UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Web scraping per l'atletica: estrazione dati, elaborazione ed archiviazione per l'analisi statistica delle manifestazioni FIDAL

Relatore:

Prof. Di Nunzio Giorgio Maria

Laureando:

Pellegrini Luca

ANNO ACCADEMICO 2022 – 2023

Data di laurea 25/09/2023

Abstract

La maggior parte dei dati presenti su internet sono distribuiti in modo non strutturato rendendo facilmente fruibile la consultazione ma difficile un’analisi complessiva ai fini statistici. Le tecniche di web scraping permettono di estrarre i dati dalle pagine web per consentirne l'elaborazione, l'archiviazione strutturata mediante database e l'analisi. Questa tesi prende in esame le manifestazioni della Federazione Italiana di Atletica Leggera e lo sviluppo del programma e del database per l'estrazione e l'archiviazione dei dati derivanti dalle gare. L'obiettivo è consentire un’analisi statistica delle manifestazioni per località, categoria degli atleti, tipologia di manifestazione, specialità e partecipazione.

Indice

Abstract iii
Introduzione 1
1 Il web e la raccolta dei dati 3
1.1 Web e l'HTML 3
1.2 Analisi pagine web 4
1.3 Web crawling 4
1.4 Web scraping 5
1.5 Data science 6
2 Calendario federale e Iscrizioni/Risultati 7
2.1 Stato dell'arte 7
2.2 Calendario federale 8
2.3 Sito Iscrizioni/Risultati 10
2.3.1 Iscrizioni 10
2.3.2 Risultati corse 12
2.3.3 Risultati concorsi 13
3 Sviluppo del programma 14
3.1 Obiettivi 14
3.2 Strategie di sviluppo 15
4 Database 16
4.1 Analisi dei requisiti 16
4.2 Progettazione concettuale 16
4.3 Progettazione logica 18
4.4 Progettazione fisica 18
5 Programma 19
5.1 Creazione database 19
5.2 Main 21
5.3 Calendario 21
5.4 Manifestazione 22
5.5 IscrizioniRisultati 23
5.6 Iscrizioni 23
5.7 Risultati 24
5.8 Funzioni aggiuntive 26
6 Conclusioni 27
Bibliografia 28

Introduzione

L'obiettivo di questa tesi di laurea è sviluppare un programma Python che estrae i dati relativi alle manifestazioni della Federazione Italiana Di Atletica Leggera tramite tecniche di web scraping e web crawling. Lo scopo della raccolta dei dati è l'analisi statistica delle manifestazioni fornendo uno strumento a dirigenti, tecnici, atleti e giudici per una più facile e completa pianificazione delle attività.

Il documento esamina le modalità di funzionamento di web crawling e web scraping presentando poi la progettazione e la realizzazione di un'applicazione che utilizza queste tecniche per l'estrazione dei dati delle manifestazioni FIDAL. Viene inoltre descritto il processo di progettazione del database utilizzato per l'archiviazione dei dati.

L'elaborato è strutturato nei seguenti capitoli:

1. il web e le tecniche di web crawling e web scraping
2. l'analisi calendario FIDAL e delle pagine di pubblicazione di iscrizioni e risultati
3. la progettazione del programma, gli obiettivi e gli strumenti utilizzati
4. la progettazione concettuale, logica e fisica del database
5. lo sviluppo dell'applicativo
6. conclusioni

1 Il web e la raccolta dei dati

Le pagine web contengono una grande quantità di informazioni che vengono rappresentate in modo da facilitarne la lettura e la comprensione dei contenuti da parte degli utenti. I contenuti vengono illustrati mediante diversi mezzi di comunicazione: testo, tabelle, grafici, presentazioni, immagini, audio, video…

Per consentire l'utilizzo delle informazioni in modo automatizzato è necessario procedere all'analisi delle pagine e alla raccolta strutturata dei dati presenti nelle stesse.

1.1 Web e l'HTML

Il World Wide Web, abbreviato in Web è uno dei servizi disponibili su internet, permette la navigazione in documenti ipertestuali che si possono raggiungere mediante un indirizzo web chiamato URL, *l'Uniform Resource Locator*, identificativo di una risorsa presente sul web.

Il primo sito web[1] è stato creato il 6 agosto del 1991 da Tim Berners-Lee informatico britannico impiegato al CERN di Ginevra con lo scopo di sviluppare lo standard per scambiare documenti mediante una rete di calcolatori. Con questo progetto negli stessi anni vennero definiti il protocollo HTTP, l *'HyperText Transfer Protocol*, dedicato al trasferimento di documenti mediante internet e il linguaggio HTML, *l'HyperText Markup Language*, utilizzato per formattare ed impaginare i documenti ipertestuali disponibili nella prima versione del web.

Figura 1.1 Il primo sito web[1]

I primi siti web erano composti da un insieme di pagine correlate tra loro accessibili tramite internet e contenevano solo testo e link: collegamenti ipertestuali cliccabili che mediante gli url rimandano ad un'altra risorsa web.

Nel corso degli anni le pagine web si sono evolute per poter rappresentare innumerevoli tipologie di contenuti multimediali grazie a successive versioni di linguaggio HTML e hanno permesso l'integrazione con altri linguaggi rendendo dinamico il contenuto e la formattazione.

[1] https://info.cern.ch/hypertext/WWW/TheProject.html

1.2 Analisi pagine web

Data la ricchezza di dati contenuti nelle pagine web per vari scopi è utile raccogliere e analizzare le informazioni presenti e la relazione tra esse. In assenza del rilascio delle informazioni in modo strutturato: mediante fogli elettronici, database o API per raccogliere e analizzare i dati è necessario esaminare direttamente le pagine web che contengono le informazioni. A seconda della finalità di analisi si possono adottare principalmente due tipologie di analisi automatizzata delle pagine: il web crawling e il web scraping.

Nell'utilizzo di queste tecniche è importate rispettare i termini e le condizioni di utilizzo dei siti web analizzati per non incorrere nell'utilizzo di materiale coperto da diritto d'autore o violazione della normativa GDPR.

Analisi ripetute delle pagine web possono anche provocare dei *Denial Of Service*, DoS, andando a sovraccaricare il server ospitante il sito web. Per non essere identificati come attacchi informatici è consigliabile limitare il numero di richieste HTTP o distribuirle in un maggiore arco di tempo.

1.3 Web crawling

Il web crawling analizza le correlazioni tra pagine web e siti: a partire da una pagina esamina i link presenti nella stessa ed in modo ricorsivo i collegamenti trovati nelle pagine individuate. Il risultato è un grafo contenente le correlazioni che permette l'analisi del sito considerato e dei siti a lui collegati. I software che eseguono queste analisi vengono chiamati, *spider*, ragni o crawler, perché a partire da un *seed*, seme, un url noto, scansionano tutta la ragnatela di collegamenti presenti. L'analisi viene effettuata periodicamente per aggiornare le informazioni raccolte ed osservare le modifiche nei siti. [2]

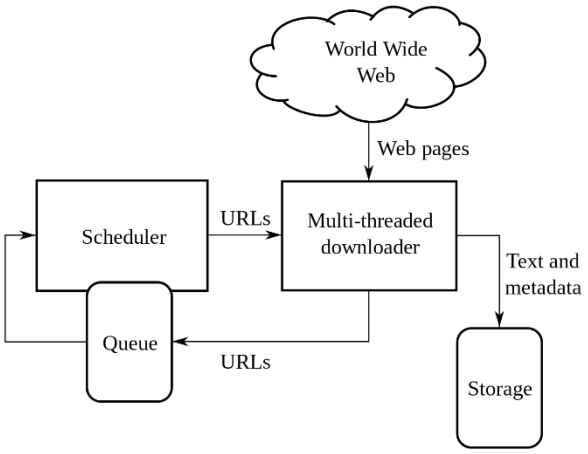


Figura 1.2 L'architettura di un web crawler[2]

[2] https://en.wikipedia.org/wiki/Web_crawler#/media/File:WebCrawlerArchitecture.svg

Questa tecnica viene utilizzata specialmente dai motori di ricerca con lo scopo di esplorare ed indicizzare le pagine presenti in un sito per catalogare i contenuti classificandoli per tipologia. La quantità e la tipologia di menzioni che ottiene un particolare sito o pagina permette anche ai motori di ricerca di valutare la qualità del contenuto per stabilire il posizionamento del sito tra i risultati delle ricerche. Molti siti facilitano questa pratica adottando strategie SEO, *search engine optimization*, che permettono di semplificare l'indicizzazione dei contenuti di un sito mediante l'utilizzo di *sitemap*, tag e codice ben strutturato. Gli amministratori dei siti web possono anche scoraggiare questa pratica indicando in un file apposito chiamato robots.txt le pagine che i software non devono analizzare ma spesso queste indicazioni vengono ignorate.
Uno dei più conosciuti web crawler è Googlebot, utilizzato dal motore di ricerca Google, analizza le pagine che compaiono nei risultati del motore di ricerca.

1.4 Web scraping

Il web scraping ha come obiettivo l'estrazione automatizzata delle informazioni dalle pagine web. È conosciuto anche come web harvesting o web data extraction. Questa tecnica permette la raccolta di grandi volumi di dati quando applicata automaticamente ma può essere impiegata anche manualmente mediante copia-incolla o riscrittura delle informazioni.
L'impiego principale è quello di raccogliere grandi quantità di dati per consentirne un'analisi statistica o per l'archiviazione strutturata mediante fogli di calcolo o database.
Il processo di web scraping, effettuato da uno scraper, software appositamente creato si compone generalmente delle seguenti parti (Ciutacu, 2022):

1. individuazione dell'url della pagina da analizzare
2. richiesta HTTP della pagina al server e ricezione della risposta
3. selezione dei dati d'interesse
4. elaborazione dei dati
5. esportazione dei dati in un formato strutturato

Gli scraper utilizzati possono utilizzare differenti strategie per estrarre i dati, i più comuni sono software realizzati in modo specifico per analizzare un sito target per il quale il programma viene sviluppato. È possibile effettuare lo scraping di pagine web anche mediante l'utilizzo di estensioni di browser o servizi online che utilizzano software non specializzato.
Modelli di intelligenza artificiale possono essere applicati per consentire la raccolta dei dati di pagine con formattazione complessa o discontinua. Inoltre, sempre mediante intelligenza artificiale, è possibile analizzare anche immagini, audio, video ed altri supporti multimediali per raccogliere anche le informazioni contenute in questi contenuti. (scrapeit, 2022)

1.5 Data science

Successivamente alla raccolta dei dati è necessaria l'analisi degli stessi per consentirne l'interpretazione, la scienza che si occupa di queste analisi è la data science. Rende possibile l'analisi dei dati, specialmente se in volumi considerevoli mediante l'utilizzo dei principi metodologici e di tecniche multidisciplinari per estrarre conoscenze dai dati. Vengono utilizzati una serie di strumenti come il data mining, il machine learning, pattern recognition, analisi statistiche e visuali per interpretare i dati.

