1506
UNIVERSITÀ
DEGLI STUDI
DI URBINO
CARLO BO

Corso di Laurea in Informatica Applicata

Reti di Calcolatori – *Progetto sessione estiva a.a. 2020/2021*

"Scraping dati di 3 siti web di previsioni metereologiche per confrontare le previsioni odierne della città italiana selezionata"

Sviluppatori:

Nicolò Santini, mat. 292404

INDICE

PARTE PRIMA – Introduzione

Oggigiorno molti utenti fanno riferimento ad un solo sito web per consultare i dati metereologici o, altrettanto spesso, utilizzano l'applicazione predefinita nel loro smartphone. Poiché i dati riguardanti le previsioni metereologiche possono variare in base all'ente che li fornisce, potrebbe risultare utile ad un utente che necessita di controllare assiduamente le previsioni metereologiche per i più vari motivi (organizzazione di una vacanza, pianificazione di un'attività all'aperto, ecc.), confrontare le previsioni metereologiche di diversi fornitori.
L'idea del presente progetto è quella di mettere a disposizione un semplice tool che a partire dalla località italiana scelta dall'utente, faccia scraping (raschiare) i dati di alcuni siti web, per poi mostrarli all'interessato tutti insieme affinchè possa confrontarli senza la necessità di effettuare diverse ricerche nel web.
I dati presi in considerazione sono quelli dei siti ilmeteo.it, 3bmeteo.com e meteo.it.
La volontà è stata anche quella di considerare i dati del sito meteoam.it (il meteo dell'Aeronautica Militare). Nella relazione verrà successivamente discussa la motivazione per cui ciò non è stato possibile.

Che cos'è il web scraping?
"*Il web scraping (detto anche web harvesting o web data extraction) è una tecnica informatica di estrazione di dati da un sito web per mezzo di programmi software. Di solito, tali programmi simulano la navigazione umana nel World Wide Web utilizzando l'Hypertext Transfer Protocol (HTTP) o attraverso browser, come Internet Explorer o Mozilla Firefox." (1).*
Tale tecnica si presta particolarmente al caso di questo progetto perché si vogliono confrontare dati riguardanti lo stesso problema che si trovano su siti diversi. Non potendo, per ovvie ragioni, avere accesso direttamente ai databases che i fornitori di previsioni meteo utilizzano per memorizzare i loro dati, non essendo nemmeno fornite delle API per recuperare i dati in maniera automatica, l'unico modo possibile per giungere all'obiettivo preposto è quello di "raschiare" i dati da ognuno dei siti in esame per poi raccoglierli assieme, ordinarli e mostrarli all'utente finale.

(1): definizione scraping da Wikipedia (https://it.wikipedia.org/wiki/Web_scraping)

PARTE SECONDA – Scelte Implementative

Linguaggio di Programmazione:

- Per lo sviluppo dello script è stato scelto il Python. La scelta è dovuta al grande numero di librerie fornite dalla community per svolgere i compiti più diversi e al fatto che Python, essendo un linguaggio interpretato, necessita solamente del suo interprete per poter funzionare su qualsiasi dispositivo (purchè supporti l'interprete);

Come recuperare i dati:

- Per recuperare i dati si è deciso di fare uso del protocollo HTTP: nello specifico il client (il tool sviluppato), ogni qualvolta deve interagire con il server, invia delle richieste. Di contro, il server, ad ogni richiesta manda delle risposte.

Che dati leggere:

