Politecnico di Torino

Corso di Laurea
Anno accademico 2021/2022
Sessione di Laurea Dicembre 2022

Simulazione di una stagione di Champions League

Relatore:
Fulvio Corno

Candidato:
Fiore Nicolò Antonio

Politecnico
di Torino

Politecnico di Torino

Sommario

Capitolo 1 : Proposta di progetto 4
1.1 Studente proponente 4
1.2 Titolo della proposta 4
1.3 Descrizione del problema proposto 4
1.4 Descrizione della rilevanza gestionale del problema 4
1.5 Descrizione dei data-set per la valutazione 4
1.6 Descrizione preliminare degli algoritmi coinvolti 5
1.7 Descrizione preliminare delle funzionalità previste per l'applicazione software 5
Capitolo 2: Descrizione dettagliata del problema affrontato 7
2.1 Contesto operativo 7
2.2 Criticità 7
Capitolo 3: Descrizione del data-set utilizzato per l'analisi 8
3.1 Ranking 8
3.2 Squadre 8
3.3 Stadi 9
Capitolo 6: Descrizione ad alto livello delle strutture dati e degli algoritmi utilizzati 10
5.1 Strutture dati 10
5.2.1 Ricorsione 14
5.2.2 Simulazione 19
Capitolo 7: Alcune videate dell'applicazione realizzata e link al video dimostrativo del software 22
Capitolo 8:Valutazioni sui risultati ottenuti 28

Capitolo 1 : Proposta di progetto

1.1 Studente proponente

s273552 Fiore Nicolò Antonio

1.2 Titolo della proposta

Simulazione di una stagione di Champions League

1.3 Descrizione del problema proposto

Il primo problema da risolvere è la creazione di un calendario che secondo le nuove modalità della Champions League dal 2024 sarà a girone unico a 36 squadre. La particolarità è che non tutte le squadre si sfideranno tra di loro, in particolare le 36 squadre saranno divise in 4 fasce da 9 squadre, ogni squadra affronterà 2 squadre appartenenti alle 4 diverse fasce. Alla fine di tutte le partite del girone le prime 8 squadre passano agli ottavi mentre dalla nona alla 24esima si sfideranno in un play off.
Il secondo problema è la simulazione delle partite per decretare un risultato per ogni match disputato. I risultati delle partite servono per stabilire la classifica del girone o il passaggio del turno, i risultati influiscono anche sui guadagni e sulla capienza dello stadio.

1.4 Descrizione della rilevanza gestionale del problema

Dal punto di vista gestionale la rilevanza consiste nella creazione del calendario che dovrà essere equilibrato in modo da avere partite di un certo spessore (sfide tra squadre di un certo livello di ranking) in ogni turno evitando che queste si concentrino tutte nella stessa giornata; inoltre è importante avere una stima dei guadagni che una squadra potrà ottenere in base ai risultati ottenuti nella competizione.

1.5 Descrizione dei data-set per la valutazione

I data-set per la valutazione saranno tre.
Il primo è rappresentato dalla tabella ranking(sito di riferimento https://it.uefa.com/) in questo data set è presente la posizione in classifica del ranking UEFA ,il nome della squadra, la sigla della federazione nazionale di cui fa parte la squadra, i punti ottenuti nelle ultime 5 stagioni, la somma

totale dei punti ,e infine il punteggio della federazione di riferimento.
Il secondo include le 36 squadre qualificate alla Champions League in questa tabella è presente : nome della squadra, campionato di appartenenza, nome dello stadio, 4 caratteristiche sulla squadra espresse in numeri interi che indicano la "forza" della squadra e infine la fascia in cui è inserita la squadra.
L'ultimo data set riguarda gli stadi per ogni riga sono presenti il nome della squadra, la capienza, la città e l'anno di costruzione, **costo medio del biglietto;**
Identificare le fonti dalle quali verranno tratti i dati utilizzati
https://it.uefa.com
https://sofifa.com/teams
https://it.wikipedia.org
https://www.stadi.online/
https://seatpick.com/

1.6 Descrizione preliminare degli algoritmi coinvolti

I principali algoritmo saranno:
ottimizzazione del calendario per renderlo più equilibrato possibile, secondo le condizioni indicate in precedenza (squadre di
fasce diverse si sfidano in egual modo, e che non tutte le partite di cartello avvengano nello stesso turno).

simulazione dei match e quindi i risultati che andranno a comporre la classifica
simulazione delle diverse condizioni di capienza dello stadio e di costo dei biglietti, ad esempio per una partita di squadre di prima la fascia la capienza dello stadio con grande probabilità sarà al completo e avrà un costo alto del biglietto

