ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

EleKtion: libreria Kotlin Multiplatform per la democrazia digitale in nuovi contesti

Tesi di laurea in:
Laboratorio di sistemi software

Relatore
Prof. Danilo Pianini

Candidato
Jacopo Corina

III Sessione di Laurea
Anno Accademico 2022-2023

Sommario

Max 2000 characters, strict.

Ai miei genitori

Indice

Elenco delle figure

Lista di esempi di codice

Capitolo 1

Introduzione

Capitolo 2

Democrazia digitale

Il termine *Democrazia*, derivante dal greco *governo del popolo*, indica un forma di governo nella quale il potere appartiene al popolo ed è esercitato attraverso forme dirette, che coinvolgono attivamente i cittadini, e attraverso forme rappresentantive, che prevedono un sistema di rappresentanza, la quale viene rinnovata ciclicamente attraverso apposite elezioni.

In una qualunque forma di *democrazia*, sono fondamentali le basi dei principi di elezioni libere in cui i cittadini sono considerati equi, partecipazione attiva dei cittadini alla vita politica, protezione dei diritti fondamentali e della libertà personale attraverso costituzioni [**?**].

Nelle epoche successive alle prime forme di democrazia "unanime" (per quanto il concetto di cittadinanza fosse assai diverso da quello attuale, ad esempio, erano escluse donne, schiavi e minori di 20 anni), complice anche la complessità delle materie da amministrare e il numero dei partecipanti fisici, la democrazia rappresentativa ha preso la maggior diffusione.

La modernizzazione tecnologica, accelerata di recente con l'avvento della pandemia da Covid-19 che ha evidenziato notevole problemi logistici, tra cui la gestione del distanziamento [**?**] e ha aperto un ventaglio di possibilità tali da permettere un ritorno ad un controllo più diretto da parte della collettività.

La *democrazia digitale* (o *e-democracy*) è definibile come "l'esercizio di processi

decisionali democratici attraverso l'uso di informazioni e tecnologie abilitanti alla comunicazione, in particolare Internet"[**?**].

Queste tecnologie, abilitanti a rompere i vincoli geo-fisici e territoriali, permettono di facilitare l'esercizio di ulteriori forme di democrazie, come quella partecipata, in cui i cittadini, piuttosto che sostituirsi ai rappresentanti, forniscono idee sullo sviluppo dell'indirizzo di governo permettendo così di creare un collettore di confronto ed analisi di situazioni differenti.

Tuttavia, è necessario considerare che un uso eccessivamente estensivo di questi strumenti può portare al rischi di coinvolgere persone che non hanno abbastanza competenze relativamente a temi delicati, ad esempio le politiche sociali.

Inoltre, nonostante la virtualizzazione della partecipazione permette alle fasce più disagiate di essere maggiormente coinvolte, introduce un motivo di scetticismo relativamente alla componente tecnologica in ambito di sicurezza informatica, usabilità, segretezza e manipolazione del voto [**?**].

Stabilire un legame di fiducia con queste tecnologie, normarne la verificabilità, stabilire principi di *accountability*, sono i fattore principali e quindi una precondizioni essenziali per la loro diffusione. Alcuni autori ([**?**]) sostengono che è impossibile avere leggitimità in quanto "la natura dei computer è tale che il loro funzionamento interno è segreto. Poiché le transazioni e i calcoli avvengono a livello elettronico, non è fisicamente possibile per gli esseri umani osservare esattamente ciò che un computer sta facendo."

L'applicazione più comune di queste tecnologie emergenti è legata al mondo della politica[**?**]: sono degni di nota casi come l'Estonia, che dal 2005 rende possibili via Internet le votazioni per elezioni e referendum, e l'Islanda che tramite un ambizioso progetto tenta di riscrivere la costituzione con l'apporto dei cittadini. In Italia possiamo rilevare l'applicativo PartecipaMi, organizzata per mettere in contatto la popolazione milanese con l'amministrazione comunale.[**?**]

Inoltre, queste soluzioni sono spesso adottate da partiti o movimenti politici per avviare iniziative e votazioni interne tra i membri, come nel caso del Partito Pirata tedesco in Germania (piattaforma LiquidFeedback), o il Movimento 5 Stelle

in Italia (piattaforma Rousseau). Lo scopo principale è quello di promuovere la discussione di proposte tra i membri di un organizzazione, le quali saranno portate in sede legislativa da un loro rappresentante[**?**].

Sistemi come LiquidFeedback, OpenDCN, Airesis, essendo strutturati come applicativi end-to-end, pongo molta enfasi sull'interazione tra utenti, la presentazione e validazione di proposte, l'eventuale presenza di deleghe, l'espressione del voto tramite una lista di preferenze, ma non permettono la variazione delle logiche di selezione del vincitore da applicare[**?**].

In altri, una fase di analisi iniziale poco trasversale ha portato a dar maggior risalto ad aspetti non funzionali, come la privacy, rispetto a quelli funzionali come il modello di democrazia da applicare, che per alcuni aspetti rimane implicito[**?**]. Ciò è dovuto anche al fatto che il maggior coinvolgimento si concentra principalmente alle fasi iniziali e finali del ciclo di votazione [**?**]. La mancanza di elementi formalizzanti, che descrivono dettagliatamente tutti gli step applicati nel processo, unita alle variazioni ed evoluzioni che vengono apportate nel tempo, portano alla divergenza di *logica del flusso* rispetto all'idea iniziale, che conseguentemente non risulterà chiara agli utilizzatori e potenzialmente potrebbe scoraggiare l'utilizzo della piattaforma[**?**].

Al di là dell'applicazione di e-democracy nel mondo della politica, che riscontra ostacoli più legati all'aspetto organizzativo e umano, che all'aspetto tecnico, vi sono esempi notabili anche nel campo della digitalizzazione pervasiva, ad esempio nell'ambito delle *smart cities*. Ricollegandosi al fenomeno dei cambiamenti climatici che stanno inesorabilmente continuando a proliferare, in certe zone del mondo l'acqua potabile sta diventando un bene sempre meno reperibile e se sempre più da salvaguardare. [**?**] pone il focus sull'adozione di sistemi di "IdroInformatica". Questo tipo di sistemi permettono di analizzare e monitorare, attraverso sensori e sistemi real-time SCADA (Supervisory Control And Data Acquisition), gli attuali flussi e pressioni nella rete di distribuzione ed effettuare predizioni sui futuri guasti. Questi strumenti di analisi possono fornire ai cittadini informazioni su come ottimizzare l'utilizzo delle risorse. L'unione di questi sistemi, ad una politica di

gestione elettronica e democratica che porti a prendere parte alle decisioni per i cambiamenti che coinvolgono il regime di gestione presente in una determinata area, permetterebbe una miglior localizzazione circa le soluzioni da intraprendere, ed ai cittadini una maggior consapevolezza, in quanto sono i principali *stakeholders* e a loro viene attribuito anche un principio di responsabilità. In altri Stati, come Canada e Paesi Bassi, la gestione dell'acqua è delegata ad un gestione locale, che promuove la collaborazione tra soggetti pubblici e privati.

Questi nuovi approcci alla gestione democratica degli interessi dei cittadini aprono nuovi scenari di trasparenza e fiducia reciproca, permettendo un maggior capacità espressione di espressione verso i rappresentanti ma anche da parte di chi è ora in grado di percepire, in ottica di miglioramento continuo, le esigenze del mondo "reale". L'intermediazione, applicata a questi nuovi contesti, continua a rimanere essenziale. Piuttosto che diventare un nuovo "medium" per imporre decisioni, risulta più efficace per favorire lo scambio costruttivo, mantenendo comunque le opportune autonomie riguardo alle decisioni finali, tutelando gli interessi collettivi e al contempo educando le persone a mantenere la libertà di pensiero[**?**].

Capitolo 3

Analisi

In questo capitolo verrà descritta l'analisi sugli aspetti che dovranno caratterizzare la libreria multipiattaforma `EleKtion`.

Verrà esposto un approfondimento sui tipi di voto ed alcuni algoritmi applicabili per ciascun tipo; successivamente sarà definito un elenco di requisiti che dovranno essere considerati nelle fasi successive e infine, sarà fornita una modellazione del dominio mediante diagramma UML.

3.1 Voto a singola preferenza

La tipologia di voto a singola preferenza impone al votante di effettuare una scelta tra un numero definito di concorrenti. Al termine della votazione, tipicamente si adotta il criterio della votazione maggioritaria: si contano i voti ottenuti da ciascun concorrente, e colui che ottiene il numero più elevato, risulta vincitore. A seconda del contesto preso in considerazione, è possibile avere più turni per diminuire di volta in volta il numero di concorrenti coinvolti. Se si arriva ad un pareggio, è possibile effettuare un nuovo ballottaggio oppure adottare criteri secondari, che variano in base al dominio trattato.

3.2 Voto a lista di preferenze

La tipologia di voto a lista di preferenze non impone al votante di effettuare un'unica scelta, bensì di fissare un ordinamento tra i concorrenti che parta dal meno preferito al più preferito o viceversa. Questa logica permette di conservare più contenuto informativo sulle preferenze rispetto alla singola scelta, e offre anche una panoramica di gradimento degli altri concorrenti. Questo tipo di voto è stato introdotto inizialmente dall'algoritmo di Condorcet ed adottato anche nelle sue varianti. Di seguito viene data una spiegazione sul funzionamento dell'algoritmo di Condorcet, e di un suo derivato, l'algoritmo di Schultze.