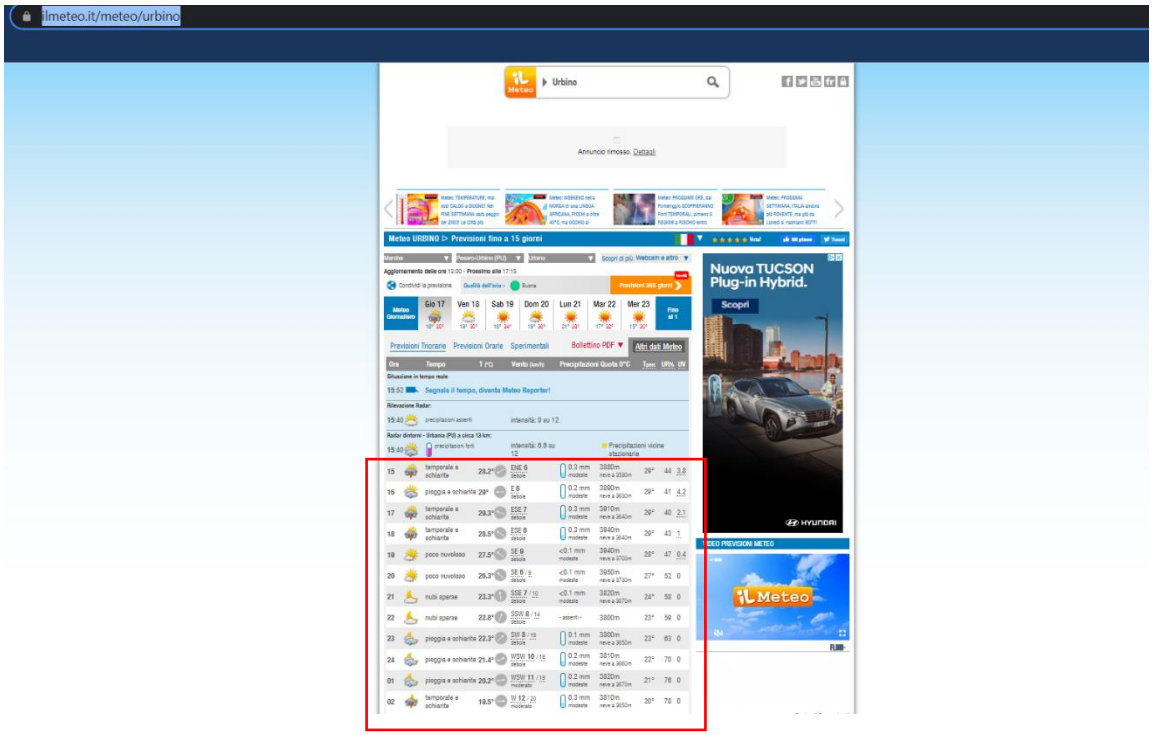
- Quasi tutte le richieste che il tool fa avranno delle risposte che ritornano del codice HTML corrispondente alla pagina da cui si vuole effettuare lo scraping (si veda a pagina 12 l'eccezione);
- Fondamentale in questo caso è l'utilizzo di una delle innumerevoli librerie di Python: per effettuare il parsing della pagina HTML e poter quindi accedere più facilmente ai suoi dati, viene usata la libreria "beautifulsoup4". Tale libreria permette di inserire il codice della pagina web passato nel costruttore della classe BeautifulSoup in un albero, rispecchiando totalmente la logica modulare dell'HTML (ogni elemento è a sua volta composto da sottoelementi che sono a loro volta composti da altri sottoelementi). Ciò permette di accedere alle parti della pagina web che interessano al tool in maniera rapida e semplice.

PARTE TERZA – Struttura dei siti Web di riferimento

3.1 – ilmeteo.it

Effettuare una richiesta HTTP che ritorni i dati meteo di una località al sito ilmeteo.it è molto semplice.
Al server, per ritornare la pagina web contenente le previsioni metereologiche per una località arbitraria, è sufficiente passare il nome della località in coda all'URL, in questo modo:
"https://www.ilmeteo.it/meteo/<località>"

Facendo la prova con Urbino, si ottiene:



Tutto ciò che interessa al tool per poter funzionare è la parte riquadrata in rosso: al suo interno sono contenute tutti i dati meteo.

