTS image (
                    img_link TEXT PRIMARY KEY,
                    caption TEXT,
                    name TEXT,
                    manufacturer TEXT,
                    FOREIGN KEY (name, manufacturer) REFERENCES device(name,
manufacturer)
                )
        ''')

        # Conferma le modifiche al database
```

```
        self.conn.commit()

    def insert_data(self, device_data):
        # Estrae i dati del dispositivo e delle immagini dal dizionario
        device = device_data['DEVICE']
        images = device_data['IMAGES']

        # Inserisci i dati nella tabella manufacturer
        self.cursor.execute('''
            INSERT OR REPLACE INTO manufacturer (name)
            VALUES (?)
        ''', (device.get('MANUFACTURER'),))

        # Inserisci dati nella tabella device
        self.cursor.execute('''
                INSERT OR REPLACE INTO device (
                    manufacturer, name, type, origin, year, quantity_built,
price, cpu, ram, rom, io_ports
                ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
            ''', (
            device.get('MANUFACTURER'),
            device.get('NAME'),
            device.get('TYPE', None),
            device.get('ORIGIN', None),
            device.get('YEAR', None),
            device.get('QUANTITY BUILT', None),
            device.get('PRICE', None),
            device.get('CPU', None),
            device.get('RAM', None),
            device.get('ROM', None),
            device.get('I/OPORTS', None)
        ))

        # Inserisci dati nella tabella images
        for image in images:
            self.cursor.execute('''
            INSERT OR IGNORE INTO image (
                img_link, caption, name, manufacturer
            ) VALUES (?, ?, ?, ?)
            ''', (
                image.get('img_link'), image.get('caption'), device.get('name'),
device.get('manufacturer')))
```

```
        # Conferma le modifiche al database
        self.conn.commit()

    def close_connection(self):
        # Chiude la connessione al database
        self.conn.close()

    def insertion(self, results):
        # Crea un'istanza del gestore del database
        db_handler = ComputerDatabase()
        for entry in results:
            # Inserisce i dati nel database
            db_handler.insert_data(entry)
        # Chiude la connessione al database
        db_handler.close_connection()
```

Questa classe presenta innanzitutto la funzione *__init__* la quale inizializza la connessione alla base dati denominata *retrocomputing_database.db* e crea le tabelle tramite la funzione *create_tables. È* presente poi la funzione *insert_data,* utilizzata dai web scraper una volta terminata l'estrazione per inserire tutti i dati nell'archivio. Si nota come alcuni attributi, ad eccezione delle chiavi primarie, possono assumere valori nulli, nel caso non fossero disponibili nei siti. Inoltre, se si verifica un tentativo di inserimento di un'istanza con stessa chiave di una già presente, essa andrà a rimpiazzare quest'ultima. Infine, è presente la funzione *close_connection,* per terminare la connessione una volta completato l'accesso e la funzione *insertion* la quale ha come argomento i risultati dell'estrazione di uno scraper da inserire nella base dati.

4.2.2 Classi di utilità

```
class ExtractionHandler:

    # Funzione per estrarre i valori da una tabella HTML
    async def get_data_from_table(self, table_data):
        device = {}

        # Estrazione dei dati dalla tabella
        for row in table_data.find_all('tr'):
            columns = row.find_all('td')
```

```
            # Estrae il valore dalla colonna 0 e lo utilizza come chiave
            key = columns[0].text.strip().replace('\xa0', '')
            value = columns[1].text.strip().replace('\xa0', '')
            device[key] = value

            try:
                # Prova a estrarre il valore dalla colonna 2 e lo aggiunge al
dizionario
                key = columns[2].text.strip().replace('\xa0', '')
                value = columns[3].text.strip().replace('\xa0', '')
                device[key] = value
            except:
                pass  # Ignora il caso in cui non ci sono ulteriori colonne

        return device
```

- *ExtractionHandler:* ha la funzione di aiutare nell'estrazione dei valori da una tabella html. Riceve come argomento il codice html ottenuto dalla richiesta alla pagina web del dispositivo e lo esamina tramite le funzioni della libreria Beautiful Soup al fine di estrapolare i valori dei vari campi della griglia. Per fare ciò divide le varie righe e colonne della tabella e salva i valori in un dizionario che restituisce una volta terminata l'estrazione.