3.2.1 Algoritmo di Condorcet

L'algoritmo di Condorcet permette di estrarre un vincitore tra n concorrenti, applicando il criterio di maggioranza nei confronti tra le coppie di concorrenti. Si generano le possibili coppie, si confrontano e si valuta quale coppia ha ottenuto il maggior numero di preferenze. Nello scontro a coppie, un concorrente è vincitore se in un voto occupa una posizione preferenziale rispetto all'altro.

Di seguito viene riportato un esempio, supponendo di avere tre possibili concorrenti, A, B, C, e 60 differenti votanti, che ordinano i concorrenti dalla posizione 1 alla posizione 3 dove la posizione 1 indica la posizione di maggior preferenza. Si ottengono delle liste di preferenze come rappresentato in tabella 3.1, in cui è mostrata l'occorrenza di ciascuna lista.

Posizione 1	Posizione 2	Posizione 3	Numero di occorrenze della lista
A	C	B	23
B	C	A	19
C	B	A	16
C	A	B	2

Tabella 3.1: Tabella delle liste di preferenze, con l'occorrenza di ciascuna lista

Ora è possibile effettuare confronti tra le coppie, partendo ad esempio dalla coppia (B,C)

- Nella prima lista C è in una posizione favorevole rispetto a B, quindi C ottiene 23 punti
- Nella seconda lista B è in una posizione favorevole rispetto a C, quindi B ottiene 19 punti
- Nella terza lista C è in una posizione favorevole rispetto a B, quindi C ottiene 16 punti
- Nella quarta lista C è in una posizione favorevole rispetto a B, quindi C ottiene 2 punti

Si ha che il concorrente C batte il concorrente B con un punteggio di 41 a 19.

Facendo i confronti tra le coppie (A,B) e (A,C) si ottiene che il concorrente B batte il concorrente A con un punteggio di 35 a 25, e il concorrente C batte il concorrente A con un punteggio di 37 a 23. Il risultato finale è quindi il seguente:

- Posizione 1: C
- Posizione 2: B
- Posizione 3: A

In alcuni casi il metodo di Condorcet può non portare ad alcun vincitore, in quel caso si ha un ciclo, ad esempio, trovandosi in una situazione in cui C $>$ B, B $>$ A, A $>$ C.

Le varianti del metodo di Condorcet ottimizzano l'algoritmo per minimizzare il rischio di indeterminazione del vincitore. Può accedere, in Condorcet come nelle sue varianti, di ottenere dei pareggi, i quali avvengono quando due o più concorrenti pareggiano l'uno con l'altro ma battono tutti i restanti. La situazione di pareggio può essere gestita in vari modi, ad esempio, estraendo una sequenza

in modo casuale, valutando altri criteri come la preferenza nei singoli voti (most first choice) oppure mantenendo il set di pareggio senza effettuare alcuna azione.

Dato il risultato della votazione, è possibile definire l'utilità individuale come il grado di soddisfazione del votante. Il grado di soddisfazione sarà tanto più alto quanto più vicino è il risultato rispetto al voto espresso.

Concludendo, l'algoritmo porta ad un risultato nel quale si tende a massimizzare l'utilità per il maggior numero di votanti, ossia a massimizzare l'utilità complessiva.

3.2.2 Algoritmo di Schultze

L'algoritmo di Schultze, conosciuto anche come Schwartz Sequential Dropping (SSD), è una variante di Condorcet basata sulla sommatoria del conteggio delle preferenze tra i concorrenti.

Ad esempio, nel caso in cui si hanno 3 candidati, A, B, C, e questi vengono ordinati in una lista di preferenze dalla posizione 1 alla posizione 3 dove la posizione 1 indica la posizione di maggior preferenza, si può avere il seguente risultato:

- Posizione 1: A
- Posizione 2: B
- Posizione 3: C

In questa lista A ottiene 2 punti, B ottiene 1 punto mentre C nessuno.

Riprendendo la situazione presente in tabella 3.1 e moltiplicando i punti per il numero di occorrenze della lista, si ha un totale dei punti pari a 48 per la lista in oggetto. Proseguendo con l'algoritmo, la situazione finale è mostrata in tabella 3.2

concorrente	x 23	x 19	x 16	x 2	Sommatoria
A	2	0	0	1	48
B	0	2	1	0	54
C	1	1	2	2	78

Tabella 3.2: Tabella delle sommatorie ottenute con l'applicazione dell'algoritmo di Schultze

Prendendo in considerazione le sommatorie in ordine decrescente, otteniamo, il seguente risultato:

- Posizione 1: C
- Posizione 2: B
- Posizione 3: A

Il risultato di Schultze non coincide sempre con il risultato di Condorcet.

3.3 Requisiti

La libreria `EleKtion` fornirà la possibilità di creare ed effettuare votazioni. Essa non tratterà la parte preliminare ad una votazione, quale fase di proposta, discussione, eventuali pre-votazioni o quorum, ma porrà l'obiettivo sulla definizione e gestione dei concorrenti coinvolti, la tipologia di algoritmo utilizzato per ottenere il risultato finale ed i voti presenti nella fase di votazione. Al termine della stessa, sarà disponibile una classifica che riporta informazioni circa le specifiche impostate e le caratteristiche dei concorrenti.

Ciascuna votazione avrà associata una competizione con un metrica che caratterizza l'operato del concorrenti. Prendendo esempi di riferimento legati al mondo del *motor sport*, si potrebbe pensare ai punti totalizzati per ogni gara, come nel caso di *Formula 1* o *Moto GP*.

Seppur in molti casi le regole non cambiano in modo drastico, ciascuno sport è influenzato da molti fattori: i concorrenti, gli ambienti di gara, i tifosi, e altri aspetti, che contribuiscono a rendere ogni campionato come un capitolo a sè stante.

Non per ultimo, ognuno di essi ha un proprio metodo di calcolo ed organizzazione delle competizioni. Soprattutto in caso di pareggi, è opportuno avere una logica per risolvere il conflitto e determinare un ordine specifico per le classifiche. Se si considerasse l'esito di ciascuna gara come un votazione, cambiando l'algoritmo utilizzato in un campionato, si otterrebbero esiti differenti, o ancora, se si cambiasse la logica di vittoria nelle fasi di un campionato, si potrebbe simulare come sarebbe stato l'esito finale con regole differenti.

3.3.1 Requisiti funzionali

- Definizione di un gestore di votazioni, intesi come contenitore delle stesse. Ciascuna votazione dovrà essere indipendente dalle altre. L'eventuale collegamento tra le stesse sarà demandato all'utilizzatore della libreria.

- Definizione dei concorrenti che rappresenteranno i candidati disponibili, delle relative caratteristiche e di eventuali punteggi, che potranno essere presenti a

seconda del dominio della votazione. Gli eventuali punteggi saranno collegati ad una specifica metrica di interesse.

- Definizione dell'algoritmo da utilizzare. L'algoritmo dovrà determinare la logica con cui saranno confrontati e selezionati i concorrenti. L'algoritmo potrà utilizzare la metrica disponibile per effettuare ulteriori valutazioni in occasione di pareggio o altre casistiche definibili dall'utilizzatore. L'algoritmo potrà essere parametrizzato, per impattare il proprio funzionamento.

- Definizione dei voti presenti nella votazione. La tipologia di voto dovrà essere dipendente dall'algoritmo utilizzato, e in ogni caso potrà essere dei seguenti tipi:

  1. Voto a singola preferenza
  2. Voto a lista di preferenze

  Dovrà essere data la possibilita si anonimizzare il votante, in modo tale che successivamente non sia possibile risalire al suo identificativo.

- Le fasi di definizione dovranno essere correlate ad opportuni controlli di congruità, al fine di elaborare la votazione in modo coerente.

- La classifica finale dovrà essere disponibile e visualizzabile dall'utilizzatore. Dovrà esser data evidenza di eventuali situazioni di pareggio, di quale algoritmo è stato utilizzato e delle caratteristiche dei concorrenti.

3.3.2 Requisiti non funzionali

- Dovrà essere possibile definire tutti gli elementi necessari attraverso l'uso di costrutti che agevolino la rapidità di creazione, evitando che l'utilizzatore effettui manualmente i controlli minimi ed obbligatori.

- La libreria dovrà essere disponibile su molteplici piattaforme, facilmente importabile in esse ed avere una documentazione a corredo.

3.4 Modello del dominio

`EleKtion` avrà a disposizione uno o più gestori di votazioni. I gestori sono indipendenti tra loro e così anche le votazioni all'interno degli stessi, fatto salvo l'uso coerente della tipologia di votazione e della metrica di punteggio, che rimane stabile in ogni gestore.

Non è contemplata l'ipotesi di più turni nella stessa votazione, quindi in tal caso dovranno essere gestiti come votazioni separate.

Una votazione avrà associata una determinata competizione, alla quale è associato l'elenco dei concorrenti.

Ciascun concorrente può avere una serie di punteggi, e la tipologia di punteggio è strettamente collegato alla tipologia di competizione.

La votazione, inoltre, ha associato l'elenco dei voti, i quali hanno riferimento sia del votato che del votante, in caso di votazione non anonima. È importante che un voto, per essere considerato valido, sia riferito ad un candidato esistente e sia della tipologia ammessa dalla votazione.

Infine, ci può essere un solo algoritmo collegato alla votazione, il quale può contenere dei parametri per effettuare variazioni nella configurazione per il suo funzionamento. La votazione avrà un' unica classifica finale, la quale può essere consultata.

In figura 3.1 viene rappresentato il diagramma UML con le interfacce che modellano il dominio descritto in questo capitolo.