1.7 Descrizione preliminare delle funzionalità previste per l'applicazione software

L'applicazione sarà in grado, tramite un'interfaccia dotata di bottoni, di permettere all'utente di sfruttare alcune funzionalità.
"Crea Calendario Partite" crea il calendario delle partite del girone.
"Simula Fase Girone" dopo la creazione delle sfide del girone sarà possibile simulare le partite del girone da cui si otterrà la classifica che verrà mostrata a video
"Simula Fase eliminazione Diretta" crea gli accoppiamenti per la fase a eliminazione diretta e ne calcola il risultato ottenendo così il vincitore

Politecnico di Torino

Le seguenti funzionalità sono state modificate e organizzate diversamente.
Saranno poi a disposizioni altre funzionalità in grado di ricercare informazioni richieste dall'utente:
-Sarà possibile selezionare da un menù a tendina una squadra e tramite un bottone sarà possibile visualizzare tutti i risultati relativi a quella squadra con un messaggio che indica la posizione raggiunta(es: vincitore Competizione o eliminata agli Ottavi di finale)
Sarà possibile selezionare da un altro menù un turno della competizione e sempre tramite bottone sarà possibile vedere tutti i risultati relativi a quel turno

Capitolo 2: Descrizione dettagliata del problema affrontato

2.1 Contesto operativo

Dal 2024 l'UEFA applicherà la nuova riforma per la Champions League per far fronte alle richieste dei club di avere un maggiore introito economico dalla competizione.
Per far questo ha deciso di aumentare le squadre partecipanti passando dalle attuali 32, divise in 8 gironi da 4 squadre a ciascuno, a 36 in un girone unico, aumentando così le partite da 6 a 8.
La particolarità del girone unico sarà quella di avere una maggiore combinazione di scontri tra le squadre, ogni partecipante affronterà 8 avversarie diverse, 2 per ogni fascia, mentre nella modalità attuale ogni squadra incontra 3 avversarie una volta in casa e una volta in trasferta.
Dal punto di vista economico sono predisposti alcuni premi in base al merito sportivo durante la competizione, ma anche dovuti ai risultati passati, infatti ogni squadra guadagna una quota dovuta al proprio 'ranking storico' ad esempio il Bayern München che occupa la prima posizione nel ranking guadagna la quota base moltiplicata per 36 mentre l'ultima squadra del ranking guadagnerà solo la quota base.
I guadagni non dipendono poi solo dai risultati del singolo team, ma anche da quelli delle compagne di federazione, c'è infatti un ammontare di denaro chiamato 'Market Pool' che ogni turno viene diviso tra le federazioni ancora presenti ponderatamente al punteggio che la federazione ha ottenuto negli anni passati, se ad esempio i quarti di finale sono raggiunti da 3 squadre italiane, 2 spagnole,2 inglesi e una tedesca la quota di denaro è divisa prima tra le quattro federazione in base a chi ha il punteggio più alto e poi distribuito egualmente tra le squadre della stessa federazione.

2.2 Criticità

La criticità principale è dovuta alla creazione del calendario del girone, come detto ogni squadra dovrà affrontare 8 diverse avversarie rispettando però alcuni vincoli:

- Nessuna squadra potrà giocare più di una partita nell'arco della stessa giornata
- Tra le 8 partite metà dovranno essere giocate in casa e metà in trasferta
- Ogni squadra dovrà giocare contro 2 squadre per ognuna delle 4 fasce una volta come padrona di casa e l'altra come ospite.
- Due squadre non potranno scontrarsi due volte nella fase a gironi
- In ogni giornata ci deve essere almeno una partita tra due squadre di prima fascia

Nome	Tipo di dati
Posizione	INT
Nome_Squadra	VARCHAR
Campionato	VARCHAR
17/18	DOUBLE
18/19	DOUBLE
19/20	DOUBLE
20/21	DOUBLE
21/22	DOUBLE
Punti	DOUBLE
Federazione	DOUBLE

Capitolo 3: Descrizione del data-set utilizzato per l'analisi

3.1 Ranking

L' attributo 'Posizione' indica la posizione del ranking
L'attributo 'Nome_squadra' è univoco e permette di collegare questa tabella alla tabella 'squadre'.
L'attributo 'Campionato' riporta la sigla della federazione in cui la squadra gioca
Gli attributi '17/18'….'21/22' rappresentano i punti guadagnati nelle passate stagioni
L'attributo 'Punti' è la somma totale delle 5 precedenti stagioni
L'attributo 'Federazione' indica il punteggio della federazione