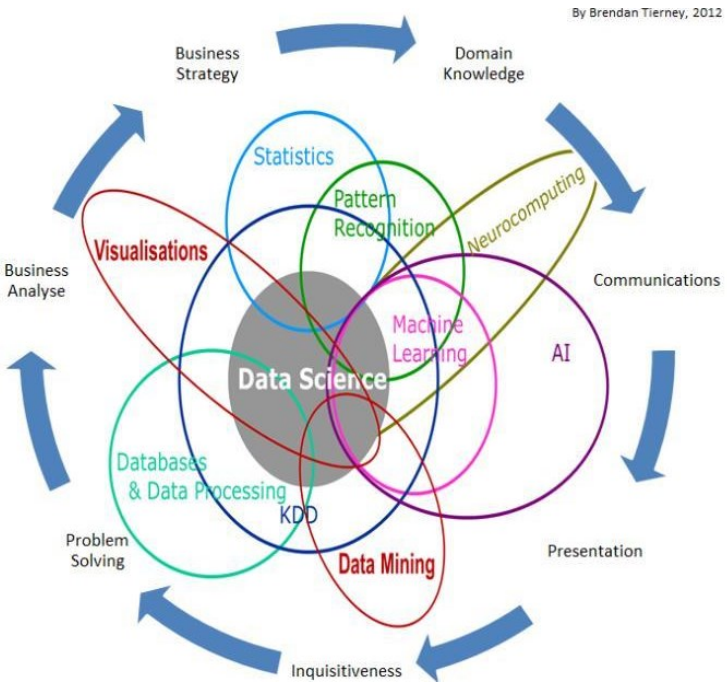


Figura 1.3 Campi Data Science[3]

[3] https://vitolavecchia.altervista.org/differenza-tra-data-mining-e-data-science/

2 Calendario federale e Iscrizioni/Risultati

Nel capitolo seguente viene esaminata la sezione del sito FIDAL contenente il calendario, le iscrizioni ed i risultati.

2.1 Stato dell'arte

La FIDAL, Federazione Italiana di Atletica Leggera, è tra le più importanti e partecipate federazioni a livello nazionale 244.125 atleti tesserati[4] e 2.872 società[5] al 31/12/2022. Sono migliaia le manifestazioni che si svolgono ogni anno sul territorio nazionale organizzate dalla federazione nazionale, dai comitati regionali e provinciali e dalle società. Il calendario federale raccoglie tutte le manifestazioni ufficiali a tutti i livelli, dall'attività giovanile a quella master passando per l'assoluta. La maggior parte delle manifestazioni vengono gestite dalla segreteria FIDAL e le pubblicazioni di tutte le informazioni relative alla manifestazione vengono pubblicate su una sezione apposita del calendario federale accessibile online. WISE, il gestionale utilizzato progressivamente a partire da gennaio 2022, consiste in una web app che utilizza un database conservato sul server FIDAL non accessibile da internet e provvede alla pubblicazione di iscrizioni e risultati mediante la generazione automatica e la pubblicazione di pagine web statiche contenenti le informazioni selezionate dall'operatore della segreteria.
Il precedente gestionale SIGMA che sta venendo progressivamente dismesso e pertanto il programma non prende in analisi le pagine generate dal passato applicativo gestionale.
Al termine di ogni manifestazione le sole prestazioni degli atleti vengono caricate nella graduatoria nazionale. Non viene utilizzato un sistema centralizzato per la raccolta di tutti i dati riguardanti le manifestazioni e pertanto allo stato attuale gli stakeholder interessati ad analizzare le manifestazioni e le gare svolte devono provvedere alla raccolta manuale delle informazioni d'interesse che risulta particolarmente difficoltosa dato il numero elevato di appuntamenti.

[4] https://www.fidal.it/upload/files/Statistiche/2022/TessAtl2022.pdf

[5] https://www.fidal.it/upload/files/Statistiche/2022/DGTM_2022.pdf

2.2 Calendario federale

La pagina relativa al calendario comprende un form che permette l'inserimento dei parametri di ricerca desiderati popolando la tabella sottostante. La pagina viene richiesta tramite un metodo GET che trasmette all'interno dell'url i parametri del form:
https://www.fidal.it/calendario.php?anno=2023&mese=9&livello=COD&new_regione=&new_tipo=0&new_categoria=&submit=Invia

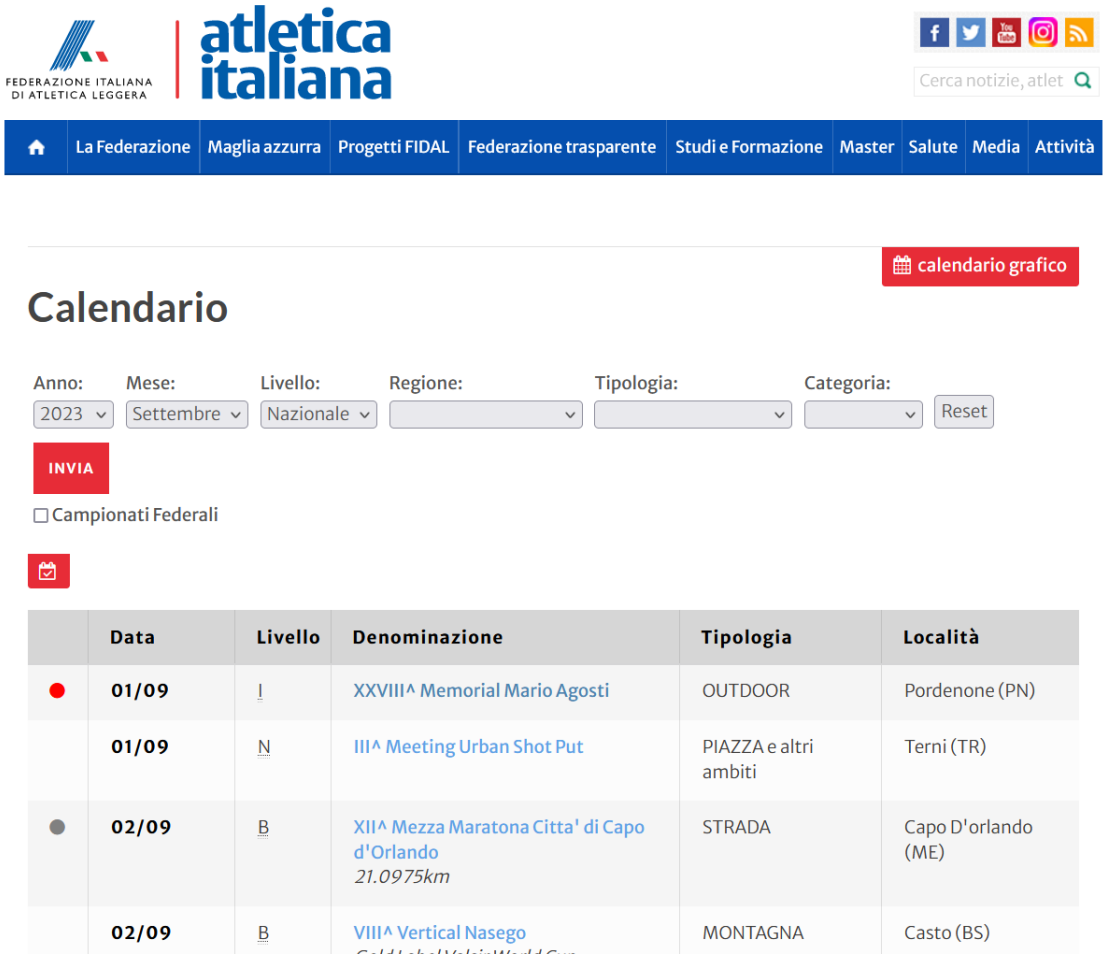


Figura 2.1 Pagina web del calendario federale relativa al mese di settembre 2023

Nella tabella sono contenute le informazioni di interesse come la data, il nome, la tipologia, la località e il collegamento della pagina individuale della manifestazione sul calendario federale.

Figura 2.2 Codice pagina web del calendario federale relativa al mese di settembre 2023

Ispezionando il codice si può esaminare la struttura della tabella: ad ogni manifestazione corrisponde una riga <*tr*> e tutti i dati utili sono contenuti nelle relative celle <*td*>.

Nella pagina della manifestazione sono presenti ulteriori informazioni utili come il livello, le categorie partecipanti. Inoltre, il pulsante "Iscrizioni/Risultati" rimanda alla sezione contenente le iscrizioni ed i risultati della manifestazione con i dati relativi alle singole gare.

Figura 2.3 Pagina web del calendario federale relativa ad una manifestazione

2.3 Sito Iscrizioni/Risultati

Tramite il pulsante indicato in precedenza si accede al sito contente tutte le informazioni relative alla manifestazione. In particolare, da questa pagina è possibile navigare nelle pagine delle iscrizioni e dei risultati delle singole gare.

Figura 2.4 Home page sito Iscrizioni/Risultati

2.3.1 Iscrizioni

Prendendo in esame la sezione relativa alle "Iscrizioni per Gara" è possibile vedere come le gare vengono divise per specialità ed elencate nella parte sottostante.

