Politecnico di Torino

Corso di Laurea Triennale in Ingegneria Gestionale
Classe L-8: Ingegneria dell'Informazione
A.A. 2025/2026

Metodologie di simulazione di competizioni sportive: un caso studio applicato alla scelta dei giocatori

Relatore
Prof. Giuseppe Bruno Averta

Candidato
Alessio Ussia

Indice

Elenco delle figure

Capitolo 1

Proposta di progetto

1.1 Studente Proponente

282376 Alessio Ussia

1.2 Titolo della proposta

Metodologie di simulazione di competizioni sportive: un caso studio applicato alla scelta dei giocatori

1.3 Descrizione del problema proposto

Il progetto nasce dalla volontà di superare i limiti dei simulatori calcistici basati su logiche puramente casuali, proponendo un modello data-driven che rifletta i valori reali della Serie A. L'obiettivo del progetto è simulare l'intero campionato di Serie A, monitorando l'evoluzione delle classifiche e le statistiche individuali (marcatori/assistman) giornata per giornata. La sfida centrale affrontata riguarda la modellazione della valutazione del giocatore: ho sviluppato un algoritmo che collega dinamicamente gli eventi del match (gol, assist, cartellini) al voto finale. Questo sistema di rating non si limita alla prestazione singola, ma integra un coefficiente correttivo basato sull'esito della partita, premiando o penalizzando il giocatore in base al risultato di squadra. Tale meccanismo è fondamentale per identificare la 'Squadra della Settimana' (Top 11), offrendo una sintesi meritocratica delle prestazioni migliori di ogni turno.

1.4 Descrizione della rilevanza gestionale del problema

La rilevanza gestionale del progetto risiede nella proposta di un modello di supporto decisionale per il management sportivo, capace di trasformare i dati di performance in indicatori strategici. Il simulatore offre una base quantitativa e scientifica per valutare il rendimento dei calciatori, superando le analisi puramente soggettive: questo approccio risulta cruciale per ottimizzare processi come lo scouting, la valutazione della rosa e la definizione dei budget ingaggi. Fornendo metriche oggettive e scenari predittivi, il modello permette ai manager di ridurre l'incertezza decisionale e massimizzare l'efficienza delle risorse tecniche ed economiche.

1.5 Descrizione dei data-set per la valutazione

I dati utilizzati per la simulazione provengono dall'integrazione di due fonti distinte, elaborate per costruire un unico database relazionale. Nello specifico, sono stati utilizzati:

- Il dataset *Serie A Matches 2024-2025*, reperito su Kaggle[1], contenente il calendario e i risultati delle partite;
- Le statistiche individuali dei calciatori, estratte dal portale *FBref*[2], fondamentali per calcolare i pesi probabilistici di ogni giocatore.

Il database finale, implementato su MariaDB, è strutturato secondo lo schema mostrato in Figura 1.1. Il modello è composto da 5 entità principali:

1. **teams**: contiene l'anagrafica delle 20 squadre di Serie A;
2. **players**: memorizza i dati dei calciatori, inclusi ruolo e pesi statistici (weight_goal, weight_assist, ecc.) calcolati sulla base delle performance reali;
3. **matches**: registra il calendario, i risultati (simulati e reali) e lo stato della partita (giocata/da giocare);
4. **events**: traccia i singoli eventi di ogni match (gol, assist, cartellini) collegando i giocatori alle partite;

[1] `https://www.kaggle.com/datasets/marcelbiezunski/serie-a-matches-dataset-2020-2025`

[2] `https://fbref.com/en/comps/11/2024-2025/stats/2024-2025-Serie-A-Stats`

5. **performances**: storicizza i voti assegnati ai giocatori al termine di ogni simulazione, fondamentali per la generazione delle classifiche individuali.

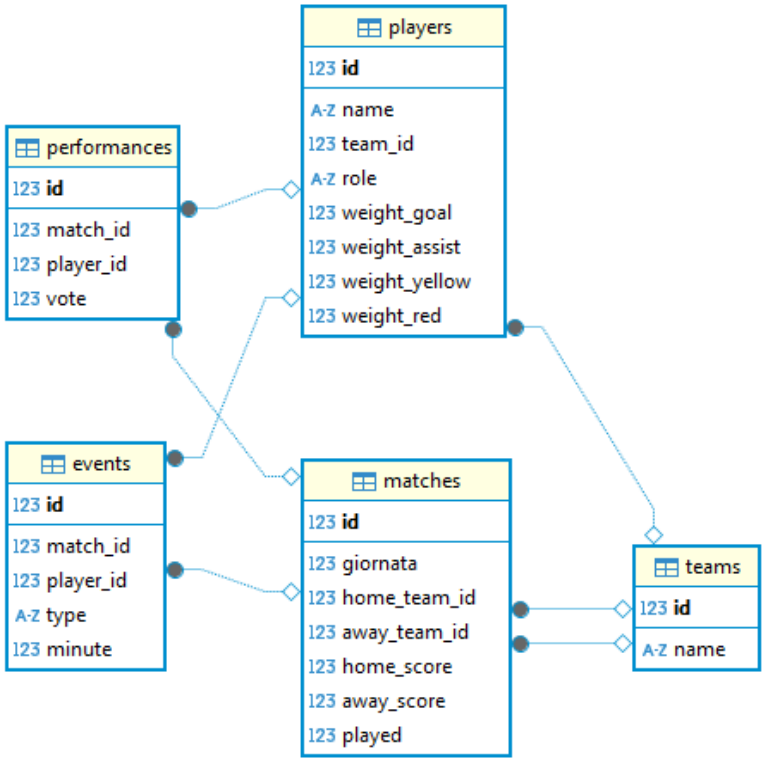

Figura 1.1: Schema ER del database utilizzato per la simulazione.