```
class RequestHandler:
    async def connect_async(url, session):
        try:
            # Effettua una richiesta asincrona all'URL specificato
            async with session.get(url) as response:
                # Se la richiesta ha successo (status code 200)
                if response.status == 200:
                    # Legge i dati grezzi della risposta e li analizza con
BeautifulSoup
                    raw_data = await response.read()
                    return BeautifulSoup(raw_data, 'html.parser')
                else:
                    # Restituisce None se la richiesta non ha avuto successo
                    return None
        except:
```

```
            # Restituisce None in caso di errore durante la richiesta
            return None
```

- *RequestHandler*: ha lo scopo di gestire le richieste asincrone svolte durante l'esecuzione del programma. Dopo aver stabilito la connessione all'URL tramite la sessione, ricevuti entrambi come argomento, ritorna i dati elaborati da *Beautiful Soup* se la richiesta è andata a buon fine, *None* altrimenti.

4.2.3 Scraper di www.vintage-computer.com

La struttura dei tre web scraper è molto simile varia solo di alcune righe, di conseguenza andremo ad analizzare nel dettaglio solo quello del sito www.vintage-computer.com il cui codice è riportato qui di seguito.

```
class Scraper2:
    url = 'https://www.vintage-computer.com/'

    # Ottiene i dati e le immagini di ogni dispositivo
    async def get_data(self, data):
        # Estrae la tabella contenente i dati del computer
        table = data.find('table')

        # Estrae tutte le immagini del computer, escludendo la prima
        all_images = data.find_all('img')
        all_images.pop(0)

        device_images = []
        for i in all_images:
            image = {}
            # Costruisce il link completo dell'immagine
            image['img_link'] = self.url + i.get('src')
            try:
                # Tenta di estrarre la didascalia dell'immagine
                image['caption'] = i.find_next('p').text
            except:
                image['caption'] = ''  # Se non ci sono didascalie, imposta una
stringa vuota

            device_images.append(image)

        # Esegue l'estrazione dei dati dalla tabella
```

```
        device = await ExtractionHandler.get_data_from_table(self, table)

        # Adattamento del dizionario a quello degli altri scraper
        device.pop('Description')
        device['MANUFACTURER'] = device.pop('Manufacturer')
        device['NAME'] = device.pop('Model')
        device['TYPE'] = 'Home Computer'
        device['ORIGIN'] = device.pop('Country of Origin')
        device['PRICE'] = device.pop('Price')
        device['QUANTITY BUILT'] = device.pop('Number Produced')
        device['CPU'] = device.pop('Processor')
        device['SPEED'] = device.pop('Speed')
        device['I/OPORTS'] = device.pop('I/O')

        return {'DEVICE': device, 'IMAGES': device_images}

    async def get_links(self):
        async with aiohttp.ClientSession() as session:
            links = []
            data = await RequestHandler.connect_async(self.url, session)

            # Estrae tutti i link che contengono "machines"
            links = [self.url + a['href'] for a in data.find_all('a', href=True)
if 'machines' in a['href']]

            return links

    async def scrape(self):
        # Logging per segnalare l'inizio dell'estrazione dal sito
        logging.info('Estrazione da www.vintage-computer.com')
        results = []

        # Ottiene tutti i link alle pagine dei dispositivi
        links = await self.get_links()

        async with aiohttp.ClientSession() as session:
            # Crea una lista di task asincroni per ciascun link
            tasks = [RequestHandler.connect_async(l, session) for l in links]

            # Esegue tutte le richieste in parallelo e attende i risultati
            responses = await asyncio.gather(*tasks)
```

```
            # Ricava le informazioni di ogni computer
            for res in responses:
                if res is not None:
                    results.append(await self.get_data(res))

        # Logging per segnalare la fine dell'estrazione e il numero di risultati
ottenuti
        logging.info(f'Estrazione terminata per www.vintage-computer.com -
Risultati ottenuti: {len(results)}')

