**POLITECNICO DI TORINO**

Laurea di 1° Livello in Ingegneria Gestionale

Classe L-8 Ingegneria dell’Informazione

A.A. 2019-2020



**Simulatore Serie A post Covid-19**

Relatore

Prof. Fulvio Corno

Candidato

Omar Panebianco

INDICE

|  |  |
| --- | --- |
| 1. Proposta di progetto | 4 |
| 1. Descrizione del problema affrontato | 6 |
| 1. Data-set utilizzati | 7 |
| 1. Strutture dati e algoritmi | 11 |
| 1. Diagramma delle classi principali | 17 |
| 1. Videate dell’applicazione e video dimostrativo | 18 |
| 1. Valutazione dei risultati e conclusioni | 20 |

1. PROPOSTA DI PROGETTO

**1.1 Studente proponente**

s234164 Panebianco Omar

**1.2 Titolo della proposta**

Simulatore Serie A post Covid-19

**1.3 Descrizione del problema proposto**

La pandemia di Covid-19 in Italia ha compromesso il regolare svolgimento del campionato di calcio di Serie A. Quest'ultimo è stato sospeso fino a tempo indeterminato dal Consiglio Federale della FIGC in seguito al DPCM del 9 Marzo 2020. Di recente, il Governo Italiano e la FIGC hanno concordato la ripresa del campionato. Nel caso in cui sia istituito un nuovo lockdown il campionato non verrà terminato e la classifica finale della stagione 2019-20 sarà stilata tramite l'utilizzo di un algoritmo.

L'applicazione ha lo scopo di simulare le 124 partite di Serie A non ancora disputate in seguito alla prima sospensione decretando la classifica finale del campionato.

**1.4 Descrizione della rilevanza gestionale del problema**

Il calcio è un business che vale miliardi di Euro. L'annullamento del campionato comporterebbe per le società la perdita di ingenti somme di denaro. Come sostenuto dagli organizzatori, è necessario decretare una classifica finale - seppur virtuale - per la corretta distribuzione dei premi e per permettere alle società di programmare la stagione successiva.

**1.5 Descrizione dei data-set per la valutazione**

L'applicazione utilizzerà dei data-set costruiti appositamente. Uno di questi conterrà i dati statistici dettagliati relativi ad ogni squadra di Serie A aggiornati all'ultima giornata disputata che costituiranno le variabili fondamentali per l'algoritmo di simulazione. Tali dati verranno reperiti dal portale WhoScored ([https://it.whoscored.com](https://it.whoscored.com/)).

Un altro data-set conterrà invece il calendario delle 124 partite di Serie A non disputate suddivise per giornata. Il calendario verrà reperito dal sito ufficiale della Lega Serie A ([https://www.legaseriea.it](https://www.legaseriea.it/)).

**1.6 Descrizione preliminare degli algoritmi coinvolti**

L'algoritmo principale dell'applicazione è la simulazione. Sulla base dei dati statistici ricavabili dal primo data-set, l'applicazione effettuerà una simulazione per ognuna delle 124 partite non disputate aggiornando la classifica del campionato di giornata in giornata.

In particolare il risultato di ogni partita verrà decretato considerando le statistiche in casa e in trasferta o assolute di entrambe le squadre tra cui la media dei tiri in porta effettuati a partita e la percentuale di realizzazione degli stessi ma anche la media dei tiri in porta subiti a partita e la percentuale di reti incassate per tiro in porta subito.

**1.7 Descrizione preliminare delle funzionalità previste per l’applicazione**

All'avvio, l'applicazione mostrerà la classifica della Serie A aggiornata alla 26a giornata. Per avviare la simulazione all'utente verrà data la scelta di utilizzare il *fattore casa* ed un secondo fattore chiamato *fattore punti*. Inoltre all'utente verrà offerta la possibilità di assistere alla simulazione di ogni giornata e la simulazione del turno successivo avverrà su comando dell'utente. Alternativamente l'utente può sceglie di visualizzare direttamente la classifica finale.

2. DESCRIZIONE DEL PROBLEMA AFFRONTATO

L’applicazione si propone, mediante l’impiego di algoritmi di simulazione, di prevedere il risultato delle partite del campionato di calcio di Serie A non ancora disputate in seguito alla sospensione del mese di Marzo decretando così la classifica finale del campionato.

Il sistema calcio può essere paragonato sotto molti aspetti ad una vera e propria azienda. Le cifre che ogni anno fluiscono nelle casse delle squadre di Serie A superano complessivamente il miliardo di Euro. Alcune di queste sono addirittura quotate in borsa.

L'annullamento di un campionato comporterebbe quindi per le società coinvolte, per i detentori dei diritti televisivi e per tutte le attività e le persone all’interno del sistema, perdite economiche non indifferenti.

Per la stagione 2019-2020, la necessità di avere a disposizione la classifica finale per la corretta distribuzione dei premi e per permettere alle società di programmare la stagione successiva ha spinto la Lega Serie A a prendere in considerazione l’utilizzo di un algoritmo che potesse simulare le partite non disputate a causa della pandemia di Covid-19 qualora non fosse possibile terminare il campionato sul campo.

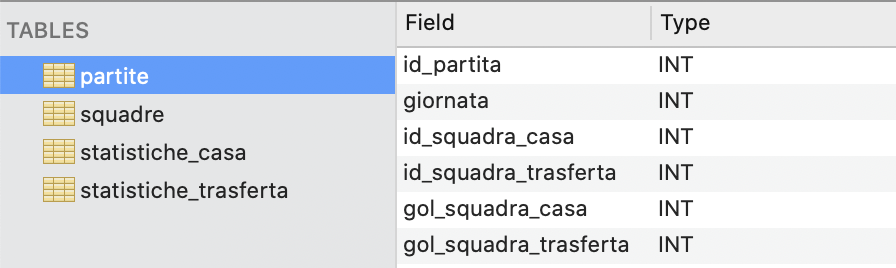
La simulazione degli ultimi turni di campionato, infatti, renderebbe possibile l’assegnazione dello scudetto, determinerebbe le squadre qualificate in Champions League e le squadre qualificate in Europa League per la stagione successiva e individuerebbe le squadre costrette alla retrocessione in Serie B.