3.2 Squadre

Nome	Tipo di dati
Nome_Squadra	**VARCHAR**
Campionato	VARCHAR
Nome_Stadio	VARCHAR
OVR	INT
ATT	INT
MID	INT
DEF	INT
FASCIA	INT

L'attributo 'Nome_squadra' è univoco e permette di collegare questa tabella alla tabella 'squadre'.
L'attributo 'Campionato' riporta la sigla della federazione in cui la squadra gioca.
L'attributo 'Nome_Stadio' è univoco e rappresenta il nome dello stadio in cui la squadra gioca le sue partite in casa.
Gli attributi 'OVR','ATT','MID','DEF' sono dei parametri numerici che indicano il valore della squadra,sono stati presi dal videogioco FIFA 22
L'attributo 'Fascia' indica la fascia di appartenenza della squadra

3.3 Stadi

Nome	Tipo di dati
Nome_Stadio	VARCHAR
Capienza	INT
Città	VARCHAR
Anno_Costruzi...	INT
Prezzo_bigliet...	INT

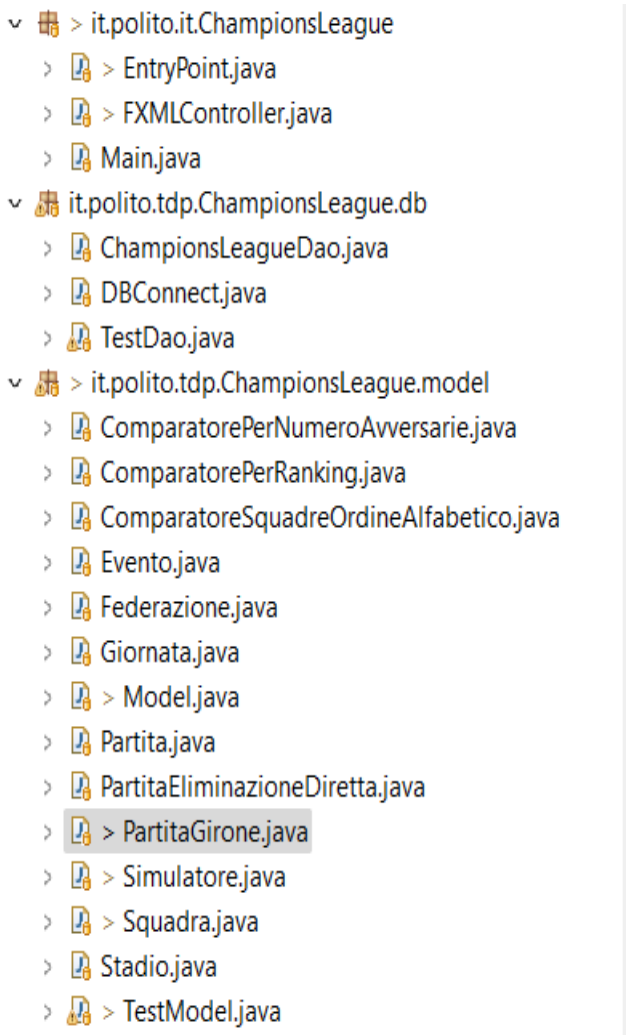
L'attributo 'Nome_Stadio' è univoco e rappresenta il nome dello stadio in cui la squadra gioca le sue partite in casa.
L'attributo 'Capienza' indica la capienza massima dello stadio
L'attributo 'Città' indica la città di appartenenza della squadra
L'attributo 'Anno_Costruzione indica l'anno in cui lo stadio è stato costruito
L'attributo 'Prezzo_biglietto' indica il prezzo medio per assistere ad una partita