1.6 Descrizione preliminare degli algoritmi coinvolti

Il cuore del sistema è un motore di simulazione stocastico che abbandona la generazione puramente casuale in favore di un approccio *data-driven*. L'architettura logica si basa su tre algoritmi sequenziali: il calcolo del risultato esatto tramite distribuzione statistica, l'assegnazione ponderata degli eventi ai singoli giocatori e l'algoritmo di selezione della "Squadra della Settimana" (TOTW).

1.6.1 Simulazione del Risultato: Il Modello di Poisson

Per determinare il numero di gol segnati da ciascuna squadra in un match, è stato utilizzato un modello basato sulla **Distribuzione di Poisson**, standard di riferimento per la modellazione di eventi rari in un intervallo di tempo definito.

La probabilità che una squadra segni un numero k di gol è data dalla formula:

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{1.1}$$

Dove il parametro λ (lambda) rappresenta il numero atteso di gol e viene calcolato dinamicamente per ogni partita attraverso il confronto della *Team Strength* (Forza Squadra):

$$\lambda_{home} = MediaGolCasa \times \left(\frac{\sum_{i=1}^{11} w_{off}(Home_i)}{\sum_{j=1}^{11} w_{def}(Away_j)} \right)^{\gamma} \tag{1.2}$$

Dove:

- w_{off} e w_{def} sono i pesi offensivi e difensivi dei migliori 11 giocatori titolari;
- γ è un fattore di smorzamento (damping factor) per evitare punteggi tennistici in caso di forte divario tecnico.

1.6.2 Algoritmo di Assegnazione degli Eventi (Weighted Choice)

Una volta stabilito il risultato (es. 2-1), il sistema deve attribuire i gol, gli assist e i cartellini ai singoli calciatori. Per evitare che un difensore segni con la stessa probabilità di un attaccante, è stato implementato un algoritmo di **Weighted Random Choice** (Scelta Casuale Ponderata).

Ogni giocatore p possiede una probabilità di essere selezionato come marcatore proporzionale al suo peso storico W_p:

$$P(Event = p) = \frac{W_p}{\sum_{i \in Team} W_i} \tag{1.3}$$

Questo garantisce che giocatori con uno storico di gol elevato (es. Lautaro Martinez) abbiano una frequenza di marcatura nella simulazione coerente con la realtà, pur mantenendo un margine di imprevedibilità stocastica.

1.6.3 Algoritmo di Valutazione e Selezione Top XI

Al termine di ogni giornata, il sistema calcola il *Match Rating* per ogni giocatore partendo da un voto base (6.0) e applicando bonus/malus deterministici:

$$Voto_{finale} = 6.0 + (1 \cdot Gol) + (0.5 \cdot Assist) - (0.5 \cdot Giallo) - (2 \cdot Rosso) + Bonus_{risultato} \tag{1.4}$$

Per la selezione della Top 11, l'algoritmo esegue i seguenti passaggi:

1. **Raggruppamento:** I giocatori vengono divisi in liste basate sul ruolo (GK, DF, MF, FW).
2. **Ordinamento:** Ogni lista viene ordinata in modo decrescente in base al $Voto_{finale}$. In caso di parità, viene utilizzato un "tie-breaker" basato sul peso offensivo del giocatore.
3. **Selezione Modulare:** Vengono selezionati i migliori k giocatori per riempire un modulo 4-3-3 (1 GK, 4 DF, 3 MF, 3 FW).

1.7 Descrizione preliminare delle funzionalità previste per l'applicazione software

L'applicazione è stata progettata per offrire un'esperienza utente intuitiva attraverso un'interfaccia grafica (GUI) sviluppata con la libreria *Tkinter*. Le funzionalità principali del sistema possono essere suddivise in tre macroaree logiche: gestione della simulazione, consultazione dei dati e analisi delle performance.

1.7.1 Gestione della Simulazione e Calendario

Il nucleo operativo del software permette all'utente di avanzare nel campionato giornata per giornata. Le funzionalità chiave includono:

- **Navigazione Cronologica:** L'utente può scorrere tra le 38 giornate di campionato tramite controlli di navigazione (*Next/Previous Day*).
- **Simulazione Stocastica:** Attraverso il comando "Simula Giornata", il sistema calcola i risultati di tutti i match non ancora disputati, utilizzando l'algoritmo basato sulla distribuzione di Poisson descritto nella sezione precedente.

- **Controllo di Integrità:** Il sistema impedisce la simulazione di una giornata futura se quelle precedenti non sono state completate, garantendo la coerenza temporale della classifica.

- **Persistenza dei Risultati:** Una volta simulata, una giornata viene marcata come "Giocata" nel database e i risultati diventano immutabili per preservare lo storico della stagione.

1.7.2 Visualizzazione Classifiche e Statistiche

L'aggiornamento dei dati avviene in tempo reale al termine di ogni processo di simulazione. È possibile visualizzare:

- **Classifica Dinamica:** Una tabella completa che mostra punti, partite giocate, vittorie/pareggi/sconfitte, gol fatti e subiti. L'ordinamento è automatico e gestisce le priorità (Punti $\rightarrow$ Differenza Reti $\rightarrow$ Gol Fatti $\rightarrow$ Ordine Alfabetico).

- **Dettaglio Match:** Tramite interazione, cliccando due volte, sulla singola partita, si apre una finestra di dettaglio che mostra il tabellino cronologico degli eventi (marcatori, assist, cartellini gialli e rossi).

- **Statistiche Individuali:** Due graduatorie sempre aggiornate mostrano la classifica dei Top Marcatori e Top Assist-man del campionato.

1.7.3 Analisi delle Prestazioni

Una funzionalità distintiva del progetto è l'analisi qualitativa delle prestazioni dei singoli calciatori:

- **Top XI Settimanale:** Un algoritmo seleziona automaticamente i migliori 11 giocatori della giornata appena conclusa, disponendoli in un modulo 4-3-3 basato sui voti ottenuti.

- **Team of the Year (TOTY):** Al termine della stagione (o su richiesta durante il campionato), il sistema aggrega le medie voto ponderate per individuare la formazione ideale dell'anno, filtrando i giocatori che non hanno raggiunto un numero minimo di presenze significative.