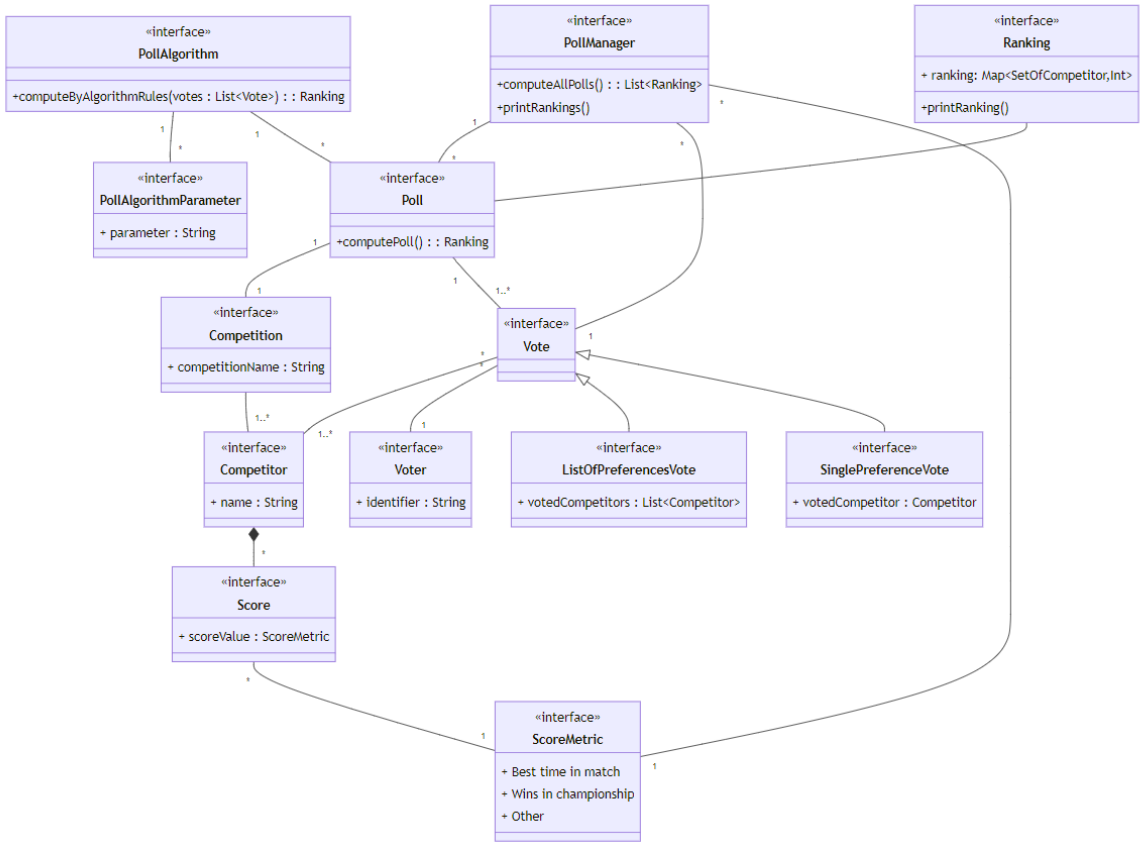

Figura 3.1: Diagramma UML del modello di dominio di `EleKtion`

Capitolo 4

Design

In questo capitolo, partendo dalla modellazione del dominio descritta nel capitolo 3.4, sarà sviluppato un approfondimento sulla definizione delle metriche di punteggio e sulla porzione impattata dall'introduzione di DSL, volto al miglioramento dell'utilizzabilità della libreria `EleKtion`. Per ciascuna funzionalità verrà evidenzata un'apposita descrizione e un diagramma UML che esplichi come saranno organizzate le interfacce e le classi astratte.

Questi diagrammi sono realizzati con il presupposto di poter adottare l'utilizzo dei seguenti costrutti:

- Tipi `generici` [**?**];
- `Overloading` degli `operatori` [**?**];
- `Higher order functions` (in particolare `function literal with receiver`) [**?**];
- `Companion object` [**?**].

Si rimanda alle fonti per l'approfondimento specifico in `Kotlin`, la sintassi utilizzata nei diagrammi è generica ed astrae dal linguaggio specifico.

4.1 Domain Specific Language per la definizione del gestore di votazioni

Questo DSL è pensato per facilitare la creazione del gestore di votazioni. Un gestore di votazioni viene vincolato ad adottare uno specifico tipo `S`, sottotipo di `ScoreMetric`, ed uno specifico tipo di voto `V`, sottotipo di `Vote`.

Una metrica è definibile come il criterio tale per cui un concorrente può essere messo a confronto con un altro, sulla base della sua prestazione nella competizione. Pertanto, è rappresentabile come una comparazione tra valori. Dall'interfaccia `ScoreMetric` possono essere definiti numerosi sottotipi utili alla competizione trattata, ad esempio il punteggio di un concorrente può essere legato ai punti assegnati in una gara, alle vittorie ottenute in campionato, oppure a criteri legati al tempo. Per facilitarne la creazione è utile definire una funzione di supporto contenuta in un `companion object`, come mostrato in figura 4.1.

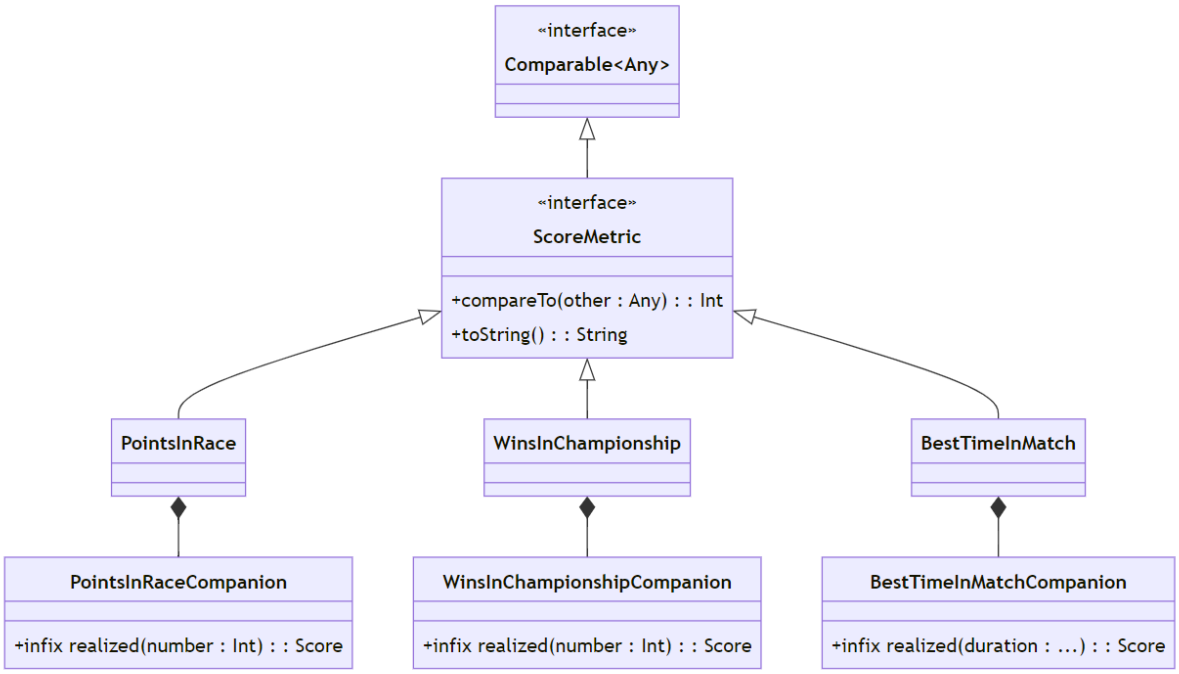

Figura 4.1: Diagramma UML dell'organizzazione di una metrica

I tipi S e V vincolano i seguenti componenti che verranno trattati, pertanto non è possibile adottare diversi metriche di punteggio tra i concorrenti, o diversi tipologie di voto, tra le varie votazioni presenti nel medesimo gestore.

Il punto di ingresso dell'intero flusso è rappresentato dalla funzione `initializedAs`, al suo interno è possibile inizializzare una votazione attraverso la funzione `poll`, e l'utilizzo dell'operatore unario `+`, per l'aggiunta di quest'ultima alla lista di votazioni. In figura 4.2 viene mostrato il diagramma UML delle interfacce e della classe astratta.