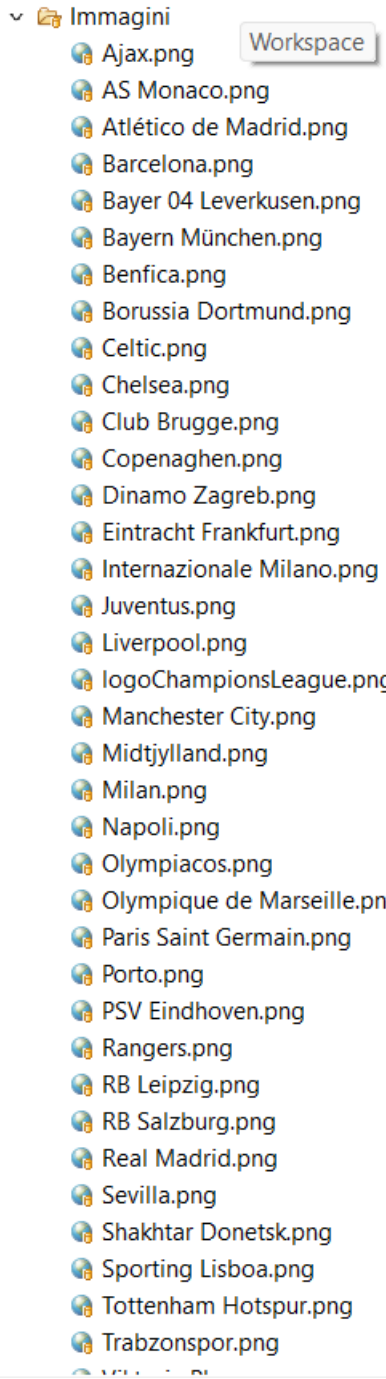
Capitolo 6: Descrizione ad alto livello delle strutture dati e degli algoritmi utilizzati

5.1 Strutture dati

L'intero progetto è stato sviluppato utilizzando regole di buona progettazione, seguendo quindi i pattern MVC (Model View Controller) per dividere il codice in blocchi dalle funzionalità ben distinte e DAO (Data Access Object) per separare la logica di business dalla logica di accesso ai dati.

Politecnico
di Torino





Il package it.polito.it.ChampionsLeague contiene la classe EntryPoint in cui viene settata la scena, la classe FXMLController riceve i comandi dell'utente attraverso il View e li trasmette al model e infine il Main che lancia l'esecuzione del programma.

Politecnico di Torino

Il package it.polito.it.ChampionsLeague.db è il package che permette il collegamento al database tramite la classe DBConnect e il passaggio dei dati presenti in esso al model tramite la classe ChampionsLeagueDAO,e anche presente una classe dotata di un proprio main utile a controllare in modo veloce che il collegamento con il database sia stato effettuato correttamente.
Il package it.polito.it.ChampionsLeague.model è il package dove viene sviluppata la logica applicativa, in particolare sono presenti le classi model dove sono sviluppati gli algoritmi ricorsivi e quella Simulatore dove avvengono le simulazioni, sono presenti poi altre classi utili allo sviluppo del programma
Nella cartella immagini sono presenti tutti i loghi delle squadre partecipanti alla competizione e anche il logo della Champions League

Politecnico di Torino

5.2 Algoritmi utilizzati

5.2.1 Ricorsione

Politecnico di Torino

```
public boolean ricorsione(Giornata giornata, List<Squadra> squadreParziale) {
    interTempo=System.nanoTime();
    double intervallo=(interTempo-start)/1000000000;
    if(intervallo>0.2) {
        this.initiGiornate();
        return false;
    }
    else {
      if(numPartite(giornata)==18) {
          if(giornata.getCont11()>=1
                  && giornata.getCont22()>=1 )  {
          if(giornata.getIdGiornata()<6) {
              partiteGironi=new ArrayList<>(parziale);
          squadreParziale=new ArrayList<>(squadre);
          boolean continual=ricorsione(giornate.get(giornata.getIdGiornata()+1),squadreParziale);
            if(!continual)
                return false;
          }
          if(giornata.getIdGiornata()==6) {
             partiteGironi=new ArrayList<>(parziale);
              squadreParziale=new ArrayList<>(squadre);
              this.calcolaPossibiliAvversarie(giornata, squadreParziale);
              boolean continual=ricorsioneUltimeGiornate(giornate.get(giornata.getIdGiornata()+1),squadreParz
                if(!continual)
                    return false;
          }

          boolean flag=false;
          for(Giornata g: this.giornate.values()) {
              if(g.getPartite().size()!=18) {
                  flag=true;
              }
          }
          if(flag==false) {
              return false;
```

```
 if(giornata.getContll()<1) {
Collections.shuffle(primaFasciaCasa);
    for(Squadra casa: primaFasciaCasa) {
        for(Squadra ospite: primaFasciaCasa) {
    if(!casa.equals(ospite)) {
            PartitaGirone p=new PartitaGirone(casa,ospite, giornata);
            if(!parziale.contains(p)) {
                if(controlloPartita(casa,ospite)) {
                    if(controlloGiornata(p,giornata)) {
                          giornata.aggiungiPartita(p);
                          parziale.add(p);
                          this.incrementaPartita(casa, ospite);
                          //p.setPartitaAggiunta(true);
                          squadreParziale.remove(casa);
                          squadreParziale.remove(ospite);
                          casa.aggiungiPartita(p);
                          ospite.aggiungiPartita(p);
                          boolean continua=ricorsione(giornata,squadreParziale);
                            if(!continua)
                                return false;
                        giornata.getPartite().remove(p);
                        decrementa(p,giornata);
                        parziale.remove(p);
                        this.decrementaPartita(casa, ospite);
                        squadreParziale.add(casa);
                        squadreParziale.add(ospite);
                        casa.rimuoviPartita(p);
                        ospite.rimuoviPartita(p);
                          }
                    }
                }
            }
        }
    }
}
```