Capitolo 2

Descrizione del problema affrontato

2.1 Contesto operativo

Il progetto si inserisce nel contesto della simulazione statistica applicata al campionato di Serie A. Nello specifico, il problema affrontato riguarda la costruzione di un sistema software in grado di replicare l'andamento di una stagione calcistica partendo da dati storici reali, con l'obiettivo di valutare oggettivamente le performance di squadre e singoli giocatori.
Il contesto operativo è definito dalla necessità di superare i limiti delle valutazioni soggettive (tipiche delle pagelle giornalistiche o dei videogiochi tradizionali) attraverso un modello matematico rigoroso che operi su tre livelli interconnessi:*Simulazione di Squadra (Team Level)*, *Valutazione Individuale (Player Level)*, *Selezione Meritocratica (Top XI)*.
L'obiettivo finale dell'applicazione è quella di trasformare dataset statici (rose e storici) in una narrazione dinamica della Serie A, dove ogni giornata simulata aggiorna classifiche e statistiche, offrendo all'utente uno strumento di analisi predittiva e di monitoraggio delle performance.

2.2 Rilevanza del problema scelto

La rilevanza del problema affrontato risiede nella crescente necessità, all'interno dell'industria calcistica moderna, di strumenti oggettivi per la valutazione delle performance. Il calcio è un settore caratterizzato da alta incertezza e ingenti investimenti economici; tuttavia, molte decisioni strategiche (rinnovi contrattuali, acquisti di mercato, scelte di formazione) sono ancora spesso influenzate da bias cognitivi o valutazioni soggettive basate sulla "percezio-

ne" piuttosto che sui fatti.

La soluzione sviluppata trasforma il dato grezzo in informazione strategica, colmando il divario tra la semplice raccolta statistica e la simulazione predittiva necessaria per una gestione efficiente del club.

2.3 Criticità e sviluppi futuri

L'analisi del sistema realizzato ha permesso di evidenziare sia i punti di forza dell'architettura proposta, sia le limitazioni intrinseche dovute alle scelte progettuali e alla complessità del dominio calcistico.

2.3.1 Criticità del Modello Attuale

Nonostante l'efficacia della simulazione stocastica, il modello presenta alcune semplificazioni che ne limitano il realismo assoluto:

- **Staticità dei Pesi Statistici:** Attualmente, la *Team Strength* e i pesi dei singoli giocatori sono calcolati su dataset storici aggregati (stagione 2024/2025). Il sistema non prevede un aggiornamento dinamico dello "stato di forma": un giocatore che segna per 5 partite di fila nella simulazione non vede aumentare la sua probabilità di segnare nella sesta.
- **Assenza della Componente Tattica e Sostituzioni:** Il simulatore seleziona i "migliori 11" per calcolare la forza della squadra, ma ignora le sostituzioni a partita in corso, che nel calcio reale sono decisive (es. inserire un difensore per proteggere il risultato). Inoltre, non viene modellata l'interazione tra moduli tattici diversi (es. il vantaggio di un 3-5-2 contro un 4-3-3).
- **Limiti della Distribuzione di Poisson:** La distribuzione di Poisson assume che gli eventi (i gol) siano indipendenti tra loro. Nella realtà, un gol segnato cambia la psicologia del match, portando la squadra in svantaggio ad attaccare di più e scoprirsi, dinamica che il modello attuale approssima solo parzialmente tramite il "fattore pressione".

2.3.2 Potenzialità e Sviluppi Futuri

La struttura modulare del software, basata sul pattern MVC, si presta a significative evoluzioni che potrebbero trasformare il progetto in un prodotto gestionale completo:

- **Integrazione di Algoritmi di Machine Learning:** Sostituire il modello di Poisson con reti neurali o algoritmi di *Random Forest* addestrati su dati storici più ampi (es. ultime 10 stagioni). Questo permetterebbe di catturare pattern non lineari, come la difficoltà di giocare in trasferta in stadi specifici o la tendenza al pareggio di certe squadre.
- **Espansione del Modello Economico-Gestionale:** Per aumentare la rilevanza manageriale, si potrebbe integrare un modulo economico che associ ai voti dei giocatori un valore di mercato dinamico (algoritmo di rivalutazione). Questo trasformerebbe il simulatore in un vero tool di *Scouting*, utile per simulare l'impatto di un acquisto sul bilancio e sulla classifica futura.

Capitolo 3

Descrizione del data-set utilizzato per l'analisi

Il database, ospitato su MariaDB, è strutturato secondo uno schema a 5 entità, progettato per supportare query efficienti durante la simulazione in tempo reale, come già visto nel sottocapitolo 1.5.
Le tabelle sono così definite:

1. **teams (20 record):** Rappresenta l'anagrafica delle società. Ogni squadra funge da aggregatore per i giocatori e da chiave esterna per le partite.
2. **players (∼550 record):** Contiene gli attributi statici (Nome, Ruolo, Squadra) e, soprattutto, i **pesi probabilistici** calcolati in fase di pre-elaborazione:
    - `weight_goal`: Propensione al gol (basata su xG storici).
    - `weight_assist`: Propensione all'assist.
    - `weight_cards`: Aggressività e rischio ammonizione.

	id	name	team_id	role	weight_goal	weight_assist	weight_yellow	weight_red
	Unique key: PRIMARY KEY id	bankwah	19	DF,MF	0	0	1	0
2	2	Saud Abdulhamid	17	DF	0	1	0	0
3	3	Oliver Abildgaard	4	MF	0	0	0	0
4	4	Tammy Abraham	17	FW	0	0	0	0
5	5	Tammy Abraham	13	FW	3	4	1	0
6	6	Federico Accornero	7	MF	0	0	0	0
7	7	Francesco Acerbi	9	DF	0	1	0	0
8	8	Che Adams	18	FW	9	3	2	0
9	9	Yacine Adli	6	MF	4	6	3	1
10	10	Vasilije Adžić	10	MF,FW	0	0	0	0
11	11	Michel Aebischer	2	MF	0	0	2	0
12	12	Honest Ahanor	7	DF,MF	0	0	0	0
13	13	Junior Ajayi	8	FW	0	0	0	0
14	14	Jean-Daniel Akpa-Akpro	14	MF	0	0	4	0
15	15	David Akpan Ankeye	7	FW,MF	0	0	0	0
16	16	Faride Alidou	8	FW,MF	0	0	0	0
17	17	Dele Alli	4	MF	0	0	0	1
18	18	Pontus Almqvist	16	FW,MF	1	1	5	0
19	19	Giorgio Altare	20	DF	0	0	2	0
20	20	Sofyan Amrabat	6	MF	0	0	0	0
21	21	Angeliño	17	DF	2	1	1	0

Figura 3.1: Esempio calciatori e statistiche.