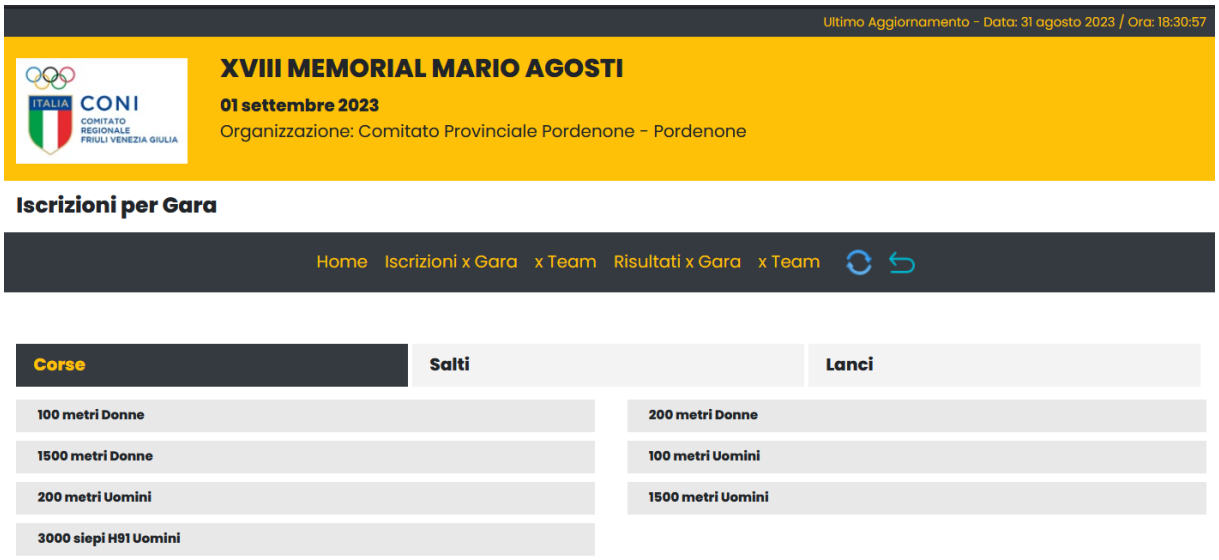
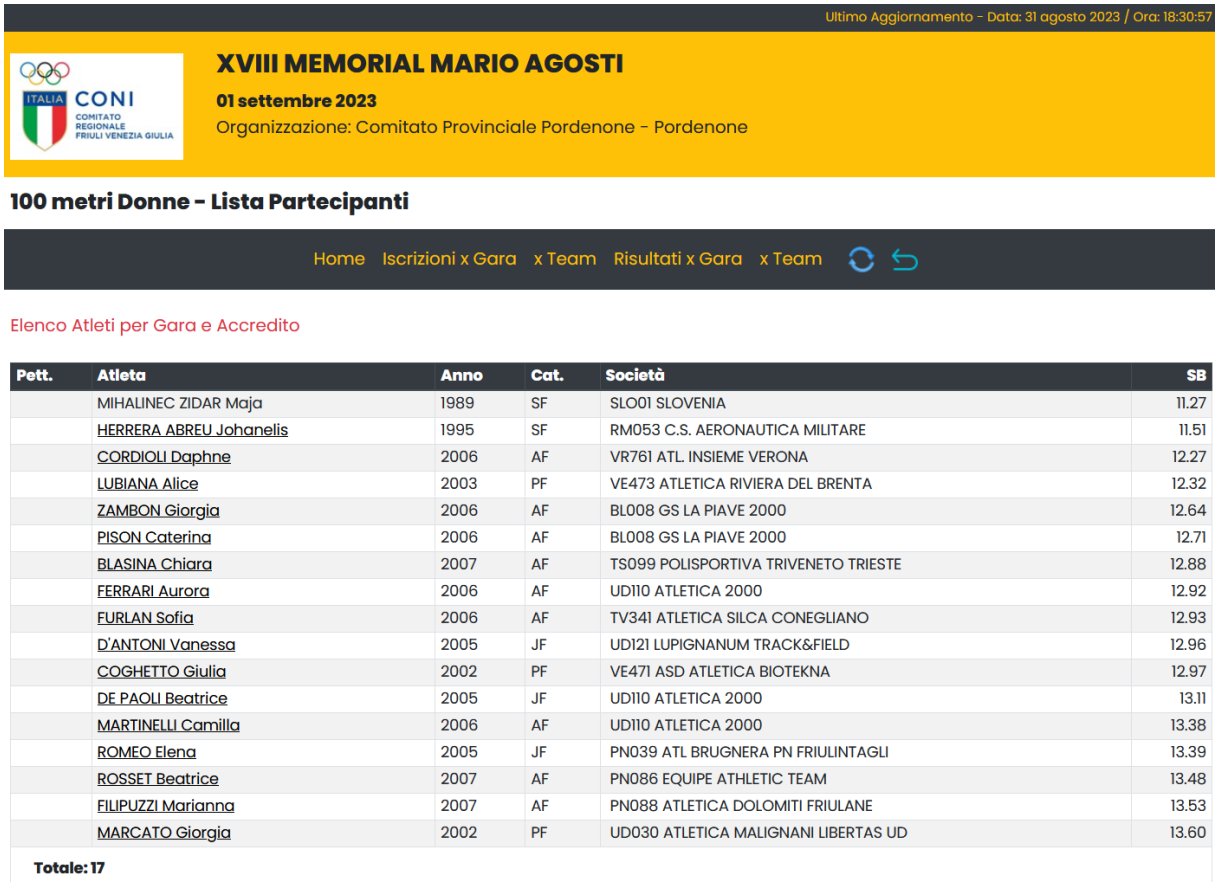


Figura 2.5 Sezione iscrizioni per Gara relativa alle corse

Per ogni gara nella sezione iscrizioni vengono elencati gli atleti partecipanti alla gara e riportato in calce il totale. Per ogni atleta sono riportate anche informazioni come l'anno di nascita e la società/nazione, gli atleti tesserati in Italia presentano anche un collegamento che dà accesso alla pagina personale sul sito federale contenente i dettagli della carriera.



Ultimo Aggiornamento - Data: 31 agosto 2023 / Ora: 18:30:57

CONI COMITATO REGIONALE FRIULI VENEZIA GIULIA

XVIII MEMORIAL MARIO AGOSTI

01 settembre 2023

Organizzazione: Comitato Provinciale Pordenone - Pordenone

100 metri Donne - Lista Partecipanti

Home Iscrizioni x Gara x Team Risultati x Gara x Team

Elenco Atleti per Gara e Accredito

Pett.	Atleta	Anno	Cat.	Società	SB
	MIHALINEC ZIDAR Maja	1989	SF	SLO01 SLOVENIA	11.27
	HERRERA ABREU Johanelis	1995	SF	RM053 C.S. AERONAUTICA MILITARE	11.51
	CORDIOLI Daphne	2006	AF	VR761 ATL. INSIEME VERONA	12.27
	LUBIANA Alice	2003	PF	VE473 ATLETICA RIVIERA DEL BRENTA	12.32
	ZAMBON Giorgia	2006	AF	BL008 GS LA PIAVE 2000	12.64
	PISON Caterina	2006	AF	BL008 GS LA PIAVE 2000	12.71
	BLASINA Chiara	2007	AF	TS099 POLISPORTIVA TRIVENETO TRIESTE	12.88
	FERRARI Aurora	2006	AF	UD110 ATLETICA 2000	12.92
	FURLAN Sofia	2006	AF	TV341 ATLETICA SILCA CONEGLIANO	12.93
	D'ANTONI Vanessa	2005	JF	UD121 LUPIGNANUM TRACK&FIELD	12.96
	COGHETTO Giulia	2002	PF	VE471 ASD ATLETICA BIOTEKNA	12.97
	DE PAOLI Beatrice	2005	JF	UD110 ATLETICA 2000	13.11
	MARTINELLI Camilla	2006	AF	UD110 ATLETICA 2000	13.38
	ROMEO Elena	2005	JF	PN039 ATL BRUGNERA PN FRIULINTAGLI	13.39
	ROSSET Beatrice	2007	AF	PN086 EQUIPE ATHLETIC TEAM	13.48
	FILIPUZZI Marianna	2007	AF	PN088 ATLETICA DOLOMITI FRIULANE	13.53
	MARCATO Giorgia	2002	PF	UD030 ATLETICA MALIGNANI LIBERTAS UD	13.60

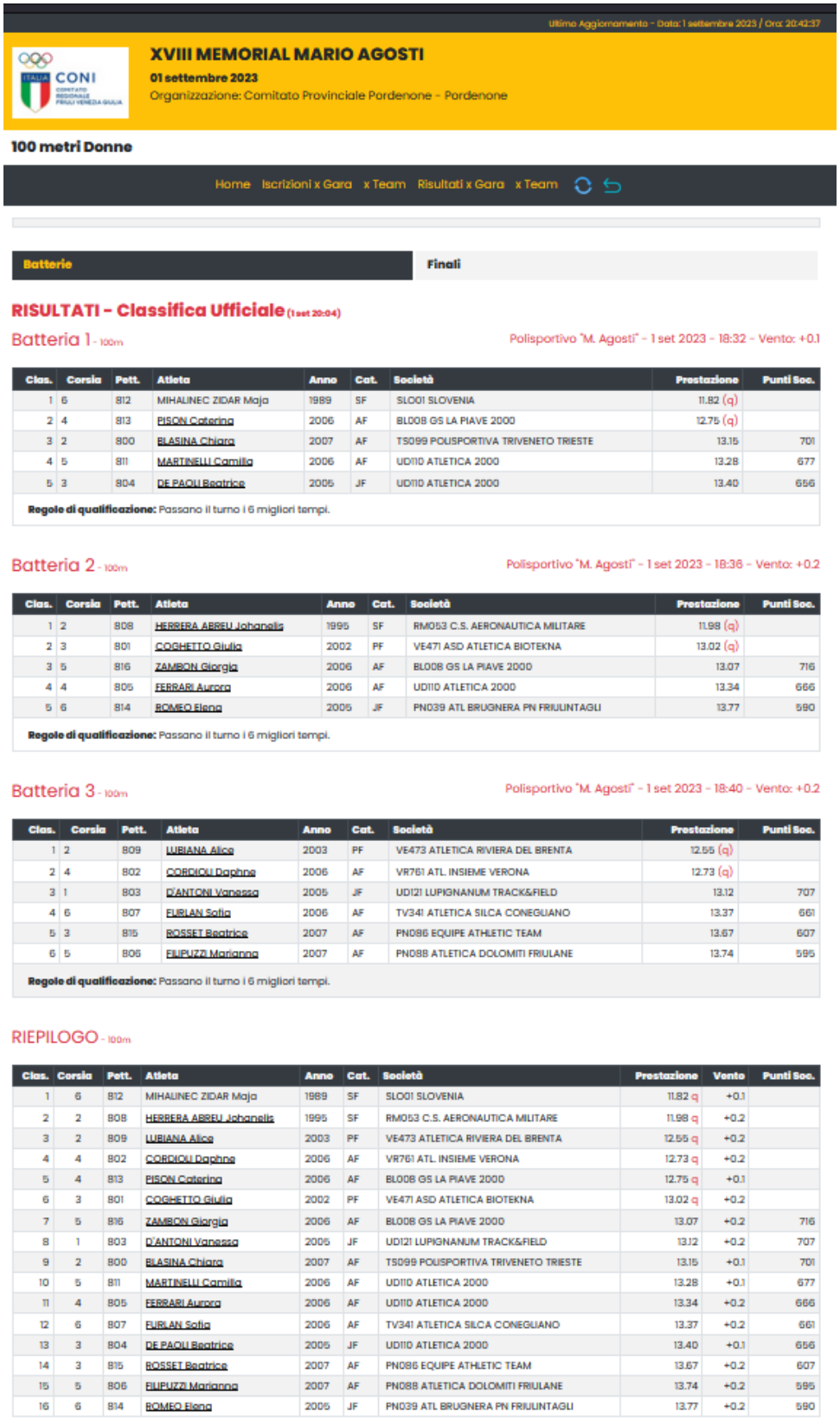
Totale: 17

Figura 2.6 Lista partecipanti ad una gara

I risultati vengono riportati in modo differente a seconda della tipologia di specialità avendo ognuna caratteristiche differenti.