L’algoritmo principale del programma è quello della creazione del girone, è presente una condizione di terminazione dovuta al tempo perché essendo un algoritmo ad alta complessità computazionale ho deciso di adottare un sistema che mi permettesse di evitare di imbattermi in loop infiniti dovuti incastri di partite che non permettano di trovare una soluzione e in questi casi ho capito che la soluzione migliore era quella di ricominciare da capo la ricorsione.
Ci sono poi altre condizioni di terminazione che determinano la soluzione ottimale, quando una giornata ha raggiunto la sua saturazione (18 partite) si passa alla successiva fino alla sesta giornata dove si passa dal primo algoritmo ricorsivo ad un secondo.


```
private boolean ricorsioneUltimeGiornate(Giornata giornata, List<Squadra> squadreParziale) {
    interTempo=System.nanoTime();
    double intervallo=(interTempo-start)/1000000000;
    if(intervallo>0.2) {
        this.initiGiornate();
        return false;
    }
    else {
    if(numPartite(giornata)==18) {
          if(giornata.getCont11()>=1)  {
          if(giornata.getIdGiornata()<8) {
             partiteGironi=new ArrayList<>(parziale);
              squadreParziale=new ArrayList<>(squadre);
              this.calcolaPossibiliAvversarie(giornata, squadreParziale);
       //     System.out.println(giornata.getPartite().toString());
              boolean continual=ricorsioneUltimeGiornate(giornate.get(giornata.getIdGiornata()+1),squadreParziale);
                if(!continual)
                    return false;
          }
          if(giornata.getIdGiornata()==8)
              partiteGironi=new ArrayList<>(parziale);

          boolean flag=false;
          for(Giornata g: this.giornate.values()) {
              if(g.getPartite().size()!=18) {
                  flag=true;
              }
          }
          if(flag==false) {
              return false;
          }
      }
          return true;
      }
```


```
this.calcolaPossibiliAvversarie(giornata, squadreParziale);
Collections.sort(squadreParziale, new ComparatorePerNumeroAvversarie());
List<Squadra> possibiliAvversarie;
List<Squadra> sp=new ArrayList<>(squadreParziale);
for(Squadra s : sp) {
    if(s.possibiliAvversarie.size()==0)
        return true;
    possibiliAvversarie=new ArrayList<>(s.getPossibiliAvversarie());
    for(Squadra s2: possibiliAvversarie) {
        if(!s.equals(s2)) {
        PartitaGirone p=new PartitaGirone(s,s2, giornata);
        if(!parziale.contains(p)) {
            if(controlloPartita(s,s2)) {
                if(controlloGiornata(p,giornata)) {
                    giornata.aggiungiPartita(p);
                      parziale.add(p);
                      this.incrementaPartita(s, s2);
                    //  p.setPartitaAggiunta(true);
                      squadreParziale.remove(s);
                      squadreParziale.remove(s2);
                      s.aggiungiPartita(p);
                      s2.aggiungiPartita(p);
                      boolean continua=ricorsioneUltimeGiornate(giornata,squadreParziale);
                        if(!continua)
                            return false;
                  giornata.getPartite().remove(p);
                 //  p.setPartitaAggiunta(false);
                    decrementa(p,giornata);
                    parziale.remove(p);
                      this.decrementaPartita(s, s2);
                      squadreParziale.add(s);
                      squadreParziale.add(s2);
                      s.rimuoviPartita(p);
                      s2.rimuoviPartita(p);
                      this.calcolaPossibiliAvversarie(giornata, squadreParziale);
```

I problemi principali della ricorsione sono giunti nella creazione delle ultime due giornate quando i vincoli da rispettare rendevano molto più stringenti la possibile aggiunta di nuove partite, perciò ho deciso di implementare un secondo algoritmo ricorsivo con una logica molto simile al precedente,ma dotato di un ulteriore metodo che permette di calcolare per ogni squadra le possibili avversarie in quel momento del girone,questo mi ha permesso di dare priorità alle squadre che avevano la scelta più ‘obbligata’ avendo poche avversarie disponibili contro cui poter giocare.