3. **matches (380 record):** Rappresenta il calendario completo. Include attributi di stato (`played`: booleano) che permettono al simulatore di distinguere tra storico (risultati fissati) e futuro (da simulare).

4. **events (Dinamica):** Tabella transazionale che registra ogni singolo evento generato dal motore di Poisson (Gol, Assist, Giallo, Rosso), collegando un *match_ id* a un *player_ id* con il relativo minuto di gioco.

	id	match_id	player_id	type	minute
1	7.169	153	115	YELLOW	35
2	7.170	153	64	YELLOW	43
3	7.171	153	186	YELLOW	76
4	7.172	153	333	YELLOW	64
5	7.173	153	156	YELLOW	41
6	7.174	153	289	YELLOW	23
7	7.175	267	182	GOAL	65
8	7.176	267	495	ASSIST	65
9	7.177	267	162	YELLOW	66
10	7.178	267	466	YELLOW	80
11	7.179	267	335	YELLOW	85
12	7.180	267	257	YELLOW	16
13	7.181	267	557	YELLOW	54
14	7.182	324	196	GOAL	83
15	7.183	324	205	GOAL	41
16	7.184	324	88	ASSIST	41
17	7.185	324	248	YELLOW	16
18	7.186	324	463	YELLOW	32
19	7.187	324	463	RED	34
20	7.188	229	367	GOAL	90
21	7.189	229	45	ASSIST	90
22	7.190	229	601	YELLOW	69
23	7.191	229	34	YELLOW	36
24	7.192	229	161	YELLOW	12

Figura 3.2: Serie di eventi.

5. **performances (Dinamica):** Memorizza il voto finale calcolato per ogni giocatore al termine di ogni partita. Questa tabella è la base dati per le query analitiche che generano la *Top XI* settimanale e la *Team of the Year*.

Nome colonna	#	Data Type
123 id	1	int(11)
123 match_id	2	int(11)
123 player_id	3	int(11)
123 vote	4	float

Figura 3.3: Struttutura tabella performances.

Capitolo 4

Strutture dati e algoritmi utilizzati

4.1 Struttura del progetto

L'architettura software adottata segue il pattern architetturale **MVC (Model-View-Controller)**, standard per lo sviluppo di interfacce utente, che garantisce la separazione delle responsabilità. Il progetto è organizzato in tre moduli principali:

1. **Database (DAO):** Il modulo `database/DAO.py` gestisce la persistenza dei dati. Utilizza la libreria `mysql-connector` per eseguire query SQL su MariaDB, astraendo la complessità del database dal resto dell'applicazione.

2. **Model (Business Logic):** Il file `model/model.py` incapsula la logica matematica della simulazione. Qui risiedono gli algoritmi di calcolo della forza squadra, la distribuzione di Poisson e la gestione degli eventi. Il modello non ha conoscenza dell'interfaccia grafica.

3. **View (User Interface):** Il modulo `UI/view.py`, sviluppato con *Tkinter*, gestisce l'interazione con l'utente, la visualizzazione delle classifiche e la validazione degli input prima di invocare il modello.

Questa struttura modulare permette di modificare l'algoritmo di simulazione (Model) senza dover riscrivere il codice dell'interfaccia grafica, o viceversa.

4.2 Logica applicativa

La logica applicativa orchestra il flusso di esecuzione, garantendo la coerenza temporale della simulazione. Un aspetto critico affrontato è la gestione sequenziale delle giornate di campionato: il sistema deve impedire salti temporali che corromperebbero la classifica.

Di seguito è riportato il frammento di codice (tratto dalla classe `SerieAApp` nel modulo View) che implementa il "posto di blocco" logico prima di avviare una simulazione. Si noti il controllo sulla giornata precedente tramite il metodo `check_giornata_completata`:

Logica di controllo sequenziale per la simulazione della giornata.

```
def simula_giornata_corrente(self):
    g = self.current_giornata

    # 1. Controllo giornata: Se giocata, stop.
    if self.db.check_giornata_completata(g):
        messagebox.showinfo("Info", f"Giornata {g} gia' giocata.")
        return

    # 2. Controllo Coerenza Temporale
    if g > 1:
        prev_day = g - 1
        # Interroga il DB per sapere se le giornate precedenti sono
            state giocate
        if not self.db.check_giornata_completata(prev_day):
            messagebox.showwarning(
                "Errore Sequenza",
                f"Impossibile simulare la G{g} prima della
                    G{prev_day}."
            )
            return

    # 3. Avvio Simulazione (se i controlli passano)
    self.sim.simulate_day(g)
    self.update_ui()
```

Questo approccio difensivo previene stati inconsistenti del database, garantendo che l'utente segua il flusso cronologico corretto della stagione.

4.3 Algoritmi

Il cuore computazionale del progetto risiede nel modulo `model.py`. Qui vengono trasformati i dati statici in risultati dinamici attraverso tre passaggi algoritmici distinti.

4.3.1 Calcolo della Team Strength

Per determinare le probabilità di vittoria, l'algoritmo calcola la forza offensiva di una squadra sommando i pesi ponderati (`weight_goal`) dei migliori 11 giocatori disponibili. Questo metodo, a differenza della media semplice, premia le squadre con "top player" in grado di decidere la partita.