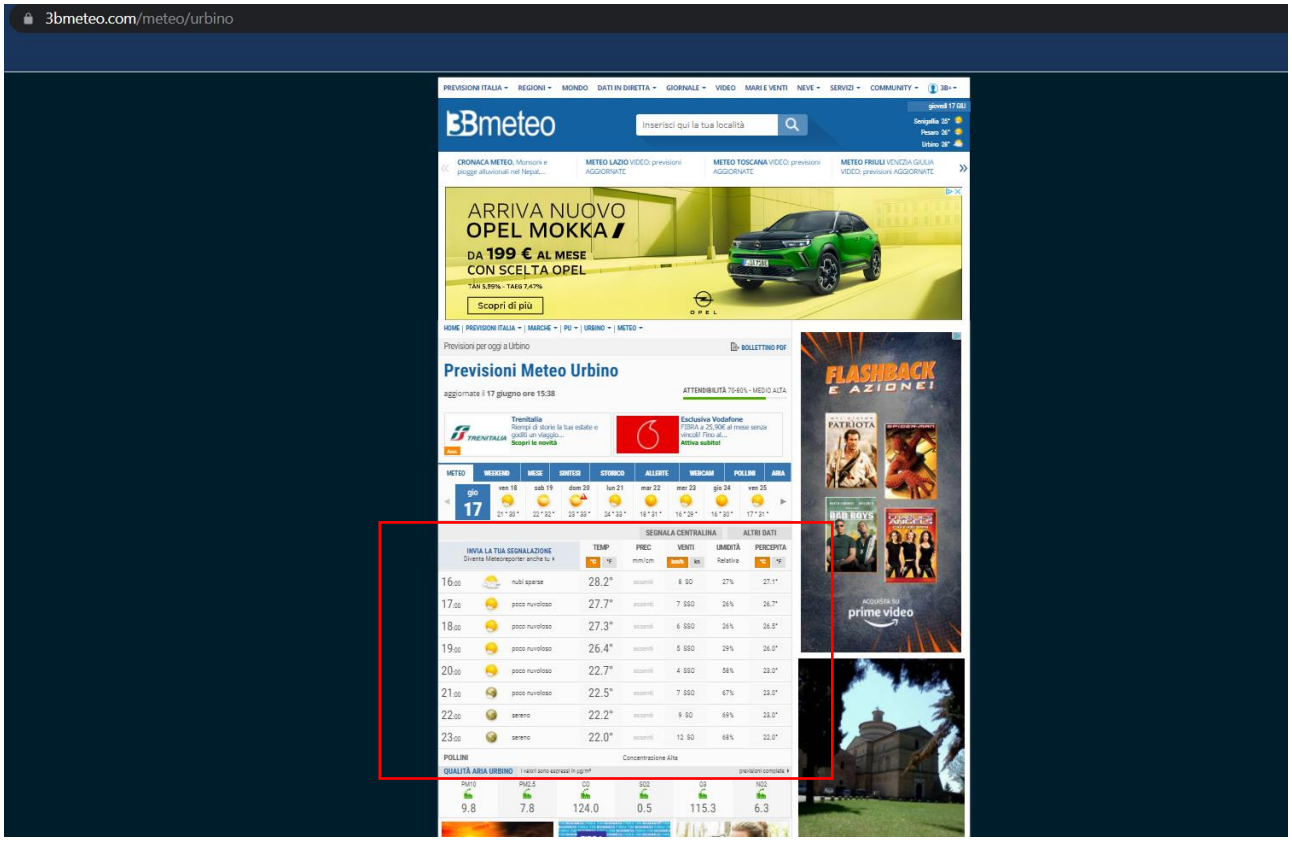
3.2 – 3bmeteo.com

Come per ilmeteo.it, anche effettuare una richiesta HTTP a 3bmeteo.com che ritorni i dati meteo di una località è molto semplice.

Al server, per ritornare la pagina web contenente le previsioni metereologiche per una località arbitraria, è sufficiente passare il nome della località in coda all'URL, in questo modo:

"https://www.3bmeteo.com/meteo/<località>"

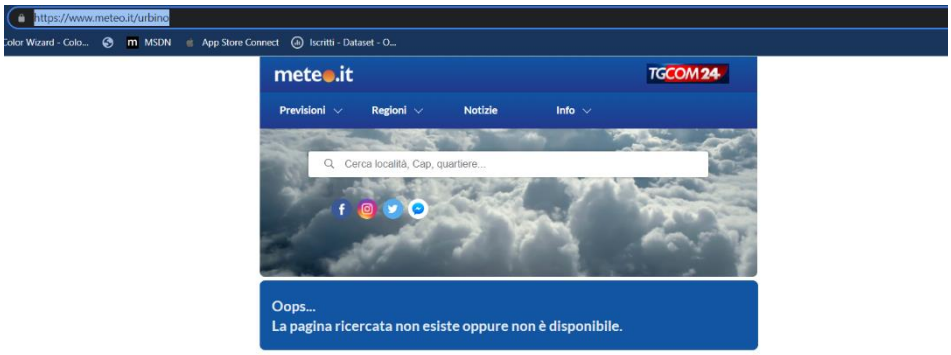
Facendo anche in questo caso la prova con Urbino otteniamo:



Tutto ciò che interessa al tool per poter funzionare è la parte riquadrata in rosso: al suo interno sono contenute tutti i dati meteo.