5.2.2 Simulazione

```
private void calcolaRisultato(Squadra casa,Squadra ospite,Partita p) {
    double perc=Math.random();
    double valoreOffensivoCasa=(casa.getATT()+2+(casa.getMID()+2)*0.5)/1.5;
    double valoreDifensivoCasa=(casa.getDEF()+2+(casa.getMID()+2)*0.5)/1.5;
    double valoreOffensivoOspite=(ospite.getATT()+ospite.getMID()*0.5)/1.5;
    double valoreDifensivoOspite=(ospite.getDEF()+ospite.getMID()*0.5)/1.5;
    double diffCasa=valoreOffensivoCasa-valoreDifensivoOspite;
    double diffOspite=valoreOffensivoOspite-valoreDifensivoCasa;
    if(diffCasa<=21 && diffCasa>=19) {
        if(perc<=0.0125) {
            casa.setGolFatti(0);
            p.setGolCasa(0);
        }
        else if(perc<=0.03) {
            casa.setGolFatti(1);
            ospite.setGolSubiti(1);
            p.setGolCasa(1);
        }
```

I risultati dei match sono calcolati grazie ai parametri definiti nel database, le squadre in casa ricevono un boost di due punti sui loro attributi di attacco,difesa e centrocampo, grazie a questi valori ho stimato un valore offensivo dato dalla media ponderata tra il valore di attacco e il 50% del valore di centrocampo,l'altro 50% del valore del centrocampo è stato sommato all'attributo difensivo ottenendo così il valore difensivo.
La differenza tra valore offensivo e difensivo implica la quota di gol segnati,più la differenza è grande e maggiore è la probabilità di segnare più gol.


```java
public void calcolaMarketPool(List<Squadra> squadre,Map<String,Federazione> idFederazioni,String turno) {
    double perc=0;
    if(turno.equals("Ottavi"))
        perc=percMarketPoolOttavi;
    if(turno.equals("Quarti"))
        perc=percMarketPoolQuarti;
    if(turno.equals("Semifinale"))
        perc=percMarketPoolSemi;
    if(turno.equals("Finale"))
        perc=percMarketPoolFinale;

    Set<Federazione> fedPresenti=new HashSet<Federazione>();
    for(Federazione fed: idFederazioni.values()) {
        fed.setNumSquadrePerTurno(0);
    }
    for(Squadra s: squadre) {
        Federazione f=idFederazioni.get(s.getCampionato());
        f.aggiornaNumSquadrePerTurno(1);
        if(!fedPresenti.contains(f))
        fedPresenti.add(f);
        }
    double totalePuntiFederazione=0;
    for(Federazione f: fedPresenti) {
        totalePuntiFederazione+=f.getPunteggio();
        //idMap.put(f.getCampionato(), f);
    }
    for(Federazione ff: fedPresenti) {
        ff.setIndicatore(ff.getPunteggio()/totalePuntiFederazione);
    }
    for(Squadra ss: squadre ) {
        Federazione fed=idFederazioni.get(ss.getCampionato());
        ss.setMarketPool((int)(marketPool*perc*fed.getIndicatore()/fed.getNumSquadrePerTurno()));
    }

}
```

In questo algoritmo viene calcolata la quota spettante ad ogni squadra del Market Pool, in particolare vengono sommati i punteggi di tutte le federazioni presenti in quel turno e viene calcolare una fattore moltiplicativo che indicherà la quota di soldi destinata ad ogni federazione che viene divisa in egual modo nelle squadre ancora in corsa nella competizione della stessa federazione.


```
private void calcolaInfoStadio(Squadra casa, Squadra ospite, String turno) {
    Stadio s=idStadi.get(casa.getNomeStadio());

    double random=Math.random();
    if(turno.equals("Girone")) {
        if(ospite.getPosizioneRanking()>0 && ospite.getPosizioneRanking()<=10) {
            if(random<0.9) {
                int min_val = (int) (0.95*s.getCapienza());
                int max_val = s.getCapienza();
                Random ran = new Random();
                int capienza = ran.nextInt(max_val-min_val+1) + min_val;
                s.setSommaGuadagni((int)(s.getPrezzoBiglietto()*1.25)*capienza);
                casa.setBigliettiG((int)(s.getPrezzoBiglietto()*1.25)*capienza);
                s.setSommaCapienzaG(capienza);
            }
            else if(random<0.95) {
                int min_val = (int) (0.90*s.getCapienza());
                int max_val =(int) (0.94*s.getCapienza());
                Random ran = new Random();
                int capienza = ran.nextInt(max_val-min_val+1) + min_val;
                s.setSommaGuadagni((int)(s.getPrezzoBiglietto()*1.25)*capienza);
                s.setSommaCapienzaG(capienza);
                casa.setBigliettiG((int)(s.getPrezzoBiglietto()*1.25)*capienza);
            }
            else {
                int min_val = (int) (0.80*s.getCapienza());
                int max_val =(int) (0.89*s.getCapienza());
                Random ran = new Random();
                int capienza = ran.nextInt(max_val-min_val+1) + min_val;
                s.setSommaGuadagni((int)(s.getPrezzoBiglietto()*1.25)*capienza);
                casa.setBigliettiG((int)(s.getPrezzoBiglietto()*1.25)*capienza);
                s.setSommaCapienzaG(capienza);
```