2.3.2 Risultati corse

Divisi per turni di gara che possono variare in base al regolamento in serie oppure batterie con turni successivi come semifinali e finali. Per ogni turno viene pubblicata la suddivisione in serie/batterie con le relative prestazioni ed in calce il riepilogo dei risultati



Ultimo Aggiornamento - Data: 1 settembre 2023 / Ora: 20:42:37

CONI

XVIII MEMORIAL MARIO AGOSTI

01 settembre 2023

Organizzazione: Comitato Provinciale Pordenone - Pordenone

100 metri Donne

Home Iscrizioni x Gara x Team Risultati x Gara x Team

Batterie **Finali**

RISULTATI - Classifica Ufficiale (1 set 20:04)

Batteria 1 - 100m

Polisportivo "M. Agosti" - 1 set 2023 - 18:32 - Vento: +0.1

Clas.	Corsia	Pett.	Atleta	Anno	Cat.	Società	Prestazione	Punti Soc.
1	6	812	MIHALINEC ZIDAR Maja	1989	SF	SLO01 SLOVENIA	11.82 (q)	
2	4	813	PISON Caterina	2006	AF	BL008 GS LA PIAVE 2000	12.75 (q)	
3	2	800	BLASINA Chiara	2007	AF	TS099 POLISPORTIVA TRIVENETO TRIESTE	13.15	701
4	5	811	MARTINELLI Camilla	2006	AF	UD110 ATLETICA 2000	13.28	677
5	3	804	DE PAOLI Beatrice	2005	JF	UD110 ATLETICA 2000	13.40	656

Regole di qualificazione: Passano il turno i 6 migliori tempi.

Batteria 2 - 100m

Polisportivo "M. Agosti" - 1 set 2023 - 18:36 - Vento: +0.2

Clas.	Corsia	Pett.	Atleta	Anno	Cat.	Società	Prestazione	Punti Soc.
1	2	808	HERRERA ABREU Johanelis	1995	SF	RM053 C.S. AERONAUTICA MILITARE	11.98 (q)	
2	3	801	COGHETTO Giulia	2002	PF	VE471 ASD ATLETICA BIOTEKNA	13.02 (q)	
3	5	816	ZAMBON Giorgia	2006	AF	BL008 GS LA PIAVE 2000	13.07	716
4	4	805	FERRARI Aurora	2006	AF	UD110 ATLETICA 2000	13.34	666
5	6	814	ROMEO Elena	2005	JF	PN039 ATL BRUGNERA PN FRIULINTAGLI	13.77	590

Regole di qualificazione: Passano il turno i 6 migliori tempi.

Batteria 3 - 100m

Polisportivo "M. Agosti" - 1 set 2023 - 18:40 - Vento: +0.2

Clas.	Corsia	Pett.	Atleta	Anno	Cat.	Società	Prestazione	Punti Soc.
1	2	809	LUBIANA Alice	2003	PF	VE473 ATLETICA RIVIERA DEL BRENTA	12.55 (q)	
2	4	802	CORDIOLI Daphne	2006	AF	VR761 ATL. INSIEME VERONA	12.73 (q)	
3	1	803	D'ANTONI Vanessa	2005	JF	UD121 LUPIGNANUM TRACK&FIELD	13.12	707
4	6	807	FURLAN Sofia	2006	AF	TV341 ATLETICA SILCA CONEGLIANO	13.37	661
5	3	815	ROSSET Beatrice	2007	AF	PN086 EQUIPE ATHLETIC TEAM	13.67	607
6	5	806	FILIPUZZI Marianna	2007	AF	PN088 ATLETICA DOLOMITI FRIULANE	13.74	595

Regole di qualificazione: Passano il turno i 6 migliori tempi.

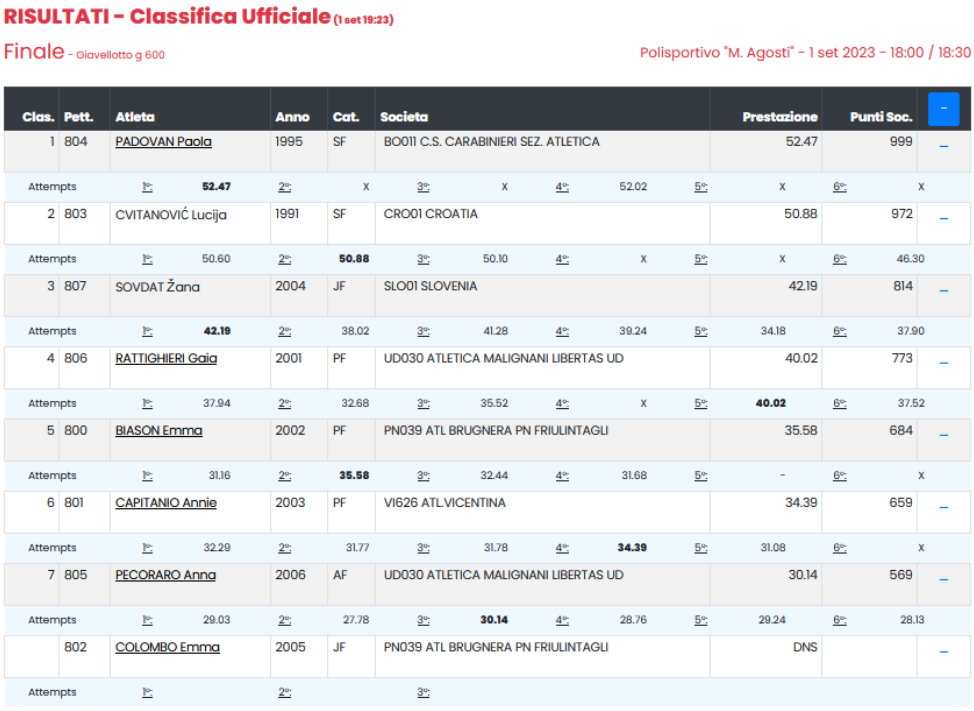
RIEPILOGO - 100m

Clas.	Corsia	Pett.	Atleta	Anno	Cat.	Società	Prestazione	Vento	Punti Soc.
1	6	812	MIHALINEC ZIDAR Maja	1989	SF	SLO01 SLOVENIA	11.82 q	+0.1	
2	2	808	HERRERA ABREU Johanelis	1995	SF	RM053 C.S. AERONAUTICA MILITARE	11.98 q	+0.2	
3	2	809	LUBIANA Alice	2003	PF	VE473 ATLETICA RIVIERA DEL BRENTA	12.55 q	+0.2	
4	4	802	CORDIOLI Daphne	2006	AF	VR761 ATL. INSIEME VERONA	12.73 q	+0.2	
5	4	813	PISON Caterina	2006	AF	BL008 GS LA PIAVE 2000	12.75 q	+0.1	
6	3	801	COGHETTO Giulia	2002	PF	VE471 ASD ATLETICA BIOTEKNA	13.02 q	+0.2	
7	5	816	ZAMBON Giorgia	2006	AF	BL008 GS LA PIAVE 2000	13.07	+0.2	716
8	1	803	D'ANTONI Vanessa	2005	JF	UD121 LUPIGNANUM TRACK&FIELD	13.12	+0.2	707
9	2	800	BLASINA Chiara	2007	AF	TS099 POLISPORTIVA TRIVENETO TRIESTE	13.15	+0.1	701
10	5	811	MARTINELLI Camilla	2006	AF	UD110 ATLETICA 2000	13.28	+0.1	677
11	4	805	FERRARI Aurora	2006	AF	UD110 ATLETICA 2000	13.34	+0.2	666
12	6	807	FURLAN Sofia	2006	AF	TV341 ATLETICA SILCA CONEGLIANO	13.37	+0.2	661
13	3	804	DE PAOLI Beatrice	2005	JF	UD110 ATLETICA 2000	13.40	+0.1	656
14	3	815	ROSSET Beatrice	2007	AF	PN086 EQUIPE ATHLETIC TEAM	13.67	+0.2	607
15	5	806	FILIPUZZI Marianna	2007	AF	PN088 ATLETICA DOLOMITI FRIULANE	13.74	+0.2	595
16	6	814	ROMEO Elena	2005	JF	PN039 ATL BRUGNERA PN FRIULINTAGLI	13.77	+0.2	590

Figura 2.7 Risultati di una corsa

2.3.3 Risultati concorsi

Anche in questo caso possono essere previsti più turni di gara. I risultati sono riportati con una tabella con gli atleti ordinati come da ordine di classifica e per ogni atleta un'ulteriore tabella con la prestazione di ogni tentativo. I salti in elevazione di differenziano dai restanti concorsi in quanto le misure sono prestabilite e ogni atleta ha 3 prove per misura, vengono indicati con O, X e – gli esiti del salto.