Algoritmo per il calcolo della forza offensiva della squadra.

```
def _calculate_team_strength(self, players):
    # Ordina i giocatori per peso offensivo decrescente
    sorted_players = sorted(
        players,
        key=lambda p: p['weight_goal'],
        reverse=True
    )

    # Seleziona solo i migliori (Top 11)
    top_11 = sorted_players[:11]

    # Somma i pesi per ottenere la forza totale
    strength = sum(p['weight_goal'] for p in top_11)

    # Normalizzazione minima per evitare divisioni per zero
    return max(strength, 0.5)
```

4.3.2 Generazione Stocastica (Poisson)

Una volta ottenute le forze delle due squadre, il sistema calcola il parametro λ (numero atteso di gol) e utilizza la Distribuzione di Poisson per generare il risultato esatto. L'uso di Poisson è preferito rispetto a una distribuzione uniforme perché modella correttamente la rarità dell'evento-gol.

Implementazione della distribuzione di Poisson per la generazione dei gol.

```
def _poisson(self, lam):
    """
```

```
    Genera un numero casuale k secondo la distribuzione di Poisson:
    P(k) = (lambda^k * e^-lambda) / k!
    """
    L = math.exp(-lam)
    k = 0
    p = 1.0
    while p > L:
        k += 1
        p *= random.random()
    return k - 1
```

4.3.3 Weighted Random Choice per i Marcatori

Infine, per assegnare i gol ai singoli giocatori, non viene utilizzata una scelta casuale uniforme (che darebbe a un difensore le stesse chance di un attaccante), ma una scelta pesata. Viene costruito un "pool" di candidati dove ogni giocatore appare un numero di volte proporzionale al suo storico realizzativo.

Algoritmo di assegnazione pesata dei marcatori.

```
def _assign_goals(self, match_id, num_goals, team_players, events):
    scorers_pool = []

    # Costruzione del pool ponderato
    for p in team_players:
        # Maggiore alto weight_goal, maggiori sono biglietti che ha
            il giocatore
        weight = int(p['weight_goal'] * 1000)
        scorers_pool.extend([p] * weight)

    for _ in range(num_goals):
        # Estrazione casuale dal pool pesato
        scorer = random.choice(scorers_pool)

        # Registrazione evento e aggiornamento voto
        events.append((match_id, scorer['id'], 'GOAL'))
        self.votes[scorer['id']] += 3.0 # Bonus Gol
```

Capitolo 5

Diagramma delle classi principali

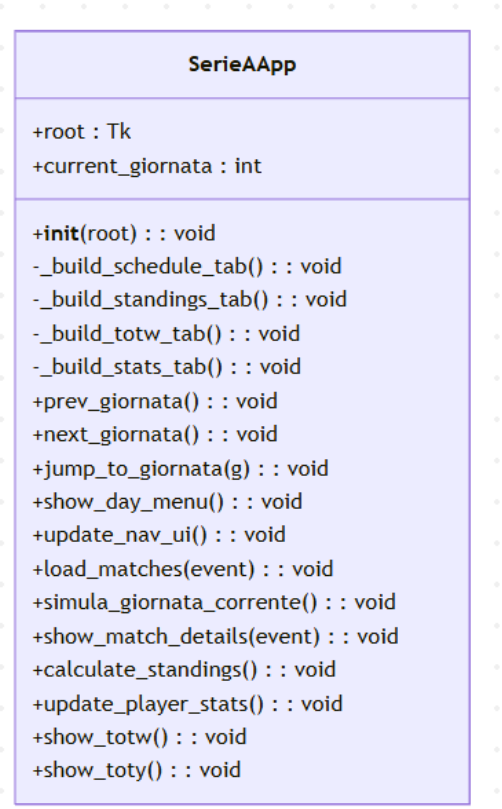

Figura 5.1: Diagramma UML della classe view.

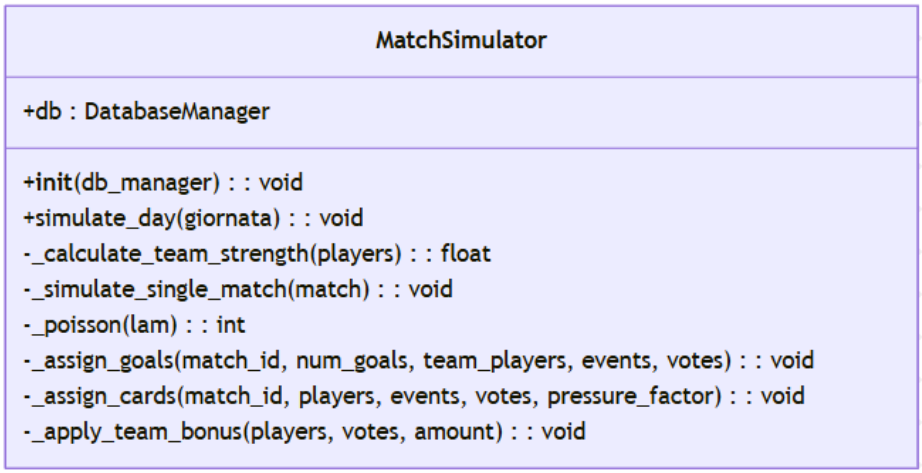

Figura 5.2: Diagramma UML della classe model.