Le informazioni sullo stadio sono calcolate in base alla squadra che si affronta: se si affronta una squadra in una posizione alta di ranking i biglietti subiranno un rincaro e lo stadio con molta probabilità sarà vicino ad essere tutto esaurito,viceversa se si gioca contro una squadra nelle ultime posizioni del ranking.
Man mano che si raggiungono fasi avanzate della competizione i biglietti saranno sempre più costosi e gli stadi sempre più pieni.

Capitolo 7: Alcune videate dell'applicazione realizzata e link al video dimostrativo del software

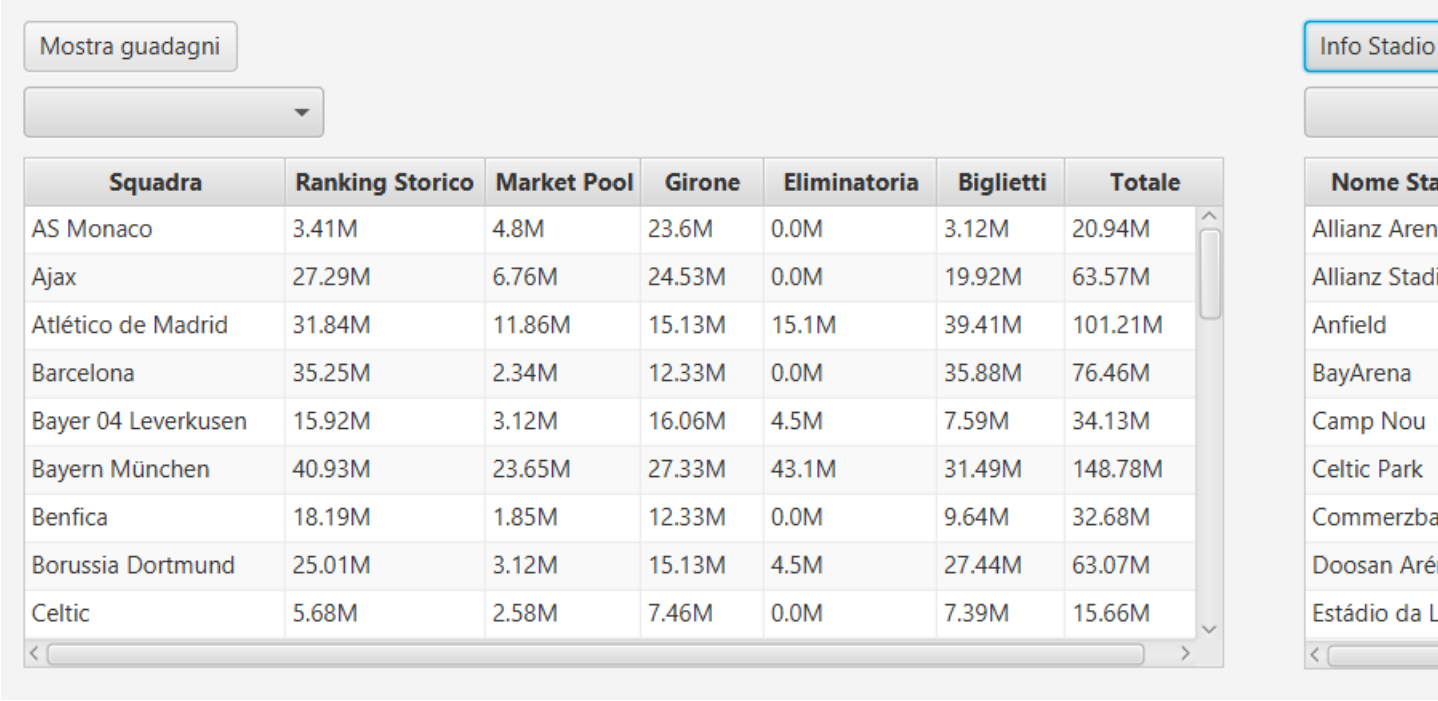
Giornata	Casa	Risultato	Ospite
1	Eintracht Frankfurt	1 - 4	Bayern München
1	Chelsea	3 - 2	Barcelona
1	Borussia Dortmund	2 - 1	Trabzonspor
1	Olympiacos	2 - 3	Tottenham Hotspur
1	Viktoria Plzen	2 - 3	Porto
1	Milan	4 - 1	Rangers
1	Juventus	0 - 2	Bayer 04 Leverkusen
1	Dinamo Zagreb	2 - 2	Napoli
1	RB Salzburg	2 - 1	Sevilla