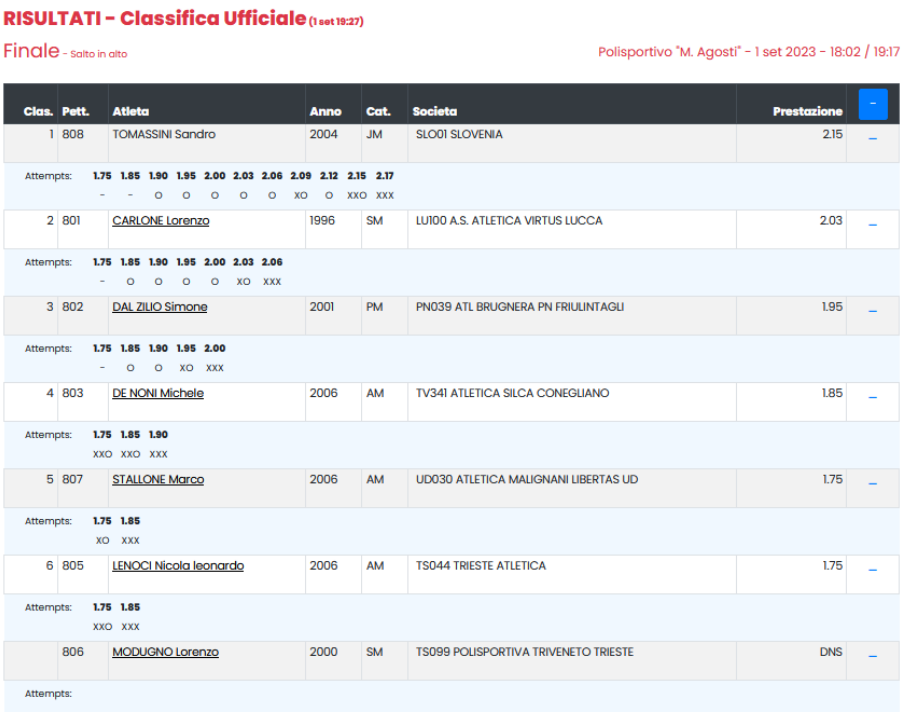
RISULTATI - Classifica Ufficiale (1 set 19:23)

Finale - Giavellotto g 600

Polisportivo "M. Agosti" - 1 set 2023 - 18:00 / 18:30

Clas.	Pett.	Atleta	Anno	Cat.	Società	Prestazione	Punti Soc.
1	804	PADOVAN Paola	1995	SF	BO011 C.S. CARABINIERI SEZ. ATLETICA	52.47	999
Attempts		1°: **52.47**	2°: x	3°: x	4°: 52.02	5°: x	6°: x
2	803	CVITANOVIĆ Lucija	1991	SF	CRO01 CROATIA	50.88	972
Attempts		1°: 50.60	2°: **50.88**	3°: 50.10	4°: x	5°: x	6°: 46.30
3	807	SOVDAT Žana	2004	JF	SLO01 SLOVENIA	42.19	814
Attempts		1°: **42.19**	2°: 38.02	3°: 41.28	4°: 39.24	5°: 34.18	6°: 37.90
4	806	RATTIGHIERI Gaia	2001	PF	UD030 ATLETICA MALIGNANI LIBERTAS UD	40.02	773
Attempts		1°: 37.94	2°: 32.68	3°: 35.52	4°: x	5°: **40.02**	6°: 37.52
5	800	BIASON Emma	2002	PF	PN039 ATL BRUGNERA PN FRIULINTAGLI	35.58	684
Attempts		1°: 31.16	2°: **35.58**	3°: 32.44	4°: 31.68	5°: -	6°: x
6	801	CAPITANIO Annie	2003	PF	VI626 ATL.VICENTINA	34.39	659
Attempts		1°: 32.29	2°: 31.77	3°: 31.78	4°: **34.39**	5°: 31.08	6°: x
7	805	PECORARO Anna	2006	AF	UD030 ATLETICA MALIGNANI LIBERTAS UD	30.14	569
Attempts		1°: 29.03	2°: 27.78	3°: **30.14**	4°: 28.76	5°: 29.24	6°: 28.13
	802	COLOMBO Emma	2005	JF	PN039 ATL BRUGNERA PN FRIULINTAGLI	DNS	
Attempts		1°:	2°:	3°:			

Figura 2.8 Risultati di un lancio o di un salto in estensione



RISULTATI - Classifica Ufficiale (1 set 19:27)

Finale - Salto in alto

Polisportivo "M. Agosti" - 1 set 2023 - 18:02 / 19:17

Clas.	Pett.	Atleta	Anno	Cat.	Società	Prestazione
1	808	TOMASSINI Sandro	2004	JM	SLO01 SLOVENIA	2.15
Attempts:		**1.75** -	**1.85** -	**1.90** O	**1.95** O	**2.00** O	**2.03** O	**2.06** O	**2.09** XO	**2.12** O	**2.15** XXO	**2.17** XXX
2	801	CARLONE Lorenzo	1996	SM	LU100 A.S. ATLETICA VIRTUS LUCCA	2.03
Attempts:		**1.75** -	**1.85** O	**1.90** O	**1.95** O	**2.00** O	**2.03** XO	**2.06** XXX
3	802	DAL ZILIO Simone	2001	PM	PN039 ATL BRUGNERA PN FRIULINTAGLI	1.95
Attempts:		**1.75** -	**1.85** O	**1.90** O	**1.95** XO	**2.00** XXX
4	803	DE NONI Michele	2006	AM	TV341 ATLETICA SILCA CONEGLIANO	1.85
Attempts:		**1.75** XXO	**1.85** XXO	**1.90** XXX
5	807	STALLONE Marco	2006	AM	UD030 ATLETICA MALIGNANI LIBERTAS UD	1.75
Attempts:		**1.75** XO	**1.85** XXX
6	805	LENOCI Nicola leonardo	2006	AM	TS044 TRIESTE ATLETICA	1.75
Attempts:		**1.75** XXO	**1.85** XXX
	806	MODUGNO Lorenzo	2000	SM	TS099 POLISPORTIVA TRIVENETO TRIESTE	DNS
Attempts:						

Figura 2.9 Risultati di un salto in elevazione

3 Sviluppo del programma

In questo capitolo vengono analizzati gli obiettivi del progetto e il processo di sviluppo del programma.

3.1 Obiettivi

Il progetto si pone come obiettivo quello di automatizzare la raccolta dei dati utili per le analisi statistiche delle manifestazioni creando un database che può essere utilizzato per le utili valutazioni dell'attività praticata. I dati che verranno conservati comprenderanno il numero di atleti iscritti ed effettivamente praticanti, in numero di tentativi differenziando le prove valide da quelle fallose, la durata della gara e la specialità e la tipologia della stessa. I dati verranno inoltre catalogati per data, tipologia di manifestazione e luogo. Queste informazioni permetteranno l'analisi dell'attività a tutte le figure presenti nell'atletica: atleti, dirigenti, tecnici e giudici con lo scopo di migliorare la programmazione delle attività future.
Sarà possibile effettuare valutazioni come:

- la frequenza delle specialità praticate per concentrare l'attività tecnica su di esse o favorire la pratica delle discipline meno diffuse;
- il numero di atleti partecipanti al fine di ipotizzare la partecipazione nella pianificazione delle manifestazioni, prevedendo anche una percentuale di atleti iscritti ma non partecipanti;
- la durata delle gare per pianificare gli orari delle manifestazioni sulla base delle edizioni precedenti o degli eventi della stessa tipologia;
- il rapporto tra prove valide e prove nulle;

Queste analisi potranno comprendere manifestazioni specifiche, periodi temporali o luoghi geografici per consentire valutazioni specifiche ai parametri scelti.
Non rientra tra gli obiettivi del progetto l'analisi delle prestazioni di un singolo atleta o di una particolare società. Nomi, data di nascita o ulteriori informazioni riconducibili ad un singolo atleta non vengono conservate. I dati vengono archiviati in modo aggregato per non incorrere in problematiche inerenti il trattamento dati personali o GDPR.

3.2 Strategie di sviluppo

Il progetto si compone da più parti:

- un web crawling appositamente creato per esplorare a partire dal calendario federale le pagine relative alle manifestazioni, alle iscrizioni ed ai risultati e alle singole gare, si limita all'analisi delle pagine relative alle iscrizioni ed ai risultati delle manifestazioni
- un web scraper per estrarre i dati dalle pagine di iscrizioni e risultati
- un database per l'archiviazione dei dati

Il linguaggio utilizzato per il programma è il Python per la versatilità e la diffusione che lo rende particolarmente semplice anche nell'utilizzo in cloud.

Il Database Management System è PostgreSQL soluzione open source che permette un hosting locale del database.

Le librerie utilizzate oltre alle librerie standard sono:

- psycopg2 [6]
- requests [7]
- BeautifulSoup [8]

Il programma non è parallelizzato per limitare il numero di richieste contemporanee e non sovraccaricare i server che ospitano il sito FIDAL. Per questo motivo il tempo di esecuzione è considerevole e proporzionale al periodo di analisi delle manifestazioni ma data la specifica di aggiornamento settimanale/mensile del database sono sufficienti un paio di minuti per l'esecuzione.

[6] https://www.psycopg.org/

[7] https://requests.readthedocs.io/en/latest/

[8] https://www.crummy.com/software/BeautifulSoup/

4 Database

Nel paragrafo è descritta la progettazione del database motivando le scelte implementative.

4.1 Analisi dei requisiti

I dati che verranno raccolti nel database proverranno dallo scraping del calendario FIDAL e dalle iscrizioni e risultati delle gare che utilizzano WISE per la gestione della segreteria. Dal calendario federale verranno raccolte le informazioni riguardanti i dati generali delle manifestazioni: codice, nome, data, luogo, url, tipologia; classificandole per regione e provincia. Dalla sezione "Iscrizione/Risultati" verranno raccolti tutti i dati riguardanti le singole gare senza mai archiviare dati personali dei partecipanti. I dati di interesse sono: nome gara, durata, ritardo, numero gruppi, numero iscritti, numero partecipanti, numero classificati, prove buone e prove nulle.
L'inserimento dei dati a regime potrà avvenire a cadenza giornaliera o settimanale con la sola necessità di aggiungere le manifestazioni svolte nel periodo individuato. Le interrogazioni avranno frequenza maggiore e saranno molteplici a seconda dei dati da analizzare.