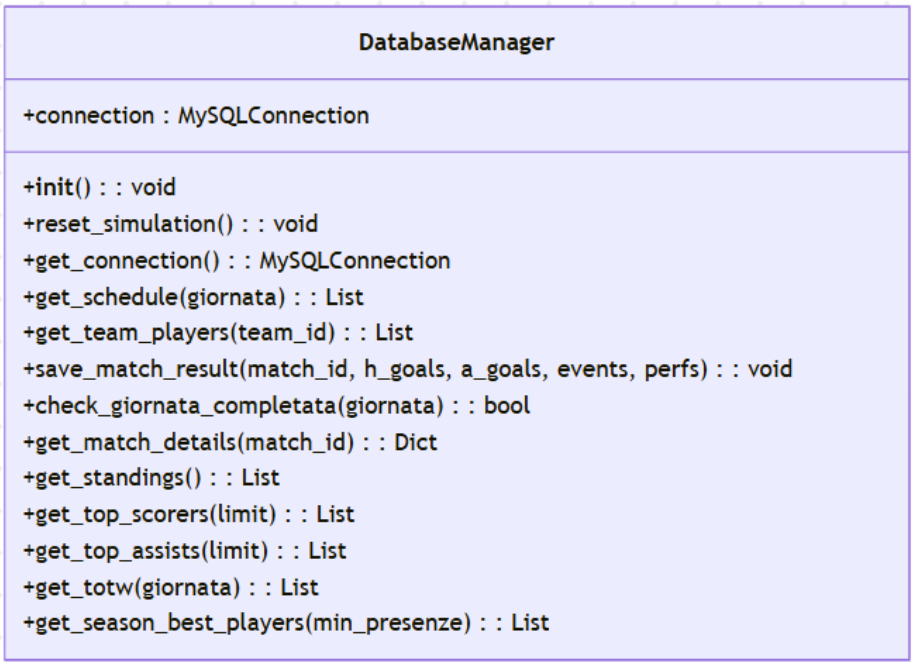

Figura 5.3: Diagramma UML della classe DAO.

Capitolo 6

Interfaccia dell'applicazione

All'avvio, l'applicazione presenta la dashboard principale (Figura 6.1), progettata per offrire all'utente una visione immediata del calendario e dello stato della simulazione.

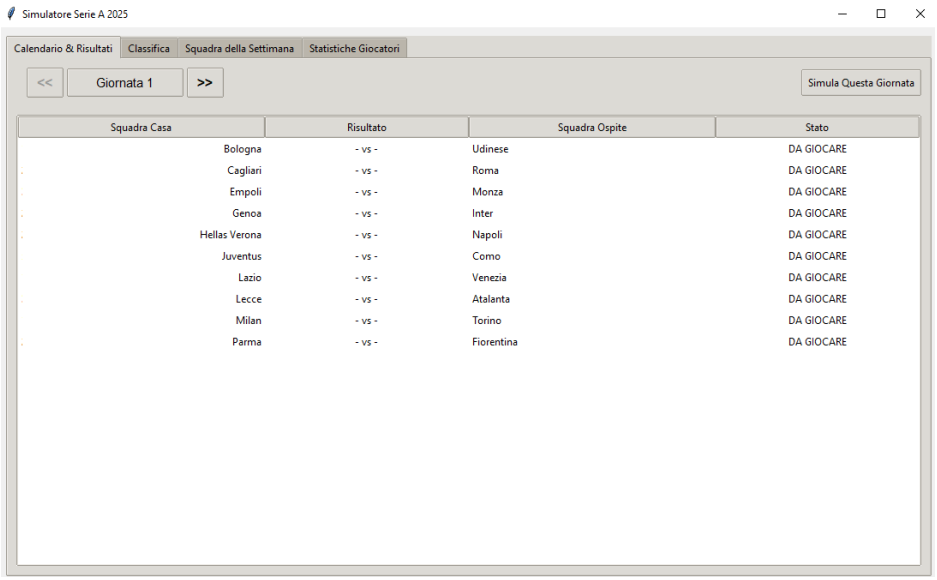

Figura 6.1: Interfaccia iniziale dell'applicazione.

La navigazione temporale è gestita tramite il pannello superiore:

Utilizzando le frecce direzionali poste ai lati dell'etichetta "Giornata", l'utente può scorrere cronologicamente il calendario. In alternativa, cliccando sul box centrale, è possibile selezionare direttamente un turno specifico dal menu a tendina per una navigazione rapida.

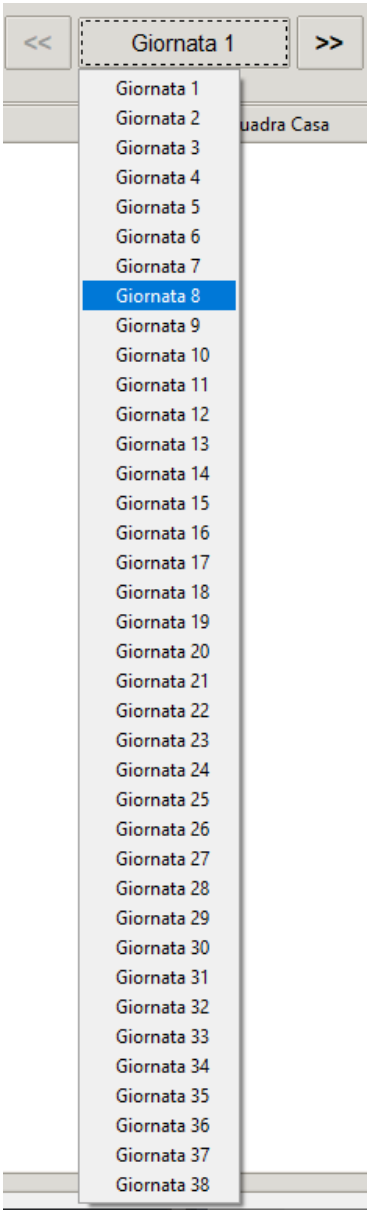

Figura 6.2: Dettaglio del selettore della giornata.