3. DATA-SET UTILIZZATI

L’applicazione, per funzionare in modo appropriato, necessita del supporto di data-set specifici difficilmente reperibili in un formato compatibile con il linguaggio di programmazione utilizzato. Si è resa dunque indispensabile la costruzione apposita di ben quattro data-set.

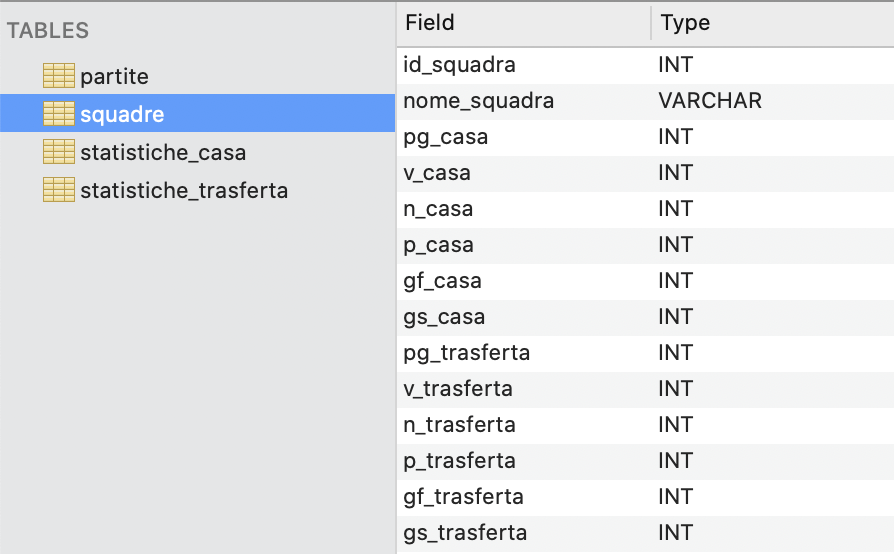
Il primo data-set utilizzato contiene il calendario delle 124 partite di Serie A non ancora disputate suddivise per giornata. Il secondo contiene invece le principali informazioni sulle squadre del campionato.

Il terzo ed il quarto data-set contengono, infine, i dati statistici dettagliati rispettivamente in casa ed in trasferta relativi ad ogni squadra di Serie A aggiornati all’ultimo turno di campionato disputato prima della sospensione.



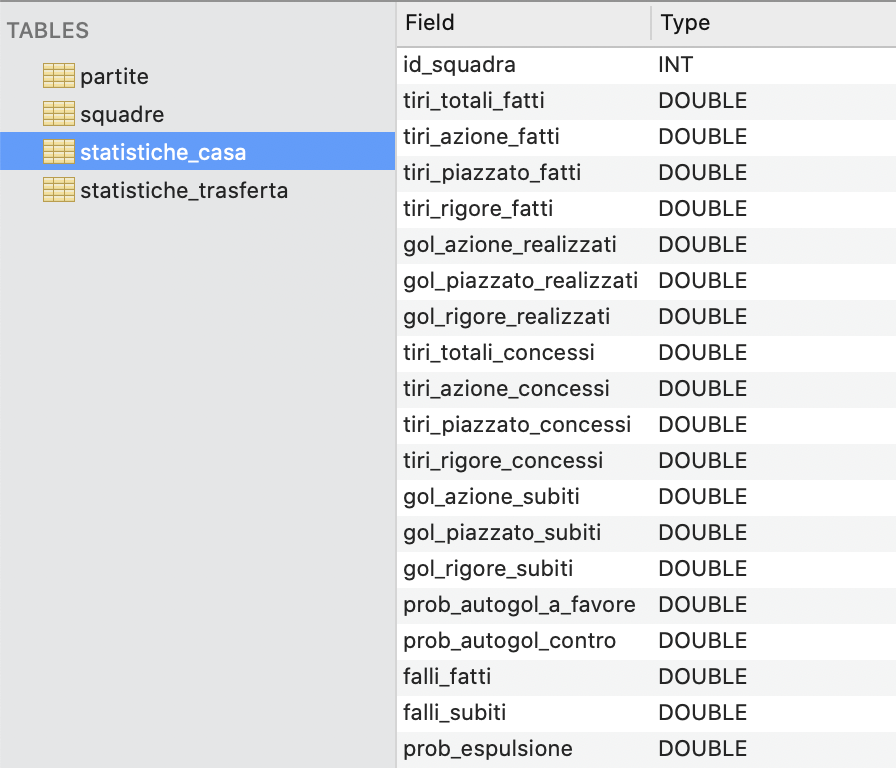
Il data-set denominato *partite* è costituito da 6 campi:

* + id\_partita identificativo della partita
  + giornata numero del turno di gioco
  + id\_squadra\_casa identificativo della squadra in casa
  + id\_squadra\_trasferta identificativo della squadra in trasferta
  + gol\_squadra\_casa numero reti della squadra in casa
  + gol\_squadra\_trasferta numero reti della squadra in trasferta