4.2 Progettazione concettuale

Scelte implementative effettuate nella modellazione:

- La Provincia è descritta come entità per permettere la correlazione con la sigla unica informazione geografica presente nelle manifestazioni per poter risalire a Provincia e Regione. È possibile aggiungere nuove provincie o modificare quelle esistenti senza perdere i dati e le associazioni passate
- In una località può essere presente nel database anche senza manifestazioni associate. Il calendario federale non prevede distinzioni tra campi diversi nella stessa località.
- Una manifestazione può non avere alcuna gara nel caso in cui sia stata programmata la manifestazione ma non inserite le gare componenti.
- In ogni manifestazione le gare hanno sempre codici diversi per permettere la gestione della segreteria e la pubblicazione dei risultati.

Di seguito il modello Entità-Associazione che descrive il progetto

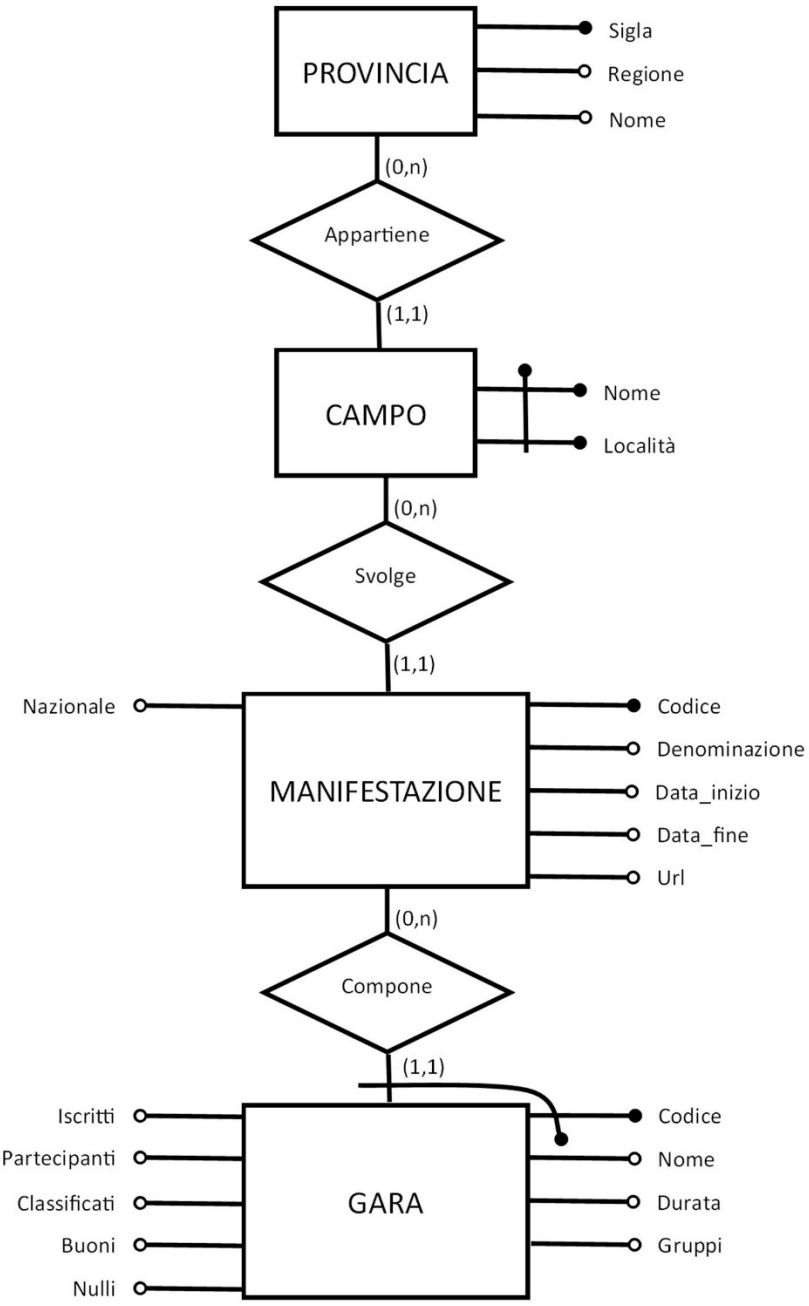


Grafico 4.1 Modello Entità-Associazione

4.3 Progettazione logica

Nel seguente grafico viene rappresentato il modello relazionale a partire dallo schema ER descritto nel paragrafo precedente.

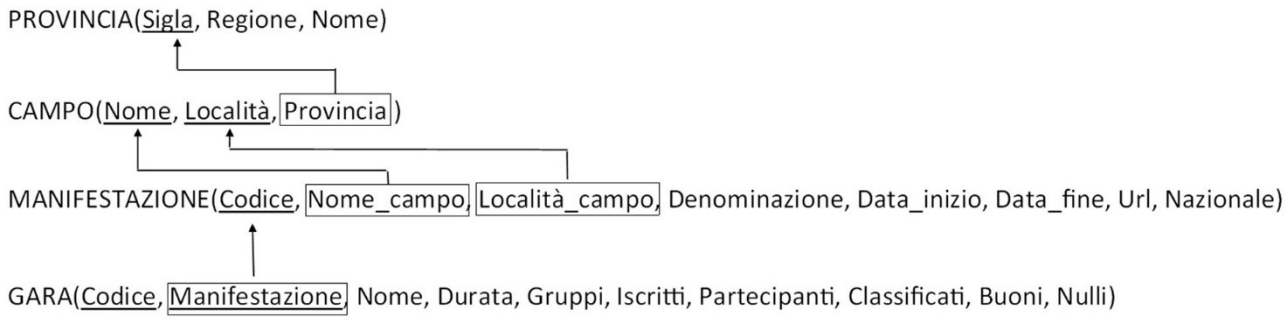


Grafico 4.2 Modello relazionale

4.4 Progettazione fisica

Di seguito le istruzioni SQL per implementare la soluzione descritta in precedenza:

```
CREATE DATABASE local;
CREATE TABLE Provincia (
        Sigla CHAR(2) NOT NULL UNIQUE,
        Regione VARCHAR(50) NOT NULL,
        Nome VARCHAR(50) NOT NULL,
        PRIMARY KEY(Sigla));
CREATE TABLE Località (
        Nome VARCHAR(128) NOT NULL,
        Provincia CHAR(2),
        PRIMARY KEY(Nome),
        FOREIGN KEY(Provincia) REFERENCES Provincia(Sigla));
CREATE TABLE Manifestazione (
        Codice CHAR(8) NOT NULL UNIQUE,
        Località VARCHAR(128) NOT NULL,
        Denominazione VARCHAR(255) NOT NULL,
        Data_inizio DATE NOT NULL,
        Data_fine DATE NOT NULL,
        Url VARCHAR(256) NOT NULL,
        Nazionale BOOLEAN NOT NULL,
        PRIMARY KEY(Codice),
        FOREIGN KEY(Località) REFERENCES Località(Nome));
CREATE TABLE Gara (
        Codice CHAR(3) NOT NULL,
        Manifestazione CHAR(8) NOT NULL,
        Nome VARCHAR(255),
        Durata SMALLINT,
        Gruppi SMALLINT,
        Iscritti SMALLINT,
        Partecipanti SMALLINT,
        Classificati SMALLINT,
        Buoni SMALLINT,
        Nulli SMALLINT,
        PRIMARY KEY(Codice, Manifestazione),
        FOREIGN KEY(Manifestazione) REFERENCES Manifestazione(Codice));
```

5 Programma

Il software si compone di più funzioni:

- creazione iniziale del database con il popolamento di regioni e provincie

5.1 Creazione database

Con questa funzione vengono create le tabelle SQL e popolata la tabella con i dati di tutte le provincie italiane. La struttura delle tabelle è quella discussa nei precedenti paragrafi. I campi delle provincie e regioni sono in funzione della visualizzazione sul sito federale, pertanto, le provincie autonome di Trento e Bolzano sono considerate come regioni, così come la Repubblica di San Marino.

```
import psycopg2

def createDB():
    try:
        connection = psycopg2.connect(host="localhost", dbname="local", user="postgres",
password="admin", port="5432")
        print("Connessione al database effettuata")
    except:
        print(f"Errore di connessione")

    #creazione Provincia
    try:
        cursor = connection.cursor()
        cursor.execute("""CREATE TABLE Provincia (
                    Sigla CHAR(2) NOT NULL UNIQUE,
                    Regione VARCHAR(50) NOT NULL,
                    Nome VARCHAR(50) NOT NULL,
                    PRIMARY KEY(Sigla)
                    );
                    """)
        connection.commit()
        print(f"Create Provincia")
    except:
        print(f"Errore Provincia")

    #creazione Campo
    try:
        cursor = connection.cursor()
        cursor.execute("""CREATE TABLE Località (
                    Nome VARCHAR(128) NOT NULL,
                    Provincia CHAR(2),
                    PRIMARY KEY(Nome),
                    FOREIGN KEY(Provincia) REFERENCES Provincia(Sigla)
                    );
                    """)
        connection.commit()
        print(f"Create Località")
    except:
        print(f"Error Località")
```

```
    #creazione Manifestazione
    try:
        cursor = connection.cursor()
        cursor.execute("""CREATE TABLE Manifestazione (
                    Codice CHAR(8) NOT NULL UNIQUE,
                    Località VARCHAR(128) NOT NULL,
                    Denominazione VARCHAR(255) NOT NULL,
                    Data_inizio DATE NOT NULL,
                    Data_fine DATE NOT NULL,
                    Url VARCHAR(256) NOT NULL,
                    Nazionale BOOLEAN NOT NULL,
                    PRIMARY KEY(Codice),
                    FOREIGN KEY(Località) REFERENCES Località(Nome)
                    );
                    """)
        connection.commit()
        print(f"Create Manifestazione")
    except:
        print(f"Errore Manifestazione")

    #creazione Gara
    try:
        cursor = connection.cursor()
        cursor.execute("""CREATE TABLE Gara (
                    Codice CHAR(3) NOT NULL,
                    Manifestazione CHAR(8) NOT NULL,
                    Nome VARCHAR(255),
                    Durata SMALLINT,
                    Iscritti SMALLINT,
                    Partecipanti SMALLINT,
                    Classificati SMALLINT,
                    Gruppi SMALLINT,
                    Buoni SMALLINT,
                    Nulli SMALLINT,
                    PRIMARY KEY(Codice, Manifestazione),
                    FOREIGN KEY(Manifestazione) REFERENCES Manifestazione(Codice)
                    );
                    """)
        connection.commit()
        print(f"Create Gara")
    except:
        print(f"Errore Gara")

    #inserimento Provincie
    try:
        cursor = connection.cursor()
        cursor.execute("""INSERT INTO Provincia (Regione, Nome, Sigla) VALUES
                    ('Piemonte', 'Torino', 'TO'),
                    ('Piemonte', 'Vercelli', 'VC'),
                    […] #restanti provincie non riportate
                    ('San Marino', 'San Marino', 'SM')
                    """)
        connection.commit()
        print(f"Insert Provincia")
    except:
        print(f"Errore Insert Provincia")

    connection.close()

if __name__ == "__main__":
    #creazione tabelle
    createDB()
```