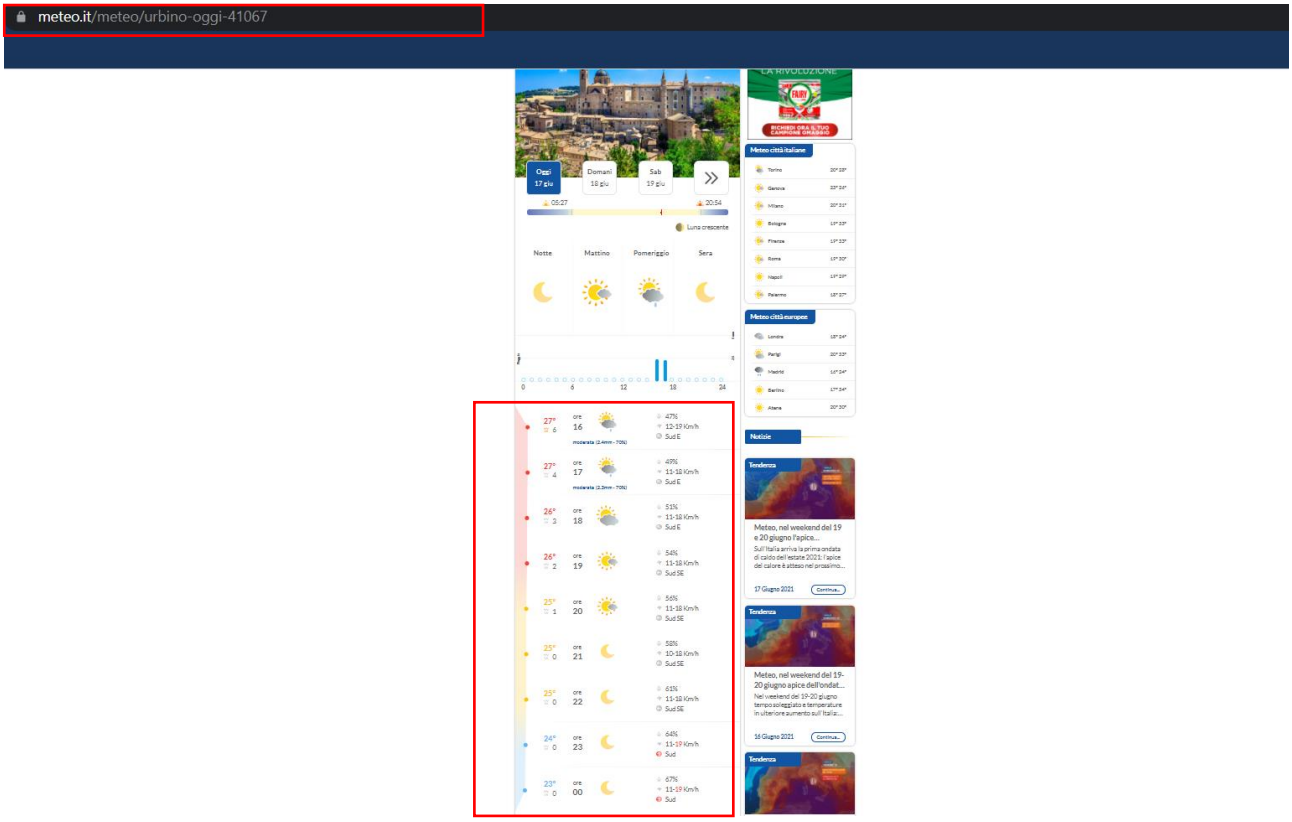
3.3 – meteo.it

Diversamente dagli altri siti analizzati fino ad ora, effettuare una richiesta HTTP passando solamente il nome della località dopo l'URL, per questo sito non funziona:



Il problema a questo punto è capire come deve essere fatto l'URL per poter effettuare correttamente la richiesta affinchè lo script abbia a disposizione la pagina HTML da interpretare.

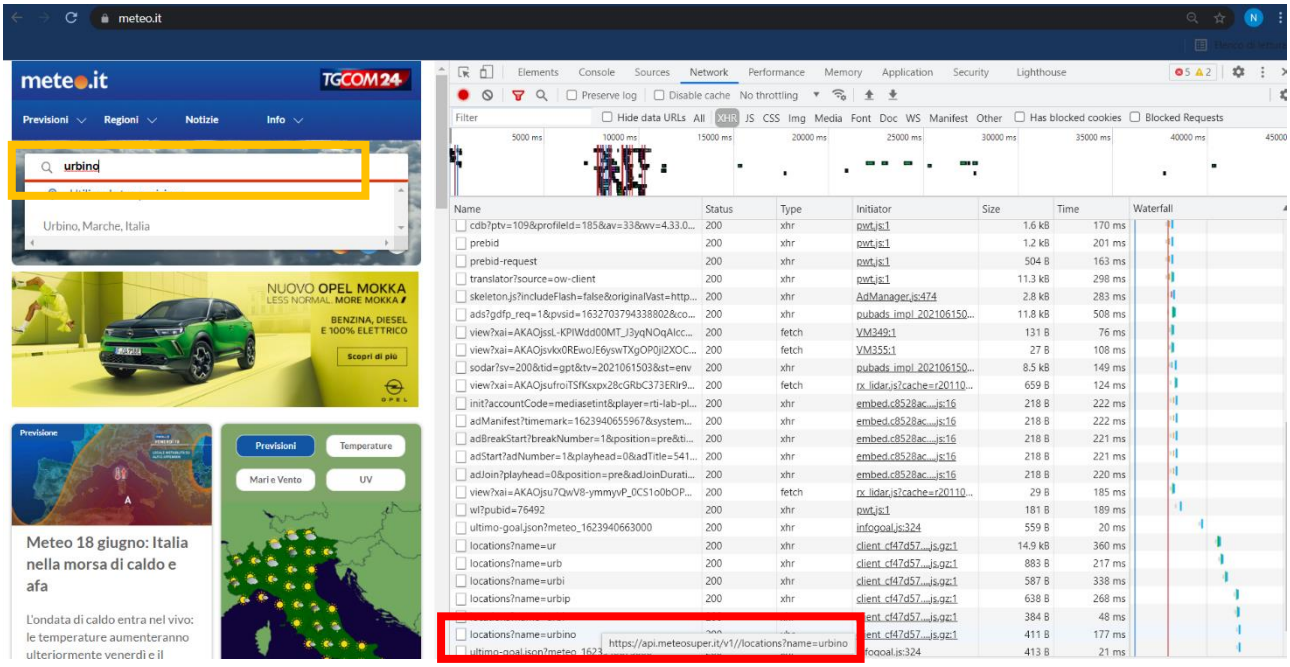
Cercando Urbino dalla barra di ricerca della home page del sito, l'URL su cui si viene reindirizzati contiene il nome della città, seguito da un "-", seguito da "oggi", seguito nuovamente da "-" e per ultimo si vede comparire un codice, diverso per ogni località scelta.