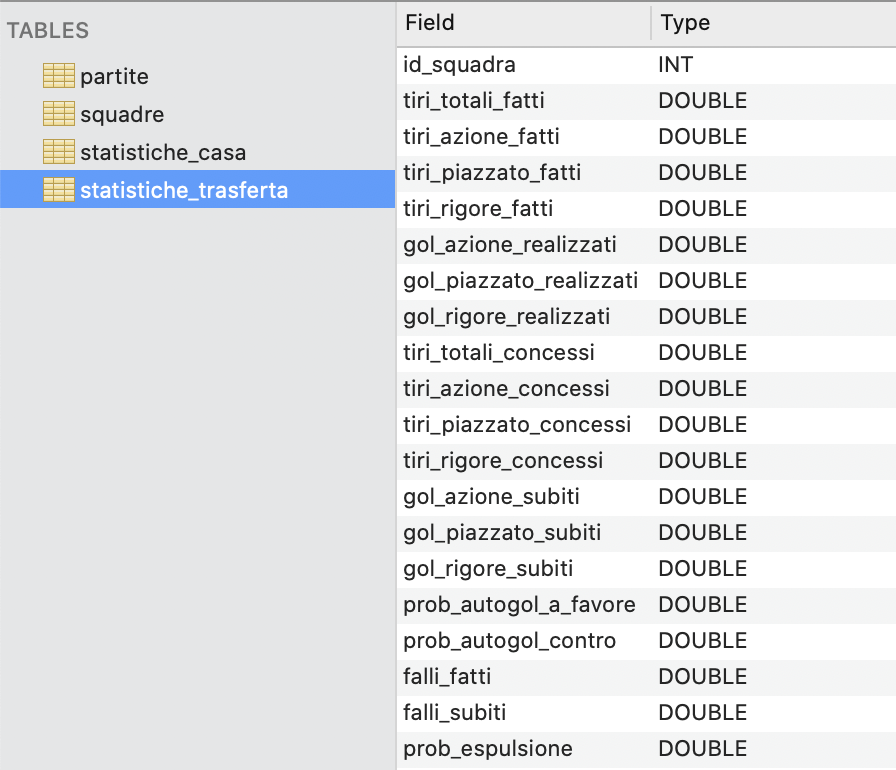
Il data-set denominato *squadre* è costituito da 14 campi:

* id\_squadra identificativo della squadra
* nome\_squadra nome della squadra
* pg\_casa partite giocate in casa
* v\_casa vittorie in casa
* n\_casa pareggi in casa
* p\_casa sconfitte in casa
* gf\_casa reti fatte in casa
* gs\_casa reti subite in casa
* pg\_trasferta partite giocate in trasferta
* v\_trasferta vittorie in trasferta
* n\_trasferta pareggi in trasferta
* p\_trasferta sconfitte in trasferta
* gf\_trasferta reti fatte in trasferta
* gs\_trasferta reti subite in trasferta



Il data-set denominato *statistiche\_casa* è costituito da 20 campi:

* id\_squadra identificativo della squadra
* tiri\_totali\_fatti media tiri totali fatti
* tiri\_azione\_fatti media tiri su azione fatti
* tiri\_piazzato\_fatti media tiri su calcio piazzato fatti
* tiri\_rigore\_fatti media tiri su calcio di rigore fatti
* gol\_azione\_realizzati media gol su azione fatti
* gol\_piazzato\_fatti media gol su calcio piazzato fatti
* gol\_rigore\_fatti media gol su calcio di rigore
* tiri\_totali\_concessi media tiri totali subiti
* tiri\_azione\_concessi media tiri su azione subiti
* tiri\_piazzato\_concessi media tiri su calcio piazzato subiti
* tiri\_rigore\_concessi media tiri su calcio di rigore subiti
* gol\_azione\_subiti media gol su azione subiti
* gol\_piazzato\_subiti media gol su calcio piazzato subiti
* gol\_rigore\_subiti media gol su calcio di rigore subiti
* prob\_autogol\_a\_favore probabilità di un autogol a favore
* prob\_autogol\_contro probabilità di un autogol a sfavore
* falli\_fatti media falli fatti
* falli\_subiti media falli subiti
* prob\_espulsione probabilità di espulsione per fallo fatto



Il data-set denominato *statistiche\_trasferta* è costituito dagli stessi 20 campi del data-set *statistiche\_casa* ma i valori concernano i dati statistici in trasferta di ogni squadra.

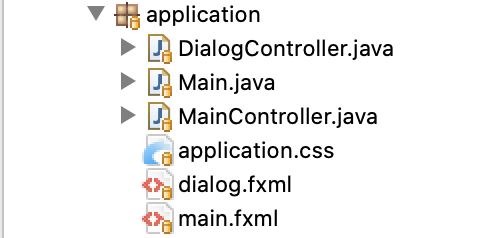
4. STRUTTURE DATI E ALGORITMI

**4.1 Descrizione delle strutture dati utilizzate**

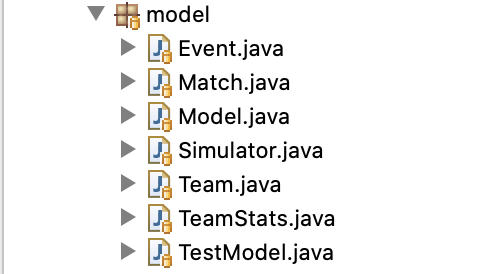
L’applicazione è realizzata mediante l’utilizzo del linguaggio di programmazione Java ed in particolare mediante l’utilizzo degli applicativi JavaFX. Sono stati altresì implementati il pattern MVC (Model View Controller), per garantire una corretta divisione delle operazioni all’interno del software delegando al Model la logica applicativa mentre il Controller ha il compito di gestire l’interazione con l’utente e mostrare a video i risultati, e il pattern DAO (Data Access Object) per la gestione delle interazioni con il database.

Strutturalmente l’applicazione è suddivisa nei seguenti tre package:

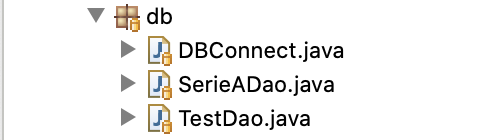
* Il package *application* contiene tutte le classi e i file che si occupano di realizzare l’interfaccia grafica e di interagire con l’utente ricevendo i parametri in input da inviare al model e restituendo i risultati.



* Il package *model* è il cuore dell’applicazione e contiene tutte le classi necessarie all’elaborazione dei dati. Il package contiene anche una classe dotata di *main* per testare gli algoritmi.



* Il package *db* contiene invece le classi utili alla gestione delle connessioni al database e all’estrapolazione dei dati necessari per l’implementazione del model.



È presente anche un quarto package denominato *img* che contiene tutte le immagini utilizzate dall’interfaccia grafica in formato *.png*.