5.2 Main

La seguente funzione, invocata all'apertura del programma, differenzia le modalità di esecuzione dell'applicativo a seconda dell'argomento indicato nel comando di esecuzione del programma permette:

- l'analisi di tutto il calendario federale a partire dal 1° gennaio 2021, nessun argomento
  python3 app.py
- l'analisi a partire da una specifica data alla data odierna, gg/mm/aaaa
  python3 app.py "01/09/2023"
- l'analisi in un intervallo indicato, gg/mm/aaaa-gg/mm/aaaa
  python3 app.py "01/01/2023-01/09/2023"

```
def main():
    #nessun argomento
    inizio = date(year=2022, month=1, day=1)
    fine = date.today()
    #un argomento gg/mm/aaaa
    if (len(sys.argv) == 2) and re.fullmatch("[0-3][0-9]\/[0-1][0-9]\/20[0-9][0-9]",
sys.argv[1]):
        sp = sys.argv[1].split("/")
        inizio = date.fromisoformat(str(sp[2]+"-"+sp[1]+"-"+sp[0]))
    #un argomento gg/mm/aaaa-gg/mm/aaaa
    elif (len(sys.argv) == 2) and re.fullmatch("[0-3][0-9]\/[0-1][0-9]/20[0-9][0-9]\-[0-
3][0-9]\/[0-1][0-9]/20[0-9][0-9]", sys.argv[1]):
        sp = sys.argv[1].split("-")
        sp0 = sp[0].split("/")
        sp1 = sp[1].split("/")
        inizio = date.fromisoformat(str(sp0[2]+"-"+sp0[1]+"-"+sp0[0]))
        fine = date.fromisoformat(str(sp0[2]+"-"+sp1[1]+"-"+sp0[1]))
        if (inizio<fine):
            print("La data iniziale deve essere antecedente alla data finale")
        else:
            #analisi calendario
            calendario(inizio, fine)
```

5.3 Calendario

La funzione, ricevuti come parametri la data di inizio e la data di fine del periodo da esaminare, naviga ciclicamente le pagine mensili del calendario estraendo le righe della tabella calendario contenenti le informazioni sulle manifestazioni. Per ogni riga invoca la funzione manifestazioni() per permettere l'elaborazione della manifestazione.

```
def calendario(inizio, fine):
    attuale = inizio
    #scraping un mese alla volta fino al raggiungimento del mese finale
    while (attuale < fine):
        print("calendario {mese}/{anno}".format(anno=attuale.year, mese=attuale.month))
        #calendario nazionale
        nazionale = True
        page = requests.get(
'https://www.fidal.it/calendario.php?anno={anno}&mese={mese}&livello=COD&new_regione=&new_t
ipo=0&new_categoria=&submit=Invia'.format(anno=attuale.year,mese=attuale.month))
        soup = BeautifulSoup(page.text, 'html.parser')
        righe_tabella_calendario = soup.tbody.find_all("tr")
```

```
        #analizza ogni riga della tebella calendario che corrisponde ad una manifestazione
        for riga in righe_tabella_calendario:
            #se la manifestazione è annullata non viene elaborata
            if re.search('annullat', str(riga), re.IGNORECASE):
                continue
            manifestazione(riga, attuale, nazionale)
        #calendario regionale
        nazionale = False
        page =
requests.get('https://www.fidal.it/calendario.php?anno={anno}&mese={mese}&livello=REG&new_r
egione=&new_tipo=0&new_categoria=&submit=Invia'.format(anno=attuale.year,
mese=attuale.month))
        soup = BeautifulSoup(page.text, 'html.parser')
        righe_tabella_calendario = soup.tbody.find_all("tr")
        #analizza ogni riga della tebella calendario che corrisponde ad una manifestazione
        for riga in righe_tabella_calendario:
            #se la manifestazione è annullata non viene elaborata
            if re.search('annullat', str(riga), re.IGNORECASE):
                continue
            manifestazione(riga, attuale, nazionale)
        attuale = attuale + relativedelta(months=1)
    print(f"Done scraping calendario")
    conn.commit()
```

5.4 Manifestazione

La funzione estrae dalla riga della pagina del calendario federale i dati riguardanti la manifestazione: codice, denominazione, luogo, località, provincia, carattere nazionale o regionale, url, data di inizio e data di fine. Questi dati grazie all'esecuzione dell'istruzione di inserimento seguente vengono istanziati nella tabella. Nel caso in cui la manifestazione sia già presente nella tabella vengono aggiornati gli attributi.

INSERT INTO Manifestazione (Codice, Località, Denominazione, Data_inizio, Data_fine, Url, Nazionale) VALUES <elenco valori da inserire> ON CONFLICT(Codice) DO UPDATE SET (Località, Denominazione, Data_inizio, Data_fine, Url, Nazionale)

```
def manifestazione(riga, anno, nazionale):
    #analizza il contenuto di ogni riga del calendario per ricavare i dati da inserire
nelle tabelle Località e Manifestazione
    cursor = conn.cursor()
    codice = riga.find_all("td")[3].a["href"].split("/")[-1]
    denominazione = riga.find_all("td")[3].a.text
    data = calcola_data(riga.b.text, anno.year)
    #se la manifestazione non è ancora terminata viene ignorata, i dati non sono completi
    if data[1] < date.today():
        return
    url = riga.find_all("td")[3].a["href"]
    luogo = riga.find_all("td")[5].string
    località = luogo.rsplit(' ', 1)[0]
    provincia = luogo.rsplit(' ', 1)[1].strip("() -")
    #la provincia non viene indicata per le manifestazioni all'estero
    if provincia == "" or provincia.__sizeof__() != 2:
        provincia = None
    #inserimento istanze
    cursor.execute(cursor.mogrify("INSERT INTO Località(Nome, Provincia) VALUES (%s,%s) ON
CONFLICT (Nome) DO NOTHING", (località, provincia)))
    cursor.execute(cursor.mogrify("INSERT INTO Manifestazione (Codice, Località,
Denominazione, Data_inizio, Data_fine, Url, Nazionale) VALUES (%s,%s,%s,%s,%s,%s,%s) ON
CONFLICT(Codice) DO UPDATE SET (Località, Denominazione, Data_inizio, Data_fine, Url,
Nazionale) = (%s,%s,%s,%s,%s,%s)", (codice, località, denominazione, data[0], data[1], url,
nazionale, località, denominazione, data[0], data[1], url, nazionale)))
    url_iscrizionirisultati(url, codice_manifestazione)
```

5.5 IscrizioniRisultati

Questa funzione individua gli url delle pagine contenenti le iscrizioni ed i risultati della manifestazione e ne invoca l'elaborazione.

```
def iscrizionirisultati(url, codice_manifestazione):
    #individua l'url della pagina contenente le iscrizioni ed i risultati della
manifestazione
    page = requests.get(url)
    soup = BeautifulSoup(page.text, 'html.parser')
    if (testo_pulsante := soup.find(string="Iscrizioni/Risultati")):
        pulsante = testo_pulsante.find_parent("a")
        link = pulsante["href"]
        print(codice_manifestazione)
        #indivudua le pagine delle iscrizioni e dei risultati per ogni manifestazione
        try:
            page = requests.get(link)
            soup = BeautifulSoup(page.text, 'html.parser')
            soup.find_all(class_="col-md-6")
            iscrizioni(urljoin(link,soup.find(string="ISCRIZIONI PER
GARA").find_parent("a")["href"]), codice_manifestazione)
            risultati(urljoin(link,soup.find(string="START LISTS E RISULTATI PER
GARA").find_parent("a")["href"]), codice_manifestazione)
        except:
            pass
```

5.6 Iscrizioni

La seguente funzione individuati gli url della gare componenti la manifestazione richiede ed elabora la pagina delle iscrizioni di ogni gara inserendo nella relativa tabella i dati d'interesse: codice gara, codice manifestazione, nome e numero di iscritti. Anche in questo caso se la gara è già presente vengono aggiornati gli attributi.

INSERT INTO Gara (Codice, Manifestazione, Nome, Iscritti) VALUES <elenco valori da inserire> ON CONFLICT(Codice, Manifestazione) DO UPDATE SET (Nome, Iscritti) = <elenco valori da aggiornare>

```
def iscrizioni(link, codice_manifestazione):
    #elabora le iscrizioni della manifestazione
    cursor = conn.cursor()
    page = requests.get(link)
    soup = BeautifulSoup(page.text, 'html.parser')
    #individua tutte le gare presenti nella manifestazione
    gare = []
    try:
        gare.extend(soup.find_all(class_="link-style"))
    except:
        pass
    for g in gare:
        #per ogni gara individua le informazioni presenti nella pagina delle iscrizioni e
le insirisce nella relativa tabella
        url = urljoin(link,g["href"])
        gara = requests.get(url)
        soup_gara = BeautifulSoup(gara.text, 'html.parser')
        nome = soup_gara.find(string=re.compile(" - Lista Partecipanti"))[:-21]
        atleti = int(soup_gara.find(string=re.compile('Totale: *'))[8:])
        cursor.execute(cursor.mogrify("INSERT INTO Gara (Codice, Manifestazione, Nome,
Iscritti) VALUES (%s,%s,%s,%s) ON CONFLICT(Codice, Manifestazione) DO UPDATE SET (Nome,
Iscritti) = (%s,%s)", (url[-8:-5],codice_manifestazione,nome,atleti,nome,atleti)))
    conn.commit()
```