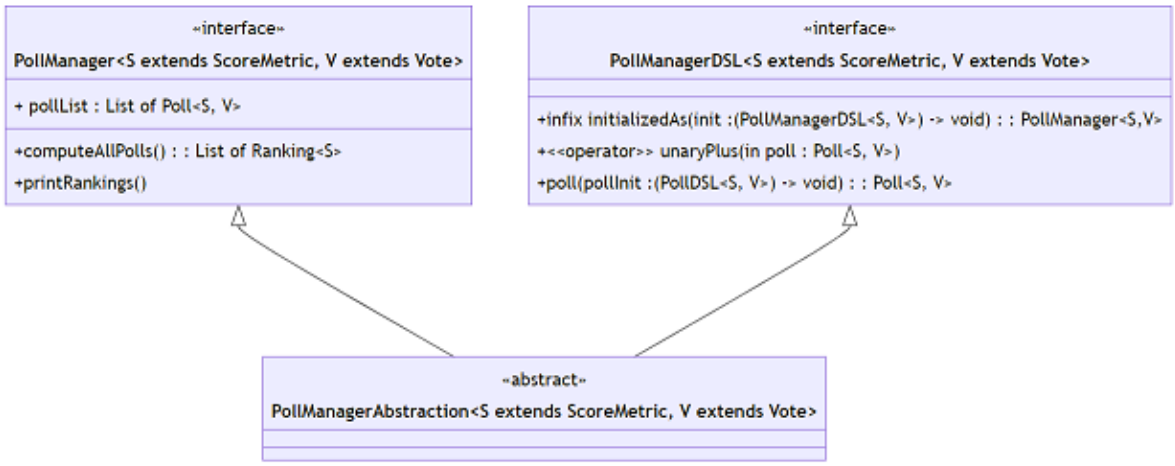

Figura 4.2: Diagramma UML del DSL per la definizione del gestore di votazioni

4.2 Domain Specific Language per la definizione della votazione

Questo DSL è pensato per facilitare la creazione di una votazione. In una votazione è obbligatorio scegliere l'algoritmo con cui sarà elaborato l'esito, l'algoritmo può ricevere parametri utili al uso funzionamento, attraverso l'utilizzo dell' operatore unario `+` presente in `PollAlgorithmDSL`. In base al tipo di voto (vedi capitolo 3.3.1) è utile definire un costrutto per agevolarne la creazione. Nel caso in cui il voto sia a singola preferenza, è necessario specificare l'identificativo del votato e l'identificativo del votante (vedi funzione `votedBy` in `SPVoteAlgorithmDSL`, figura 4.3). Nel caso in cui quest'ultimo non sia richiesto, corrisponderà ad un identificativo casuale (vedi funzione `asAnonymousVote` in `SPVoteAlgorithmDSL`, figura 4.3). Nel caso in cui il voto sia a lista di preferenze, per costruire la lista è necessario specificare l'elenco degli identificativi dei votati, utilizzando la funzione `then` in concatenazione (vedi funzione `then` in `LOPVoteAlgorithmDSL`, figura 4.3)). Allo stesso modo del voto a singola preferenza, è prevista la possibilità di anonimizzare il votante. Come visibile in figura 4.3, sono previsti metodi per semplificare l'inizializzazione dei possibili algoritmi:

- `majorityVotesAlgorithm`: rappresenta la creazione di un algoritmo a maggioranza, in cui il vincitore è colui che ricevuto più voti;
- `majorityVotesHScoreAlgorithm`: rappresenta la creazione di un algoritmo a maggioranza, in cui il vincitore è colui che ricevuto più voti. Se si verificano pareggi, viene selezionato il concorrente che ha il punteggio maggiore, secondo la metrica di punteggio definita;
- `majorityVotesHScoreAlgorithm`: rappresenta la creazione di un algoritmo a maggioranza, in cui il vincitore è colui che ricevuto più voti. Se si verificano pareggi, viene selezionato il concorrente che ha il punteggio minore, secondo la metrica di punteggio definita;
- `condorcetAlgorithm`: rappresenta la creazione dell'algoritmo di Condorcet;

- `schultzeAlgorithm`: rappresenta la creazione dell'algoritmo di Schultze.

In figura 4.4 vengono mostrati gli altri metodi utili alla creazione di una votazione: La funzione `competition`, utile alla creazione della competizione, richiede che venga definito un nome alla competizione e un inizializzatore in cui saranno definiti i concorrenti. Quest'ultimo, così come l'algoritmo desiderato, sono effettivamente assegnati alla votazione attraverso l'uso dell'operatore unario `-`. Infine è possibile aggiungere un elenco di voti, preferibilmente creati utilizzando i metodi descritti nella prima parte di questo sotto capitolo. Ciascun voto richiede di essere effettivamente aggiunto alla lista di voti attraverso l'uso dell' operatore unario `+`.

Per motivi di spazio, nelle figure 4.3 e 4.4 sono stati utilizzati gli acronimi *SP* e *LOP*, al posto di, rispettivamente, *SinglePreference* e *ListOfPreferences*.

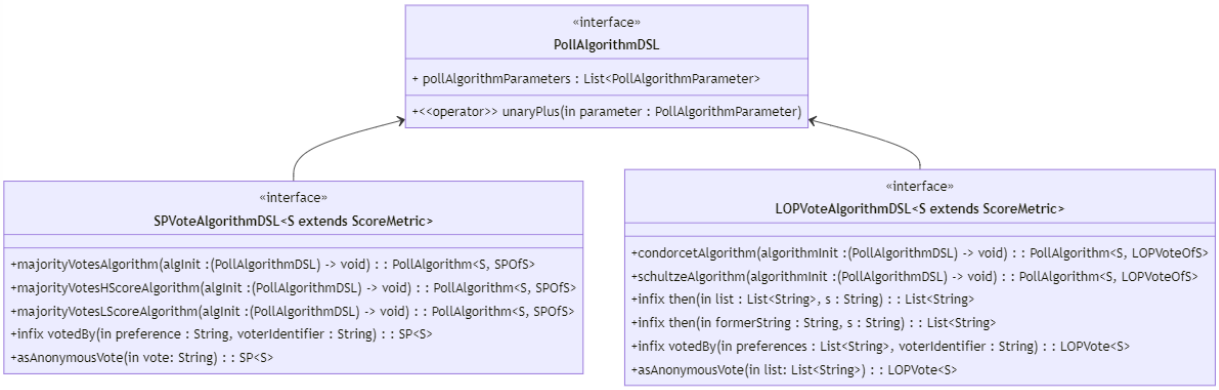

Figura 4.3: Diagramma UML del DSL per la porzione di definizione dell'algoritmo

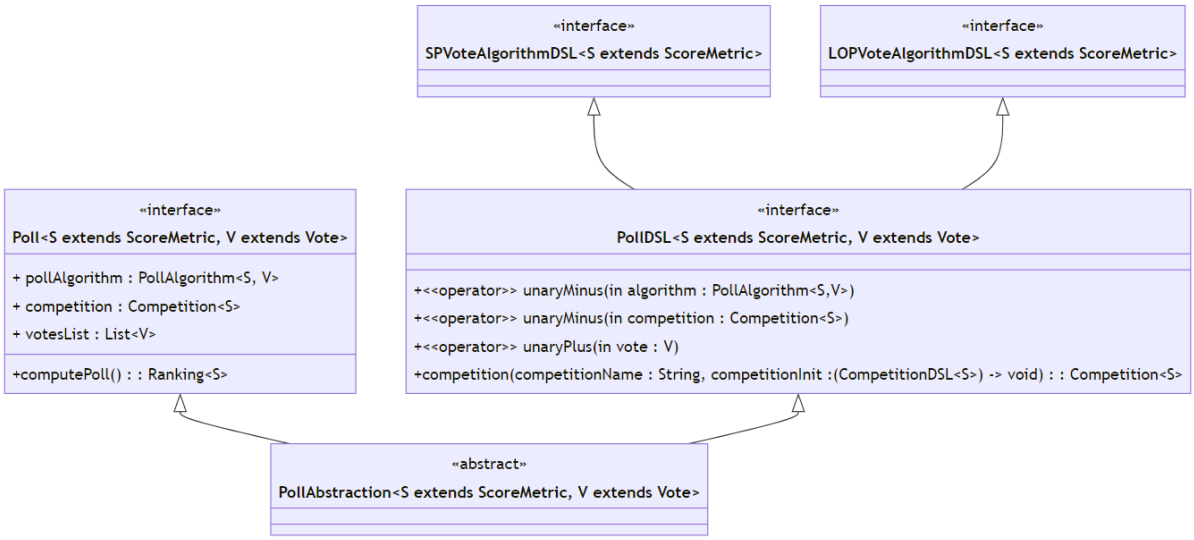

Figura 4.4: Diagramma UML del DSL per la porzione di definizione della votazione

4.3 Domain Specific Language per la definizione della competizione

Questo DSL è pensato per facilitare la creazione ed il popolamento di una competizione. La funzione `competitor`, utile alla creazione del concorrente, richiede che venga definito l'identificativo del concorrente e un inizializzatore in cui saranno definiti i punteggi realizzati. Il concorrente istanziato, viene aggiunto alla lista dei concorrenti grazie all'uso dell'operatore unario `+`.

In figura 4.5 viene mostrato il diagramma UML delle interfacce e della classe astratta.

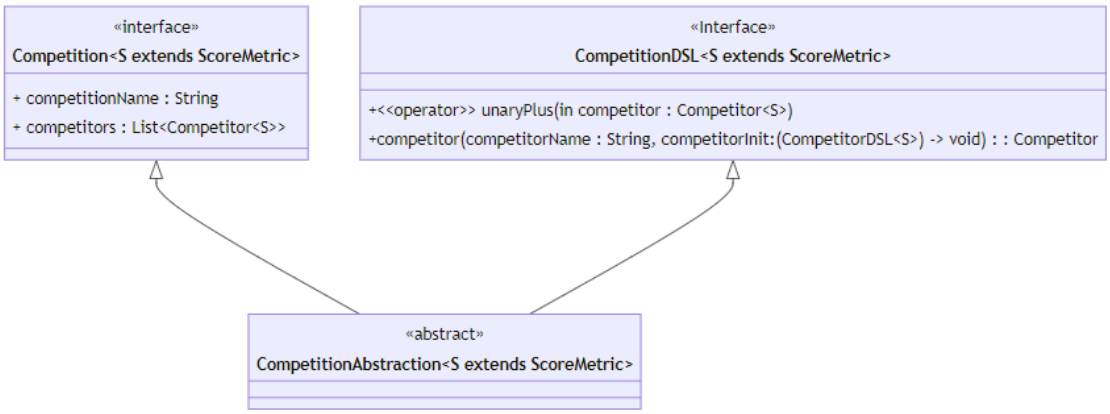

Figura 4.5: Diagramma UML del DSL per la definizione della competizione

4.4 DSL per la definizione del concorrente

Questo DSL è ideato per facilitare la creazione di un concorrente. Ciascun concorrente può avere un elenco di punteggi, e ciascuno di essi può essere assegnato al concorrente (vedi capitolo 4.3) attraverso l'uso dell'operatore unario +. .