Anche in questo caso come negli altri siti, l'obiettivo è quello di recuperare solamente i dati meteo, contenuti all'interno del riquadro rosso.

Tuttavia, non si è direttamente a conoscenza del codice della città da inserire nell'URL. Non potendo, come affermato nell'introduzione accedere in alcun modo ai databases del sito e non avendo a disposizione delle API dichiarate esplicitamente da cui recuperare i dati, si è reso necessario fare uso dello strumento per gli sviluppatori del browser, auspicando di trovare degli "indizi" che consentano di arrivare alla conclusione e trovare quindi l'id della città di cui l'utente vuole conoscere le previsioni meteo.

Nell'immagine in basso si vede aperto lo strumento per gli sviluppatori di Google Chrome. In particolare nella sezione Network->XHR è possibile vedere le richieste HTTP effettuate dal browser al server per ottenere una pagina web dinamica. Tra queste, mentre viene digitata nella barra di ricerca della home page del sito 3bmeteo.com (riquadro giallo), compare una richiesta (riquadro rosso) che fa al caso del tool.



Cercando quindi dalla barra di ricerca (nella pratica si fa anche in questo caso una richiesta HTTP) il link che compare nel popup dentro il riquadro rosso, ossia "https://api.meteosuper.it/v1/locations?name=<localita>", viene restituito un array di JSON contenenti le località che nel nome contengono la stringa digitata finora nella barra di ricerca (nel caso specifico del tool la città intera). Nel caso di Urbino viene restituita questa stringa:

```
[{"name":"Urbino","idLocation":"0041067","region":"Marche","nation":"Italia","rating":1,"
rank":6,"englishVersion":false}]
```

Il campo "idLocation" è il campo che interessa al tool per poter effettuare la richiesta: esso è infatti l'ID da mettere in coda all'URL per poter ricevere la pagina web con le previsioni per la località scelta.

3.4 – Conclusione

In conclusione a questa sezione, si riassumono il formato degli URL accettati dai server delle previsioni metereologiche per mostrare il meteo della giornata odierna e a fianco il numero delle richieste HTTP necessarie al tool sviluppato per poter leggere correttamente la pagina e interpretarla:

	Formato URL	**Numero Richieste**
ilmeteo.it	https://www.ilmeteo.it/meteo/citta	1
3bmeteo.com	https://www.3bmeteo.com/meteo/citta	1
meteo.it	https://www.meteo.it/citta-oggi-idCitta	2

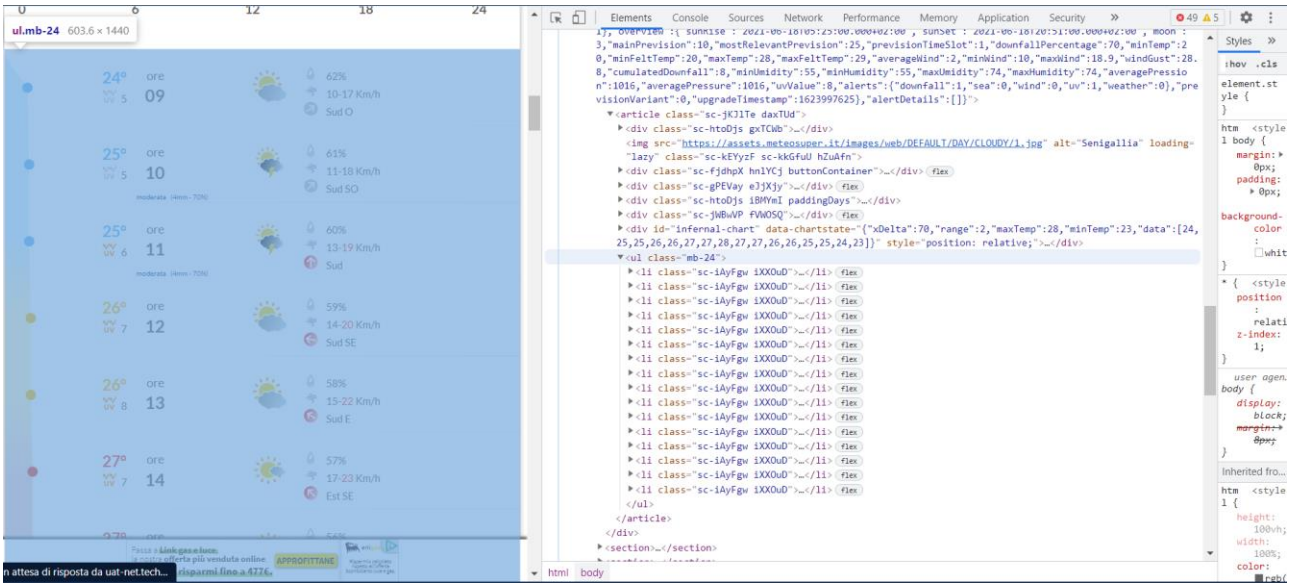
Per ilmeteo.it e per 3bmeteo.com la richiesta va fatta all'URL in tabella sostituendo il parametro "citta" con il nome della città che il tool ha preso in input, mentre per meteo.it vanno fatte due richieste: la prima all'API descritta in precedenza per ricavare l'ID della città, poi va sostituita il parametro "citta" con la città presa in input da tool e il parametro "idCitta" con l'ID recuperato dalla richiesta effettuata in precendenza.