Dai due bottoni 'Crea Girone' e 'Simula Girone' è possibile ottenere i risultati della prima parte della cometizione, e poi possibile tramite i menù a tendina selezionare le giornate che si vogliono visualizzare oppure scegliere una squadra e vedere tutti i suoi risultati.
E' sempre possibile decidere di creare un nuovo girone oppure calcolare nuovi risultati

In modo simile il bottone ‘Simula fase eliminazione diretta’
la simulazione dei risultati della fase finale anche qui è possibile selezionare un turno e vedere le partite oppure scegliere una squadra e poter visualizzare i suoi risultati.
E’ sempre possibile effettuare una nuova simulazione avendo così nuovi risultati e un nuovo vincitore.

A/R	Turno	Casa	Risultato	Ospite	Totale	Supera turno
A	PlayOff	Napoli	0 - 4	Sporting Lisboa		
R	PlayOff	Sporting Lisboa	5 - 2	Napoli	9 - 2	Sporting Lisboa
A	PlayOff	Olympique de Mars...	2 - 3	Milan		
R	PlayOff	Milan	1 - 0	Olympique de Mars...	4 - 2	Milan
A	PlayOff	Real Madrid	3 - 2	Juventus		
R	PlayOff	Juventus	2 - 3	Real Madrid	4 - 6	Real Madrid
A	PlayOff	Internazionale Milano	4 - 1	Eintracht Frankfurt		
R	PlayOff	Eintracht Frankfurt	2 - 0	Internazionale Milano	3 - 4	Internazionale Milar
A	PlayOff	Bayer 04 Leverkusen	3 - 0	PSV Eindhoven		

Politecnico di Torino



In questa immagine è presente la parte informativa e statistica dell'applicazione dove vengono riportati i guadagni di ogni team e le statistiche riguardanti lo stadio.
I menù a tendina permettono di scegliere una squadra o uno stadio per vederne singolarmente guadagni oppure statistiche.

Classifica

Pos	Squadra	Pti	GF	GS	DR
1	Tottenham Hotspur	22	24	12	12
2	Bayern München	19	30	14	16
3	Manchester City	19	24	9	15
4	Paris Saint Germain	18	25	11	14
5	Liverpool	18	23	10	13
6	Chelsea	17	20	15	5
7	Ajax	16	19	16	3
8	AS Monaco	15	20	13	7
9	Napoli	15	20	14	6
10	Olympique de Marsei...	15	17	14	3
11	Real Madrid	14	24	16	8
12	Internazionale Milano	14	23	17	6
13	Bayer 04 Leverkusen	14	18	13	5
14	Atlético de Madrid	13	21	18	3
15	Borussia Dortmund	13	14	14	0
16	Porto	13	12	12	0
17	RB Leipzig	11	24	17	7
18	Barcelona	10	18	17	1

La squadra vincitrice è:

In quest'ultima parte sono presenti i risultati dell'applicazione in alto è presente la classifica generata in seguito alla simulazione del girone,
mentre in basso è presente lo scudetto della squadra vincitrice della competizione.

Politecnico
di Torino

Politecnico
di Torino

Capitolo 8:Valutazioni sui risultati ottenuti

I risultati ottenuti sono in linea con quelli attesi, è stato possibile ottenere una buona simulazione di una stagione, dal punto di vista economico i guadagni delle squadre vincitrici sono in linea con l'ultima vincitrice effettiva che è stata il Real Madrid che con la vittoria della Champions League nella stagione 21/22 si è portata a casa un bottino di circa 130 milioni di euro se si osservano gli screen nelle pagine precedenti si vede come il Bayern Munchen con la vittoria della competizione si è aggiudicata circa 150 milioni.
Inoltre simulando più volte la competizione si ottiene una buona varietà vincitrici diverse il che dal mio punto di vista rende non banale e scontato l'applicazione.

Questa relazione tecnica è rilasciata con licenza Creative Commons BY-NC-SA.