5.7 Risultati

Questa funzione in analogia con la precedente si occupa di elaborare i risultati delle gare componenti la manifestazione aggiornando le istanze già presenti in tabella. L'esecuzione del codice all'interno di un blocco try-except è resa necessaria dalla possibilità che i dati presenti nelle pagine html dei risultati siano difformi dagli standard del sistema gestionale WISE che consente

```
def risultati(url, codice_manifestazione):
    #elabora i risultati della manifestazione
    page = requests.get(url)
    soup = BeautifulSoup(page.text, 'html.parser')
    #individua tutte le gare presenti nella manifestazione, divise nelle sezioni a seconda
della tipologia
    gare = []
    try:
        gare.extend(soup.find_all(class_="link-style"))
    except:
        pass
    for g in gare:
        #per ogni gara individua le informazioni presenti nella pagina dei risultati e le
inserisce nella relativa tabella
        try:
            url = urljoin(url,g["href"])
            partecipanti = 0
            classificati = 0
            nulli = 0
            buoni = 0
            gruppi = 1
            gara = requests.get(url)
            soup = BeautifulSoup(gara.text, 'html.parser')
            soup_gara = soup.find("table", class_="table table-striped table-sm table-
bordered h6-7")
            #la gara si svolge in serie/batterie/gruppi
            if (soup.find_all("table", class_="table table-striped table-sm table-bordered
h6-7").__len__() > 1):
                gruppi = soup.find_all("table", class_="table table-striped table-sm table-
bordered h6-7").__len__() - 1
                temp = BeautifulSoup(gara.text[gara.text.index("RIEPILOGO"):],
'html.parser')
                soup_gara = temp.find("table", class_="table table-striped table-sm table-
bordered h6-7")
            trs = soup_gara.tbody.find_all("tr", role="tabpanel", class_="panel-collapse
collapse table-borderless")
            #la gara si svolge in una serie/gruppo unico
            if (trs.__len__() == 0):
                trs = soup_gara.tbody.find_all("tr")
            partecipanti = trs.__len__()
            for tr in trs:
                if (str(tr).find("Attempts:") != -1):
                    trs2 = tr.find_all("tr")
                    for tr2 in trs2:
                        if (str(tr2).find("Attempts:") != -1):
                            continue
                        #calcolo buoni e nulli salti in elevazione
                        tds = tr.find_all("td", width="25px")
                        for td in tds:
                            for letter in str(td.string):
                                if (letter == "X"):
                                    nulli = nulli + 1
                                elif (letter == "O"):
                                    buoni = buoni + 1
```

```
                #calcolo buoni e nulli salti in estensione e lanci
                elif (str(tr).find("Attempts") != -1):
                    tds = tr.find_all("td", width="80px")
                    for td in tds:
                        if (td.string == "X"):
                            nulli = nulli + 1
                        elif (td.string != "-" and td.string != None):
                            buoni = buoni + 1
            #individuazione atleti classificati
            trs = soup_gara.tbody.find_all("tr", class_="")
            for tr in trs:
                if (str(tr).find("class") != -1 and str(tr.td.string).strip().isnumeric()):
                    temp = int(tr.td.string)
                    if (temp == classificati):
                        classificati = classificati +1
                    elif (temp > classificati):
                        classificati = temp
            #calcolo durata effettiva, le corse in un unica serie hanno durata 0
            orario = soup.find("div", id="g1").find_all("p", class_="h6 text-right text-
danger mb-4 mt-4 pt-1")
            durata = timedelta()
            prima_partenza = timedelta()
            ultima_partenza = timedelta()
            for o in orario:
                ora = str(o.string).split('-')[2]
                #concorsi
                if (ora.find("/") > 0):
                    ora_split = ora.split("/")
                    ora_inizio = timedelta(hours=int(ora_split[0].split(":")[0]),
minutes=int(ora_split[0].split(":")[1]))
                    ora_fine = timedelta(hours=int(ora_split[1].split(":")[0]),
minutes=int(ora_split[1].split(":")[1]))
                    temp = ora_fine - ora_inizio
                    durata = durata + temp
                #corse
                else:
                    if prima_partenza.total_seconds() == 0:
                        prima_partenza = timedelta(hours=int(ora.split(":")[0]),
minutes=int(ora.split(":")[1]))
                    ultima_partenza = timedelta(hours=int(ora.split(":")[0]),
minutes=int(ora.split(":")[1]))
            if durata.total_seconds() == 0:
                durata = ultima_partenza - prima_partenza
            cursor = conn.cursor()
            cursor.execute(cursor.mogrify("""UPDATE Gara SET (Durata, Partecipanti,
Classificati, Gruppi, Buoni, Nulli) = (%s,%s,%s,%s,%s,%s) WHERE Codice=%s AND
Manifestazione=%s""",
((durata.total_seconds()//60).__int__(),partecipanti,classificati,gruppi,buoni,nulli,url[-
8:-5],codice_manifestazione)))
            conn.commit()
        except:
            pass
```

5.8 Funzioni aggiuntive

La funzione calcola_data(stringa, anno) permette la conversione della stringa presente nella pagina web del calendario in data di inizio e data di fine della manifestazione.

```
def calcola_data(stringa, anno):
    #analisi stringa calendario per individuare la data di inizio e fine manifestazione
    #due giorni mesi diversi
    if re.fullmatch("[0-3][0-9]\/[0-1][0-9][0-3][0-9]\/[0-1][0-9]",stringa):
        stringa = stringa[:5] + '-' + stringa[5:]
    if re.fullmatch("[0-3][0-9]\/[0-1][0-9]\-[0-3][0-9]\/[0-1][0-9]",stringa):
        sp = stringa.split("-")
        sp0 = sp[0].split("/")
        sp1 = sp[1].split("/")
        data_inizio = date.fromisoformat(str(f'{anno}' +"-"+sp0[1]+"-"+sp0[0]))
        data_fine = date.fromisoformat(str(f'{anno}'+"-"+sp1[1]+"-"+sp0[1]))
        return [data_inizio, data_fine]
    #un giorno
    elif re.fullmatch("[0-3][0-9]\/[0-1][0-9]",stringa):
        sp = stringa.split("/")
        data_inizio = data_fine = date.fromisoformat(str(f'{anno}'+"-"+sp[1]+"-"+sp[0]))
        return [data_inizio, data_fine]
    #due giorni stesso mese
    elif re.fullmatch("[0-3][0-9]\-[0-3][0-9]\/[0-1][0-9]",stringa):
        sp = stringa.split("/")
        sp1 = sp[0].split("-")
        data_inizio = date.fromisoformat(str(f'{anno}'+"-"+sp[1]+"-"+sp1[0]))
        data_fine = date.fromisoformat(str(f'{anno}'+"-"+sp[1]+"-"+sp1[1]))
        return [data_inizio, data_fine]
```

La funzione get_cursor() permette la connessione al database locale di PostgressSQL

```
def get_cursor():
    #connessione al database locale
    try:
        connection = psycopg2.connect(host="localhost", dbname="local", user="***",
password="***", port="***")
        print("Connection to PostgressSQL DB successful")
        return connection
    except:
            print(f"Errore connessione DB")
```

Il seguente codice inserito nelle ultime righe del file consente l'esecuzione dell'applicativo mediante riga di comando consentendo anche l'utilizzo dello stesso in ulteriore programma Python mediante l'invocazione della funzione main()

```
if __name__ == "__main__":
    #connessione al database PostgresSQL
    conn = get_cursor()
    main()
    #chiusura connessione database
    conn.close()
```

6 Conclusioni

Il programma sviluppato mediante il salvataggio dei dati individuati nel database consente di utilizzare praticamente le informazioni ricavate mediante l'interrogazione del database.
L'esecuzione dell'applicativo a partire da gennaio 2022 fino a settembre 2023 ha permesso di prendere in esame 6.673 manifestazioni con 11.911 gare inserite nel database. L'interrogazione del database permette un utile impiego delle informazioni raccolte e con la maggior diffusione nei prossimi anni del gestionale WISE permetterà una più completa capillare analisi delle manifestazioni.
In sporadici casi i dati non risultano uniformi a causa di problematiche riguardanti le pagine web come la non corrispondenza del codice della gara tra la pagina delle iscrizioni e quella dei risultati o l'assenza della pubblicazione dei risultati. Ci sono alcuni casi dove il numero di partecipanti è maggiore del numero di iscritti oppure la durata non risulta attendibile a causa degli orari errati riportati nella pagina della gara.
Le informazioni presenti sono abbondantemente sufficienti per consentire un'analisi statistica approfondita dell'attività, ulteriori classificazioni come categoria e numero di turni potrebbero permettere lo studio ancor più granulare delle manifestazioni a discapito dell'uniformità dei dati a causa della diversità di pubblicazione degli stessi sul territorio nazionale.
A partire dal database popolato realizzando questo progetto è possibile la creazione di un interfaccia web per l'interrogazione semplificata del database permettendo agli stakeholder una fruizione semplificata con i dati raccolti.

Bibliografia

Atzeni Paolo, S. C. (2013). *Basi di dati - Modelli e linguaggi di interrogazioni.* McGraw-Hill.

Azzalini, A., & Scarpa, B. (2004). *Analisi dei dati e data mining.* Milano: Springer.

Beautiful Soup Documentation. Tratto da https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Ciutacu, L. (2022). *How to scrape data from a website*. Tratto il giorno 8 30, 2023 da n8n: https://blog.n8n.io/how-to-scrape-data-from-a-website/

Di Nunzio, G. M., & Di Buccio, E. (2017). *Basi di Dati, Manuale per la Progettazione Concettuale, Logica e SQL v3.0.* Società Editirice Esculapio.

FIDAL. Tratto da https://www.fidal.it

Patel, J. M. (2020). *Getting Structured Data from the Internet.* Apress Berkeley, CA.

Psycopg 2.9.7 documentation. Tratto da https://www.psycopg.org/docs/

Python 3.11.5 documentation. Tratto da https://docs.python.org/3/index.html

Requests: HTTP for Humans. Tratto da https://requests.readthedocs.io/en/latest/

scrapeit. (2022). *How Artificial Intelligence Is Used In Web Scraping*. Tratto da scrapeit: https://www.scrapeit.io/blog/how-artificial-intelligence-is-used-in-web-scraping

Web Scraping History: The Origins of Web Scraping. (2022). Tratto da scraping robot: https://scrapingrobot.com/blog/web-scraping-history/