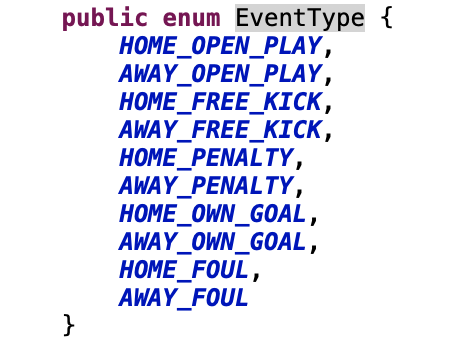
**4.2 Descrizione degli algoritmi utilizzati**

Ogni qualvolta si avvia una simulazione tramite apposito pulsante viene richiamato il metodo startSimulation() della classe *Model* che aggiorna i parametri di simulazione personalizzabili in input – qualora siano stati modificati quelli impostati di default – e verifica quale sia il tipo di simulazione scelto dall’utente.

La scelta del tipo di simulazione determina il numero di ripetizioni dell’algoritmo di simulazione principale situato nella classe *Simulator*. Per ogni partita da simulare, infatti, il simulatore viene inizializzato, aggiornato con i dati delle squadre coinvolte e successivamente avviato.

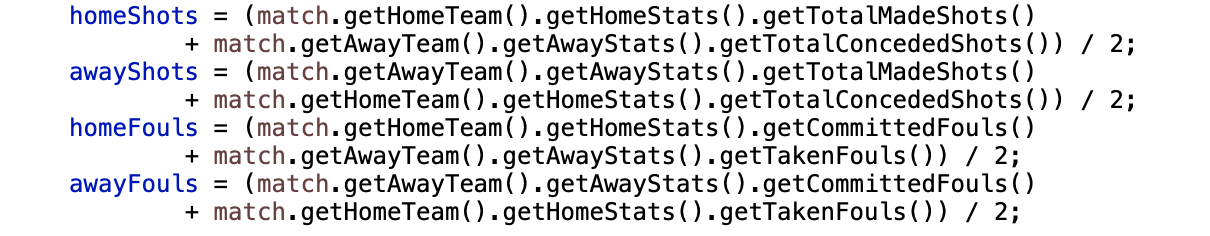
I risultati di ogni singola partita, suddivisi per giornata, vengono poi memorizzati in una mappa denominata simulatedDays all’interno della classe *Model*.

Il simulatore utilizza una PriorityQueue per la gestione degli eventi di una partita inizializzata in modo tale da estrarre gli stessi eventi in modo casuale. Si possono verificare dieci tipi di evento diversi per match:

**

L’inizializzazione del simulatore è molto complessa. Oltre a resettare tutte la variabili di output, il metodo init(Match match) è responsabile del corretto inserimento di tutti gli eventi nella *queue* e della loro probabilità di realizzazione.

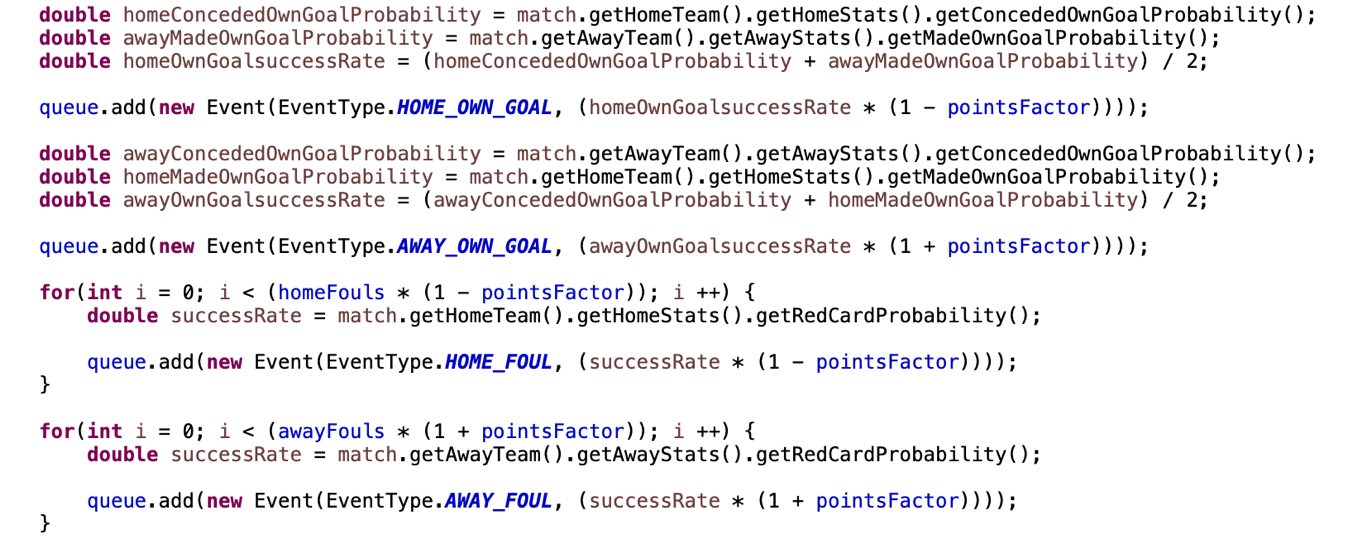
Il numero di occasioni a favore della squadra in casa viene determinato effettuando una media pesata del numero di tiri fatti a partita dalla squadra in casa e del numero di tiri subiti a partita dalla squadra in trasferta. Il numero di occasioni a favore della squadra in trasferta segue la stessa logica. Analogo è anche il calcolo del numero di falli fatti e subiti da entrambe le squadre.



Il metodo inserisce quindi tanti eventi in coda fino a raggiungere il numero medio pesato di occasioni per squadra appena calcolati. Viene inoltre calcolata la probabilità di realizzazione di ogni singolo evento: ad esempio il *successRate* di un *HOME\_OPEN\_PLAY*,ovvero la probabilità che la squadra di casa realizzi una rete in seguito ad un tiro su azione,è determinato effettuando una media pesata tra il numero di reti fatte su azione per tiro su azione dalla squadra in casa e il numero di reti subite su azione per tiro su azione dalla squadra in trasferta.