Il cuore operativo del software risiede nella funzionalità di simulazione. Cliccando il pulsante *Simula Questa Giornata*, il sistema avvia l'algoritmo di calcolo che elabora istantaneamente i risultati di tutte le partite in programma, aggiornando il database e l'interfaccia grafica:

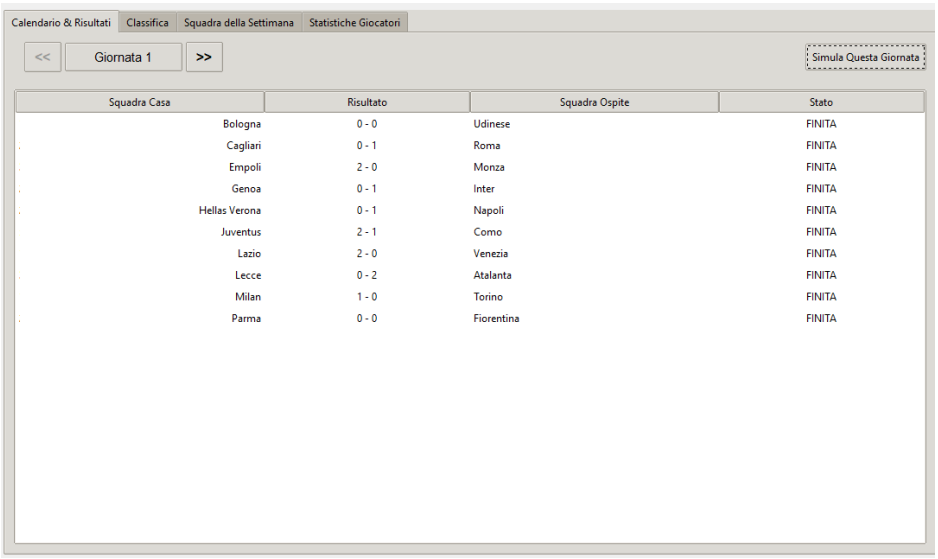

Figura 6.3: Risultati simulati per la 7ª Giornata.

Al termine della simulazione, i dati vengono aggregati nelle tabelle statistiche. La vista *Classifica* (Figura 6.4) riepiloga la situazione del campionato mostrando, per ogni squadra: punti in classifica, partite giocate, bilancio (vittorie, pareggi, sconfitte) e differenza reti.

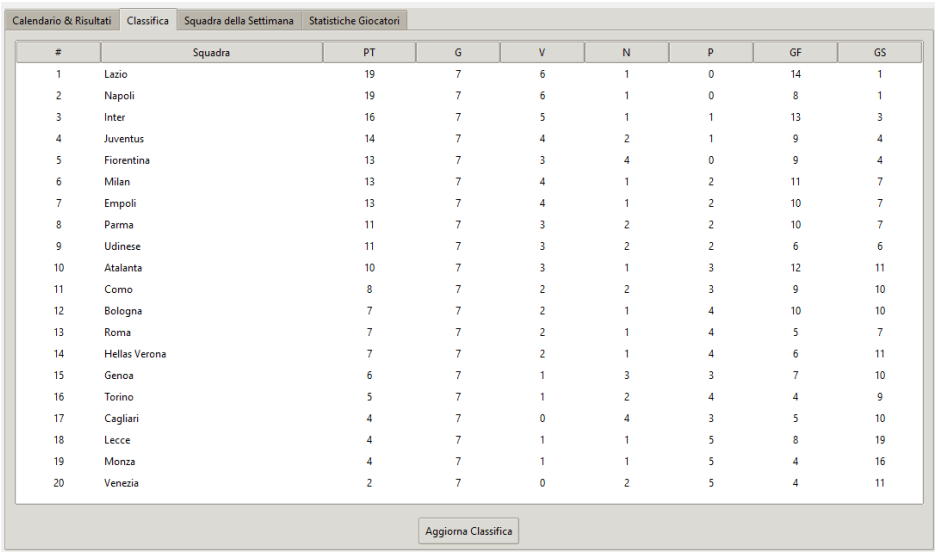

Figura 6.4: Tabella della classifica aggiornata alla 7ª Giornata.

Oltre alle performance di squadra, il sistema monitora le prestazioni individuali. Le schermate seguenti mostrano le graduatorie aggiornate dei migliori

marcatori e degli assist-man, oltre alla selezione automatica della *Squadra della Settimana* (Team of the Week), calcolata sui voti ottenuti dai singoli giocatori.

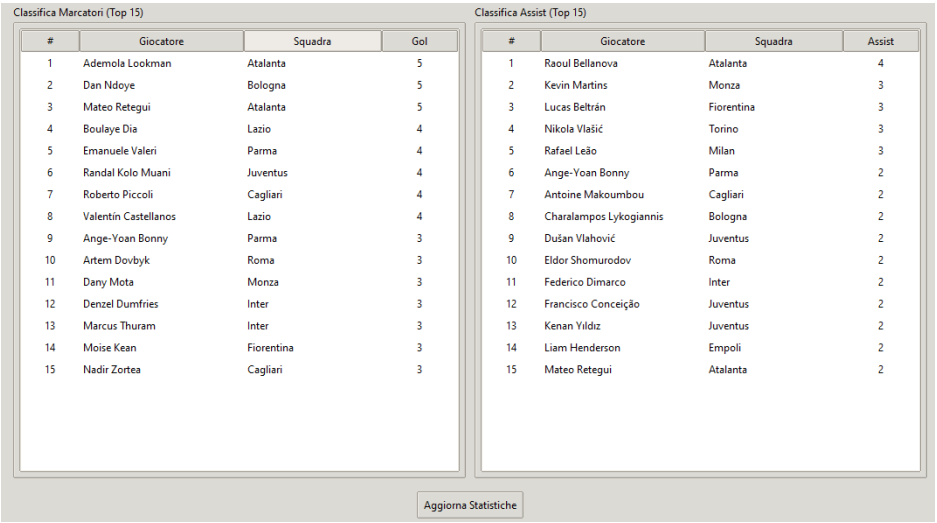

Figura 6.5: Classifiche Top Marcatori e Assist-man.