PARTE QUARTA – Il pacchetto beautifulsoup4

Il codice HTML è un codice modulare: tutti gli elementi visuali di cui la pagina web è costituita sono fatti da tag o sono contenuti in tag. Ogni tag può essere quindi un tag genitore e contenere altri tag al suo interno che sono detti tag figli. Questa particolare struttura del codice HTML può essere equiparata ad un albero i cui nodi con dei figli sono i tag genitori, mentre i figli sono i tag figli. Le foglie corrispondono invece ai tag senza alcun figlio.
Lo scopo del pacchetto beautifulsoup4, dopo aver preso in input del codice HTML, è quello di "parsare" il codice e renderlo disponibile in una struttura ad albero al programmatore che necessita di analizzarlo.
Il compito del tool sviluppato è quello di estrapolare da ogni pagina web i dati riguardanti la situazione metereologica, l'ora di riferimento e la temperatura. Il pacchetto beautifulsoup4, viste le premesse, è uno strumento ideale per effettuare la ricerca di questi dati all'interno del "parsing tree" senza dover ricorrere ad analizzare carattere per carattere tutta la stringa contenente il codice HTML della pagina.

Come usarlo nella pratica?

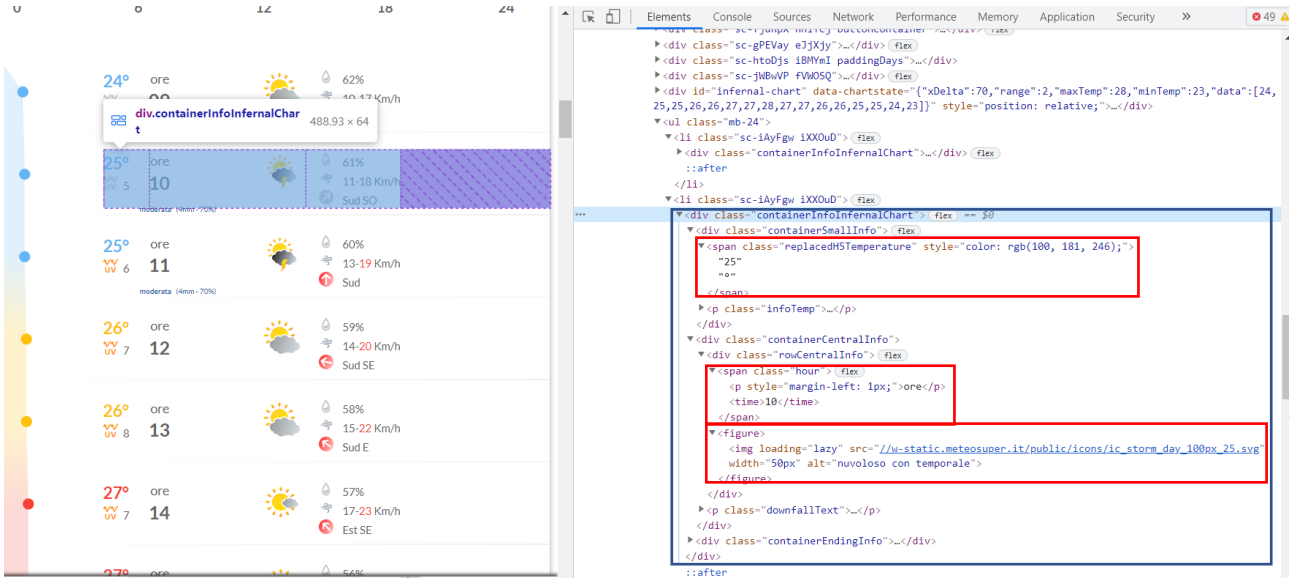
Mediante lo strumento di sviluppo fornito da Google Chrome, per ogni pagina si è analizzato il codice HTML delle parti che interessano al tool per funzionare (quelle riquadrate in rosso nelle immagini del capitolo precedente) e sono stati ricercati i tag nei quali compare l'informazione riguardante l'ora, la situazione metereologica e la temperatura.
Viene di seguito riportato l'esempio di un solo sito web (meteo.it) il procedimento è analogo per gli altri, con la sola differenza che i tag ovviamente hanno altro nome e altri attributi.