Analogamente, vengono inseriti tanti eventi in coda fino a raggiungere il numero medio pesato di falli a favore per squadra. In questo caso il *successRate* rappresenta la probabilità che l’evento di un fallo di gioco provochi un’espulsione. Viene inoltre inserito in coda un evento di tipo *OWN\_GOAL* – autogol – per squadra.



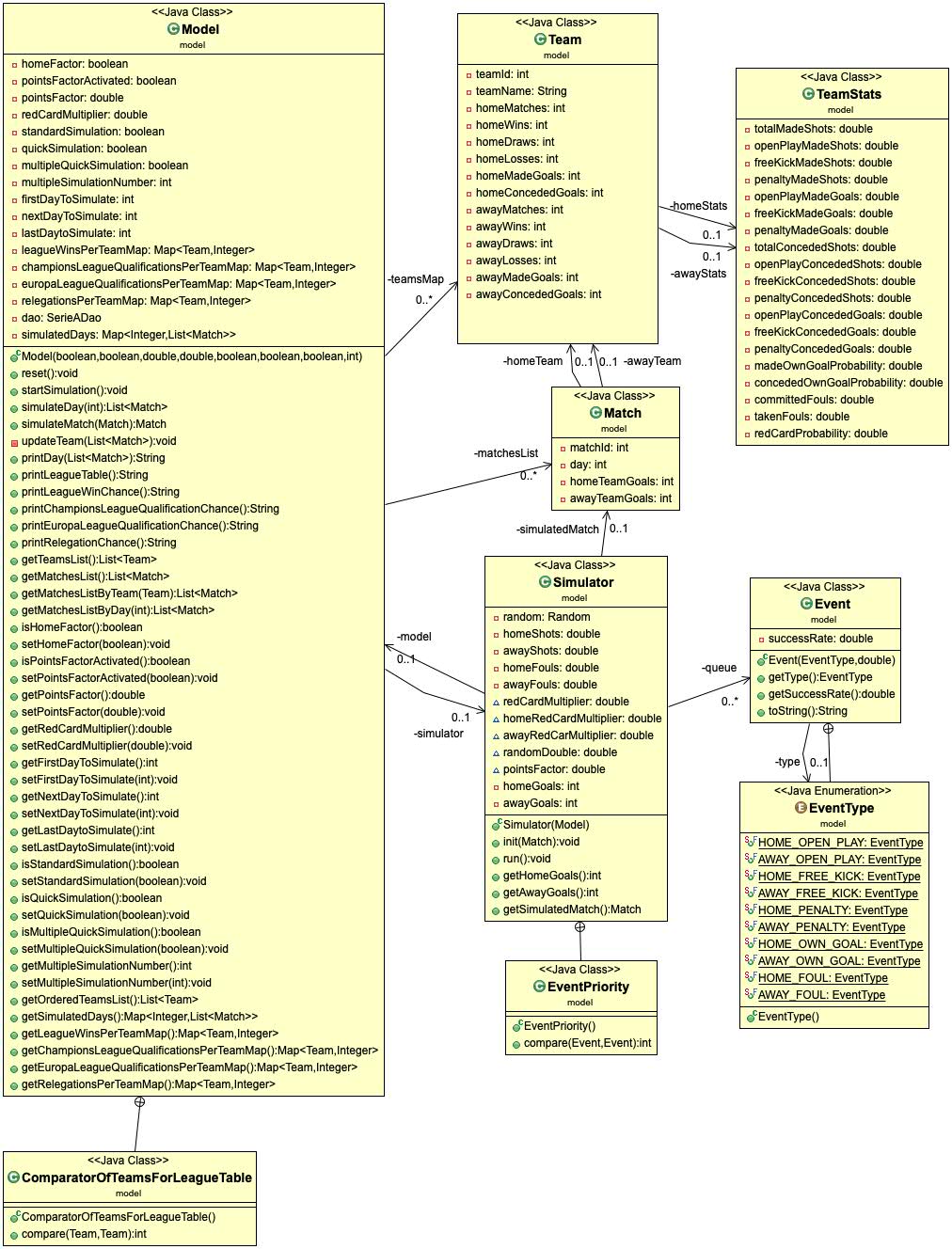
I moltiplicatori *homeFactor* e *pointsFactor* incidono sul numero e sulla percentuale di realizzazione delle occasioni da rete e dei falli commessi e subiti di entrambe le squadre coinvolte nella simulazione. Maggiore è il loro valore, maggiore è l’impatto sui dati coinvolti.

Il *redCardMultiplier*, invece, determina un aumento del solo numero di occasioni da rete per la squadra che ne beneficia e una diminuzione dello stesso per la squadra che la subisce in seguito al verificarsi di un espulsione.