        # Chiama il metodo di inserimento dei dati nel database dalla classe
ComputerDatabase
        ComputerDatabase.insertion(self, results)
```

Lo schema della classe di ciascun web scraper è il seguente:

- Una variabile *url* che fornisce l'indirizzo web di base del sito.
- La funzione asincrona *get_data* che riceve come argomento i dati ottenuti dalla richiesta internet e gli elabora per estrarre i valori di interesse e le immagini con le rispettive eventuali descrizioni.
  Una volta terminata la sua esecuzione ritorna i parametri salvandoli in un dizionario.
- La funzione asincrona *get_links* la quale crea una sessione ed estrae tutti gli indirizzi web dei vari dispositivi per poi salvarli nella lista *links* che ritorna.
- La funzione asincrona *scrape* che gestisce l'intera esecuzione dello scraper e chiama le altre funzioni. Per prima cosa inizializza la variabile *results,* che andrà a contenere i risultati finali, e avvisa tramite il modulo *logging* che è iniziata la sua esecuzione. Successivamente ottiene i link dei vari dispositivi tramite l'apposita e funzione e, dopo aver creato un oggetto sessione, realizza una lista di task asincroni per ogni link ed esegue tutte le richieste in parallelo. Una volta ottenuti i risultati, estrae i dati da quelli diversamente nulli e li salva nella lista *results.* Essa andrà poi inserita nel database tramite l'apposita funzione, non prima di aver avvisato che l'estrazione è terminata.

4.3 Esecuzione del codice

Per eseguire il codice bisogna innanzitutto assicurarsi di avere installato Python nel proprio terminale. Una volta fatto ciò si usa l'istruzione "*pip install -r requirements.txt"* nel terminale per

installare i moduli necessari per il funzionamento del programma e lo si può eseguire tramite l'istruzione "*python main.py"*. L'output che si andrà ad ottenere sarà simile al seguente:

2024-08-03 18:30:46 - INFO - Inizio estrazione
2024-08-03 18:30:46 - INFO - Estrazione da www.1000bit.it
2024-08-03 18:30:46 - INFO - Estrazione da www.vintage-computer.com
2024-08-03 18:30:46 - INFO - Estrazione da www.thepcmuseum.net
2024-08-03 18:30:53 - INFO - Estrazione terminata per www.vintage-computer.com - Risultati ottenuti: 100
2024-08-03 18:31:09 - INFO - Estrazione terminata per www.thepcmuseum.net - Risultati ottenuti: 347
2024-08-03 18:35:48 - INFO - Estrazione terminata per www.1000bit.it - Risultati ottenuti: 1334
2024-08-03 18:35:54 - INFO - Fine estrazione
2024-08-03 18:35:54 - INFO - Tempo impiegato: 0:05:08.255761

Vediamo quindi che in questa esecuzione vengono estratti 1781 risultati totali in poco più di 5 minuti.

5 Conclusioni

È bene notare che il tempo di esecuzione del programma e il numero di risultati ottenuti può variare a seconda della connessione internet, della disponibilità dei siti e della loro eventuale modifica. Inoltre, il numero di risultati ottenuti non coincide poi direttamente con il numero di risultati aggiunti alla base dati in quanto alcuni dispositivi sono catalogati in più siti e avendo imposto il vincolo di unicità di nome del costruttore e modello, essi verranno catalogati una sola volta.
Il programma realizzato è robusto e può essere sviluppato ulteriormente in qualsiasi momento aggiungendo ulteriori classi per estrarre dati da nuovi siti di interesse così da ampliare la collezione di istanze nel database. Un ulteriore fatto di interesse è il recente attacco hacker subito dal sito *www.old-computers.com* uno degli archivi web di vecchi dispositivi più popolare fra gli appassionati. Contattando i suoi amministratori è emerso che il sito è stato reso completamente inagibile, a tal punto da impedirne la navigazione in esso per molto tempo se non addirittura definitivamente. Esistono comunque siti come *archive.org* che salvano i backup di molti indirizzi web i quali però non sempre si presentano come aggiornati e completi. Questo avvenimento ci fa ulteriormente capire come sia importante proteggere e preservare la cultura informatica da ciò che la minaccia, che sia il passare del tempo o l'agire dell'uomo, per impedire che essa finisca irreparabilmente nel dimenticatoio.

Sitografia

Definizione di Retrocomputing - *https://it.wikipedia.org/wiki/Retrocomputing*

Definizione di web scraping - *https://it.wikipedia.org/wiki/Web_scraping*

Informazioni sul web scraping - *https://kinsta.com/it/knowledgebase/web-scraping/*

Elenco di siti sul Retrocomputing - *https://www.computerhistory.it/*

Siti utilizzati per l'estrazione dei dati:

https://www.thepcmuseum.net

https://www.1000bit.it

https://www.vintage-computer.com

Documentazione:

https://www.selenium.dev

https://www.python.org

https://www.pypi.org/project/requests

https://www.pypi.org/project/beautifulsoup4

Repository:

https://github.com/FilippoFrance/Retrocomputing_scraper