Il tag “ul” il cui attributo “class” vale “mb-24” è il tag che contiene tutta la tabella con i dati di interesse.

All’interno del tag “ul”, come nodi figli, si trovano dei tag “li” il cui parametro “class” vale “sc-iAyFgw iXXOuD”. Ciascuno di questi tag “ul” corrisponde ad una riga della tabella dei dati.

Si mostra ora come questo ultimo tag è organizzato internamente.



All’interno del rettangolo blu sono visibili 3 rettangoli rossi che sono i punti del codice HTML in cui si trovano le informazioni di interesse. E’ bene specificare che ogni riga della colonna ha i dati all’interno di tag aventi lo stesso nome e gli stessi attributi, quindi per effettuare la ricerca è sufficiente eseguire un ciclo che scorra iterativamente tutte le righe della tabella.

Per trovare l’ora di riferimento, mediante l’uso del pacchetto beautifulsoup4, bisogna cercare tutti i valori testuali contenuti nel tag “time”, contenuti a loro volta nel tag “span” con attributo “class” uguale a “hour”.

Il codice in Python per fare ciò è il seguente:

```
for ora in val.find_all('span', attrs = {'class': 'hour'}):
        ora = ora.find("time")
```

dove val è il BeautifulSoup ritornato dal pacchetto cercando all'interno della tabella.

Il procedimento è analogo per gli altri valori, notando tuttavia che per il dato sulla situazione, in questo caso, bisogna leggere il valore del attributo "id" del sottotag "img".

PARTE QUINTA – Il problema del sito meteoam.it

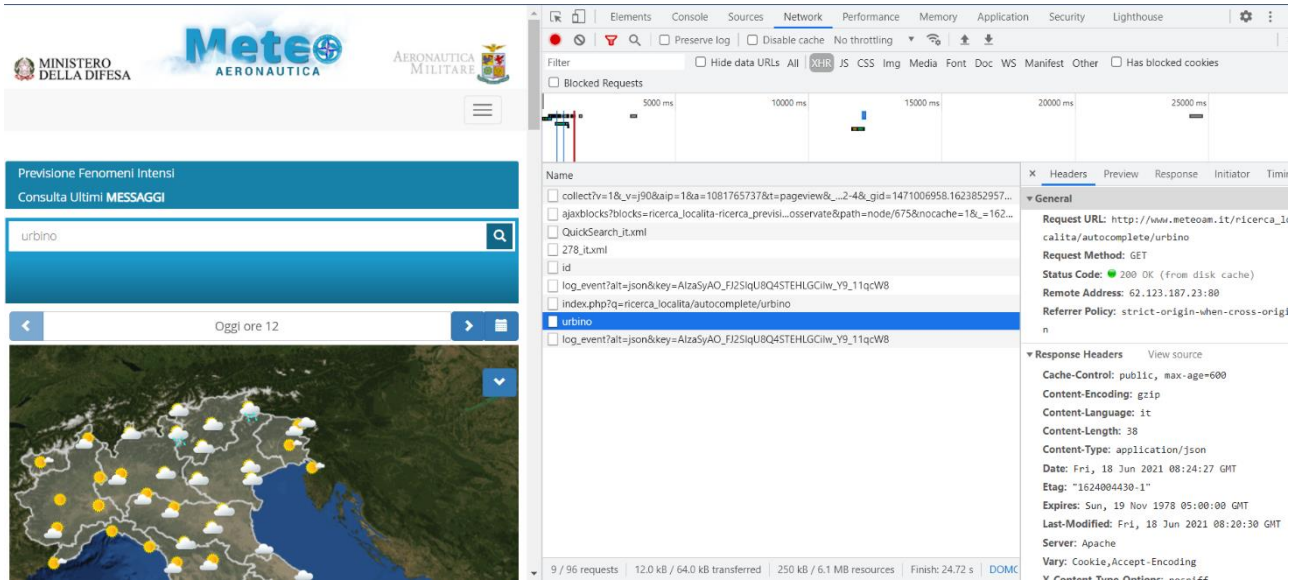
Inizialmente si è pensato di inserire nel tool anche i dati provenienti dal sito meteoam.it, ma si è presentato un problema:

navigando sul sito e cercando di conoscere il formato URL in maniera tale da poter effettuare la richiesta HTTP ed avere i dati meteo, è possibile notare che l'URL è così fatto:

"http://www.meteoam.it/ta/previsione/<id>/<localita>"

Viene spontaneo affermare che anche in questo caso esista un API fornita dal servizio che permette, a partire dalla località inserita dall'utente di risalire in maniera univoca all'id della località (come accade nel sito meteo.it).