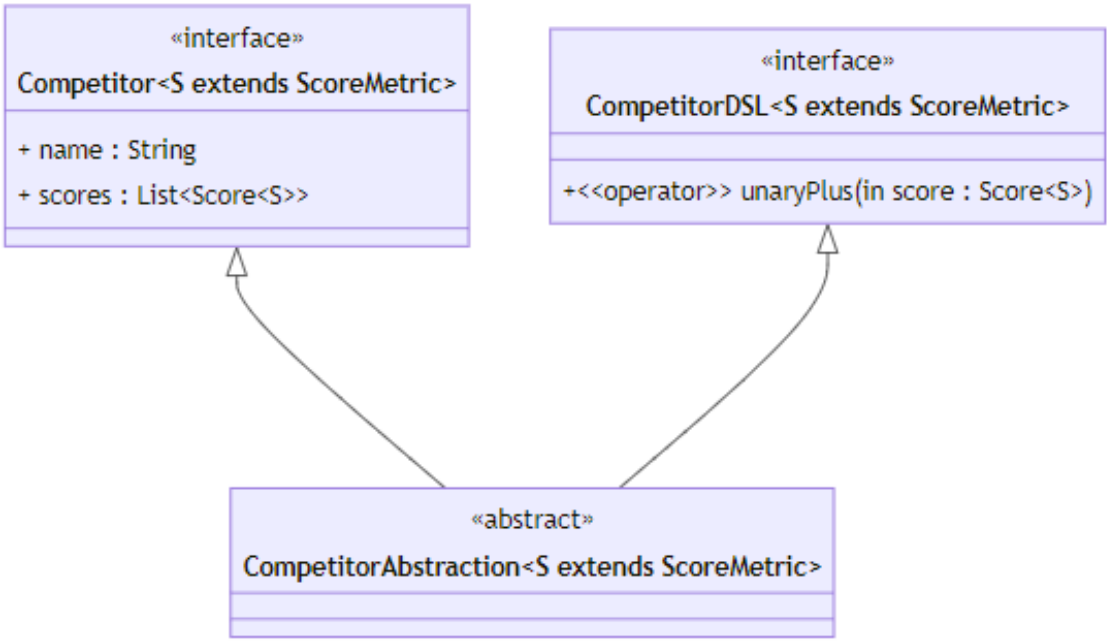

Figura 4.6: Diagramma UML del DSL per la definizione della concorrente

Capitolo 5

Implementazione

5.1 Build automation e versionamento degli artefatti

L'adozione di un sistema di build automation è un fattore chiave per la produzione efficace ed efficiente di software di qualità.

Come principio base, è importante che ciascuna modifica apportata al codice sia testata a livello unitario e abbia documentazione correlata, che sia l'aggiunta di una funzionalità piuttosto che la variazione di un campo, a titolo meramente esemplificativo.

Inoltre, queste modifiche devono essere ulteriormente verificate attraverso test di integrazione, in modo da garantire la piena compatibilità tra i componenti.

Le verifiche possono essere effettuate manualmente dallo sviluppatore, ma per ridurre il rischio di errori è possibile adottare un sistema di automazione, che attraverso una serie di task predefiniti e ripetibili, verifichi automaticamente le varie fasi, gestendo anche le dipendenze tra i task stessi. Così facendo, si può subito notare se tutto è andato a buon fine oppure se è necessario effettuare ulteriori eventi correttivi.

Una volta che il flusso di compilazione e verifica viene terminato con successo, è possibile procedere alla generazione di uno o più artefatti e della relativa

documentazione. Questi prodotti possono essere sottoposti ad un operazione di versionamento, in modo da distinguere le evoluzioni del software e semplificarne la distribuzione e la tracciabilità.

In questo progetto sono stati adottati `Kotlin` come linguaggio di programmazione e `Gradle` come build automator.

Poichè gli artefatti che vengono generati sono dipendenti dall'architettura della macchina e dal sistema operativo sui quali avviene il processo di build, è opportuno eseguire tale processo su piattorme dedicate come Github. Github fornisce ambienti standard e personalizzabili in base a direttive, gestendo il processo di Continuous Integration (CI)/Continuous Delivery (CD) che è stato descritto in questo capitolo. Infine, per favorire la distribuzione e la compatibilità verso molteplici sistemi, è utile adottare strumenti come `Kotlin Multiplatform`, che semplificano la generazione dell'output finale adattandolo alle specifiche piattaforme.

5.1.1 Kotlin Multiplatform

Kotlin Multiplatform è un tool che favorisce il riuso di codice tra molteplici piattaforme. Grazie ad esso, riutilizzando la logica applicativa scritta in *Kotlin*, è possibile produrre artefatti compatibili in molteplici piattaforme, dette anche *target*, come `JVM`, `JS`, `Android`, `iOS` e piattaforme native come `Linux`. I *target* sono disposti secondo una gerarchia predefinita, comunque modificabile in caso di necessità, e sono previsti *target* intermedi. Per ogni *target* viene definito un *source set*, ossia un insieme di file sorgente, che definisce anche le dipendenze e le compatibilità. Il *source set* predefinito è `commonMain`, e contiene il codice comune a tutti i *target*. Al suo interno è possibile utilizzare funzioni e dipendenze che siano compilabili verso tutte le piattaforme dichiarate, mentre quelle che sono legate ad uno specifico linguaggio o architettura vanno definite ed utilizzate all'interno degli opportuni *source sets*, come `jvmMain` e `jsMain`. Un esempio è visibile in figura 5.2, in cui i *target* intermedi sono colorati in azzurro: *apple* conterrà codice compatibile solamente con i *target* che lo estendono, oltre a quello ereditato dai livelli superiori.

Pertanto, è ammesso che *macos* utilizzi funzioni che non siano disponibili in *tvos* ma entrambi accedono al codice messo a disposizione da *apple*.

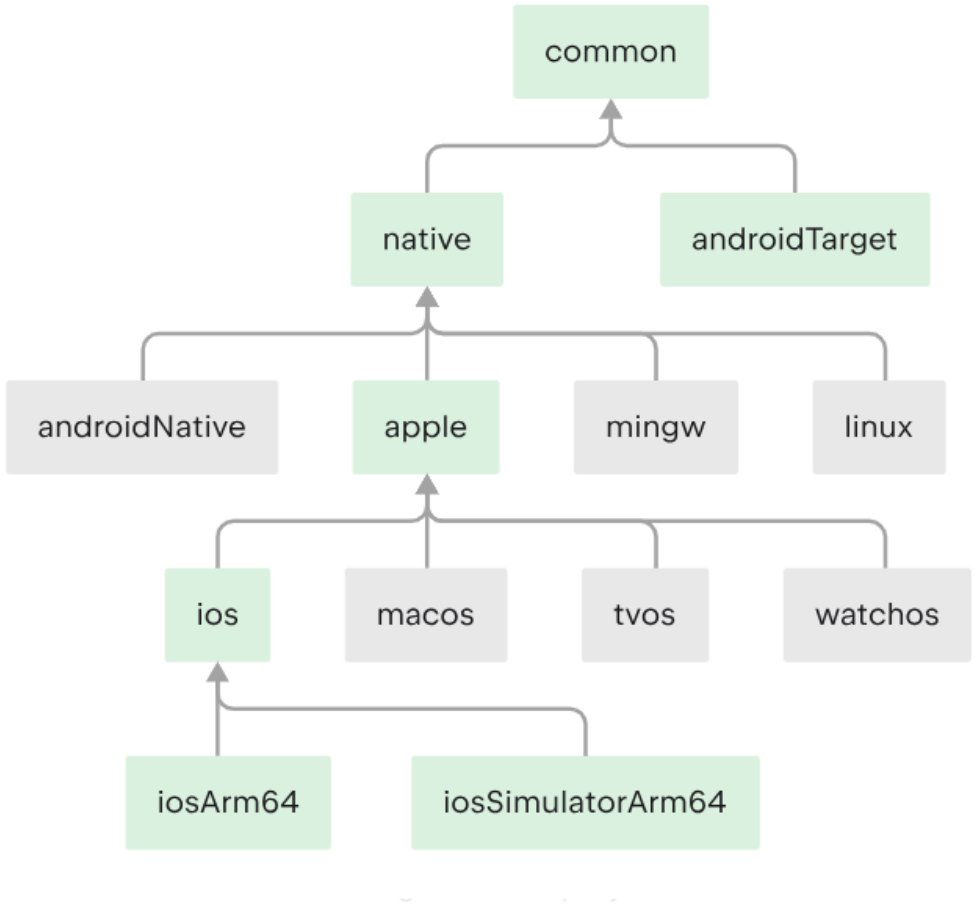

Figura 5.1: Esempio di una possibile gerarchia di *target*. Fonte: [**?**]

Inoltre, questo meccanismo permette di applicare un *template method* architetturale, grazie al quale è possibile dichiarare nel *source set* `commonMain` funzioni agnostiche tramite la direttiva `expected`, senza definirne il contenuto, che sarà valorizzato in un successivo momento. In fase di compilazione, nelle versioni platform-specific, saranno valorizzate tramite funzioni che adottano la di-

rettiva `actual`, che posso richiamare funzionalità altrimenti non adottabili in *commonMain*.