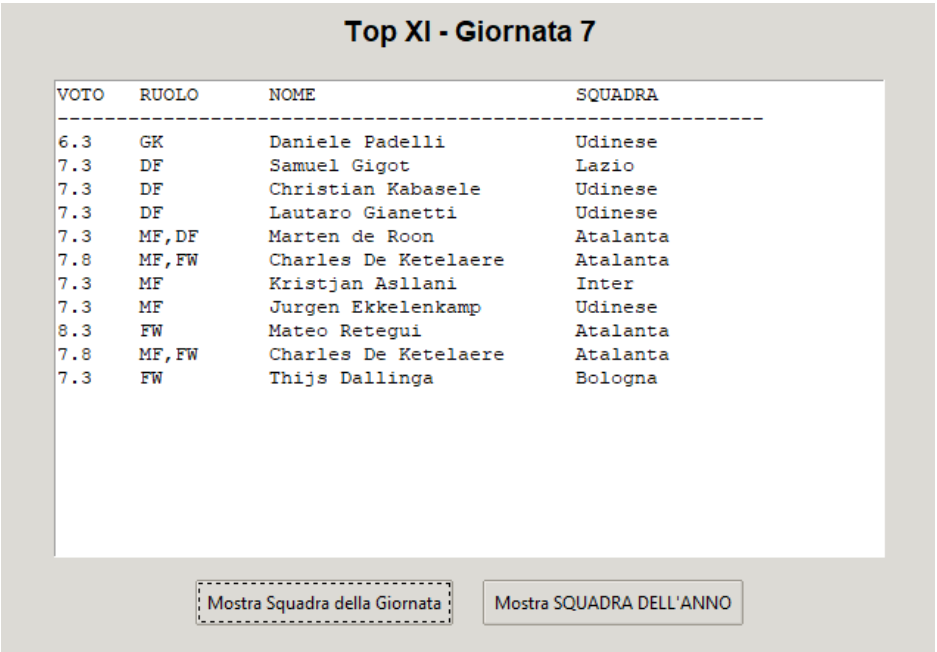

Figura 6.6: Formazione Top XI (TOTW) della 7ª Giornata.

DA CAMBIARE

Un video dimostrativo sull'uso dell'applicazione è disponibile al seguente link: https://www.youtube.com/watch?v=dSJLT8fDhbg.

Capitolo 7

Valutazioni sui risultati ottenuti e conclusioni

Il presente lavoro di tesi ha portato alla realizzazione di un sistema completo per la simulazione e l'analisi statistica del campionato di Serie A. In questo capitolo finale vengono discussi i risultati emersi dalle simulazioni, valutando l'accuratezza del modello matematico rispetto alla realtà e l'efficacia dell'architettura software implementata.

7.1 Validazione del Modello Matematico

L'obiettivo primario del progetto era creare un motore di simulazione che non fosse puramente casuale, ma *data-driven*. L'analisi dei dati generati dopo diverse iterazioni di simulazione (stagioni complete simulate) ha evidenziato i seguenti punti:

- **Coerenza delle Gerarchie:** Il calcolo della *Team Strength* basato sui pesi offensivi dei giocatori ha prodotto classifiche finali coerenti con i valori reali delle squadre. Le formazioni di vertice (es. Inter, Napoli) si posizionano costantemente nella parte alta della classifica simulata, mentre le squadre con rose tecnicamente inferiori lottano per la salvezza. Questo conferma che l'algoritmo distingue correttamente i divari tecnici.
- **Realismo della Distribuzione dei Gol:** L'utilizzo della Distribuzione di Poisson ha permesso di replicare fedelmente la frequenza dei risultati. I punteggi "tennistici" (es. 6-0 o 7-1) sono eventi rari, mentre risultati come 1-0, 2-1 e 1-1 costituiscono la maggioranza dei casi, rispecchiando le statistiche reali della Serie A.

- **Attendibilità delle Performance Individuali:** L'algoritmo di *Weighted Random Choice* per l'assegnazione dei marcatori ha funzionato come previsto. Nelle simulazioni, i capocannonieri risultano essere effettivamente gli attaccanti con i valori di xG (Expected Goals) più alti nel dataset iniziale, evitando anomalie statistiche come difensori centrali con 15 gol stagionali.

7.2 Valutazione dell'Architettura Software

Dal punto di vista ingegneristico, l'adozione del pattern MVC e la modularizzazione del codice hanno portato benefici tangibili:

- **Reattività dell'Interfaccia:** La separazione tra il thread della GUI (Tkinter) e la logica di calcolo ha garantito un'esperienza utente fluida. La simulazione di una giornata completa, che richiede centinaia di calcoli probabilistici e accessi al database, avviene in frazioni di secondo.
- **Robustezza dei Dati:** La struttura del database relazionale in Terza Forma Normale ha prevenuto ridondanze e incongruenze. I vincoli di integrità referenziale hanno impedito errori comuni, come l'assegnazione di gol a giocatori non convocati o la simulazione di giornate fuori sequenza.

7.3 Confronto con i Dati Reali

Sebbene il modello non possa prevedere l'imprevedibilità intrinseca del calcio (infortuni dell'ultimo minuto, errori arbitrali, fattori psicologici), la tendenza statistica è rispettata.
Le deviazioni osservate (es. una "big" che performa peggio del previsto in una singola run) sono attribuibili alla varianza stocastica del modello, elemento desiderato per garantire che ogni simulazione sia unica e non deterministicamente identica alla realtà.

7.4 Conclusioni

In conclusione, il progetto ha dimostrato come l'applicazione di modelli statistici consolidati (Poisson) a dataset sportivi granulari possa generare simulazioni verosimili e utili per l'analisi delle performance.

L'applicazione sviluppata non è solo un esercizio di programmazione, ma un prototipo funzionante di *Decision Support System* sportivo. Essa permette di rispondere a domande di tipo "What-If", isolando il fattore fortuna dalla qualità della rosa.
I risultati ottenuti confermano che, anche con un modello semplificato rispetto ai complessi algoritmi delle grandi società di betting, è possibile catturare l'essenza statistica del gioco del calcio, fornendo uno strumento che unisce l'intrattenimento videoludico al rigore dell'analisi dati.

Licenza d'uso