Si è andato quindi ad analizzare con lo strumento di sviluppo del browser il comportamento della pagina web quando viene inserito nella barra di ricerca il nome della località:



Anche in questo caso si nota la presenza di una richiesta effettuata direttamente dal browser contenente la località digitata. Si potrebbe pensare che come nel caso del sito meteo.it, tale richiesta restituisca una risposta in JSON contenente anche l'id per risalire alla previsione. Invece no: effettuando la richiesta ricercando direttamente l'URL

evidenziato, è possibile vedere che si, la risposta è in formato JSON, ma non contiene l'ID della città per risalire alla previsione:

```
{"URBINO":"URBINO (PU)"}
```

Un possibile motivo dell'impossibilità a reperire l'ID è che il sito utilizzi uno script PHP per recuperare l'ID: in tal caso, essendo il linguaggio interpretato direttamente dal server e non dal browser come invece accade per il Javascript, richieste HTTP e risposte vengono effettuate lato server e sono quindi invisibili al client. E' lecito pensare che quando si clicca sul bottone di ricerca venga chiamato uno script in PHP che prenda in ingresso il risultato JSON mostrato in alto, effettui una richiesta al database che a sua volta ritorni al server l'ID della località. A questo punto lo script PHP reindirizzerebbe semplicemente l'utente alla pagina giusta senza possibilità alcuna di conoscere a priori l'ID della città cercata.

Esiste un modo per risolvere questo problema?

Ci si è a questo punto chiesti se possa esistere un modo per aggirare questa limitazione del server: una possibile soluzione potrebbe essere quella di pensare che l'id sia un codice che identifica la località univocamente e che non cambi nel tempo. In tal caso è essere possibile supporre di effettuare n richieste con $n > numero\_comuni\_italiani$ considerando $x = \{0, \dots, n\}$ all'URL:
"http://www.meteoam.it/ta/previsione/<x>"
In questo modo, il tool potrebbe effettuare, magari al primo avvio tutte le n richieste e potrebbe salvare i risultati ottenuti in un file in locale, affinchè dal secondo avvio in poi sia possibile associare il nome della località all'id della località.

Nel tool è presente la possibilità di provare questo tipo di approccio: se si digita 1 vengono inviate $n = 50000$ richieste HTTP in cui a ogni iterazione la variabile i viene incrementata di 1. Viene scelto 50000 in quanto numero arbitrariamente grande che sicuramente comprende tutti i comuni italiani e le altre località (non propriamente comuni) di cui il server tiene traccia delle previsioni metereologiche.
Tale approccio "brute force" potrebbe essere utile se si sfruttasse la potenza di calcolo della propria macchina, invece viene usato per effettuare delle richieste ad un server. Ciò comporta che anche un numero irrisorio di operazioni (50000), che nel caso fossero eseguite in locale impiegherebbero ad eseguire un tempo dell'ordine dei millisecondi, risultano molto dispendiose, sia a livello di prestazioni che di dati spediti/ricevuti.

Inoltre di contro, si ha anche il problema che il server risponde alle richieste molto spesso lo Status Code 403 – Forbidden e ogni tanto lo Status Code 200 – OK. Ciò potrebbe dipendere dal fatto che il server non accetti richieste in cascata da parte di client, ma accetti solamente richieste ogni periodo di tempo definito. Come prova di ciò, se dalla barra della ricerca del browser digitiamo l'URL che ci ha restituito l'errore 403, il sito meteoam.it non darà problemi e restituirà le previsioni della località con quell'ID.

Per questi motivi nel tool la sezione di prova del sito meteoam.it permette solamente di visualizzare le risposte fornite dal server, in quanto risulterebbe impossibile, la costruzione di un file contente le associazioni tra l'ID della località e il nome della località.

PARTE SESTA – Conclusioni

Lo scraping dei dati metereologici è andato a buon fine per i siti meteo.it, ilmeteo.it, 3bmeteo.com, mentre non è andato a buon fine per il sito meteoam.it. Ciò dipende dal fatto che il sito di meteoam.it, essendo anche un sito direttamente collegato al Ministero della Difesa, ha inevitabilmente più controlli che permettono di bloccare questo tipo di tecnica automatica per recuperare i dati.
Per preservare la proprietà dei dati, da parte dei siti di cui lo scraping è riuscito, potrebbe essere utile inserire un altro parametro all'interno dell'URL che viene recuperato lato server rendendolo invisibile al client. Tale strategia è adottata in maniera corretta dal sito meteoam.it e adottata in parte dal sito meteo.it. Tuttavia, dopo un'attenta analisi, è stato possibile trovare un API non esplicitamente pubblica che permette di recuperare l'ID anche per il sito meteo.it.
Il tool sviluppato, per quanto semplice, effettua il suo dovere correttamente e confronta tutti i dati dei 3 siti di previsioni meteo di cui si è riuscito a effettuare lo scraping dei dati.

Bibliografia, Link

1. Siti meteo:
    a. https://www.ilmeteo.it
    b. https://www.3bmeteo.com
    c. https://www.meteo.it
    d. http://www.meteoam.it
2. https://it.wikipedia.org/wiki/Web_scraping
3. https://www.apogeonline.com/articoli/come-fare-web-scraping-con-python-serena-sensini/
4. https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Repository progetto:

https://github.com/nicosanti98/ScrapingMeteo