Una logica analoga è applicabile anche alla sezione dei test: è disponibile un *source set* di base, detto *commonTest*, e altri eventuali che contengono codice compatibile solo in quella determinata piattaforma. In figura **??** viene mostrato il flusso e gli output ottenuti utilizzando i *target* `JVM` e `JS`.

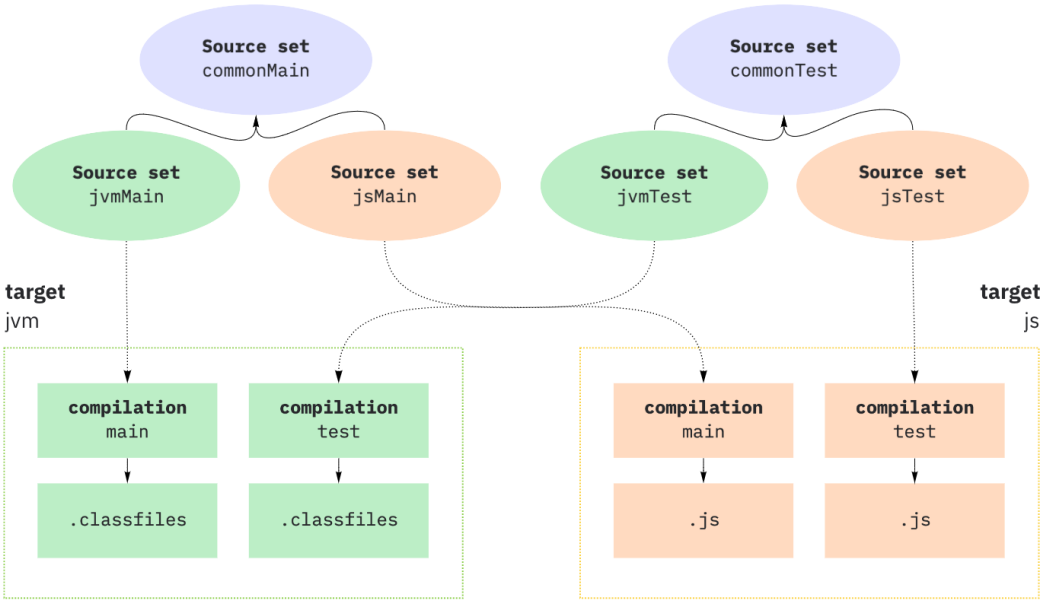

Figura 5.2: Esempio di output ottenibili in un progetto multipiattaforma. Fonte: [?]

5.1.2 GitHub Actions

GitHub Actions (GH Actions) è una piattaforma di CI/CD basata su repository GitHub (GH), ai quali possono essere agganciati degli elementi detti workflow.

Un `workflow`, definibile attraverso la sintassi `YAML`, rappresenta una sequenza di azioni che possono essere eseguite all'occorrenza di un evento, ad esempio la

creazione di un `tag` nel repository, una `richiesta pull`, un'esecuzione manuale richiesta dal proprietario del repository, ecc...

In ogni `workflow` va specificato il nome dello stesso e l'insieme di eventi che ne scatena l'esecuzione. Di seguito a ciò, è possibile definire un insieme di `job`.

Ciascun `job` è legato ad uno specifico ambiente di esecuzione: va definito che tipo di sistema operativo deve essere adottato (`Linux`, `Windows`, `macOS`) e se al termine dello stesso debba essere restituito un output, che sarà accessibile da altri `job`. È possibile specificare delle variabili in una `matrice`, in modo eseguire parallelamente molteplici istanze di uno stesso `job` in base alle combinazioni delle variabili; a titolo esemplificativo, eseguire la compilazione di codice utilizzando la stessa versione del compilatore ma su 3 sistemi operativi differenti.

Ciascun `job` è caratterizzato da un proprio flusso indipendente, di conseguenza per orchestare l'intero processo è opportuno specificare condizioni basate sulle variabili di output oppure sull'indicazione del completamento di uno o più i `jobs` desiderati.

All'interno del `job` sono definibili uno o più `step`: questi vengono eseguiti in maniera sequenziale e sono impostabili per eseguire una sequenza arbitraria di comandi `bash` oppure per eseguire una `action` pubblica. Quest'ultima rappresenta un ottimo modo di favorire il riuso e la condivisione di codice nella community GH, in quanto è possibile, ad esempio, utilizzare un workflow complesso come se fosse un singolo step (`composite action`) oppure lanciare l'esecuzione di un container senza imporre *effort* riguardo alla gestione delle dipendenze e delle configurazioni (`docker container action`).

5.1.3 GitHub Pages

GitHub Pages (GH Pages) è un servizio di hosting disponibile per ogni account ed organizzazione in GH, attraverso il quale è possibile pubblicare un sito web statico, che viene costruito e pubblica attraverso un apposito `workflow` predefinito. Nel caso di un iniziativa progettuale come una libreria, può essere utilizzato dare

informazioni su come scaricare ed installare il software oppure per pubblicare la documentazione.

5.1.4 Maven Central, GitHub Packages e NPM

Maven Central è tra i repository centralizzati più popolari per distribuire e riutilizzare artefatti basati su Java Virtual Machine (JVM). Un repository è definibile come "la directory che memorizza tutti i progetti e le librerie JAR, i plugin o qualsiasi altro artefatto specifico del progetto che può essere facilmente utilizzato da Maven"[**?**].

Il sistema di gestione delle dipendenze di Maven è in grado di risolvere automaticamente le dipendenze transitive, semplificando la gestione manuale delle dipendenze per gli sviluppatori. Gli artefatti caricati su Maven Central, tra cui librerie di codice compilato, documentazione ed elenco di dipendenze, sono immutabili e non possono essere modificati né eliminati. Le dipendenze vengono identificate in modo univoco con una tripletta `groupId:artifactId:version`, dove `groupId` identifica un gruppo di artefatti, `artifactId` si riferisce al nome della libreria e `version` identifica univocamente ogni rilascio della libreria. Ad esempio, la tripletta 'org.jetbrains.kotlinx:kotlinx-serialization-json:1.6.0' identifica la versione 1.6.0 di un serializzatore json per *Kotlin*, gestito da *JetBrains*.

Tutte le versioni di una libreria rilasciate su Maven Central, una volta pubblicate sono sempre disponibili nel repository. Gli utenti della libreria sono liberi di decidere quale versione utilizzare e per quanto tempo e a questi ultimi è delegata la responsabilità di aggiornare le proprie dipendenze per adeguarsi a problemi di sicurezza o evoluzioni dell'API. Nel lungo termine, queste decisioni determinano le librerie e le versioni più popolari nell'intero ecosistema software. Si osserva che per la maggior parte delle librerie, più di una versione è attivamente utilizzata in un dato momento [**?**].

Dallo studio empirico condotto da [**?**], è emerso che "circa il 40% delle librerie ha due o più versioni attivamente utilizzate, mentre quasi il 4% non ha mai avuto alcun utente su Maven Central. Inoltre, si è scoperto che più del 90% delle

versioni più popolari non sono le ultime versioni rilasciate, e sia le versioni attive che significativamente popolari sono distribuite lungo la storia delle versioni della libreria".

Un' alternativa all'uso di Maven Central può risiedere in GH Packages. Quest'ultimo pone una gestione semplificata per la pubblicazione degli artefatti, in quanto strettamente connesso all'uso di repository GH, il che permette di avere un controllo degli accessi efficace come quello disponibile per i repo. Al pari di Maven Central viene mantenuto il concetto di artefatti immutabili, e in più viene dato il supporto per l'archiviazione di pacchetti in numerosi linguaggi, tra cui *C#*.

Un altro famoso gestore di pacchetti è Node Package Manager (NPM), focalizzato sulle diffusione di librerie *Javascript* e *Node.js*, si differenzia in particolare dai precedenti in quanto non adotta il principio di immutabilità. In questo caso, la modifica o eliminazione di una versione di un pacchetto può avere impatti notevoli sugli utilizzatori, portando a potenziali disastri mondiali [**?**].

5.2 DSL

come ho utilizzato gli operator le lambda with receiver esempi di utilizzo del dsl tanti esempi nei listings

5.3 Librerie multiplatform utilizzate

5.4 Pubblicazione di artefatti

Spiegare brevemente i passaggi necessari che sono stati fatti per arrivare alle pubblicazioni

5.5 Pubblicazione della documentazione

spiegare la creazione di un nuovo workflow, e collegato a quelli esistenti

Capitolo 6

Valutazione

6.1 Test realizzati

Descrivere le varie situazioni che sono andato a controllare, mettendo dettagli solo per i casi più complicati