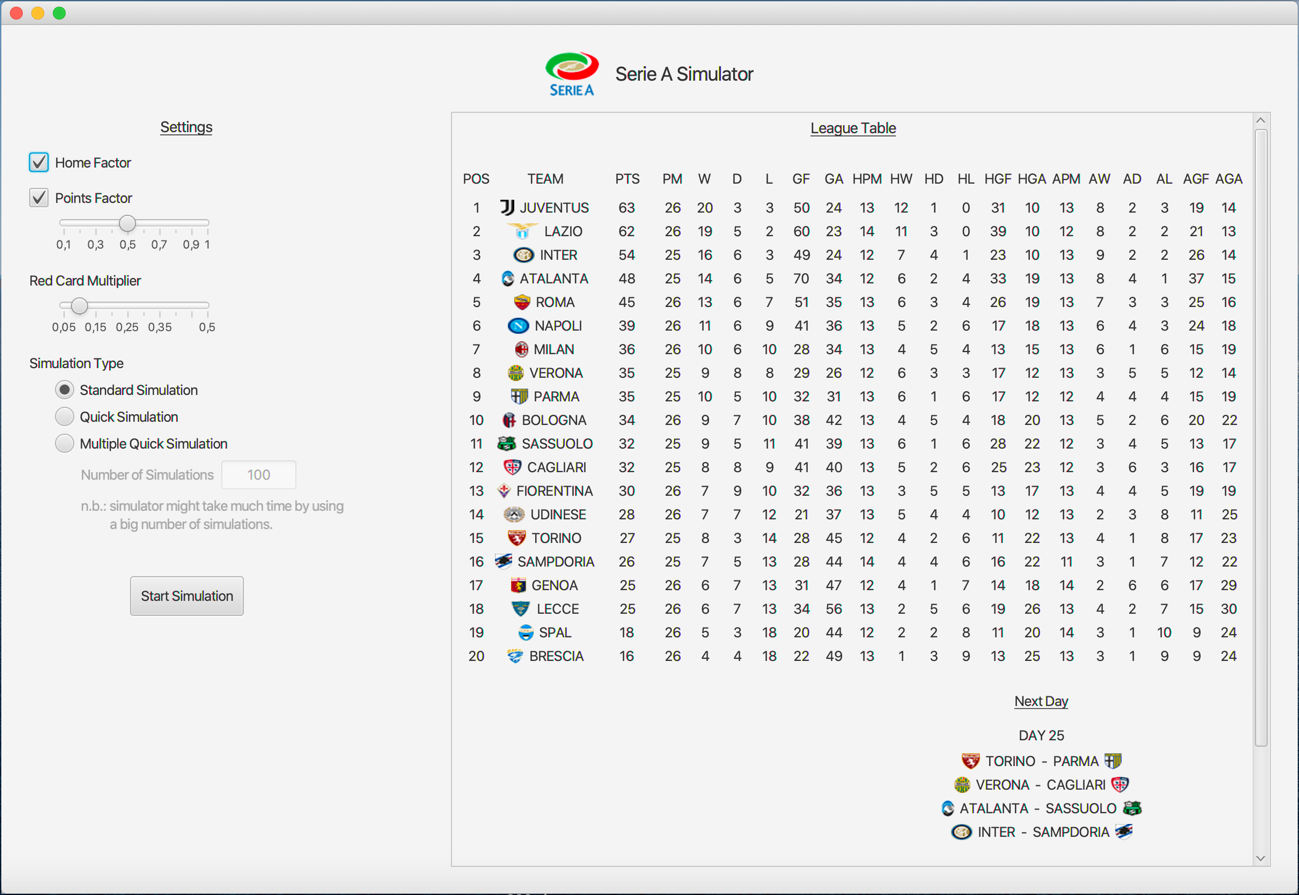
Il metodo run() avvia la simulazione estraendo uno alla volta i vari eventi dalla coda e li gestisce determinando in modo casuale la realizzazione degli stessi in base al già discusso *successRate*. Il metodo aggiorna quindi i parametri della simulazione.



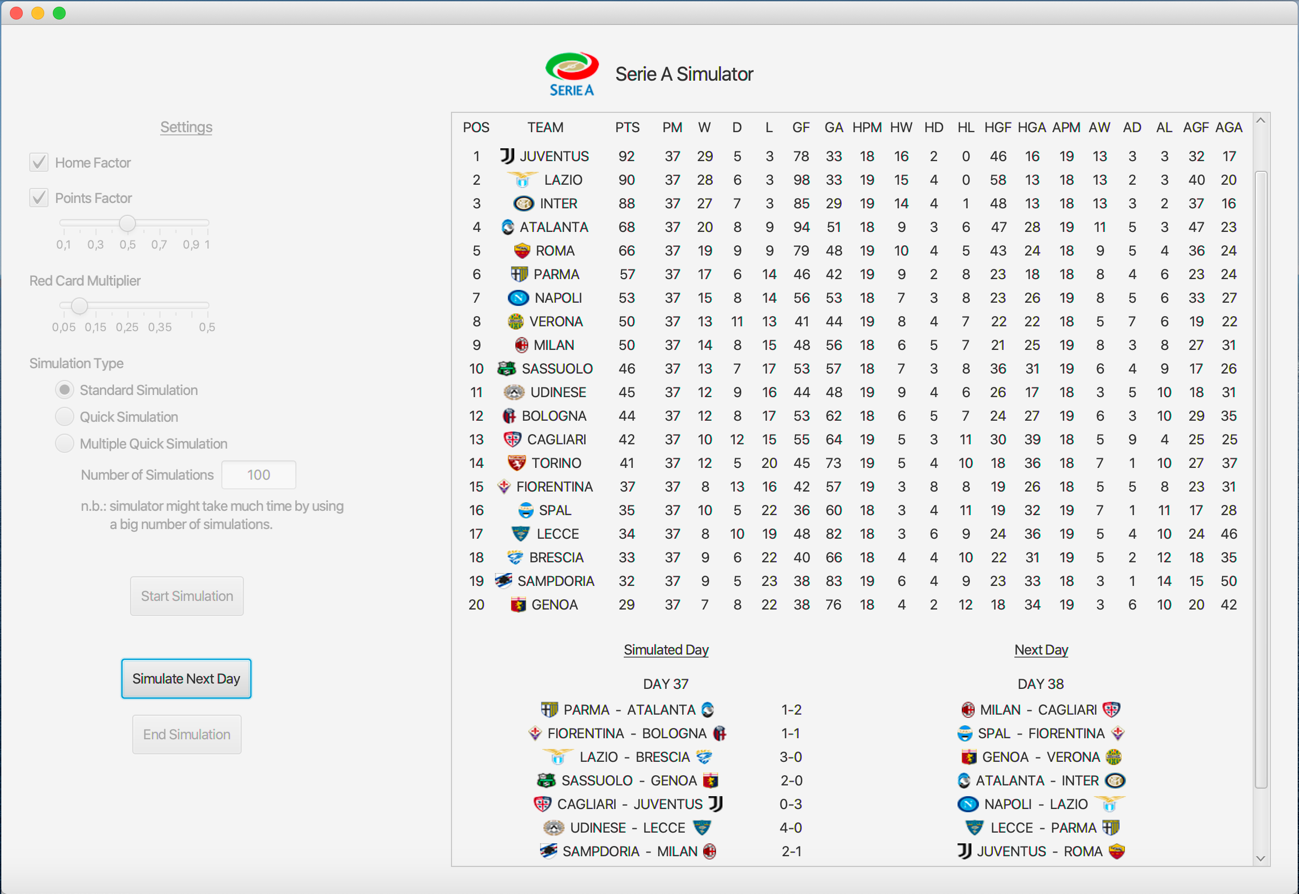
5. DIAGRAMMA DELLE CLASSI PRINCIPALI



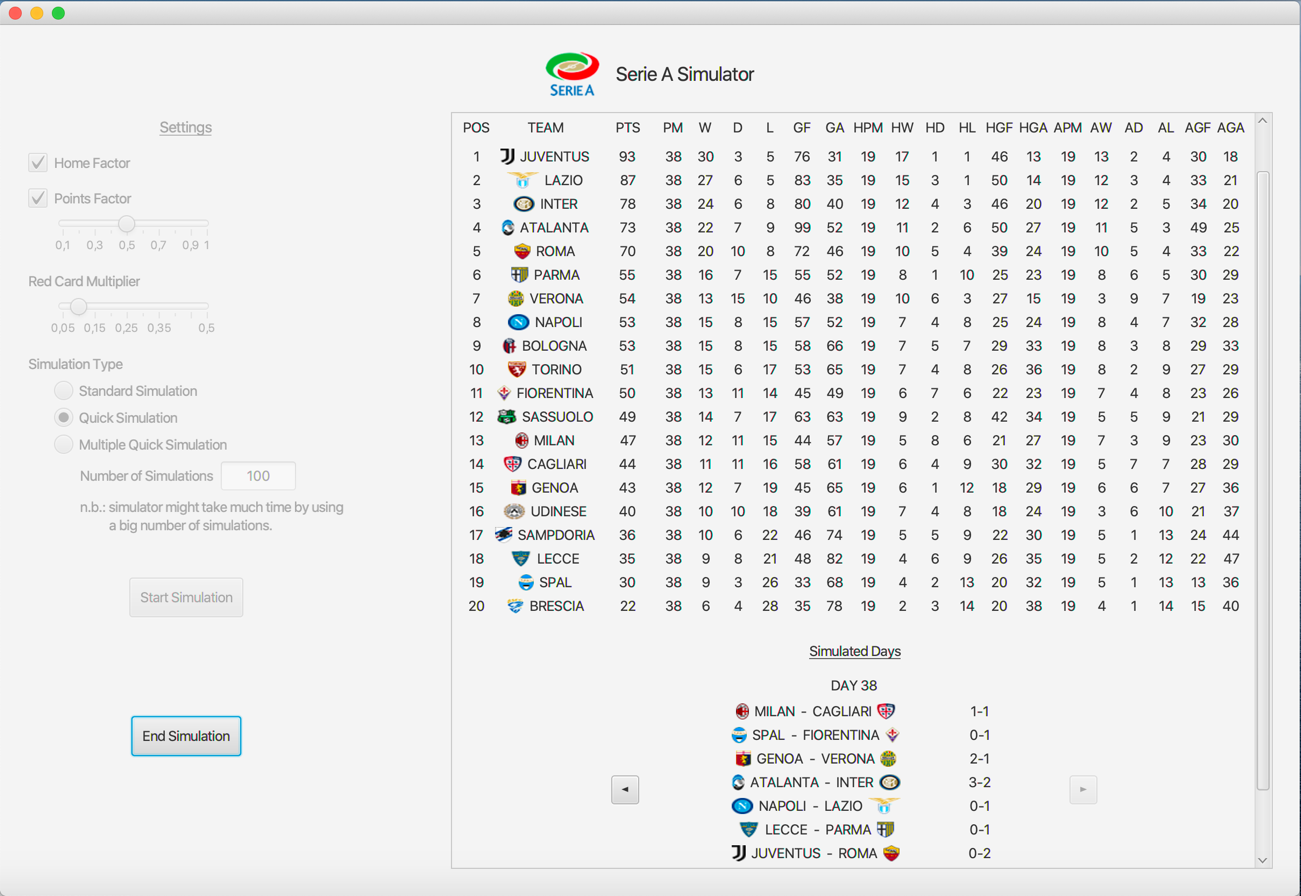
6. VIDEATE DELL’APPLICAZIONE E DIMOSTRAZIONE