6.2 Sperimentazione con Ergast API

Estratti di demo che ho realizzato usando i dati della Formula 1

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20	G21	G22
C1.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2.	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
C3.	5	5	4	4	4	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3
C4.	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	5	4	5	4
C5.	4	4	5	5	5	6	6	5	5	5	6	7	5	5	5	5	5	5	4	6	4	7
C6.	19	8	10	6	7	7	7	7	6	7	7	5	6	6	6	6	6	7	7	7	7	5
C7.	17	20	8	9	9	11	11	11	10	9	8	8	8	8	8	7	7	6	6	5	6	6
C8.	7	6	7	7	6	5	5	6	7	6	5	6	7	7	7	8	8	8	8	8	8	8
C9.	6	7	6	8	8	8	8	8	8	8	9	9	9	9	9	10	10	11	11	10	10	10
C10.	20	19	13	11	14	13	13	14	14	11	11	11	12	12	11	9	9	9	9	9	9	9
C11.	9	11	14	14	10	10	10	10	11	12	12	12	10	10	10	11	11	10	10	11	11	11
C12.	10	13	18	17	18	18	18	12	13	13	13	13	13	13	13	13	13	13	13	13	13	13
C13.	18	10	12	13	12	9	9	9	9	10	10	10	11	11	12	12	12	12	12	12	12	12
C14.	11	14	16	16	16	16	16	17	17	17	17	17	17	17	17	17	17	16	16	14	14	14
C15.	8	9	11	12	13	14	14	15	15	15	15	15	15	15	15	15	14	14	14	15	15	15
C16.	15	15	9	10	11	12	12	13	12	14	14	14	14	14	14	14	15	15	15	16	16	16
C17.	16	17	15	15	15	15	15	16	16	16	16	16	16	16	16	16	16	17	18	18	18	18
C18.	13	12	17	18	17	17	17	18	18	18	18	18	18	18	18	18	18	18	19	19	19	19
C19.	12	16	19	19	19	20	20	20	19	19	19	19	19	19	20	20	20	20	21	21	21	21
C20.	14	18	20	20	20	19	19	19	20	20	20	20	20	21	21	21	21	21	22	22	22	22
C21.	20	20	20	20	20	20	20	20	20	20	21	21	21	22	22	22	22	22	17	17	17	17
C22.	20	20	20	20	20	20	20	20	20	20	21	21	22	20	19	19	19	19	20	20	20	20

Tabella 6.1: Tabella relativa al Mondiale F1 2023, in cui per ogni gara (G*) è riportata la posizione del concorrente (C*) in classifica generale, al termine della gara stessa. I valori sono ricavati da Ergast API [**?**]

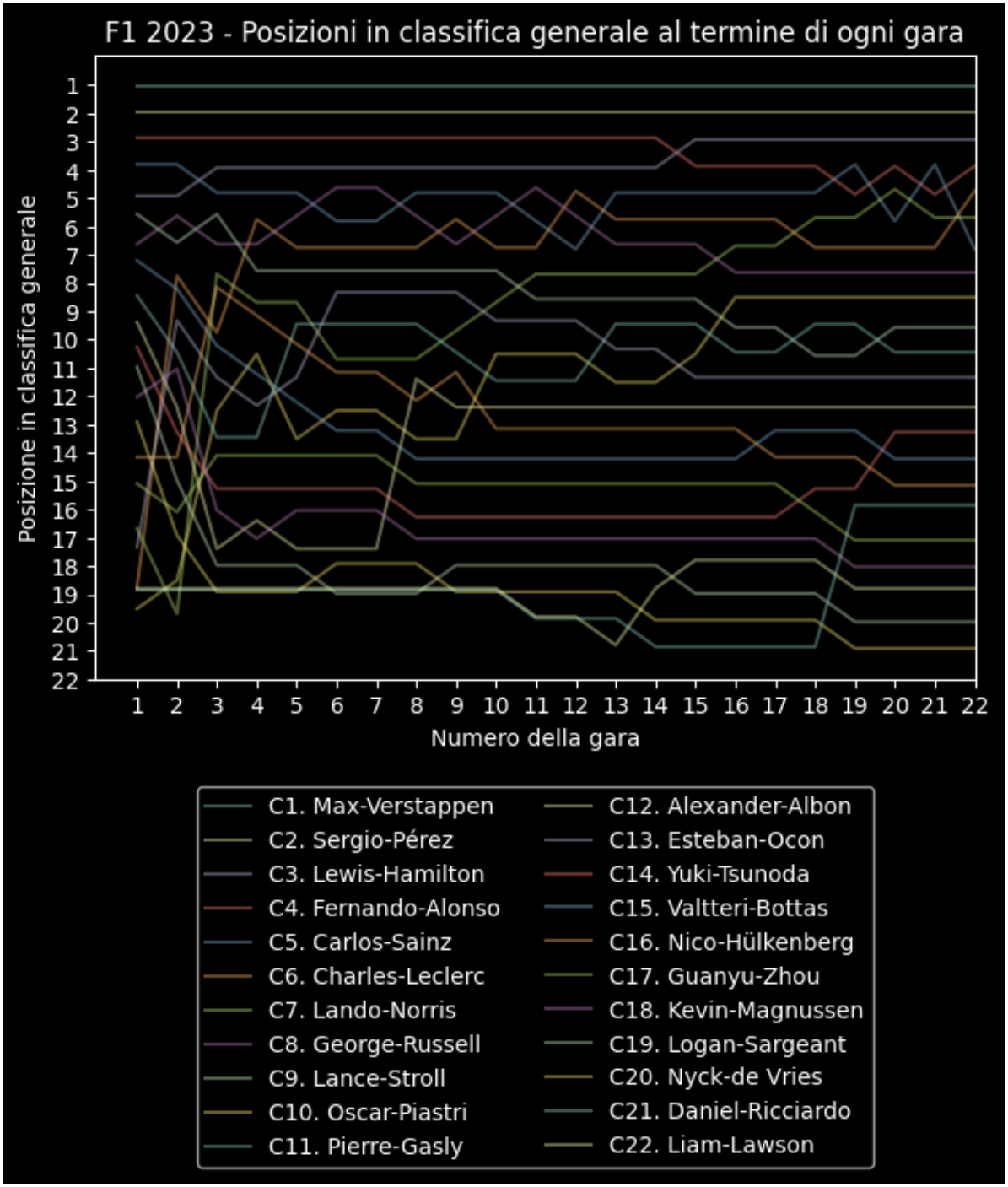

Figura 6.1: Rappresentazione grafica della tabella 6.1

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20	G21	G22
C1.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2.	2	1	2	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
C3.	5	3	4	3	5	4	4	3	4	3	4	4	4	4	4	3	4	4	4	3	4	3
C4.	3	2	3	2	3	3	3	2	3	2	3	3	3	3	3	3	3	3	3	3	3	4
C5.	4	3	5	3	4	5	6	4	5	5	7	7	7	6	5	4	5	5	5	4	5	4
C6.	19	10	14	11	6	7	7	6	7	5	6	6	6	6	7	5	5	7	5	6	5	4
C7.	17	11	13	7	11	10	11	10	11	8	10	9	8	7	8	6	6	6	6	5	6	5
C8.	7	4	7	4	6	6	5	5	6	4	5	5	5	5	6	5	7	6	7	5	7	6
C9.	6	11	6	4	6	9	7	7	8	6	8	8	8	7	9	9	9	10	9	9	8	7
C10.	20	11	14	8	15	12	12	10	13	10	11	12	9	10	11	8	8	9	8	8	9	7
C11.	9	5	8	6	7	8	8	8	9	7	9	10	8	8	10	7	8	8	8	7	8	8
C12.	10	11	14	11	13	13	16	12	14	11	11	13	9	9	11	8	9	10	9	9	10	9
C13.	18	10	11	9	9	9	9	8	10	9	11	11	9	9	11	9	9	11	9	10	11	9
C14.	11	6	8	5	8	9	10	9	12	12	13	13	11	12	13	11	11	13	11	11	13	10
C15.	8	10	11	9	12	11	13	11	13	11	12	13	10	11	12	10	10	12	10	12	12	11
C16.	15	8	9	7	10	11	11	12	15	13	14	14	12	13	14	12	12	14	12	12	14	11
C17.	16	9	10	10	13	12	14	13	15	13	15	14	13	14	15	13	13	15	13	13	15	12
C18.	13	7	12	9	9	12	13	13	15	15	16	15	14	15	16	14	14	16	14	14	16	13
C19.	12	10	13	11	16	14	17	15	17	15	18	16	15	16	17	15	15	17	15	15	17	14
C20.	14	9	12	11	14	13	15	14	16	14	17	16	16	17	18	16	16	18	16	16	18	15
C21.	21	12	15	12	17	15	18	16	18	16	19	17	17	18	19	17	17	19	17	17	19	16
C22.	22	13	16	13	18	16	19	17	19	17	20	18	18	19	20	18	18	20	18	18	20	17

Tabella 6.2: Tabella relativa al Mondiale F1 2023, in cui per ogni gara (G*) è riportata la posizione del concorrente (C*) in classifica generale, al termine della gara stessa. I valori sono ricavati dall'applicazione dell'algoritmo di Condorcet al termine di ogni gara, considerando anche le precedenti. Ogni gara rappresenta un voto, composto dagli ordini di arrivo.