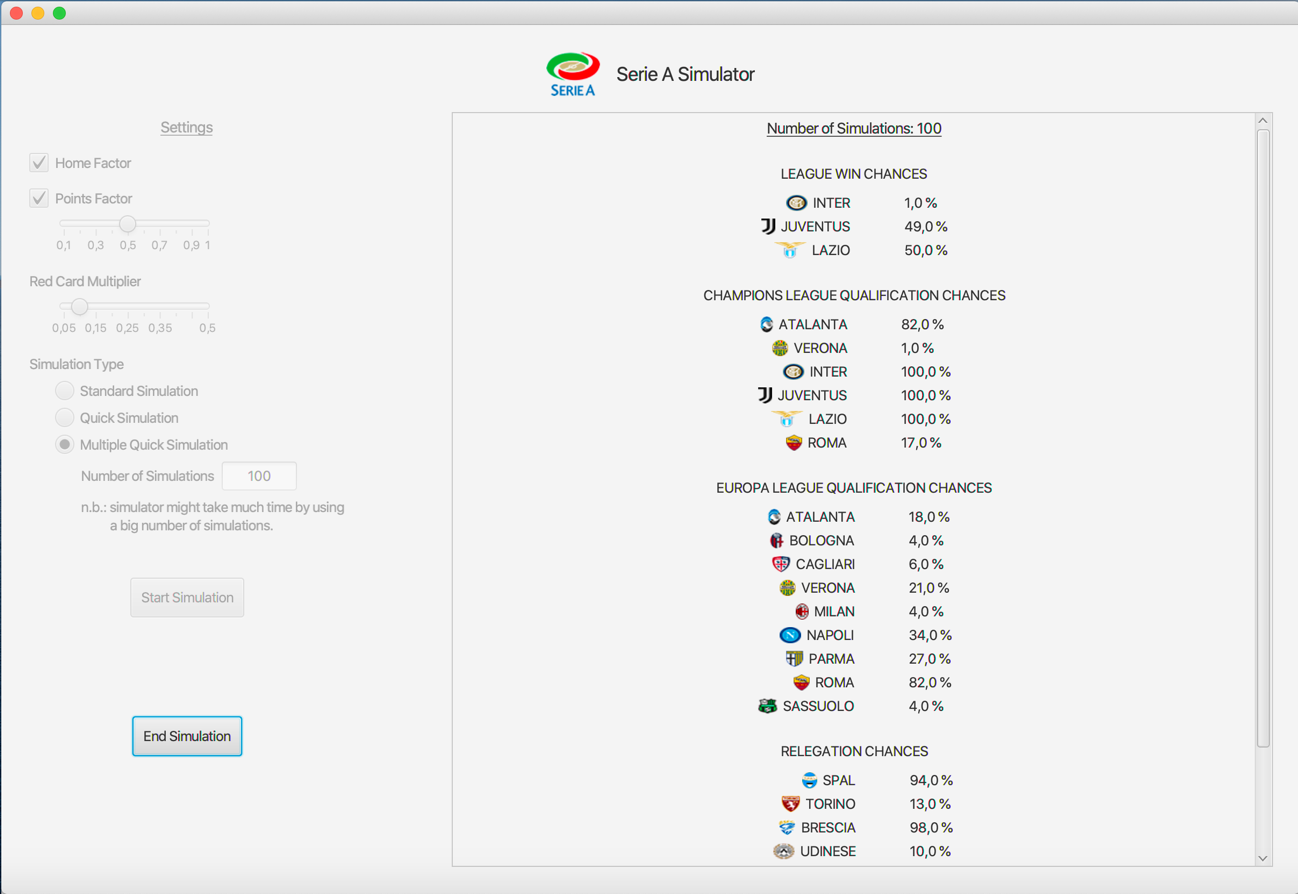
*Videata iniziale dell’applicazione*



*Videata dell’applicazione durante una Standard Simulation*



*Videata dell’applicazione al termine di una Quick Simulation*



*Videata dell’applicazione al termine di una Multiple Quick Simulation*

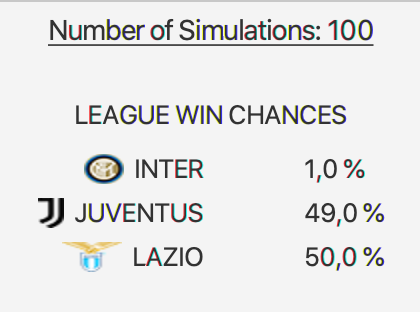
Video dimostrativo su Youtube: <https://youtu.be/0ihUSEDus1w>

7. VALUTAZIONE DEI RISULTATI E CONCLUSIONI

La simulazione veloce multipla può fornire un’idea sulla bontà algoritmica dell’applicazione. Appare chiaro che un algoritmo di simulazione per un campionato di calcio non potrebbe mai garantire la certezza assoluta del risultato così come è indiscutibile il risultato sul campo di giuoco.

Ciò viene evidenziato particolarmente quando si confronta il cammino di due squadre molto vicine in classifica. Sul campo sono i dettagli a fare la differenza e questi dettagli sono variabili casuali difficilmente computerizzabili.

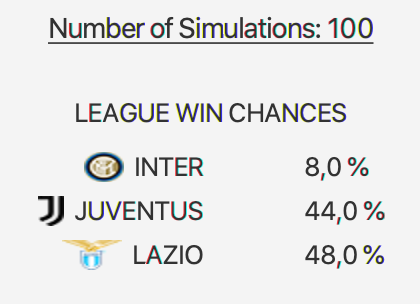
Si rende dunque fondamentale sfruttare appieno i dati a disposizione sulle squadre – tra l’atro, di non semplice reperibilità – e riuscire a dare un peso specifico ai risultati ottenuti. In base a quest’esperienza, si può affermare che, una volta scelti i parametri di simulazione, quest’ultima è tanto più accurata quanto è maggiore il numero di ripetizioni.



*homeFactor: attivo*

*pointsFactor: 0.5*

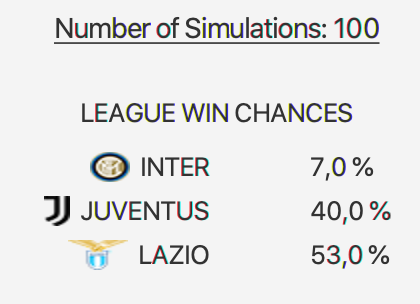
*redCardMultiplier: 0.1*



*homeFactor: attivo*

*pointsFactor: 0*

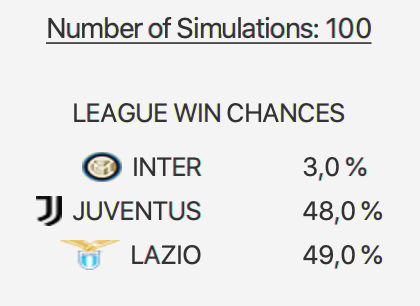
*redCardMultiplier: 0.1*



*homeFactor: attivo*

*pointsFactor: 0.2*

*redCardMultiplier: 0.1*



*homeFactor: attivo*

*pointsFactor: 0.5*

*redCardMultiplier: 0.25*

Effettuando cento simulazioni differenti si può notare come, anche variando i parametri di simulazione modificabili dall’utente nel tab *Settings*, i risultati ottenuti sono del tutto paragonabili. Il che può dimostrare una discreta affidabilità dell’applicazione.

[Licenza Creative Commons](http://creativecommons.org/licenses/by-nc-sa/4.0/)

Quest'opera è distribuita con Licenza [Creative Commons Attribuzione - Non commerciale - Condividi allo stesso modo 4.0 Internazionale](http://creativecommons.org/licenses/by-nc-sa/4.0/).