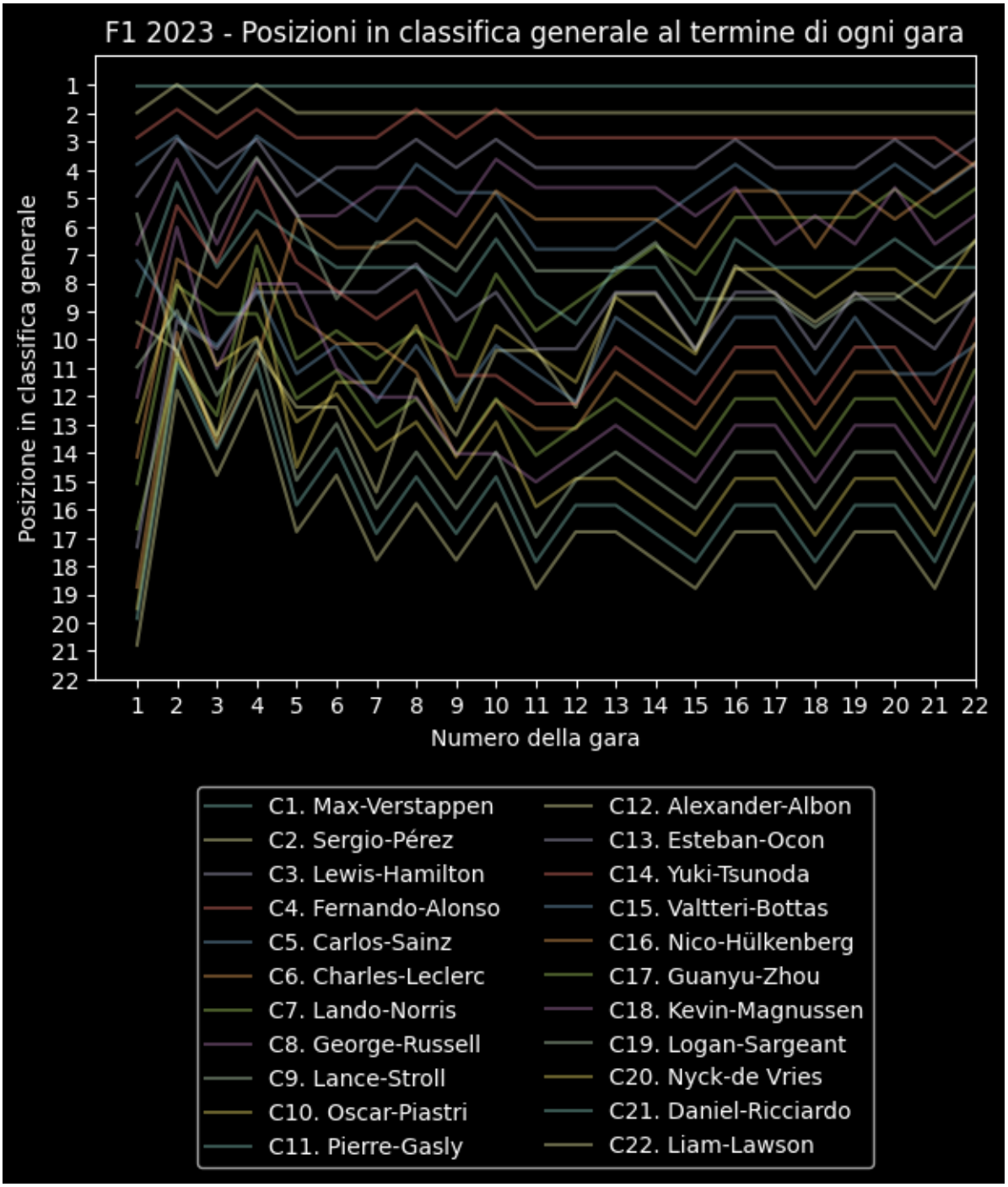

Figura 6.2: Rappresentazione grafica della tabella 6.2

	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10	G11	G12	G13	G14	G15	G16	G17	G18	G19	G20	G21	G22
C1.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
C2.	2	1	2	2	2	3	4	4	3	4	3	2	2	2	2	3	4	2	3	3	2	2
C3.	5	3	4	4	4	4	3	3	4	3	2	3	3	3	3	2	2	4	2	2	3	3
C4.	3	2	3	3	3	2	2	2	2	2	2	4	2	3	4	4	3	3	4	4	4	4
C5.	4	3	5	5	5	5	5	5	5	5	4	6	4	4	5	5	5	5	5	5	5	5
C6.	19	8	16	9	9	8	8	7	7	7	6	7	6	6	6	6	6	8	8	8	8	8
C7.	17	12	13	9	12	12	11	12	11	10	7	8	7	7	8	8	8	6	6	6	6	6
C8.	7	4	6	6	6	6	6	6	6	6	5	5	5	5	7	7	7	7	7	7	7	7
C9.	6	8	7	6	7	11	9	10	9	8	8	9	8	8	10	11	11	11	11	11	11	11
C10.	20	13	14	12	14	15	13	13	14	12	10	12	11	11	11	10	9	10	10	10	10	10
C11.	9	5	8	8	8	7	7	8	8	9	9	10	9	9	9	9	10	9	9	9	9	9
C12.	10	11	17	15	15	17	16	15	13	14	12	14	12	10	12	12	13	12	12	13	13	13
C13.	18	8	13	13	11	9	9	9	10	11	11	11	10	11	13	13	12	13	13	12	12	12
C14.	11	6	9	7	8	10	10	11	12	13	13	13	13	13	15	14	16	15	14	14	14	14
C15.	8	8	11	13	13	13	14	14	14	15	14	15	14	12	14	14	14	14	14	15	15	15
C16.	15	9	10	10	12	15	14	16	15	17	16	17	16	15	17	16	17	17	16	17	17	17
C17.	16	11	12	14	14	16	12	15	14	16	15	16	15	14	16	15	15	16	15	16	16	16
C18.	13	7	13	11	10	14	15	17	16	18	17	18	17	16	18	17	18	18	17	18	18	18
C19.	12	10	15	15	16	19	18	19	18	20	18	19	18	17	19	18	19	19	18	19	19	19
C20.	14	10	14	16	17	18	17	18	17	19	18	20	19	18	20	19	20	20	19	20	20	21
C21.	21	14	18	17	18	20	19	20	19	21	19	21	20	19	22	21	22	22	21	21	21	20
C22.	22	15	19	18	19	21	20	21	20	22	20	22	21	20	21	20	21	21	20	22	22	22

Tabella 6.3: Tabella relativa al Mondiale F1 2023, in cui per ogni gara (G*) è riportata la posizione del concorrente (C*) in classifica generale, al termine della gara stessa. I valori sono ricavati dall'applicazione dell'algoritmo di Schultze al termine di ogni gara, considerando anche le precedenti. Ogni gara rappresenta un voto, composto dagli ordini di arrivo.

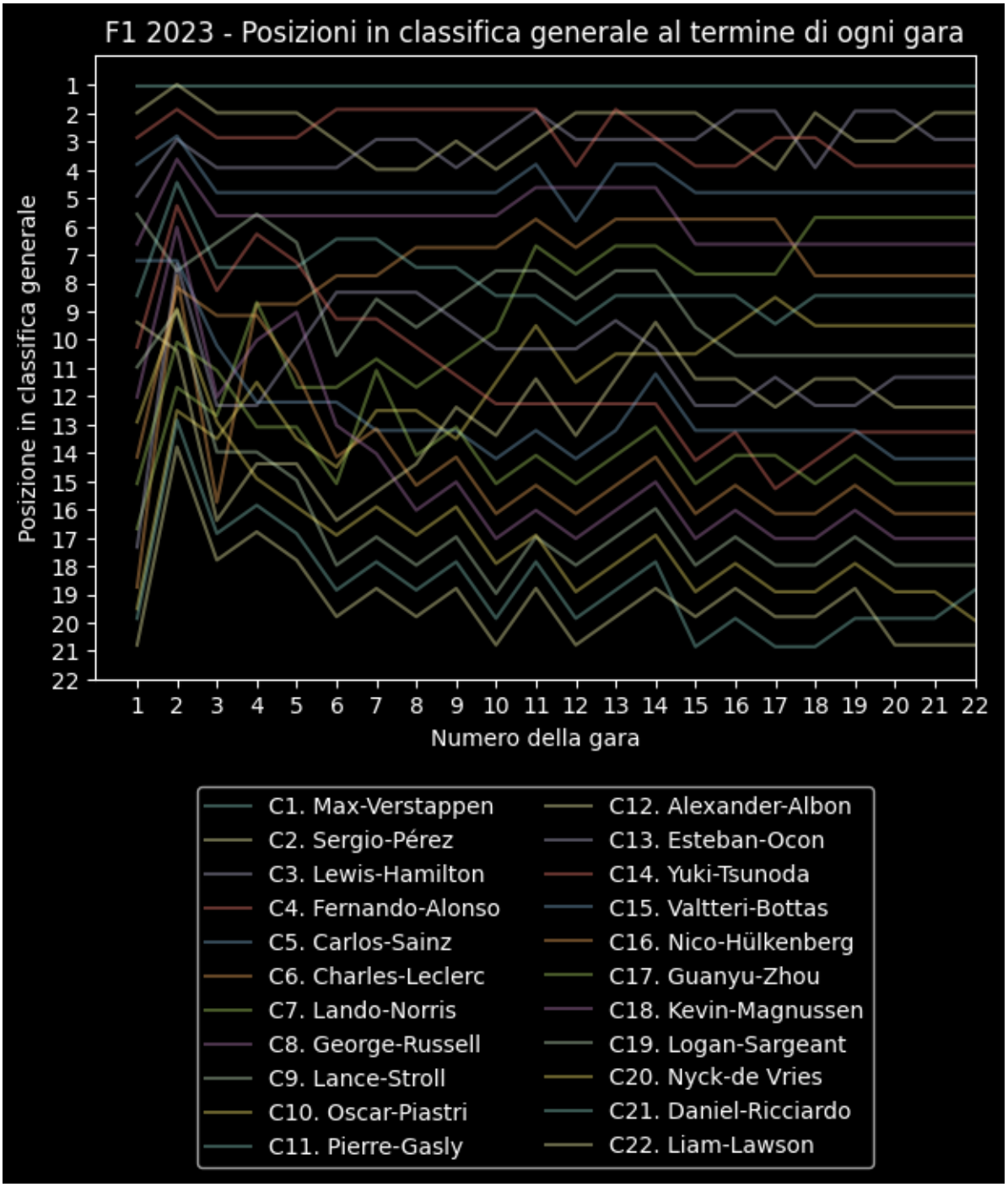

Figura 6.3: Rappresentazione grafica della tabella 6.3

Capitolo 7

Conclusioni